Data Acquisition and Data Analysis in Skin Measurements

Zhang Xu

School of Engineering

London South Bank University

Supervisors:

Dr. Perry Xiao

Submitted: April 2017

To my Parents

Content

Acknowledgement

First, I wish to express my sincere thanks to Dr. Perry Xiao, my supervisor, for his super guidance, inspiration and encouragement in my research.

I am also very grateful to the colleagues in my group for their kindly help and useful discussions.

Thanks also go to my parents for their care and encouragement.

Finally, I like to thank my friends and everyone who had ever spent effects in helping me to get this studying opportunity.

Abstract

The water in skin, particularly in stratum corneum, the outmost skin layer, is very important. However, to measure it is very difficult. OTTER (Opto-thermal transient emission radiometry), AquaFlux, and Epsilon are three novel technologies specifically developed by our research group for such measurements. This thesis describes the latest development of the technologies. The main focus is to develop a state of the art OTTER data acquisition and data analysis software programme based on Pico Technology. The new software programme offers a range of benefits, such as faster sampling rate, better ADC resolutions, more user friendly interfaces, and more functions and features. The modularized dynamic link library based approach, also means it is easier to maintain, update and expand. With the new OTTER software programme, a multiple wavelength detection is carried out, and an enhanced segmented least squares (SLS) fitting is proposed. The results show that by combining multiple wavelength detection and enhanced SLS fitting, OTTER is capable to detect different types of skin damage, and the presence of topically applied solvents as well as depth distribution of solvents within skin. The second main focus is skin characterization by using AquaFlux and Epsilon, as well as other skin measurement instruments, such as Corneometer, Moisture Checker, Hydratest Beauty Pro, and ProscopeHR2 digital microscope. The aims are to have better understanding on the instrument performances, the correlations between instruments, as well as skin damage assessments, and *in-vitro* and *in-vivo* skin solvent penetration. The results show that the combination of AquaFlux and Epsilon can be very useful for skin characterizations, and the ratio of skin hydration and TEWL can be a better index for skin barrier function. The results also show that the Epsilon capacitive occlusion curves can be potentially used for skin damage assessments, as it can detect both the scale of the damage, and the type of the damage. The skin solvent penetration results show that Epsilon can be effectively used for measuring different types of solvents, and a method to quantify solvent concentration in skin has been developed.

Chapter 1 Introduction and Skin Research Review

The human skin is the outer covering of the body. In humans, it is the largest organ of the integumentary system. The skin has multiple layers of ectodermal tissue and guards the underlying muscles, bones, ligaments and internal organs. Human skin is similar to that of most other mammals, except that it is not protected by fur. Though nearly all human skin is covered with hair follicles, it can appear hairless ("Human skin, Wikipedia, 2017"). There are two general types of skin, hairy and glabrous skin. The adjective cutaneous literally means "of the skin" (from Latin cutis, skin). There are many attempts which have been made to measure the water content in skin in-vivo, such as electrical capacitance and electrical conductance measurements, evaporimetry, infrared spectrometry, ultrasound, magnetic resonance imaging and wide-angle x-ray diffraction. In this chapter, a general introduction and review of skin structure and skin hydration will be presented.

1.1 Skin Histology

Skin is a barrier to provide protection against environmental hazards, such as radiological and microbial attacks, physical and chemical. It also works as a sensor to transmit environmental information such as pain, cool to the nerve center and depot to store excess food as fat for future use (Swarbrick et al, 1995). For an average 70 kg human being, the surface area is about $1.8m^2$. A typical square centimeter covers 10 hair follicles, 12 nerves, 15 sebaceous glands, 100 sweat glands, 3 blood vessels with 92 cm total length, 360 cm of nerves and 3×10^6 cells (Prameela, 2010).

Human skin may be subdivided into three mutually dependent layers: the fatty subcutaneous layer (hypodermis), the overlying dermis, and the epidermis, the outermost layer of the skin (Figure 1.1, from Van De Graaff KM, Fox SI, 1995). Human skin displays two main types. Hairy skin encloses hair follicles and sebaceous glands, but there are no encapsulated sense organs.

1.1.1 The Epidermis

The most superficial layer of human skin is the epidermis (Figure 1.2, from Van De Graaff KM, Fox SI, 1995). The cells of the epidermis are produced from the basal layer and migrate to the exterior, undergoing keratinization to form the outermost layer, the stratum corneum. It varies in thickness from 0.8 mm on the palm to 0.06 mm on the eyelids. The epidermis can be subdivided into five layers:

(1) The stratum basale (stratum germinativum).

The stratum basale is characterized by intense mitotic activity and is responsible. The cell division occurs in this layer. New keratinocytes which move upwards pushing the older above towards the surface of the skin is generated by the basal keratinocytes.

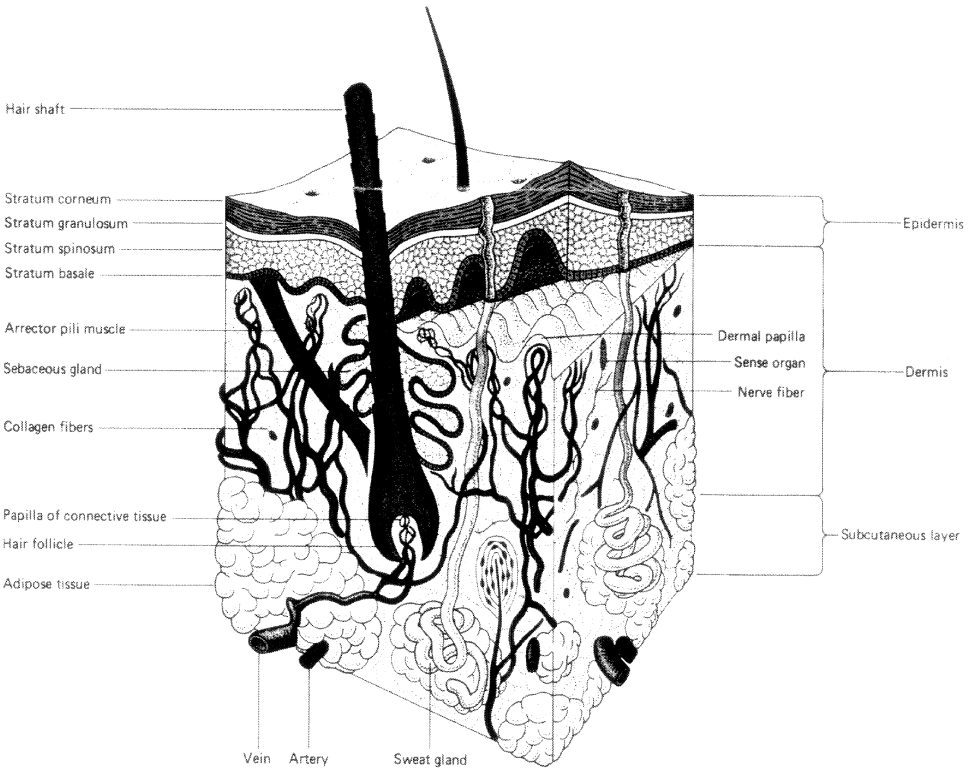


Figure 1.1 Structure of Human Skin, from Van De Graaff KM, Fox SI.

(2) The stratum spinosum.

The cells begin to flatten and their nuclei shrink in this layer. The epidermis of areas subject to continuous friction and pressure (such as the sole of the feet) has a thicker stratum spinosum with more abundant tonofibrils and desmosomes.

(3) The stratum granulosum (granular layer).

It is characterized by 3-5 layers of flattened polygonal cells containing centrally located nuclei and cytoplasm. Its shape is membrane-coating granule, ovoid or rod like. The cells generate keratohyalin granules in this layer.

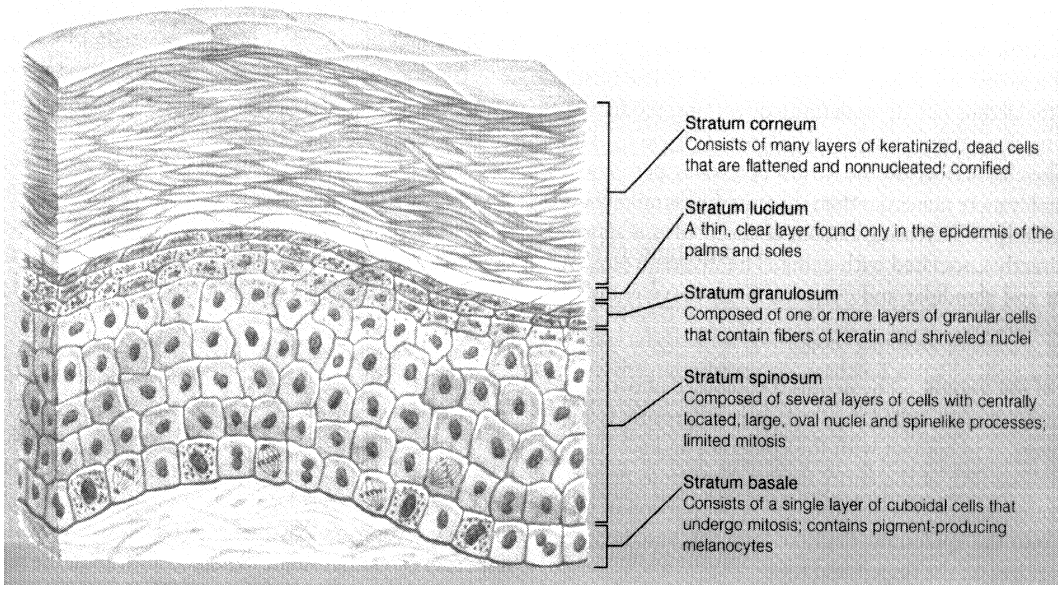


Figure 1.2, Structure of Epidermis, from Van De Graaff KM, Fox SI

(4) The stratum lucidum.

It is a thin translucent layer and is generated in the palm of the hand and the sole of the foot, an anatomically distinct, poorly staining hyaline zone.

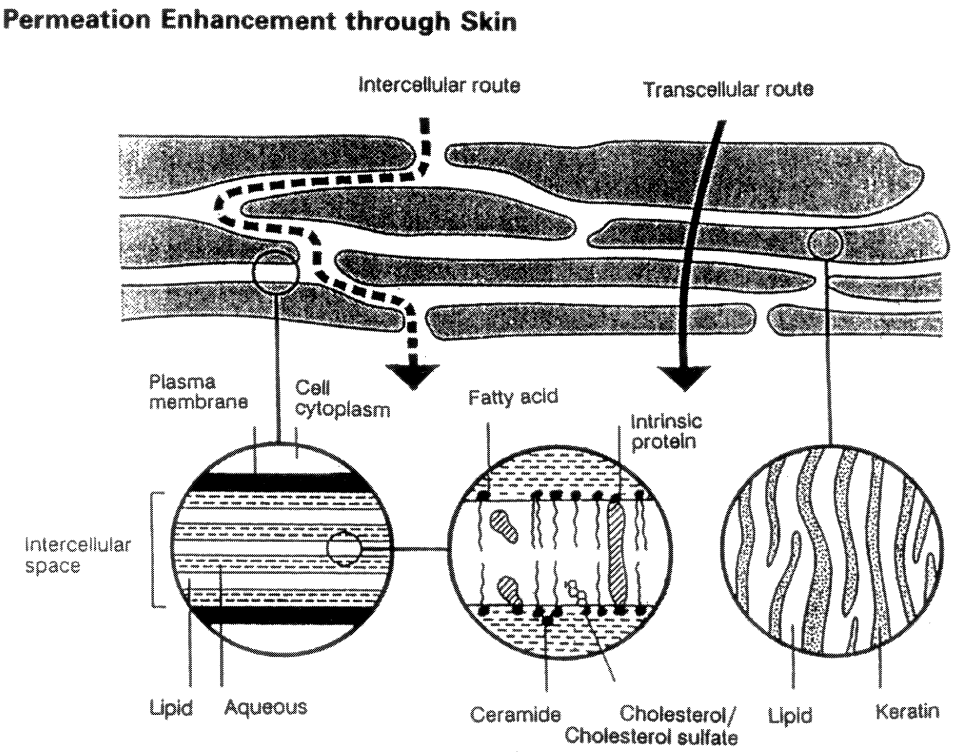


Figure 1.3, The "brick and mortar" structure of the stratum corneum (Diagram from William AC and Barry BW, 1992).

(5) The stratum corneum (The horny layer).

It is the most superficial layer of the skin and comprises 10 to 15 layers of dead cells with approximately 10-20 μm thickness. The structure of the stratum corneum may be represented as a brick and mortar model which provides two micro pathways for the trans-epidermal drug diffusion (Figure 1.3, *William AC and Barry BW, 1992*). When the cells arrive at the SC, they are fully keratinized and dead.

The Epidermis contains 3 less abundant cell types: Melanocytes, the Langerhans Cells, and the Merkel Cells.

1.1.2 The Dermis

The dermis shown in Figure 1.4 is the second layer of the skin. The connective tissue that supports the epidermis and binds it to the subjacent layer forms the dermis. The thickness of the dermis varies depending upon the region of the body, reaching its maximum of 3 mm on the soles of the feet (Junqueira LC, 1977).

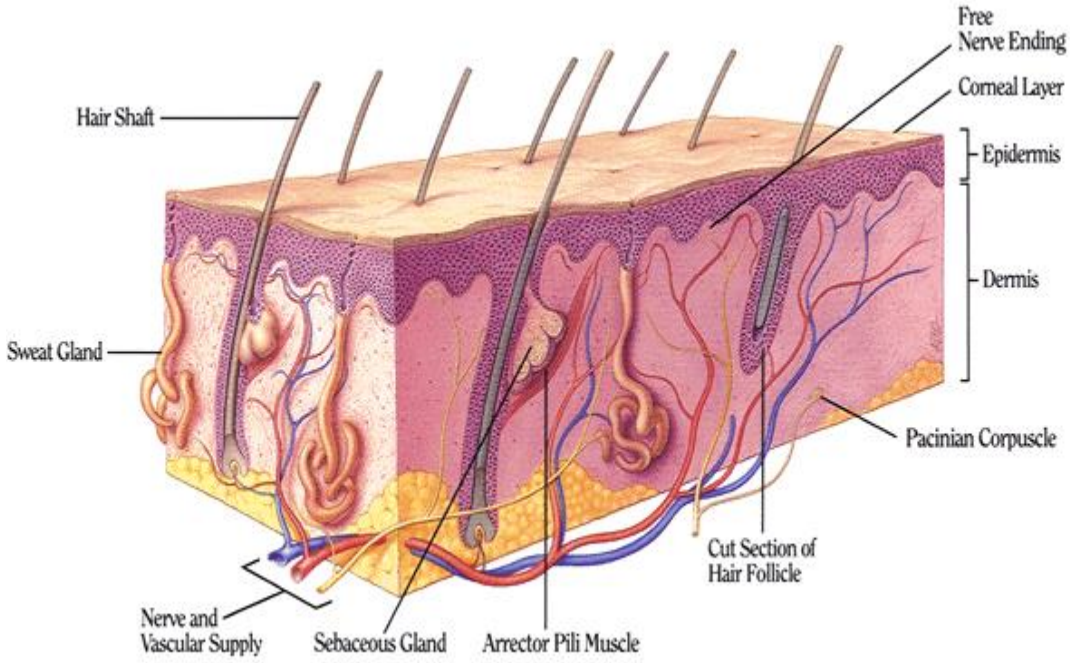


Figure 1.4 Structure of dermis

The dermis is composed of two layers. One is the immediately subepidermal papillary layer, and another is the deeper, more massive and denser reticular layer. The dermis has a rich network of blood vessels which regulates temperature and pressure, delivers nutrients to the skin and removes waste products, mobilizes defense forces, and contributes to skin color. The dermis also contains some epidermal derivatives, the hair follicles, sweat and sebaceous glands. A rich supply of nerves is found in the dermis, and the effector nerves to the skin are postganglionic fibers of the ganglia of the paravertebral chain.

1.1.3 The Subcutaneous Tissue

The subcutaneous fat (hypodermis, subcutis) spreads all over the body. The size of this layer varies throughout the body and from person to person. The subcutis provides a thermal barrier and a mechanical cushion; it is a site of synthesis and depot of readily available high-energy chemicals (Barry BW 1983). It carries the major blood vessels and nerves to the skin and may contain sensory pressure organs (William AC, 1992).

1.1.4 Skin Functions

The skin performs many varied functions, the following presents a brief digest of its biological role (Barry BW 1983).

1. To contain body fluids and tissues --the mechanical function.
2. To protect from potentially harmful external stimuli--the protective or barrier function: (a) micro-organisms; (b) chemicals; (c) radiation; (d) heat; (e) electrical barrier; or (f) mechanical shock.
3. To receive external stimuli, i.e., to mediate sensation: (a) tactile (pressure); (b) pain; or (c) heat.
4. To regulate body temperature.
5. To synthesize and to metabolize compounds.
6. To dispose of chemical wastes.
7. To provide identification by skin variations.
8. To attract the opposite sex.
9. To regulate blood pressure.

1.2 Skin Hydration

A water profile in skin calculated from Energy Dispersive Spectrometry Spectra (Warner *et al*, 1988) is shown in Figure 1.5. The water in skin, and its relationship to skin function will be introduced in the following section.

1.2.1 Skin Hydration

The relationship between relative humidity and water contents of the stratum corneum is shown in Figure 1.6 (Spencer TS et al, 1975). According to Takenouchi M. At 0% relative Humidity (RH), stratum corneum still holds about 0.05 g/g water.

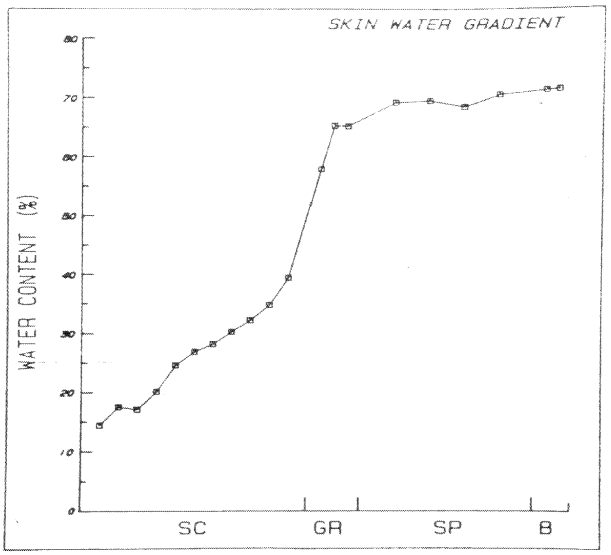


Figure 1.5 Calculated Water profile across human skin (Warner et al, 1988). Vertical scale is percent water expressed as grams of water per total grams (water plus dry mass) of tissue. SC stands for Stratum Corneum, GR stands for Stratum Granulosum, SP stands for Stratum Spinosum, and B stands for Stratum Basale.

1.2.2 Water Concentration Profile and TEWL

Stratum corneum is dry at its surface because of the dry external environment and wet at its base where it is contact with the deeper fully hydrated part of the epidermis (Blank IH, 1984). Therefore, a concentration gradient which results in a continuing diffusion of water from within the body through the skin to the environment, trans-epidermal water loss (TEWL) will generate within the stratum corneum. The TEWL for normal skin is about 4~8 g/m^2 per hour (Tsai J et al, 1990).

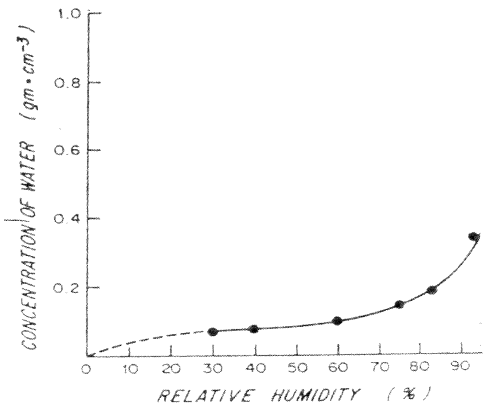


Figure 1.6 Concentration of water in stratum corneum in equilibrium with air at 30 ºC as a function of relative humidity (Spencer TS et al, 1975).

1.2.3 The Water in Stratum Corneum

In the stratum corneum, there are three types of water: tightly bound water, loosely bound water and free water. The experiments from (Anderson RL et al 1973) show that the hydration and dehydration rates of free water are lower than that of bound water. This suggests that free water is located primarily intracellularly, with the cell envelope the major barrier to its loss, while the bound water is primarily intercellular or on the cell surface.

1.3 Latest Skin Measurement Technologies

Many attempts have been made to measure the properties in skin in-vivo and in-vitro, such as infrared, trans-epidermal water loss (TEWL), electrical conductance measurements for basic, and so on. In recent years, there are also some advanced methods which can be used in skin measurement.

1.3.1 Skin Conductance Measurement Technologies

Ogorevc et al (2013) investigated the uncertainty of skin conductance measurement. Calibrated commercial devices are chosen for analysis. The uncertainty (UUT) which is the total uncertainty, was analyzed by several ways, such as $u_{reference}$ (depends on its valid calibration certificate and repeatability, producibility, resolution and drift of reference instrument), $u_{repeatability}$, $u_{producibility}$, $u_{temperature}$, and $u_{resolution}$ (all depend on skin conductance measurement by meter). Figure 1.7 shows Contributions to standard measurement uncertainty of a skin conductance meter, conductance unit is micro-Siemens (uS), the bars mean the values of each uncertainty part, large value mean the large influence to uncertainty and small value means the small influence to uncertainty. The results show that repeatability and reproducibility are the major problem because small changes in measurement method may cause big contribution to measurement uncertainty, but the resolution, temperature of skin conductance meter is not the critical problem of measurement uncertainty.

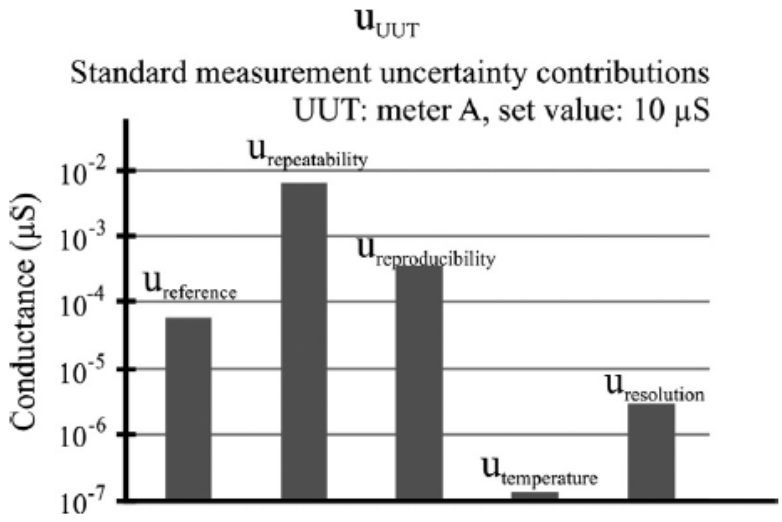
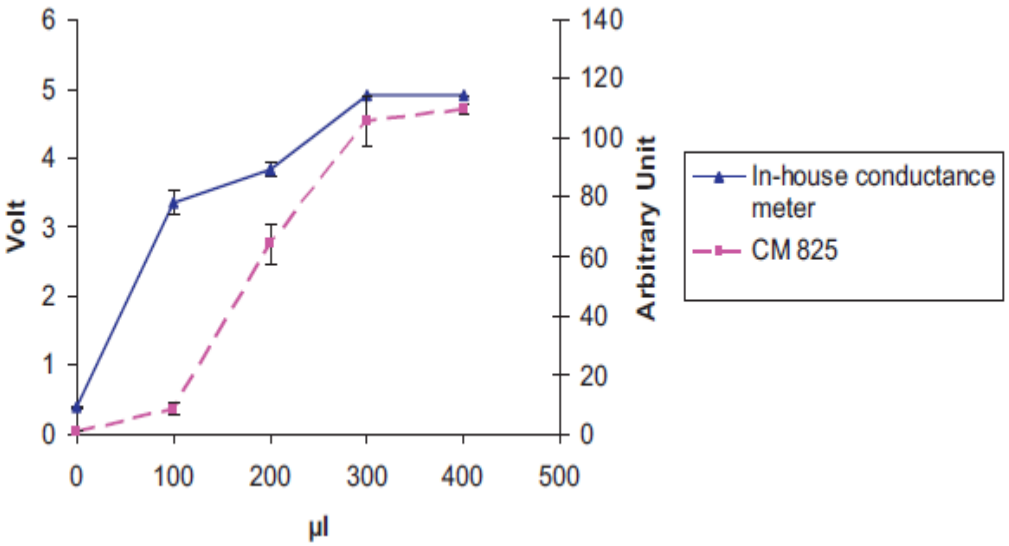
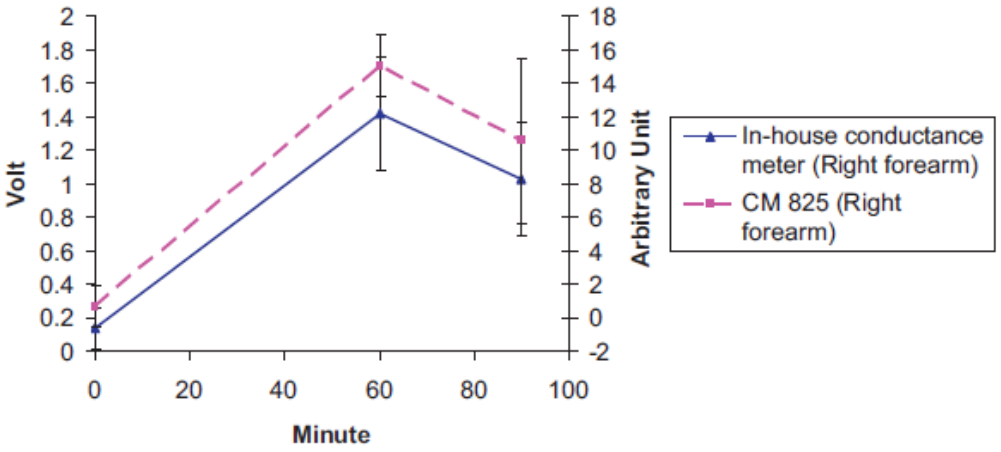


Figure 1.7 Contributions to standard measurement uncertainty of skin conductance meter A at set value 10 uS, conductance unit is micro-siemens (uS), bars mean the values of each uncertainty part, $u_{repeatability}$ had the largest value and $u_{temperature}$ had the smallest value (Qgorevc et al, 2013).

Hamed et al (2012) designed a simple conductance meter (in-house conductance meter) for vivo and vitro skin hydration measurement and investigated the correlation with Corneometer 825 (CM 825). Two measurements were chosen for the research, in-vitro water sorption test and in-vivo measurement. For the in-vitro water sorption test, a cellulose filter paper was chosen, distilled water was pipetted into the filter paper with a dose of 100ul each up to a maximum of 400ul, measurements using both instruments were taken 30s after adding the specific volume of water to ensure sorption. The result showed that the values of both in-house conductance meter and CM 825 gradually increased with water impregnating the filter paper until a steady state of both values was reached after 300ul (Figure 1.8 (a)). For the in-vivo measurement, two skin areas were selected on the forearm, each area was treated with moisturizer and measurement by both instruments were taken at baseline, 60, 90 min after product's application. The result showed that the results increased before 60 min of both instruments and slight decreased at 90 min (Figure 1.8 (b)). All the results showed that the in-house conductance meter had the similar curve trend with CM 825 in in-vitro and in-vivo which can conclude that the simple inexpensive conductance probe has the good records on all the measurements, and with a good correlation to the results of CM 825.



(a) In-vitro sorption test results, x axis represents the volume of distilled water (unit: ul), y axis represents the measured values of both instruments, the blue solid line represents the value of in-house conductance meter (unit: V), the pink dotted line represents the value of CM 825 (unit: arbitrary unit), both instruments have similar curve trend.



(b) In-vivo measurement results, x axis represents the measurement time (unit: min), y axis represents the measured values of both instruments, the blue solid line represents the value of in-house conductance meter (unit: V), the pink dotted line represents the value of CM 825 (unit: arbitrary unit), both instruments have similar curve trend.

Figure 1.8 results of all tests by using in house-conductance probe and CM 825 (Hamed et al, 2012).

Dooren et al (2012) investigated responsiveness and similarity with 16 different recording positions of skin conductance (SC) while watching emotional film fragments. Skin conductance is one of the electrical properties of skin which depends on the sweat secretion from sweat glands (Benedek, et al, 2010),

therefore, different position has different distribution of sweat glands. In this experiment, 17 volunteers were chosen, and skin conductance recordings were made from 16 different positions on the body of each volunteer. The data was recorded in 3 ways: mean skin conductance level (SCL), the number of skin conductance response per minute (SCRs), and the sum of the skin conductance response amplitudes per minute (S-AMPL). The results showed that SCL and SCRs had the high values for forehead, foot, finger, and shoulders, SCL low value was found for arm, armpit, thighbone, SCR low value was found for arm, armpit, thighbone and back, which scores were very much in line with each other. For S-AMPL, foot gave the high scores, follow by finger, wrist, and shoulders.

1.3.2 Trans-Epidermal Water Loss (TEWL) Measurement

Anthonissen et al (2013) investigated the repeatability and reproducibility of repeated elasticity and TEWL measurements on burn scars (contain grafted and spontaneously healed scars) and healthy skin and differences between burn scars and healthy skin by using DermaLab®. In this experiment, two observers did this measurement, 30 active burn scars and healthy skin were chosen and DermaLab® was used to measure the elasticity and TEWL by two separate probes. It used the means of intra - class correlation coefficient (ICC) to examine intra-observer reliability, which means first observer did the measurement and analyzed the reliability (repeatability) and inter-observer reliability which means second observer did the same measurement and analyzed all the reliability include first observer did (producibility), it was considered that the measurement had good repeatability and producibility when ICC>0.75, elasticity and TEWL were measured in 48 times (N) on grafted and spontaneously healed scars respectively and 96 times on healthy skin. The results (Figure 1.9) showed that both elasticity and TEWL had the good repeatability and producibility and there were differences between healthy skin and burn scars in elasticity and TEWL values.

	Type	ICC (95% CI)
Intra-observer reliability	Grafted scar	0.90 (0.74–0.96)
	Spontaneously healed scar	0.93 (0.81–0.97)
	Normal skin	0.93 (0.79–0.97)
Inter-observer reliability	Grafted scar	0.86 (0.66–0.95)
	Spontaneously healed scar	0.93 (0.80–0.97)
	Normal skin	0.93 (0.81–0.98)

	N	Mean
Grafted scar (MPa)	48	11.20
Spontaneously healed scar (MPa)	48	9.01
Normal skin (MPa)	96	6.15

ICCs for the levels of the factor grafted scar, spontaneously healed scar and normal skin were used to quantify the intra-observer reliability (repeatability) and inter-observer reliability(producibility) of elasticity and mean values of elasticity (MPa) measurements with the DermaLab®, N means the number of measurement.

	Type	ICC (95% CI)
Intra-observer reliability	Grafted scar	0.86 (0.63–0.95)
	Spontaneously healed scar	0.88 (0.70–0.96)
	Normal skin	0.88 (0.71–0.96)
Inter-observer reliability	Grafted scar	0.93 (0.80–0.97)
	Spontaneously healed scar	0.78 (0.42–0.92)
	Normal skin	0.78 (0.50–0.92)

	N	Mean
Grafted scar ($g/m^2/h$)	48	7.42
Spontaneously healed scar ($g/m^2/h$)	48	10.50
Normal skin ($g/m^2/h$)	96	6.48

ICCs for the levels of the factor grafted scar, spontaneously healed scar and normal skin were used to quantify the intra-observer reliability (repeatability) and inter-observer reliability(producibility) of TEWL and mean values of TEWL ($g/m^2/h$) measurements with the DermaLab®, N means the number of measurement.

Figure 1.9 results of ICC level and mean values of elasticity and TEWL measurements (anthonissen et al, 2013)

Machado et al (2010) investigated the relationship between in-vivo transepidermal water loss (TEWL) and the diffusional permeation pathlength through the stratum corneum (SC). 90 volunteers were invited to measure their TEWL and corneocyte surface areas for six skin sites (forehead, wrist, forearm 1, 2, elbow, and abdomen). The results (Figure 1.10) showed that the abdomen had the largest corneocytes and least TEWL, and forehead had the smallest corneocytes and most TEWL, other skin sites were in the middle. It can be concluded that when TEWL approached 0, corneocytes were very large, and with small corneocytes and short permeation pathlength, TEWL had high value, which had proved the relationship between TEWL and pathlength.

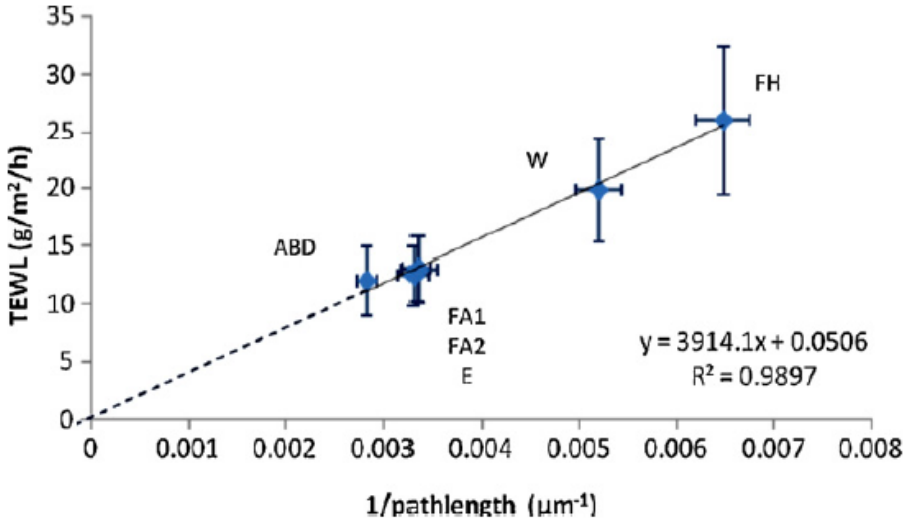


Figure 1.10 TEWL $(g/m^2/h)$ ***vs. 1/pathlength*** (um^{-1})***, the trend line is achieved trhough linear regression. (FH: forehead; W: wrist; FA: forearm; E: elbow and ABD: abdomen) (Machado et al, 2010)***

Makin A. et al (2015) used tape stripping for epidermal damage and measure TEWL to examine the variation. In the experiment, 4 different normal skin parts of 2 minipigs were chosen to tape strip, and TEWL was measured by two different ways, a conductance meter and open chamber TEWL meter. The results showed that, after tape stripping, both meters had the similar increased TEWL values, and it is correlated with the number of tape stripping occurs.

1.3.3 Infrared Technologies for Skin Measurement

Quesada et al (2015) investigated the differences of skin temperature during cycling by using infrared thermography camera (ITC) and thermal contact sensor. Seven skin sites (chest, abdomen, neck, back, waist, thigh, calf) were chosen on 14 healthy male cyclists in this experiment, the temperature was measured on all seven sites by both ITC and thermal contact sensor at 3 positions, pre-cycling (at the end of the thermal adaptation phase: 15 min), post-cycling (immediately after cycling phase: 48 min) and post-cooling (at the end of the cooling-down phase: 10 min). For ITC measurements, some skin sites were called small range of interests (ROIs) (i.e. neck), and some skin sites were called large ROIs (i.e. back) The results (Figure 1.11) showed the temperature of all body skin sites results. The results described that before cycling, IRT has the same values with the thermal contact sensor, and after the cycling, IRT has the lower values with the thermal contact sensor because water (sweat) would be formed a continuous layer and it shielded the infrared radiation, but IRT has the higher values than thermal contact sensor after cooling down because the skin kept dry after cooling down.

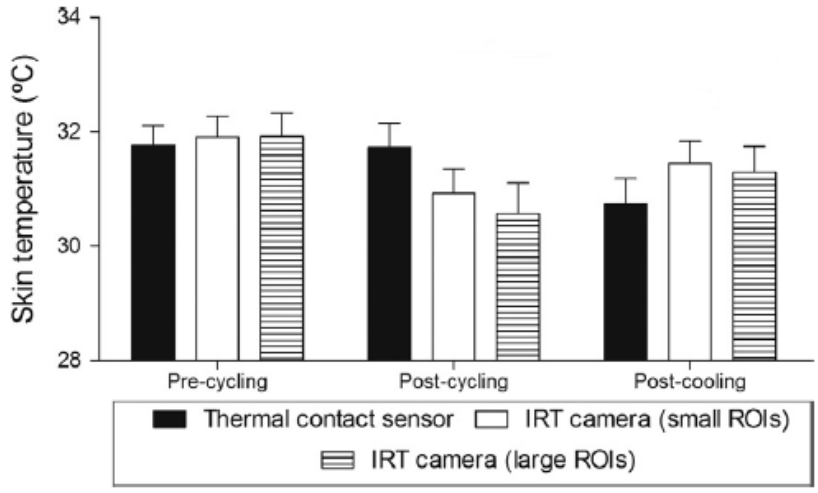


Figure 1.11 Mean with skin temperature of the all body locations analyzed in each measurement time: Pre-cycling, Post-cycling and Post-cooling. Significant differences between measurement methods (thermal contact sensors and IRT camera defined by small and by large ROIs) for each measurement time (Quesada et al, 2015)

Zamora-Rojas et al (2014) studied how light go into the agri-food tissues. In this paper, it simulated the diffuse reflectance spectra of pig skin for different

source-detector distances with the wavelength range from 1150nm to1850nm, because all the optical properties had good performance between these range of wavelength (Zamora-Rojas et al, 2013a, b). Monte Carlo method was used for this simulation. Figure 1.12 showed the average photon visit depth for different source-detector distances (from 0.6mm to 5.4mm with the step of 0.6mm). The results showed that the photon visit depth increased with the distance increasing, photon can visit the maximum depth with wavelength between 1200nm-1300nm because of the minimum absorption of water and lipids, the photon visit depth decreased to the bottom (which can be considered as the surface of skin) at wavelength nearly 1450nm which was related with the strong absorption of water, and it failed to show the surface because the water absorption was too high and no photon reached to surface when the distance was over 3.6mm. it can be concluded that photon visit depth depended on the source - detector distance, larger distance can make light going deeper into the skin tissues, and the surface of skin (dermis) response the most of the photon absorption.

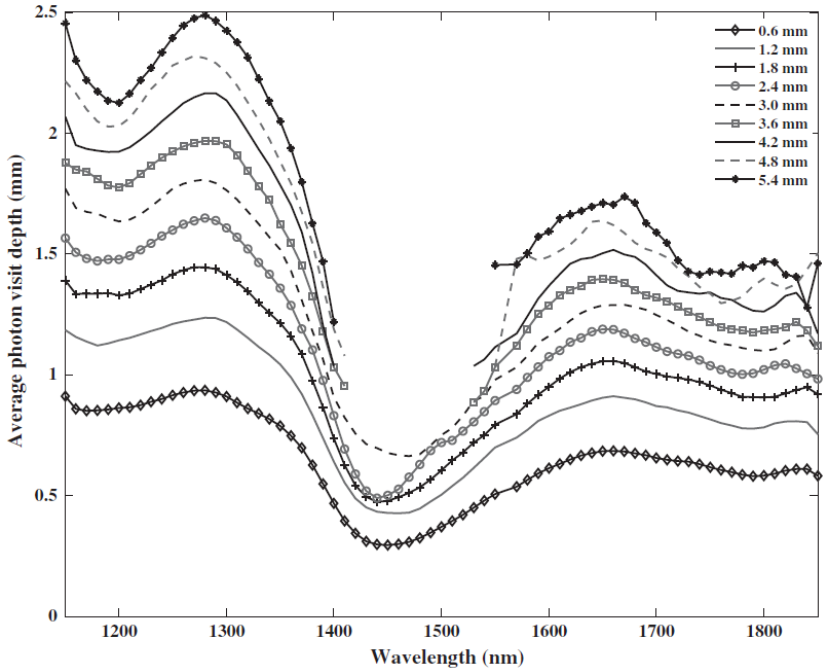


Figure 1.12 Average photon visit depth (y axis) for different source–detector distances, every curve represents the depths with different source-detector distances from 0.6mm to 5.4mm (shown upper right corner) with wavelength from 1150nm-1850nm (x axis) (Zamora-Rojas et al, 2015)

Soerensen et al (2014) investigated the emissivity of pig skin at the shoulder, ear base, and udder, in order to determine whether was an effect with hair and blood

perfusion on the emissivity. In this paper, three parts (shoulder with and without hair, ear base and udder) of ten sows were chosen to measure the emissivity by using infrared thermography camera (ITC) respectively when they were alive (with blood perfusion) and dead (without blood perfusion). The results showed that ear base and udder had the close emissivity, but the shoulder with hair had a lower emissivity than that of the ear base and udder. Emissivity of three skin parts with no blood perfusion was lower than those when perfused with blood.

1.3.4 Skin Penetration Measurements

Solvent penetration is one of the most popular and important researching area in skin measurement. The understanding on solvent penetration can be beneficial for transdermal drug delivery studies.

Paleco et al (2014) investigated the potential of microneedles combined with lipid microparticles (LMs) to improve the in vitro skin penetration of the antioxidant flavonoid, quercetin.

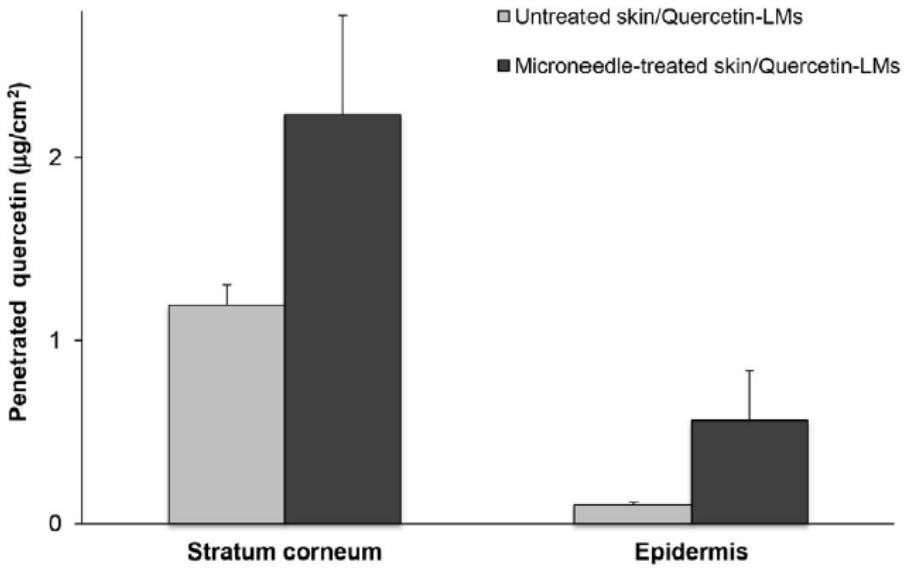


Figure 1.13 In vitro distribution of quercetin in untreated- and microneedle-treated pigskin, y axis represents the penetrated quercetin (ug/cm^2), gray bar represents skin with quercetin-LMs, black bar represents skin with quercetin-LMs combine with microneedle arrays (Paleco et al, 2014)

Figure 1.13 showed the in-vitro distribution of quercetin in untreated and microneedle-treated pig skin. The results showed that when skin was treated by the flavonoid - loaded LMs in combination with microneedle arrays, compared

with that without microneedle arrays, the contents of quercetin in test skin obviously increased in the stratum corneum nearly to 2 times and the epidermis (quercetin was located mainly in epidermis) nearly to 5 times. It can be concluded that the combination of microneedles with LMs can have an efficiency drug delivery for topical administration of quercetin.

Tachaprutimun, et al (2013) investigated the penetration depth of 4 formulation of Garcinia Mangostana Linn (GML) on stratum corneum of porcine ear skin which was a highly suitable model for human skin (Godin, et al, 2007). The four different GML: encapsulated GML in cream (GML-EN cream), free GML in cream (GML-Free cream), encapsulated GML in water base (GML-EN water), and free GML in water base (GML-Free water) were used for measurement. Figure 1.13 showed the penetration depth of GML in 4 different formulations. The results showed that the GML-EN and GML-Free cream penetrated better than GML-EN water and GML-Free water, one important reason was the hydrophobic effect of cream allowed to pass through the hydrophobic stratum corneum to deeper skin but the hydrophilic water cannot pass through stratum corneum very well. It can conclude that the GML in cream type has a good penetration in porcine skin.

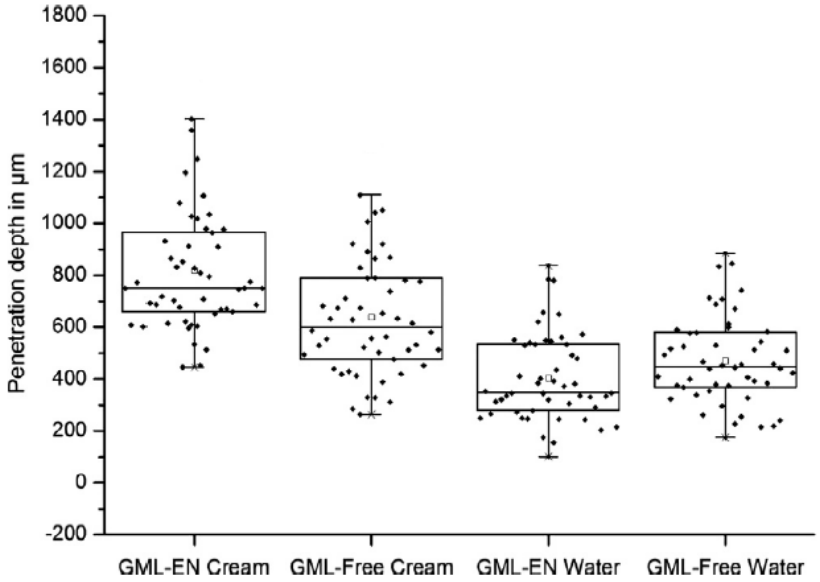
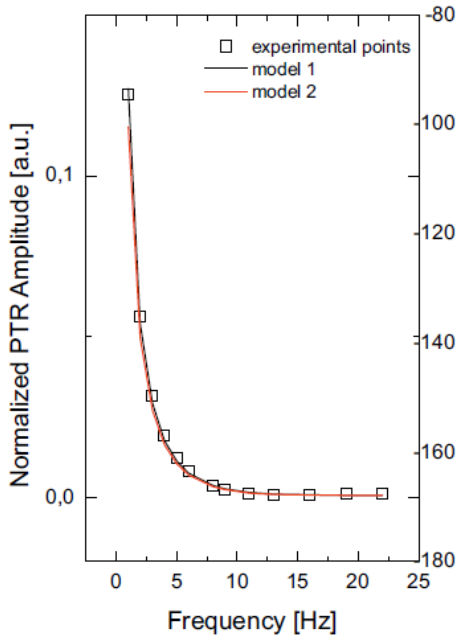


Figure 1.13 Presentation of the penetration depths of GML in 4 different formulations, points represent the penetrated GML, the rectangle represents the main area of penetrated GML and the line in the rectangle represents the average depth of GML penetrated (Tachaprutimun et al, 2013).

Römgens, et al (2016) found that a repetitive microjet injection device can be used for drug delivery, which was much better than normal jet injectors, because of more efficiency and less bruising and pain. The experiments were performed on epidermal and full - thickness skin, the results showed that microjet can deliver drug through the epidermal about 96% with the velocity of over 90m/s, but decrease to 12% when through the full - thickness skin. In vivo experiment on mouse showed that insulin can be absorbed in the systemic circulation by using microjet. Therefore, microjet injection can still use in medication delivery into the skin with good drug delivery efficiency.

1.3.5 Photothermal Technologies

Pawlak M. and Maliński M. (2015) investigated a new method to measure the effective infrared radiometry absorption coefficient and the thermal diffusivity of solid state samples by using the modulation photothermal infrared radiometry technology with the material sample $\mathrm{Cd_{0.94}Mg_{0.96}Se}$ crystal with the cover of thin aluminum foil (which absorbs the infrared radiometry). Two fitting modes: ignored and considered reflections of IR radiation in the sample were used as the theoretical references of the modulation frequency PTR amplitude. The fitting procedure used to determine the thermal diffusivity and effective infrared absorption coefficient solved nonlinear curve-fitting problems in the least-squares sense between experimental data and a theoretical model. Figure 1.14 showed Frequency PTR amplitude and Thermal diffusivities (D_t) and effective infrared absorption coefficients ($\beta_{IR,eff}$) of the sample $Cd_{0.94}Mg_{0.96}Se$. The results showed that the experimental points had the similar trend with both fitting modes, and the R^2 value (close to 1) proved the reliability of thermal diffusivities and effective infrared absorption coefficient of the sample. It can conclude that this modulation PTR method with the thin aluminum absorbing foil can be successfully applied for measurements of the thermal diffusivity and IR absorption coefficient.



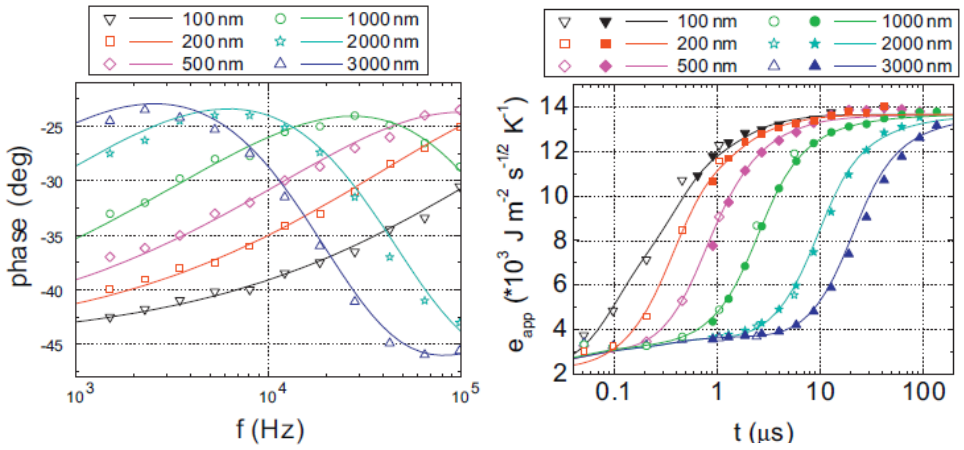
Sample

Parameter	Model 1	Model 2
$Cd_{0.94}Mg_{0.06}Se$		
D_t [cm^2/s]	0.021	0.022
$\beta_{IR,eff}$ [cm^{-1}]	37	40
R^2	0.81	0.91

Figure 1.14 modulation Frequency (x axis) PTR amplitude (y axis) and Thermal diffusivities ($D_t(cm^2/s)$) and effective infrared absorption coefficients ($\beta_{IR,eff}(cm^{-1})$) of the sample $Cd_{0.94}Mg_{0.96}Se$ square are experimental points, gray solid line is theoretical curve of mode 1 and red solid line is theoretical curve of mode 2 (Pawlak et al, 2015)

Kusiak A. et al (2013) investigated the thermal properties of 6 different thickness CrN thin nanocrystalline films by using two different techniques of photothermal radiometry (PTR). Modulated PTR was used to measure the phase, one of the most important elements of modulated photothermal radiometry, for each film thickness. Figure 1.15 (a) showed the phase of different thickness. The results showed that the phase increases with increasing modulation frequency which thickness is from 100 to 500nm, and decreases which thickness is over 1000nm, because the thermal diffusion process occurred in the measurement on thicker films (over 1000nm). Pulsed PTR with two different rise time detectors (fast and slow) was used to measure the thermal effusivity for different film thickness on silicon substrate. Figure 1.15 (b) showed the apparent thermal effusivity of different film thickness. The results showed that the thin film effusivity was the lower constant value present in time from 20ns up to 30us, and the substrate effusivity is the high constant value present in time from 10 to 200us. The effusivity variation between film and substrate value is called a transition region which depends on film thickness for lower thickness in shorter time. It can be concluded that the results showed the good thermal properties on both

technologies.



(a) (Symbols – measured values, lines – calculated values).

(b) (Empty symbols – fast detector, filled symbols – slow detector) and fitted analytical model curves (solid lines).

Figure 1.15 Phase (y axis) for different thicknesses of CrN films with modulation frequency increasing (x axis) (a) and apparent thermal effusivity evolutions (e_{app}, y axis) for different thicknesses CRN films on silicon substrates with the increasing pulsed laser heated time (x axis) (b) (Kusiak et al, 2013)

Kim. Y. J et al (2015) investigated a transdermal delivery system by using composite micelles, which was incorporated Au nanopaticles (AuNPs), the core of heating the micelles, and photothermal effect to the micelles by using light irradiation (λ = 520nm). In vitro skin penetration, 2 sample of pig skin were chosen, placed the 100uL composite micelles encapsulating Indomethacin (IM) on each sample, and one of the samples was irradiated with the LED system for 10 min. In vivo skin penetration, Albino guinea pigs were chosen, and placed the 100uL composite micelles combined with AuNPs on the dorsal portion of the skin, and irradiated the pig for 0, 5, or 10 min by using LED system. The results showed that the AuNPs composite micelles had good viability at high concentrations, and the release of IM - loaded micelles was controlled by the light irradiation time because of the photothermal effects of the encapsulated AuNPs.

1.4 Summary and Research Objectives

In summary, this chapter presents the basic introduction and review of skin characterization such as skin structure, functions, hydration and trans-epidermal water loss (TEWL). Five latest methods: skin conductance measurement,

trans-epidermal water loss (TEWL) measurement, infrared measurement, skin penetration and photothermal measurement have been described.

The literature reviews show that there are many exciting technologies that have been used for skin measurements. But there are still many questions unanswered, particularly in the areas related to skin hydration, skin TEWL, and skin solvent penetrations.

Therefore, the aim of this study is to have a better understanding in the areas of skin hydration, skin TEWL, and skin solvent penetrations. This is achieved through two steps, OTTER (opto-thermal transient emission radiometry) software development and experimental investigations. In OTTER software development, a new data acquisition and data analysis software for OTTER based on Pico technology digital oscilloscope will be designed which will significantly enhance the performances (e.g. less noise, faster, more accurate etc.,) and provide more functions. In the experimental investigations, several experiments will be done to investigate the instrument performance and correlations, skin damage assessment, capacitive imaging occlusion curve effect, solvent penetration, etc.

1.5 Organization of the Thesis

Chapter 1 describes the introduction and skin research review.

Chapter 2 describes the conceptions and principles of different types of skin measurement technologies, e.g. OTTER, Aqua Flux and Epsilon, Corneometer, Moisture Checker, Hydratest, and Proscope HR2 digital microscope, and their functions.

Chapter 3 describes the basic mathematical modelling for OTTER, signal de-noising and Fast Fourier Transform. It also proposed a new enhanced segmented least squares fitting algorithm.

Chapter 4 describes the OTTER software development based on PicoScope 4000 series and PicoScope 2204 PC oscilloscope. The main program was developed by

using C# visual studio 2012, and data analysis libraries are developed by using C# as well as MATLAB, and the results of OTTER experiments.

Chapter 5 describes seven skin experimental investigations, skin damage measurements, tape stripping measurements, instrument performances, correlations of instruments, SLS irritation measurements, capacitive imaging occlusion effects, *in-vivo* and *in-vitro* skin solvent penetration measurements.

Chapter 6 describes conclusions and further work.

Chapter 2 Skin Measurement Instruments Overview

This chapter will present the conceptions and principles of skin measurement instruments used in this study, e.g. OTTER, AquaFlux, Epsilon, Corneometer, Moisture Checker, Hydratest, ProScope HR2 digital microscope, and their functions.

2.1 OTTER

Opto-thermal transient emission radiometry (OTTER) is an infrared remote sensing technique based on photothermal radiometry (PTR) or Opto-thermal radiometry (OTR).

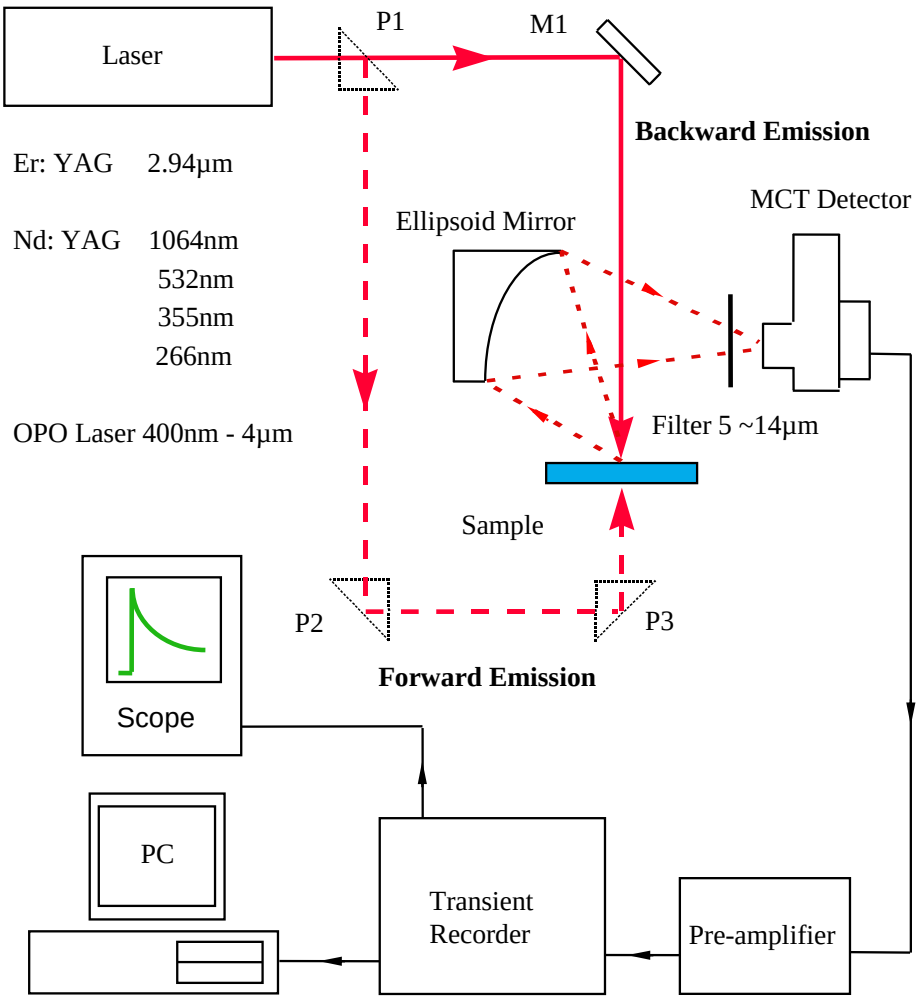


Figure 2.1 Schematic diagram of OTTER. P1, P2, P3 are prisms, and M1 is a mirror.

Figure 2.1 shows a schematic diagram of the OTTER. A Q-switched pulsed laser as the excitation source is used to heat up the sample surface and then generate a heat radiation. Heat radiation, whose wavelength region is in the mid-infrared ~5-15μm, is focused by an aluminum ellipsoidal mirror (it can also use two parabolic mirrors to focus the radiation, but it is harder for setting than using one ellipsoidal mirror) from the sample onto a high speed, liquid nitrogen cooled Mercury Cadmium Telluride (MCT) detector, whose signal is captured by a digital oscilloscope. A PC, linked to the transient recorder as digital oscilloscope through a high-speed parallel interface, is used for signal averaging, data storage, display and analysis. The shape of measured signal is dependent on sample's optical and thermal properties, the thickness of the sample and its layer structure.

OTTER has been used for measurements of paint thickness on steel panels and for monitoring the drying of paints (for example on car bodies), and it also used for measuring the properties of live human skin: its hydration level, and the diffusion of externally applied substances such as cosmetics and sunscreens. Meanwhile, following are the potential application areas:

(1) Skin Hydration and Hydration Depth Profiling.

(2) Depth Resolved FTIR.

(3) Skin Pigments Depth Profiling.

(4) Trans-dermal Drug/Solvent Diffusion.

OTTER has many advantages, such as, non-contact, non-invasive, work on arbitrary surfaces, in-sensitive to color and small movements, but OTTER is hard to move so that it can be only used in the laboratory, and it is too expensive for popularizing.

2.2 AquaFlux

AquaFlux shown in Figure 2.3 is a novel condenser based, closed chamber technology for measuring water vapor flux density from arbitrary surfaces, including *in-vivo* measurements of trans-epidermal water loss (TEWL), skin

surface water loss (SSWL) and perspiration. It uses a cylindrical measurement chamber, with one end open and attached to the sample surface, and another end closed and cooled down to below freezing point.

Figure 2.3AquaFlux

Figure 2.4 shows a simple cutout diagram of the AquaFlux condenser-chamber. When the chamber is brought in contact with the skin surface, the chamber is sealed and it can protect the diffusion zone within it from ambient air movement. The natural convection and other bulk air movements are brought down because of the internal dimensions of the chamber being very small. In fluid dynamics terms, this requires the Rayleigh Number to be below the critical value for its geometry. Under these conditions, passive diffusion remains the only transport mechanism for the water vapor entering the chamber. The condenser controls the absolute humidity in the measurement chamber independently of ambient conditions. It is similar to a vapor sink by forming ice on its surface, thus creating a zone of low humidity in its immediate vicinity. By contrast, the test surface acts as a vapor source, creating a zone of higher humidity in its immediate vicinity. This humidity difference causes water vapor to migrate from source to sink by passive diffusion and creates a diffusion vapor density gradient, from which the flux density can be measured (Imhof R. E., Xiao P. et al, 2014).

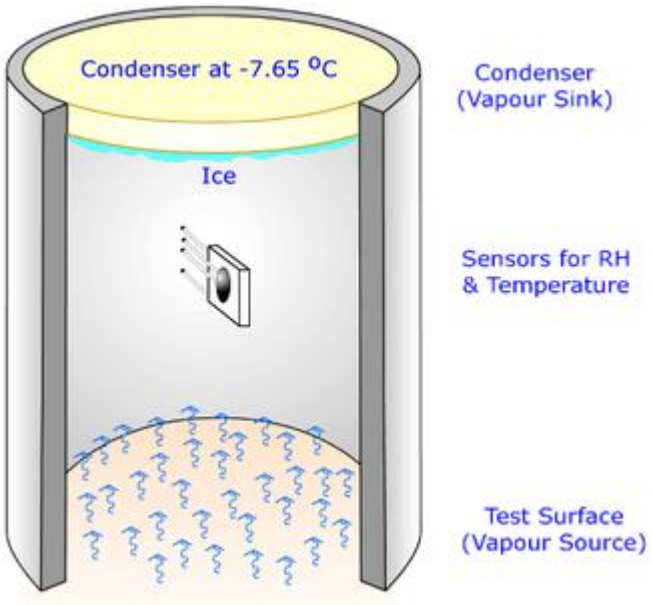


Figure 2.4 a simple cutout diagram of the AquaFlux condenser-chamber

AquaFlux can be used in the following areas:

(1) Skin barrier function, skin recovery, skin disease etc., based on TEWL (Trans-Epidermal Water Loss) measurements.

(2) Membrane penetration and permeation.

(3) Material absorption/desorption.

Compare with similar instruments in the market such as VaporMeter (Delfin Technologies Ltd, Finland), DermaLab (Cortex Technology Ltd, Denmark), etc. AquaFlux has many advantages, for example, high sensitivity, low noise, super reliability, excellent calibration ability and comparability, and above all, the measurement results are independent of external environment, but it is heavy for portable and it needs external power supply.

2.3 Epsilon Permittivity Imaging System

The water content within human skin is very important for its cosmetic properties and its barrier functions, however, to measure it is very difficult. There are some disadvantages of the commercial skin hydration measurement devices, such as, poor repeatability, reproducibility and difficult to calibrate. To address these problems, we have developed a novel hand-held probe for in-vivo skin hydration

imaging based on the capacitance measurement principle, which is called Fingerprint sensor.

Fingerprint sensor (Figure 2.5 (a)) is based on Fujistu fingerprint sensor (Fujistu Ltd), which has 256x300 pixels with 50µm spatial resolution. Each pixel is equivalent of a capacitive sensor, which measures the dielectric constant or permittivity of the sample, it has an 8-bit grey-scale capacitance resolution per pixel (0 - 255). Fingerprint sensor is uncalibrated.

(a) Fingerprint sensor

(b) Epsilon permittivity imaging system

Figure 2.5 Development of capacitive imaging system

Based on the Fingerprint sensor, the new Epsilon permittivity imaging system has been developed (Imhof R. E. and Xiao P., 2012). The Epsilon differs from other similar systems in its calibrated, linear response to near-surface dielectric permittivity, see Figure 2.6. The linear response is important because hydration is linearly related to permittivity. The calibration ensures consistency from

instrument to instrument and from time to time. With calibrated Epsilon imaging systems, we can measure the absolute dielectric permittivity of the material (Zhang. X. et al, 2016).

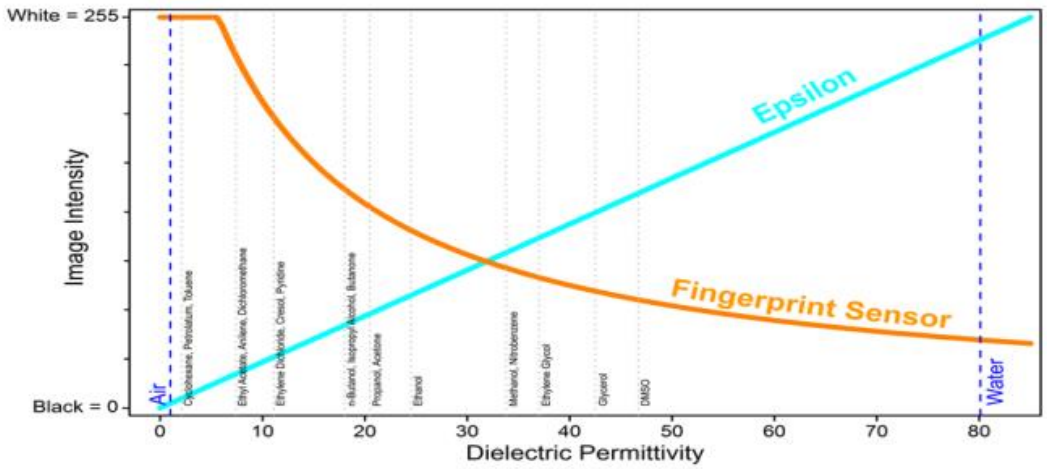


Figure 2 6The Epsilon Calibration curve and linear response to dielectric permittivity

Epsilon shown in Figure 2.5 has many functions:

(1) By processing those images which generated from Epsilon using a dedicated software programme with purposely designed mathematical algorithms, we can produce skin hydration images, skin micro relief images and skin 3D surface profile images.

(2) It can also be used for study materials.

(3) With Epsilon, we can also study membrane penetration. Apart from water, Epsilon is also sensitive to most the solvents using in pharmaceutical studies.

2.4 Corneometer

The Corneometer (Courage+Khazaka electronic GmbH, Germany) measures in arbitrary units from 0 to 120. The units have been very well establishes as so called "Corneometer units" (Corneometer®, Courage+Khazaka electronic GmbH, 2003).

The penetration depth of the electric scatter field is demonstrably very small, the

total measurement depth is approx. 40mm (Figure 2.7).

Figure 2.7Corneometer
(Source:
http://www.courage-khazaka.de/index.php/en/faq-en/faq-scientific-devices/61-corneometer)

2.5 Moisture Checker

The Moisture Checker (Scalar Corporation, Japan) measures the amount of moisture in the skin area with whom he comes into contact. It utilizes a new type of glass-coated ceramic sensor coupled to a highly sensitive micro-processor-controlled circuit which measures the capacitance at the sample site. The Moisture Checker has a special auto-zeroing circuit which minimizes drift so that the readings are both accurate and repeatable (Moisture Checker for the Skin, 1999) (Figure 2.8).

Figure 2.8 Moisture checker
(Source: http://www.tpm-online.de/tpm/webneu/index.php/MoistureChecker.html)

The specification of this moisture checker shows below (table 2.1).

Sensor	Glass-coated ceramic/capacitance/impedance
Reading	0-99.9%+/-0.2%
Operating Temperature	5-35 °C
Weight	Approximately 85 grams (including battery
Size	Approximately 190 x 35 x 19mm
Power	Two AAA batteries (last approximately 3000 hours)

Table 2.1 Specification of moisture checker

2.6HydraTest

HydraTest (BeautyPro, UK) is a device that measures hydration and oil levels so that you can keep a check on your skin when at home or traveling.

HydraTest utilizes the latest BIA technology (Bioelectrical Impedance Analysis) the same credible technology used in larger medical and beauty equipment to ensure accurate and precise information about your skin (HydraTest skin analysis guide, 2013) (Figure 2.9).

Figure 2.9Hydrotest Beauty Pro
(***Source:*** ***http://www.beautyexpress.co.uk/beautypro-hydra-test-skin-analysis/***)

2.7ProScopeHR2 Digital Microscope

The new ProScope HR2 (Bodelin Technologies, USA) features the new state-of-the-art, Aptina™ imager which was originally made for surveillance, with large pixels that take in more light than other imagers on the market. The Bodelin engineering team customized this imager to allow extensive imaging adjustment and LED intensity control. The most impressive feature is the ability to actually stream true pixels at higher resolutions.

Fig.2.10. Proscope digital microscope
(Source: https://www.bodelin.com/proscope/proscope-hr2)

The ProScope HR2 is the only digital microscope in the world that can do this. Other imagers stream at 640X480 and only interpolate to simulate higher resolutions. The total effect in the ProScope HR2 is the live video that appears to be 4 to five megapixels in clarity while only using an actual two megapixel imager. The ProScope products are currently used in tens of thousands of schools, universities, law enforcement forensic labs, manufacturing quality control, medical and tele-medicine since 2001. The model used in this study uses an optical lens of 30x zoom with polarized light that can take images from both skin surface and underneath the skin surface.

2.8 Summary

In summary, this chapter describes the conceptions and principles of different types of skin measurement instruments: Opto-thermal transient emission radiometry (OTTER) can be used for skin hydration, skin hydration and pigments depth profiling, and trans-dermal drug/solvent diffusion measurement, it is a non-contact, non-invasive technology, but it is a bench-top instrument which required extensive power supply and cooling water. It is also very expensive. AquaFlux can be used for skin recovery, barrier function, etc. based on TEWL, membrane penetration, and material absorption/desorption measurement, it is an independent of external environment technology with high sensitivity, low noise, super reliability, excellent calibration ability and comparability, but it is not very portable as it needs external power supply. Epsilon imaging system can be used for skin hydration, solvent penetration, and membrane penetration measurement, it is an image based with good repeatability and producibility technology, because of calibration, the measured results can be used for hydration calculation. Corneometer is the current industry standard, but suffers poor repeatability and is uncalibrated. Moisture Checker and Hydratest are two most low cost, most portable skin hydration measurement instruments, they can be easily used at skin clinics, point of sales, or at home. But their measurement results are not accurate, and cannot be calibrated. ProScope HR2 can take the skin images from both skin surface and underneath the skin surface.

Chapter3 Mathematical Modeling of OTTER

Opto-thermal transient emission radiometry (OTTER) is a non-invasive, remote sensing infrared technique. It is very easy to make a measurement. You can virtually get signals from any samples. However, to extract the right information you will need complex mathematical models (Long F. H., et al, 1987, Jesus de. M., 1994 and Power J. F., 1991).

This chapter first presents the theoretical background OTTER measurement technology, then presents the work carried out in this study, e.g. enhanced Segment Least-Square (SLS) fitting, signal de-nosing, and Fast Fourier Transform. all these are implemented in new OTTER software described in Chapter 4.

3.1 Mathematical Modelling for OTTER

When laser radiation is incident on a sample surface, the absorbed laser light energy will decay exponentially into deeper skin, and the **initial temperature field** (when time t=0) can be described as:

$$\theta(z,0) = \frac{E_0 \alpha}{\rho C} e^{-\alpha z} \tag{3-1}$$

where$\theta(z,0)$, is the temperature at time t = 0 of the skin at position z, with z = 0 at the surface and increasing toward the inside of the skin. $\alpha$is the absorbance for the excitation laser, C is the specific heat, ρ is the density, and E_0 is the energy density absorbed from the excitation laser (Imhof R. E., et al, 1994, 1995 and Xiao P., et al, 1998).

Figure 3.1 shows the basic schematic diagram for OTTER model. Skin is

assumed to be semi-infinite homogeneous, with a surface at Z = 0.

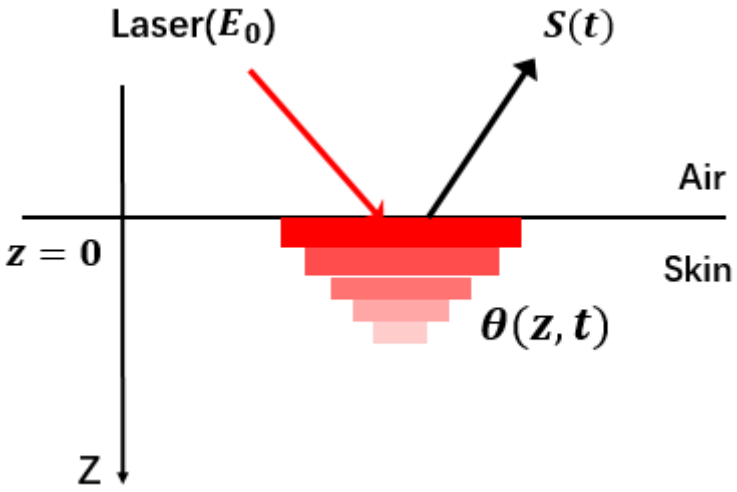


Figure 3.1 the basic OTTER model

While the absorbed laser light energy diffuses in the sample, the initial temperature will re-contribute following the diffusion law which can be described as:

$$\begin{cases} \left(\frac{\partial}{\partial z}\left[D\times\frac{\partial}{\partial z}\right]-\frac{\partial}{\partial t}\right)\theta(z,t)=0 \\ \text{Initial Condition:} \quad \theta(z,t)|_{t=0}=\theta(z,0) \\ \text{Boundary Condition:} \quad -k\frac{\partial\theta(z,t)}{\partial z}|_{z=0}=0 \end{cases} \tag{3-2}$$

where D is the thermal diffusivity, and k is thermal conductivity of the skin. Because skin measurement is mainly focused on optical properties and the thermal properties change very less, therefore, it is safe to assume that skin is thermally homogeneous (Imhof R. E., 1995), Eq. (3-2) can be solved by Green's function method to get the **time-dependent transient temperature field** is described as (Xiao P., 1997):

$$\theta(z,t)=\frac{E_0\alpha}{2C\rho}e^{\alpha^2Dt}\left\{e^{-\alpha z}\text{erfc}\left(\frac{\alpha^2Dt-\frac{1}{2}\alpha z}{\sqrt{\alpha^2Dt}}\right)+e^{\alpha z}\text{erfc}\left(\frac{\alpha^2Dt+\frac{1}{2}\alpha z}{\sqrt{\alpha^2Dt}}\right)\right\} \tag{3-3}$$

where $\text{erfc}(x)=1-\text{erf}(x)$ is the complementary error function, $\text{erfc}(x)=$

$\frac{2}{\sqrt{\pi}}\int_0^x e^{-y^2}dy$ is error function. The **OTTER signal S(t)** generated by the transient temperature $\theta(z,t)$ can be calculated by:

$$S(t) = \frac{\zeta E_0 \beta}{C\rho}\int_0^{\infty} e^{-\beta z}\,\theta(z,t)dz \qquad (3\text{-}4)$$

where β is the skin's absorption coefficient for the emitted thermal radiation, and the parameter$\zeta = \zeta(\lambda_{em})$includes factors that depend on the black body emission curve, detector sensitivity, focusing and alignment, but is independent of the properties of the sample (Imhof R. E., et al, 1984 1994, 1995).

Eq. (3-3) can be substituted into the Eq. (3-4), it is described as:

$$S(t) = \frac{\zeta E_0 \alpha\beta}{C\rho(\beta^2-\alpha^2)}\left\{\beta e^{\alpha^2 Dt}\mathrm{erfc}\sqrt{\alpha^2 Dt} + \alpha e^{\beta^2 Dt}\mathrm{erfc}\sqrt{\beta^2 Dt}\right\} \qquad (3\text{-}5)$$

Eq. (3-5) is the basic opto-thermal signal of a homogeneous material with impulse excitation.

If assuming the skin is optically **homogeneous**, which is the basic structure of the skin and under the excitation saturation condition, for example, the optical absorption coefficient for excitation, α, is much bigger than the optical absorption coefficient for emission, β (this is true for most of the bio-tissues when 2.94μm Er: YAG laser radiation is used for excitation) ($\alpha \gg \beta$), the opto-thermal signal S(t) (Eq. (3-5)) can be solved and expressed as (Imhof R. E., et al, 1984):

$$S(t) = Ae^{t/\tau}\mathrm{erfc}\sqrt{t/\tau} \qquad (3\text{-}6)$$

where$A$ is the amplitude of the signal, $\tau = \frac{1}{\beta^2 D}$is the signal decay lifetime, which is the key parameter extracted from the data.

By fitting Eq. (3-6) to OTTER signal, the best-fit value for τ can be received, which can be related to calculate the water content and solvent concentration in

the skin. This is the basic model called **semi-infinite homogeneous model**.

If assuming the skin is optically **non-homogeneous**, which is the normal properties of the skin, it can be considered that it has a linear gradient distribution, for example:

$$\beta(z) = \beta_0 + \omega z \tag{3-7}$$

whereβ_0 is the absorption coefficient of the surface of the skin, andω is the gradient of the absorption coefficient, and meet the excitation saturation condition $\alpha \gg \beta$, Eq. (3-6) can be solved and expressed as (Imhof R. E., et al, 1994 and Xiao P., et al, 1998):

$$S(t) = A\times\left(\frac{2W\sqrt{t\tau}}{\sqrt{\pi}(2Wt+1)} + \frac{1}{\sqrt{(2Wt+1)^3}}e^{\frac{t/\tau}{\left(\frac{2t}{\tau}+1\right)}}\mathrm{erfc}\left(\frac{\sqrt{t/\tau}}{\sqrt{2W+1}}\right)\right) \tag{3-8}$$

where W=ωD is the effective gradient and $\tau = \frac{1}{\beta^2 D}$is the signal decay life time. By fitting Eq. (3-8), the fitting value W and τ can be received, which can be related to calculate the water content and solvent concentration of the skin surface and the gradient of them within the skin. This is the normal model called **gradient model**.

From the fitting value τ, the value of β can be calculated. The β is considered to be comprised by dry skin and the water in the skin, and the equation of β can be written as:

$$\beta = H\times\beta_w + (1-H)\times\beta_{ds} \tag{3-9}$$

where H is assumed as the skin hydration, therefore, H can be calculated by:

$$H = \frac{\beta-\beta_{ds}}{\beta_w-\beta_{ds}} \tag{3-10}$$

whereβ_w is the absorption coefficient of water in the skin and β_{sd} is the absorption coefficient of dry skin.

When assuming the skin is optically **non-homogeneous**, Eq. (3-6) is difficult to solve, because the basic LS fitting is inadequate. Therefore, Xiao P. et al developed a new technique called SLS fitting technique (Xiao P., et al, 2001), which is described as **segmental analysis model**. For an OTTER decay signal, it comes from different depths when the laser excitation at different time, the different depths are called "mean detection depth",$\overline{Z}(t)$ can be calculated as:

$$\overline{Z}(t) = \frac{\int_0^\infty \beta e^{-\beta z} \cdot z \cdot \theta(z,t) dz}{\int_0^\infty \beta e^{-\beta z} \theta(z,t) dz} \tag{3-11}$$

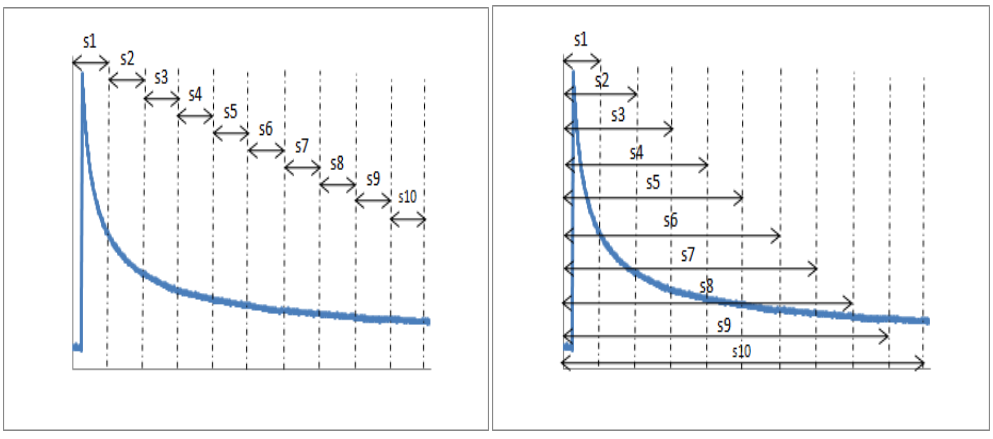
where$\theta(z,t)$ is the transient temperature field which is expressed as Eq. (3-3), and meet the excitation saturation condition $\alpha \gg \beta$, Eq. (3-11) can be expressed as:

$$\overline{Z}(t) = \frac{2}{\sqrt{\pi}} \times \frac{\sqrt{t/\tau}}{\beta e^{t/\tau} erfc\sqrt{t/\tau}} - 2 \times \frac{t/\tau}{\beta} \tag{3-12}$$

where$\overline{\beta}$ is the average of β, which calculated by using equation (3-6) to fit the whole signal. It also can use equation (3-10) to calculate the skin hydration. In the case of surface dominant absorption, we can find the relationship between the OTTER signal S(t) and its corresponding mean detection depth $\overline{Z}(t)$ for a semi-infinite homogenous sample, the initial data points give information about the surface of the sample, and later data points give information about deeper parts of the sample.

3.2 Enhanced Segmented Least Squares (SLS) Fitting Algorithm

In this work, a new Enhanced Segmented Least Squares (SLS) fitting algorithm has been developed. The existing Segmented Least Squares (SLS) fitting algorithm is developed by Xiao P., et al, 2000. The theory of SLS data analysis is shown below. **First**, divide the OTTER signal data points into several of short segments, more segments mean more accuracy but slower calculation speed. In this program, 10 segments (s1-s10) are chosen as the balance. **Second**, fit every segment by using LS fitting by Eq. (3-6) respectively, to calculate the best-fit value of the decay lifetime τ for the segment. Assuming the sample is homogeneous, the optical absorption coefficient β can be calculated as $\beta = 1/\sqrt{\tau D}$. **Finally**, put every segment's τ into its mean radiation depth by using Eq. (3-12) to calculate a depth-resolved τ profile, which can be related to the water content and solvent concentration depth in the skin (Xiao P., et al, 2001).



Model (a) ***Model (b)***

Figure 3.2 different segment analysis models

Traditionally, the OTTER signal is divided into several separated pieces, we fit every piece, and associate each piece to a certain skin depth described in Eq (3-12). See Figure 3.2 model (a). However, this approach presents several issues,

first, it is physically inaccurate, as the OTTER signal is average of all depth, not just certain depth. Secondly, the fitting can become unstable, especially for the pieces near the tail of the signal.

To improve the fitting, an enhance SLS fitting is proposed. In this case, the OTTER signal is divided into progressive pieces, e.g. every piece is starting from the same beginning, and progressively increased in length. See Figure 3.2 model (b). This approach is more physically sound, and brings better stability of the fitting. Therefore, in this work, enhanced SLS fitting is used for OTTER signal depth profiling.

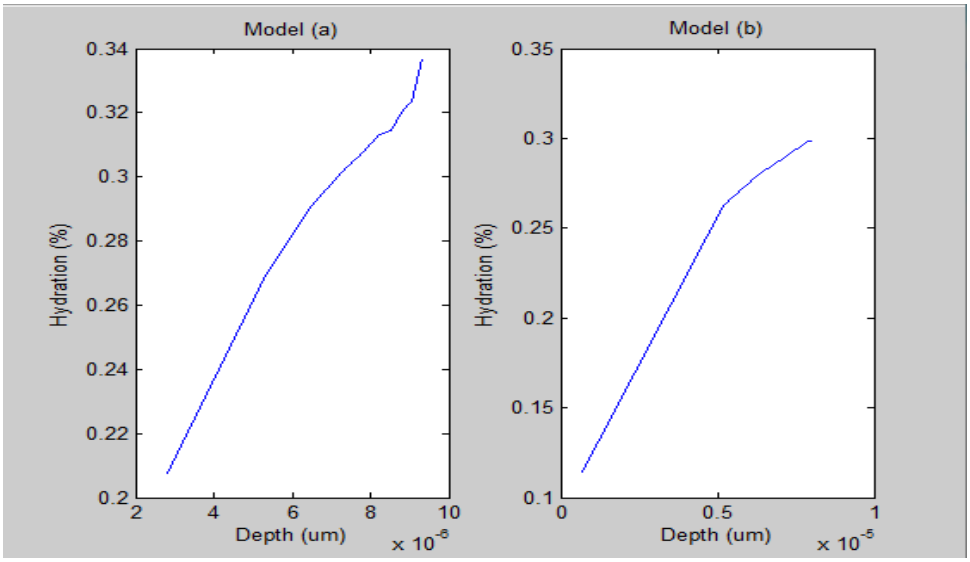


Figure 3.3 different segment model fitting results

From chapter 1, it described that the stratum corneum (surface of skin) has the least water content, and it increases with depth deeper within skin. Figure 3.3 shows the two different segment model fitting results. Both models have the similar trend, but the model (a) – original SLS fitting - has unrealistic values at deeper skin because of unstable fitting, and model (b) – enhanced SLS fitting - has much better more stable fitting results.

3.3 Signal De-noising

An OTTER signal de-noising method is also developed in this study. Signal de-nosing is very important because it can reduce the noise so that it is better for signal processing. Wavelet de-nosing is one of the most popular methods in signal analysis because it has several advantages such as low entropy, de-correlation and the flexibility of selected base wavelet (Wu. Y. F., et al, 2014).

In theory, wavelet de-noising depends on signal filtering. Assuming a limited length signal with Gaussian white noise model can be described as:

$$X(t) = F(t) + e(t) \qquad (t = 1,2, \dots, n)$$

where$X(t)$ is the original signal, $e(t)$ is the Gaussian white noise, $F(t)$ is the useful signal.

Normally the de-noising signal is considered as the low frequency part or smooth signal, while the noise signal is considered as the high frequency signal. If the original signal $X(t)$ is decomposed by a wavelet, Ca1, Ca2 and Ca3 can be assumed as the useful signal (low frequency part), Cd1, Cd2 and Cd3 (high frequency part) which include noise signal, are processed by a wavelet, and then reconstruct the both parts of the signal to receive the de-noising signal. The flow chart is shown in figure 3.4.

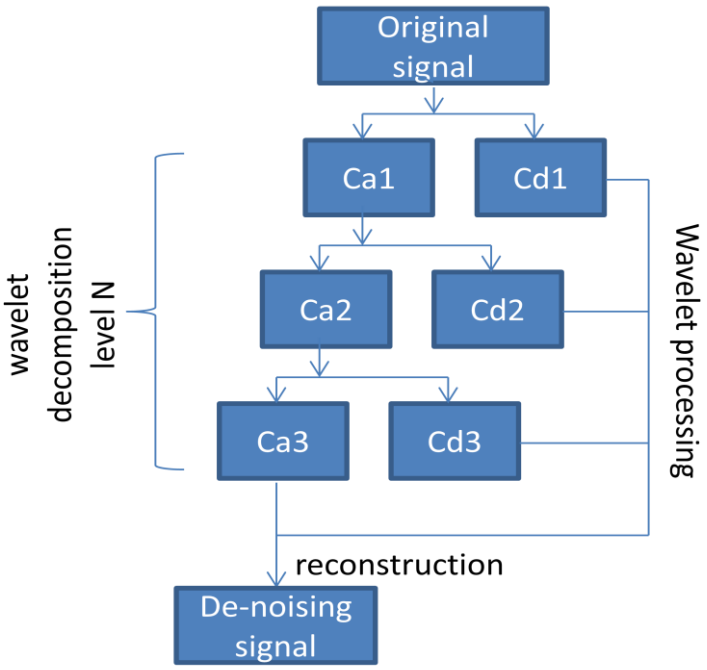


Figure 3.4 wavelet de-noising.

Therefore, the de-noising procedure can proceed in three steps:

Decomposition. Choose a wavelet, and choose a level N. Compute the wavelet decomposition of the signal s at level N.

Detail coefficients thresholding. For each level from 1 to N, select a threshold and apply soft thresholding to the detail coefficients.

Reconstruction. Compute wavelet reconstruction based on the original approximation coefficients of level N and the modified detail coefficients of levels from 1 to N.

In this program, a MATLAB function **"wden"** is used which is based on basic wavelet method (figure 3.5).

```
function y=DeNtestDN(s)
%k='D:\PhD\OX-Arm1-13\OX-arm1-13001.omt';
%s=textread(k,'' , 'headerlines', 12);

%s=s';
lev = 5;

s_den = wden(s,'sqtwolog','s','one',lev,'sym6');
y=s_den;
```

Figure 3.5 MATLAB code of "wden".

In figure 3.4, the function "wden" includes 6 parts:

"s" means the original signal;

'sqtwolog' is the threshold selection which is the universal threshold

's' is for the soft thresholding;

'one' is defined as the multiplicative threshold rescaling which is the basic model;

lev means the wavelet decomposition level;

'sym6' is one of symmetric wavelet, which has strong linear feature that suitable for wavelet de-noising;

s_den is the de-noising signal.

MATLAB receives the collected signal "s" from the main program and uses function "wden" to get the de-noising signal "s_den" and returns to the main program.

Figure 3.6 shows the original signal and the de-noising signal by using wavelet de-noising.

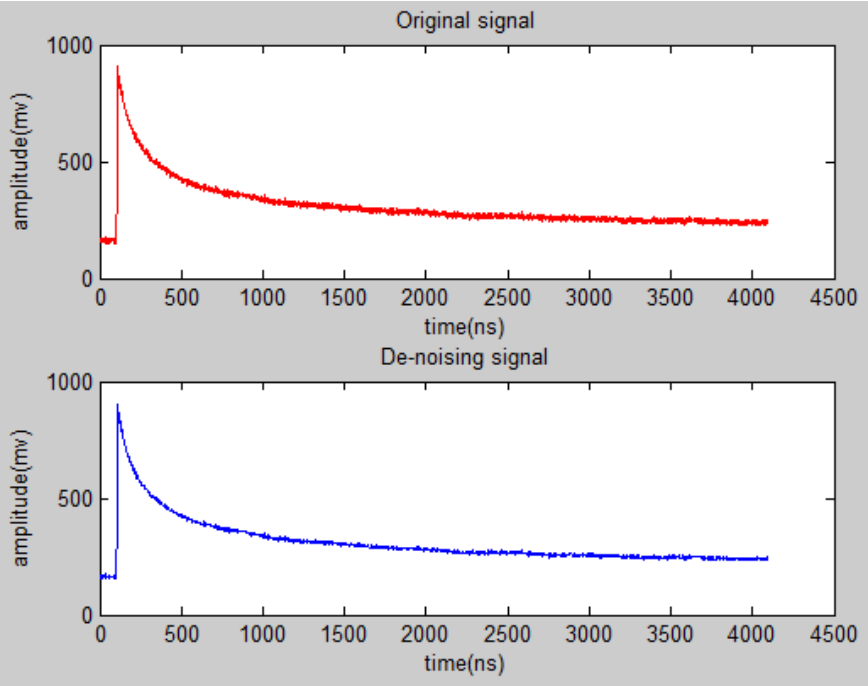


Figure 3.6 original signal and de-noising signal

Although traditional signal averaging can also effectively reduce OTTER signal noises, averaging does take longer measurement time. This is more significant when we are performing multiple measurements. Wavelet de-noising can efficiently reduce the need for signal averaging and therefore reduce measurement time.

3.4 Fast Fourier Transform

In this work, a Fast Fourier Transform (FFT) is also implemented. Signal analysis is a powerful tool which can indicate a great deal of information about the operation of any given system (Oakley. C., 2011). Although pulse laser technology is used in this thesis, modulated laser technology is also important and suitable for skin measurement which can be processed by Fourier Transform. The Fourier Transform (FT) is one of the most important methods to transfer a

time domain signal to a frequency domain signal. In this program, it uses the function from the Aforge.Math library of the Fast Fourier Transform (FFT) which is based on the Discrete Fourier Transform (DFT), because computer can only process the discrete signal with limited length.

Most Fourier transforms are based on the use of complex number, Aforge.Math library uses the "Complex" class to encapsulates the complex numbers. Sometimes, the Fourier transform can be used strictly to the real numbers, in the program, it is used the real part of a complex number with the imaginary part equal to 0 (figure 3.7).

```
AForge.Math.Complex[] input

    input[i].Re = chaA[i];
    input[i].Im = 0.0;
```

Figure 3.7 C# code of complex number only uses the real part

Figure 3.8 shows the Aforge.math function of "FFT", which can implement one dimensional Fast Fourier Transform.

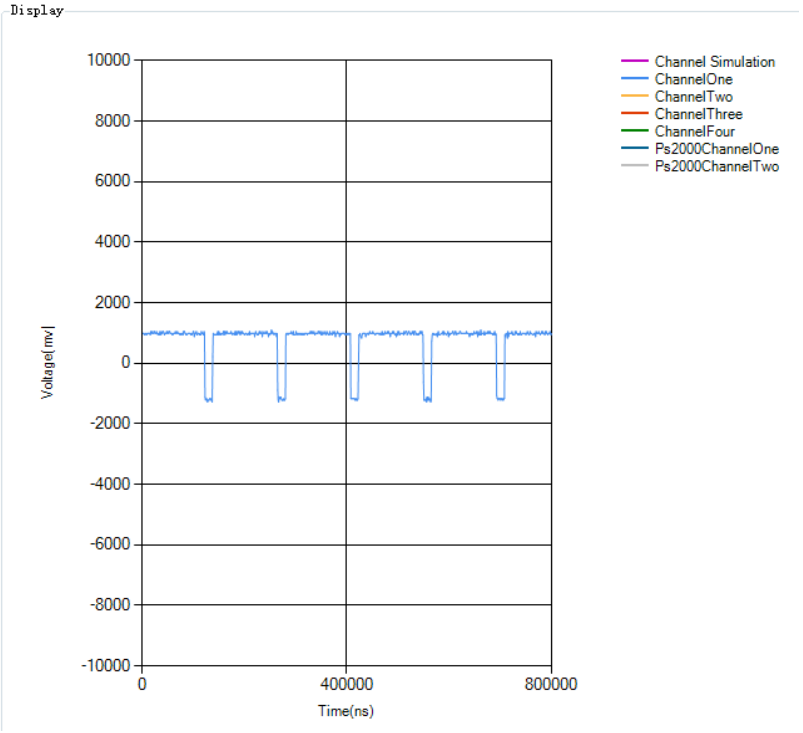
```
        c1.FFT(input, FourierTransform.Direction.Forward);

public Complex[] FFT(Complex[] y, FourierTransform.Direction n)
{

    FourierTransform.FFT(y, n);
    return y;
}
```

Figure 3.8 C# code of Aforge.math function FFT

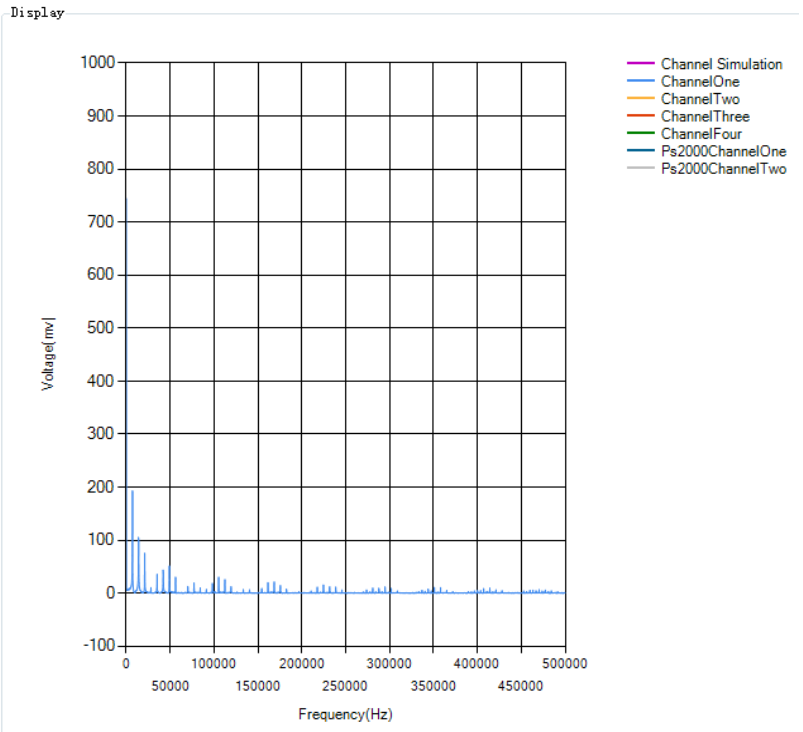
The main program transfers the parameter of complex number "input" (shown in figure 3.3) to the Aforge.math function FFT to implement the forward Fast Fourier Transform and then return the converted signal back to the main program.

Figure 3.9 shows the Fast Fourier Transform of a square wave signal by using

Aforge.math function FFT.



time domain signal



frequency domain signal

Figure 3.9 The FFT of square wave signal

3.5 Summary

This chapter first describes the theoretical background for OTTER measurements, then describes the work carried out in this study. In the OTTER theoretical background, two models are presented, semi-infinite homogeneous model and gradient model. For semi-infinite homogeneous model, it assumes the skin is homogeneous, the signal decay lifetime τ can be calculated by using Eq.(3-6). For gradient model, it assumes the skin is optically non-homogeneous, the signal decay lifetime τ and effective gradient W can be calculated by Eq.(3-8). According to τ, the water content can be calculated by both models and according to W, the gradient of them within the skin can be calculated by gradient model.

In the work carried out in this study, e.g. enhanced Segment Least-Square (SLS) fitting, signal de-nosing, and Fast Fourier Transform. The tradition Segmented Least Squares (SLS) fitting method is used for skin depth profile analysis, but it is unstable in fitting process, the other new Enhanced SLS fitting is developed to improve fitting stability. It also describes signal de-noising, which can reduce the noise by using wavelet de-noising method and Fast Fourier Transform, which can be used for modulated laser technology. these two new functions are useful for the new OTTER software. More results on using these new mathematical modeling techniques and functions will be available in next chapter.

Chapter 4 Development of OTTER Data Acquisition and Data Analysis Program

The existing OTTER (opto-thermal transient emission radiometry) software was developed based on a PCI (Peripheral Component Interconnect) data acquisition card, which only works under the obsolete Window 2000 operating system. The software is also difficult to debug and to add new features. There is a genuine need to develop new OTTER data acquisition and data analysis software.

This chapter presents the research work on developing new modularized OTTER software, which not only includes data acquisition but also includes data analysis. A new set of data analysis software libraries, Dynamic-link library (or DLL), are developed by using C# (Visual Studio 2012) and MATLAB. The modularized nature of the new software makes it easy to manage, update and add new features, and finally some OTTER measurement results, all done by using the new OTTER software.

4.1 OTTER Software Specification

The OTTER software is planned to design into two parts, "Real Time" part (data acquisition) and "Measurement" part (data analysis). In "Real Time" part, the software can implement the collected signal displaying, signal environment setting (time-interval, trigger channel etc.), signal characterization (amplitude, frequency, etc.), de-noising, fast Fourier Transform, and signal filtering. In "Measurement" part, it can implement the measurement, measurement and fitted data displaying, hydration and depth profile calculation by different modes and displaying, and saving all the results into Excel files.

4.2 PicoScope Technology and Software

Comparing with existing PCI data acquisition card, PicoScope Technology offers many advantages: USB based, higher data sampling rate (80MS/s), better ADC (Analog-to-digital converter) resolutions (up to 16 bits), multiple programming language support (C/C++, C#, Visual Basic etc.), and multiple systems support, i.e. Windows, MAC, and Linux etc.

4.2.1 Overview of Picoscope 4000 series PC oscilloscope

The PicoScope 4000 Series Automotive Oscilloscopes are a range of high-speed PC oscilloscopes for automotive diagnostic use. They are fully USB 2.0-capable and backwards-compatible with USB 1.1. There is no need for an external power supply as power is supplied from the USB port, making these oscilloscopes highly portable. The PicoScope 4000 Series Automotive Oscilloscopes are supplied with the PicoScope software, which turns your PC into a powerful PC Oscilloscope.

The 4–channel **PicoScope 4424** (figure 4.1) is high–resolution oscilloscopes that are suitable for general, scientific and field–service use. With 12–bit resolution (adjustable up to 16 bits in enhanced resolution mode) and 1% accuracy, they also make an excellent choice for noise, vibration and mechanical analysis.

Figure 4.1 PicoScope 4424 PC oscilloscope
(Source: https://www.picotech.com/oscilloscope/picoscope-4000-series)

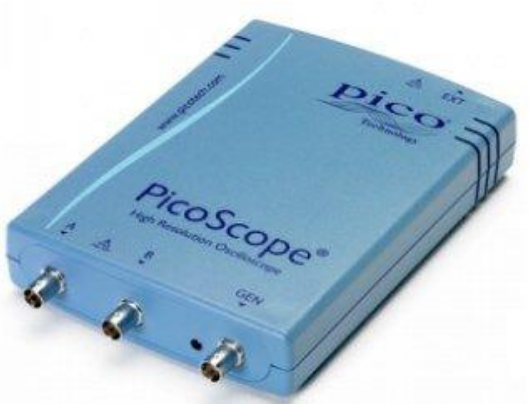


Figure 4.2 PicoScope 4262 PC oscilloscope
(Source: https://www.picotech.com/oscilloscope/picoscope-4000-series)

The **PicoScope 4262** (figure 4.2) from Pico Technology is a 2-channel, 16-bit high-resolution oscilloscope with a built-in low-distortion signal generator. With its 5 MHz bandwidth, it can easily analyze audio, ultrasonic and vibration signals, characterize noise in switched mode power supplies, measure distortion, and perform a wide range of precision measurement tasks.

4.2.2 Basic Specifications of PicoScope 4000 series

PicoScope 4424 oscilloscope has 4 channels, which bandwidth is 20MHz, sampling frequency is 80MS/s and ADC resolution is 12 bits. It also has a 32MS buffer memory.

PicoScope 4262 oscilloscope has 2 channels, which bandwidth is 5MHz, sampling frequency is 10MS/s and ADC resolution is 16 bits. It also has a 16MS buffer memory.

All the specifications show in table 4.1.

Specifications of PicoScope 4000 series		
Model	4424	4262
Channels	4	2
Bandwidth	20MHz	5MHz
Sampling Frequency	80MS/s	10MS/s
ADC Resolution	12 bits	16 bits
Buffer Memory	32MS	16MS

Table 4.1 Specifications of PicoScope 4000 series

4.2.3 Overview of Picoscope2200A PC oscilloscope

The PicoScope 2200A oscilloscope provide fast real-time sampling rates up to 1 GS/s, equivalent to a timing resolution of only 1 ns. For repetitive signals, equivalent-time sampling (ETS) mode can boost the maximum effective sampling rate up to an incredible 10 GS/s, allowing even finer resolution down to 100 ps. All scopes support configurable length pre-trigger and post-trigger capture.

Figure 4.3 PicoScope2204A oscilloscope

(Source: https://www.picotech.com/oscilloscope/2000/picoscope-2000-overview)

4.2.4 Basic specifications of PicoScope2204

PicoScope 2204 oscilloscope has 2 channels, which bandwidth is 10MHz, sampling rate is 100MS/s for one channel and the other one is 50MS/s and ADC resolution is 8 bits. It also has 8KS buffer memory. The specifications show in table 4.2.

Specifications of PicoScope 2204 oscilloscope	
Model	2204
Channels	2
Bandwidth	10MHz
Sampling Rate	Channel 1: 100MS/s Channel 2: 50MS/s
ADC Resolution	8 bits
Buffer Memory	8KS

Table 4.2 Specifications of PicoScope2204 oscilloscope

4.3 Visual Studio software

Visual Studio is a comprehensive collection of tools and services to help you create a wide variety of apps, both for the Microsoft platform and beyond. Visual Studio also connects all of your projects, teams, and stakeholders. Now your team can work with greater agility from virtually anywhere—irrespective of development tool, including Eclipse and Xcode. Whether you're designing mission-critical .NET apps, writing blazing fast code with C++ AMP, or testing and debugging a cloud-connected HTML/JavaScript app that runs on many devices, join millions of developers worldwide in choosing Visual Studio as your essential development environment.

In this case, the visual studio 2012 software with the C# language was used. Figure 4.4 shows graphical user interface of visual studio 2012.

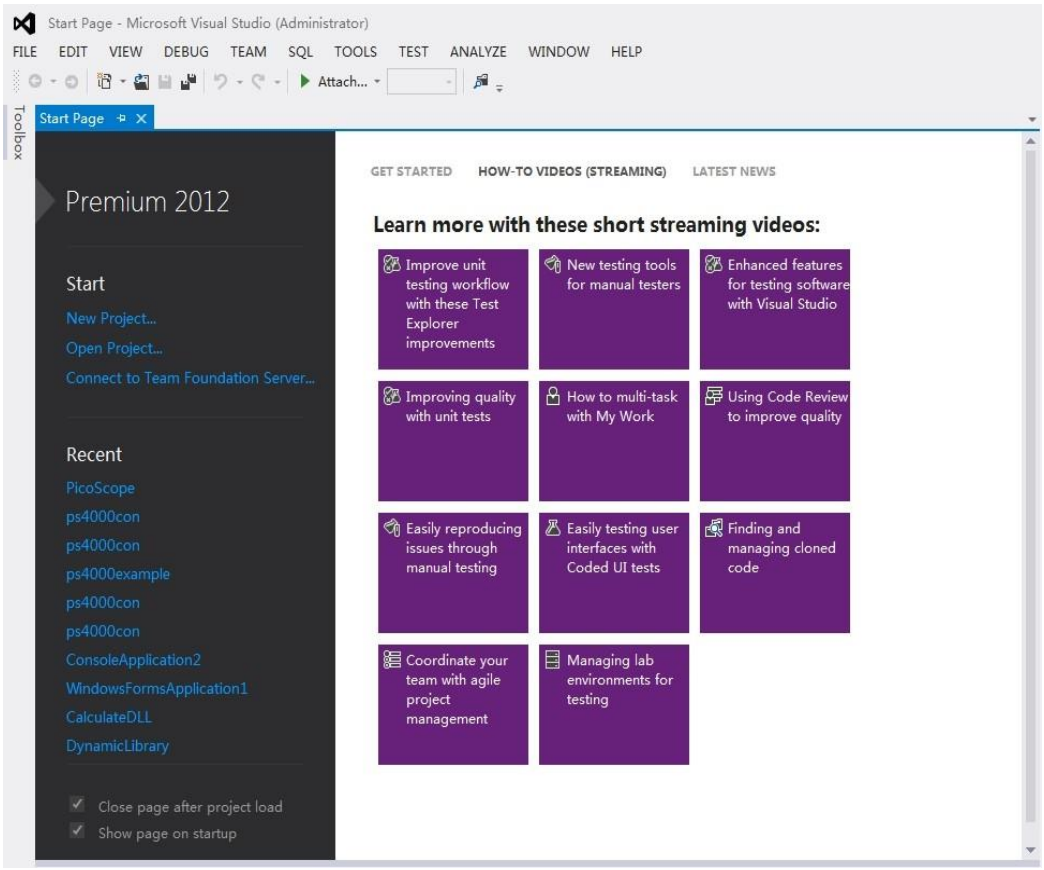


Figure 4.4 The graphical user interface of visual studio 2012

4.4 MATLAB Dynamic-link Library

MATLAB is a high-level language and interactive environment for numerical computation, visualization, and programming. Using MATLAB, you can analyze data, develop algorithms, and create models and applications. The language, tools, and built-in math functions enable you to explore multiple approaches and reach a solution faster than with spreadsheets or traditional programming languages, such as C/C++ or Java™.

MATLAB can be used for a range of applications, including signal processing and communications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers and scientists in industry and academia use MATLAB, the language of technical computing. Figure 4.5 shows a typical graphic user interface of MATLAB.

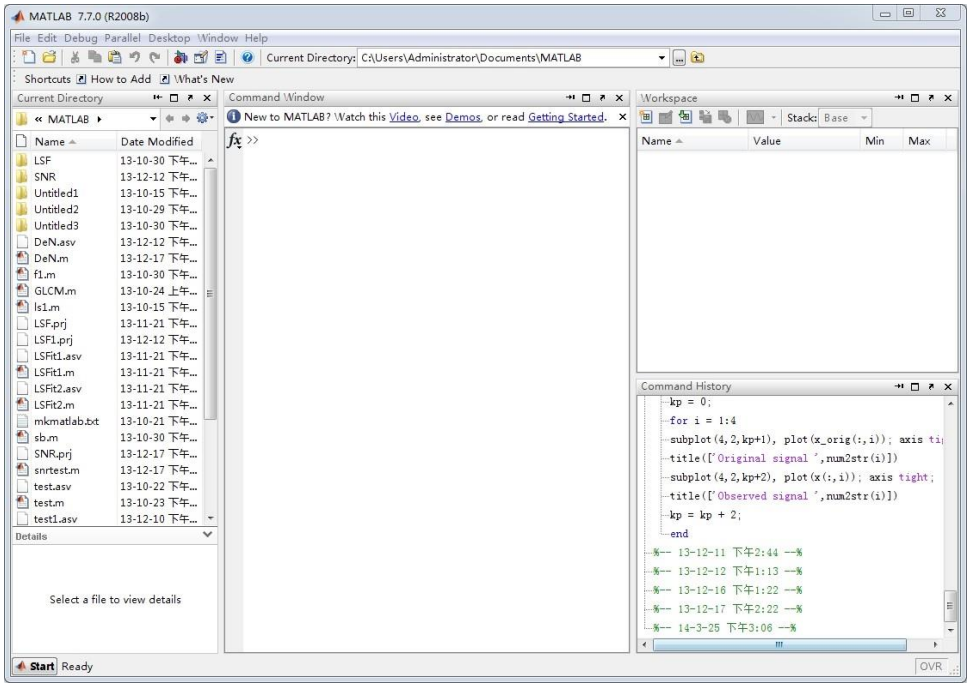


Figure 4.5Thegraphical user interface of MATLAB

In this work, MATLAB is used to create the dynamic link library (DLL) for data analysis of the main program.

4.5 Structure of the new OTTER Data Acquisition and Data Analysis program

The new OTTER program includes a main GUI program and seven data analysis dynamic link libraries (DLLs).

4.5.1 Overall Design

There are seven main parts in this program, one graphical user interface (GUI), three DLLs ("CalculateDll", "PicoScopeDll", "Pico2000Dll") which is programmed by using C#, and four DLLs ("LSF", "DeN", "Frequency", "PFilter") which is programmed by using MATLAB. The structure of the program shows in Figure 4.6, and this section will introduce the main function roughly of these parts.

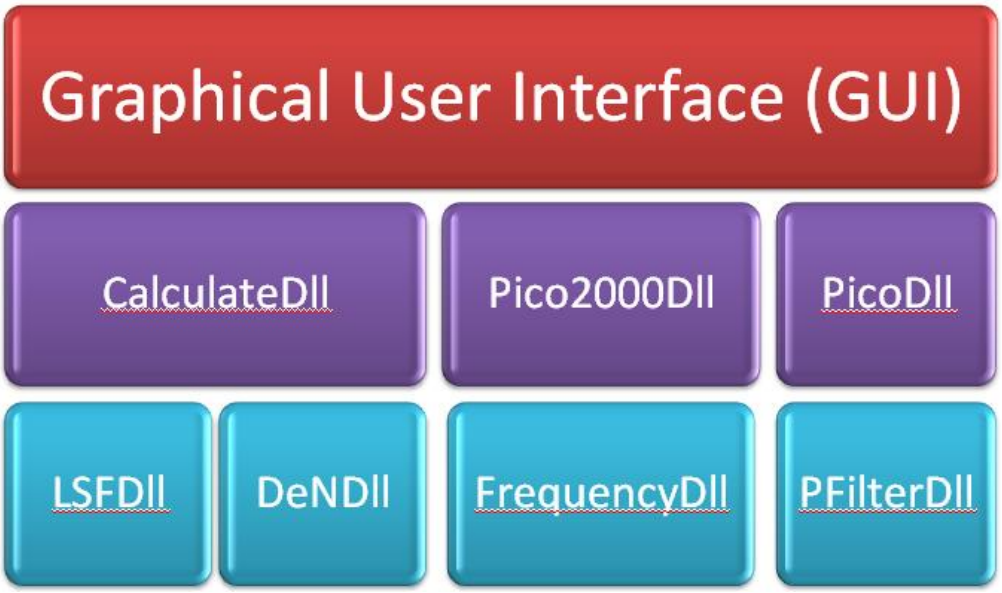


Figure 4.6 structure of the program

4.5.2 Graphical User Interface (GUI)

The Graphical user interface (GUI) is one of the most important parts in the total program, because all the functions of the rest parts have to be achieved through the

GUI (Figure 4.7). In Figure 4.7, there are two view tabs in GUI. "Real Time" view is displaying the figure which received from the oscilloscope, and it also can modify the displaying environment, such as time interval, scale range, etc. (Figure4.7 (a)). "Measurement" view can help customers to get the results by specific algorisms from the real time display, and save the results for analysis (Figure4.7 (b)).

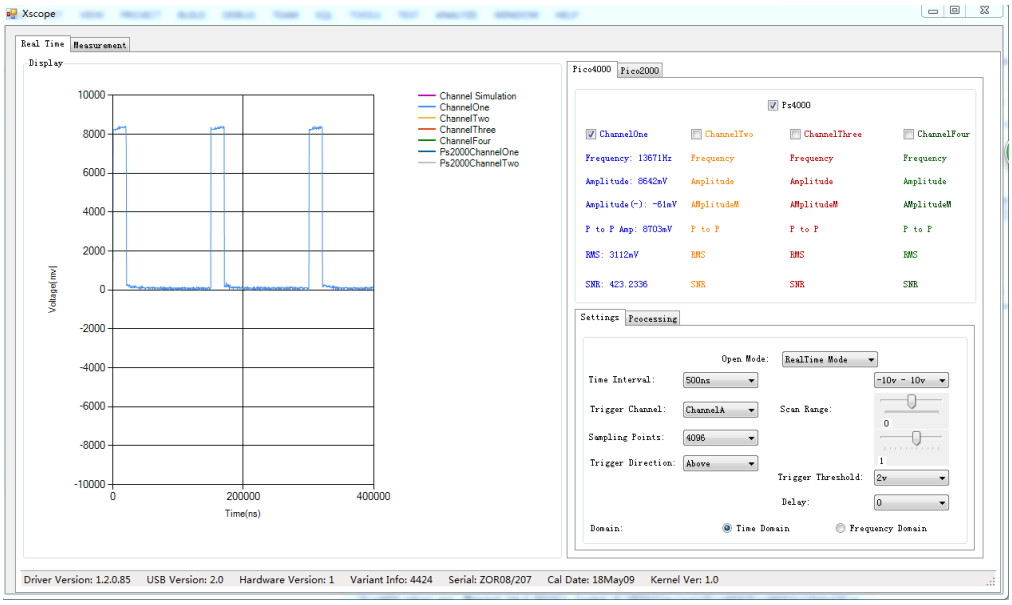


Figure 4.7 (a) "Real Time" view of GUI

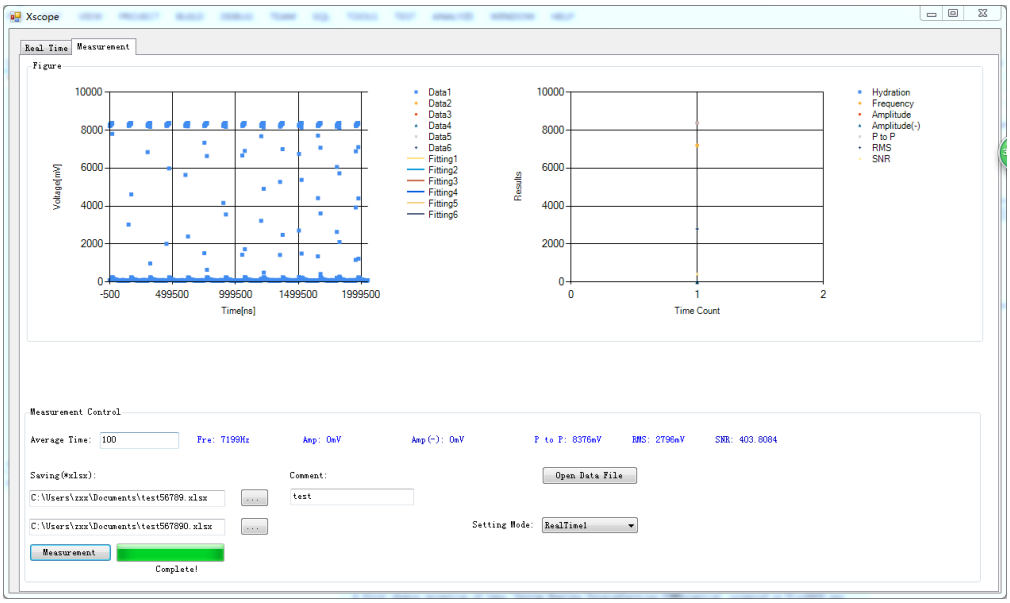


Figure 4.7 (b) "Measurement" view of GUI

4.5.3 Data Analysis Libraries

Seven new data analysis software libraries have been developed by using Visual Studio 2012 C# and MATLAB, as shown in Figure 4.6. These libraries are developed as DLLs.

DLL is the abbreviation of Dynamic Link Library, it's a code and data library which can be linked by multiple programs at the same time, but it is not an executable file. Dynamic link just provides a way to make process call its non-executable function, and the function code is in the DLL, and it contains one or more stored functions which have been compiled and linked separately. DLL also helps to share data and resources; hence it can implement multiple applications to access one DLL together.

There are two advantages by using DLL, first is DLL has a good compatibility, one type of DLL can be linked by multiple programming software. It can use the visual studio to call a MATLAB DLL, because visual studio has a good graphic designing, and MATLAB has a very strong calculation function, we can combine both of their advantages together.

Second is it easy for error correction and data updating. because each DLL can be seen as an independent module, when you correct errors or update data in one DLL, it won't influence other DLLs. for example, when you have a large project to program, you may have thousands of codes to write, if you put all codes into one application, it looks complicate, and it is very difficult to correct errors and update the program, but if you put these codes into several DLLs which are divided by different functions, it will become much better.

(1). CalculateDll

CalculateDll is a dynamic link library which is programmed by C#. It has four main functions, first and second is calculating the maximum and minimum values of the signals, third is calculating the position of the maximum number, and fourth is changing the signals from time domain to frequency domain for analysis by using fast Fourier transform (FFT) algorithm (Figure 4.8).Some of the functions are based on an open source library called "**AForge**" – which is a C# framework library designed for developers and researchers in the fields of Computer Vision and Artificial Intelligence - image processing, neural networks, genetic algorithms, machine learning, robotics, etc.

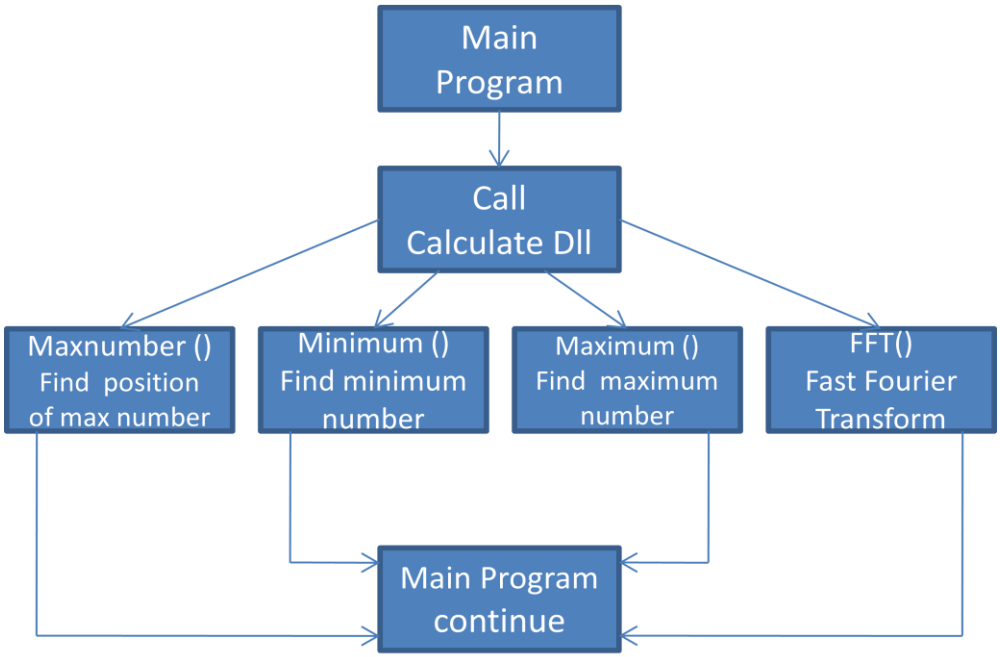


Figure 4.8 structure of CalculateDll

(2). PicoDll

PicoDll is the one of the most important parts in total program. It is based on the C# open source "ps4000CS" from PicoScope technology. The original program is a simple winform program which can implement several functions of PicoScope

4424 and 4262 oscilloscopes. In this program, it is set as a DLL for the GUI part, which can detect whether the oscilloscope has connected and the variant info of oscilloscope, display the status of them, set time-interval, collect the signal values and convert them to voltage values (unit: mv), and set trigger channel and trigger threshold (Figure 4.9).

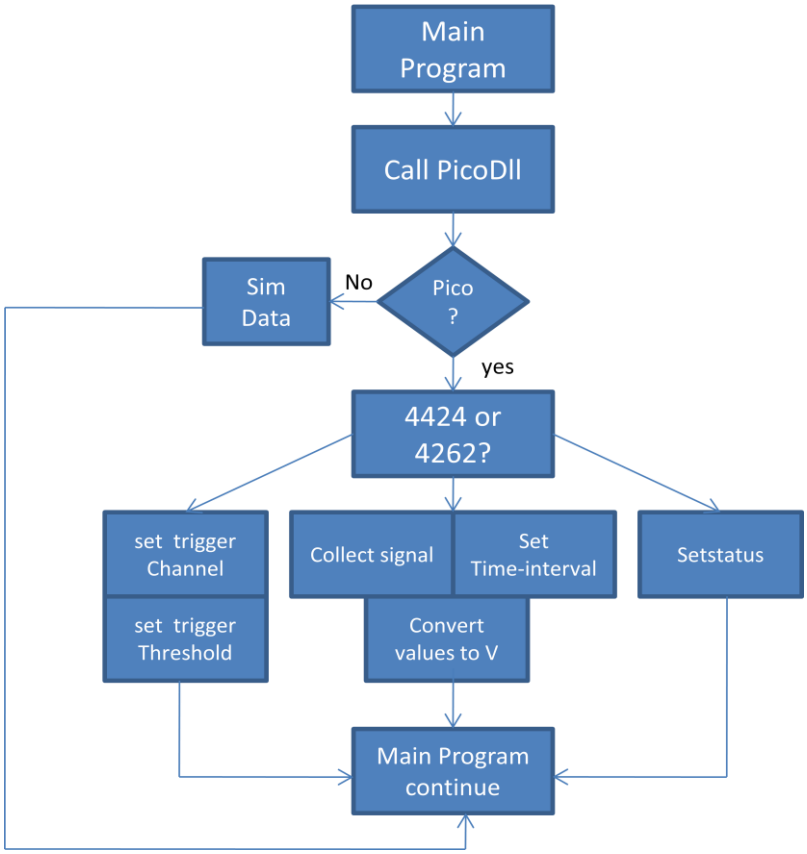


Figure 4.9structure of PicoDll

(3). Pico2000Dll

Pico2000Dll is the one of the most important parts in total program. It is based on the C# open source "ps2000CS" from PicoScope technology. The original program is a simple winform program which can implement several functions of PicoScope 2204 oscilloscope. In this program, it is set as a DLL for the GUI part, which can detect whether the oscilloscope has connected set time-interval, collect the signal values and convert them to voltage values (unit: mv), and set trigger

channel and trigger threshold, this Dll is very similar with the PicoDll, but it doesn't need to check the variant info because only one PicoScope 2204 oscilloscope existed.

(4). LSF

LSF is the DLL for collected OTTER signal analysis which is programmed by MATLAB. LSF has three MATLAB files, LSFit1.m, LSFit2.m, and LSFitdeep.m (Figure 4.10).

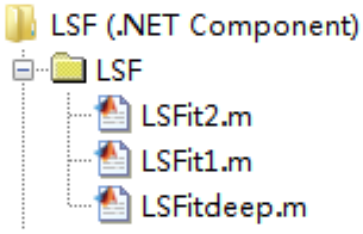


Figure 4.10LSFDll

In **LSFit1.m**, it receives the Excel filename and the sheet of saved original signal which is collected from C# main program, the number of sample points and time-interval of the signal. Least square algorithm is used for fitting equation (3-6), original signal and time-interval to calculate the signal decay lifetime$\tau$and transfer back to the main program for skin hydration calculation.

LSFit2.m is very similar with LSFit1.m, it also receives the Excel filename and the sheet of saved original signal which is collected from C# main program, the number of sample points and time-interval of the signal. Least square algorithm is used for fitting equation (3-8), original signal and time-interval to calculate the signal decay lifetime$\tau$and the effective gradient Wand transfer back to the main program for skin hydration calculation.

There is a little different between **LSFitdeep.m** and other two files above. It not

only receives Excel filename and the sheet of saved original signal which is collected from C# main program, the number of sample points and time-interval of the signal, but also receives the start and last point of a segment. Least square algorithm is used for fitting equation (3-6), original signal and time-interval to calculate the signal decay lifetimeτof a segment and transfer back to the main program for mean detection depth calculation.

Three modules are chosen by "Fitting Mode" control in main program.

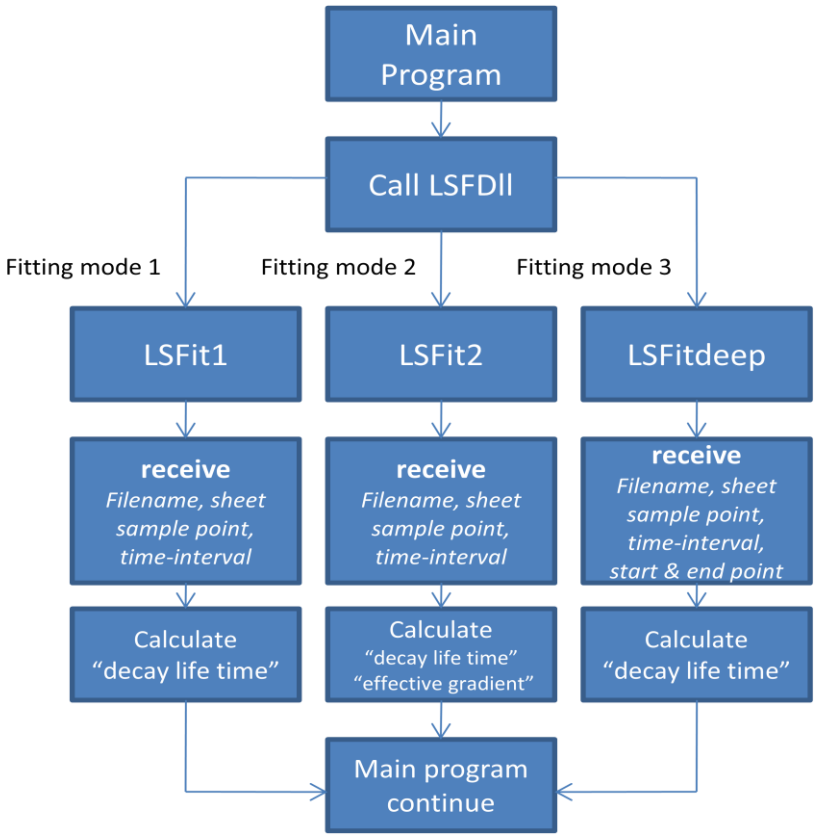


Figure 4.11structure of LSFDll

(5). DeN

DeN is the DLL for collected signal analysis which is programmed by MATLAB. DeN has two MATLAB files, DeNtest.m and DeNtestDN.m (Figure 4.12).

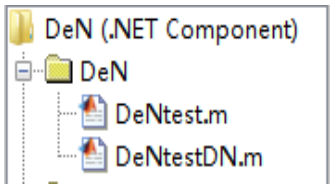


Figure 4.12DeNDll

In **DeNtest.m**, it received the original signal which is collected from C# main program and MATLAB wavelet de-noising library is used to get the de-noised signal and noise signal ("original signal"-"de-noised signal") and then calculate the signal noise ratio (SNR). In **DeNtestDN.m**, it only de-noises the original signal to get the de-noised signal by using MATLAB wavelet de-noising library.

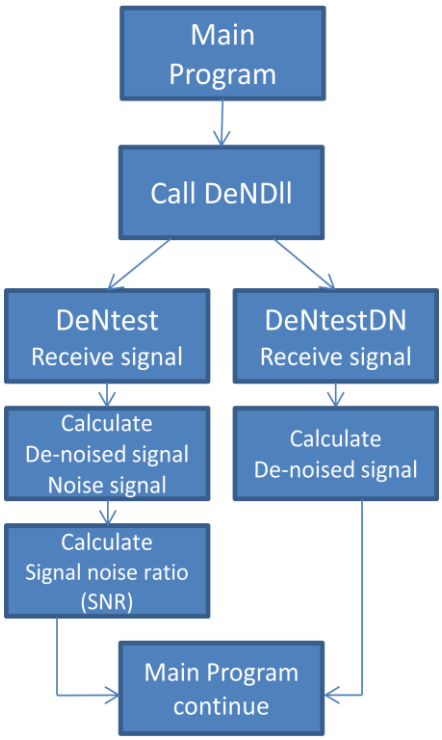


Figure 4.13 structure of DeNDll

(6). Frequency

Frequency is the DLL for collected signal analysis which is programmed by MATLAB. Frequency has only one MATLAB file, ffttest.m (Figure 4.14).

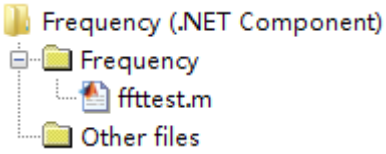


Figure 4.14 FrequencyDll

Frequency.m receives the original signal which is collected from C# main program and time-interval of the signal and then calculates the basic signal specifications such as signal frequency, signal wave crest, signal wave trough, peak to peak value, and root mean square (RMS).

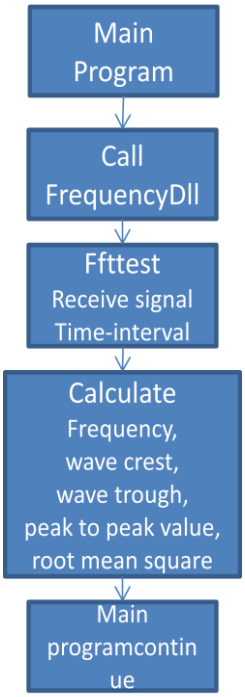


Figure 4.15 structure of frequencyDll

(7). PFilter

PFilter is the DLL for collected signal analysis which is programmed by MATLAB. PFilter has four MATLAB files, LPfilter.m, HPfilter.m, BPfilter.m and NPfilter.m (Figure 4.16).

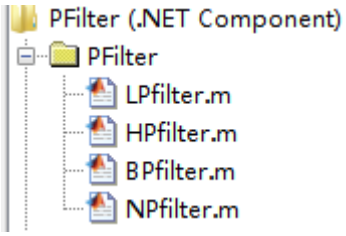


Figure 4.16 PFilterDll

LPfilter.m receives the original signal which is collected from C# main program, time-interval of the signal and the cutoff frequency and then makes the passes signal with the frequency lower than cutoff frequency. **HPfilter.m** receives the original signal which is collected from C# main program, time-interval of the signal and the cutoff frequency and then makes the passes signal with the frequency higher than cutoff frequency. **BPfilter.m** receives the original signal which is collected from C# main program, time-interval of the signal and two cutoff frequencies (high and low) and then makes the passes signal with the frequency between the two cutoff frequencies. **NPfilter.m** receives the original signal which is collected from C# main program, time-interval of the signal and the cutoff frequency and then makes the passes signal with the most frequencies except the specific cutoff frequency.

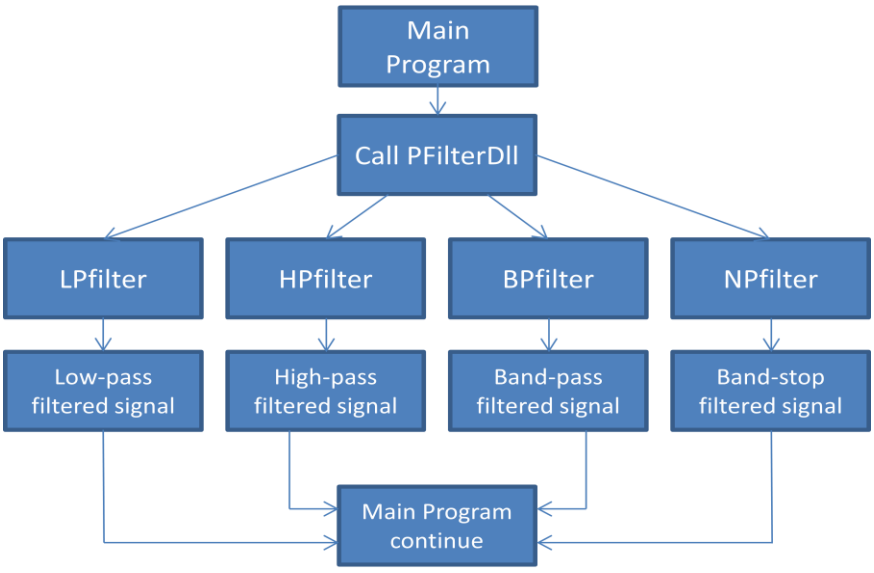


Figure 4.17 structure of PFilterDll

4.6 Functions of the New OTTER Program

Figure 4.18 shows the flow chart of the OTTER program. When it starts, if a PicoScope device is detected, it reads data directly from the device, otherwise, simulated data will be used. It then starts the graphical user interface (GUI), through which two view tabs are implemented: the "Real Time" view and "Measurement" view respectively.

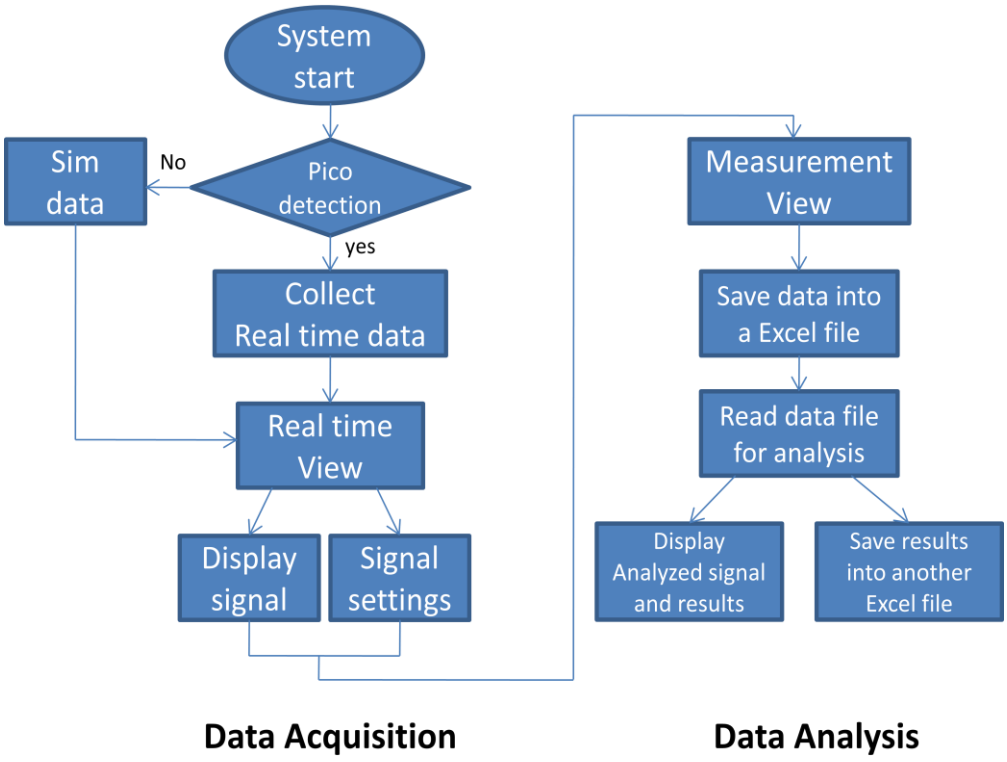


Figure 4.18 basic functions of the program

4.6.1 "Real Time" View

The "Real Time" view is for display the signals of different channels in real time. It has three big components, "Display", "Setting channel (Pico 4000 and 2000)", and "Settings (Settings and Processing)".

"Display": In this component, it has a chart component of C# toolbox to display the signal which is collected by PicoScope oscilloscope. Because of six channels

which the two oscilloscopes have, the chart can also display six channels together or separately by different colors, it can also display the simulation channel (Figure 4.19).

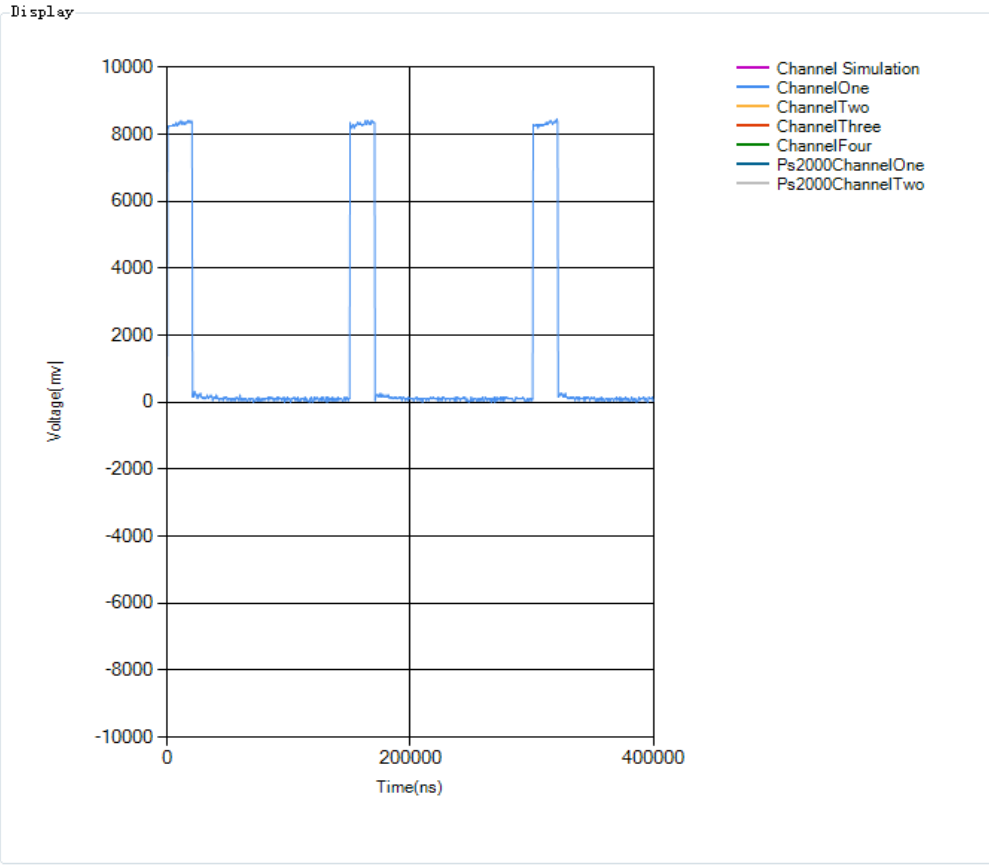


Figure 4.19The "Display" of "Realtime" part

"Setting Channel (pico 4000 and 2000)": In this component, it has 2 parts in pico 4000 and 2000, separately. In pico 4000, there are 5 checkboxes of C# toolbox (all components come from this toolbox below), one is used for controlling whether we choose PicoScope 4424 and 4262 oscilloscope, and the other 4 channels represent the 4 channels which PicoScope 4424 and 4262 oscilloscope has, it can control the figure to display any channel we want, meanwhile, in this tab, it can also display the specifications of the signal showing in the chart, such as frequency, wave crest, wave trough, peak to peak value, RMS (root mean square), and SNR (signal noise ratio). The same function in pico 2000 for PicoScope 2204 oscilloscope (Figure 4.20). This is one of the new

features of the software, it can read data from multiple channels, and multiple PicoScope devices simultaneously.

Figure 4.20The "Setting channel" of "Realtime" part

"Settings (settings and processing)": In this component, it has 4 parts (settings and processing) to control the figure properties of the "Display" component which receive from PicoScope 4424, 4262 and 2204 oscilloscopes separately. In "settings", ***Timeinterval*** has a combobox to change the time interval of the signal it collects. There are 10 options of time intervals (40ns, 80ns, 160ns, 320ns, 640ns,

1.28us, 5.12us, 10.24us, 40.96us, 163.84us, 655.36us for Pico 2204A and 50ns, 100ns, 500ns, 1us, 5us, 10us, 50us, 100us, 500us, 1ms for Pico 4424 and 4262), the figure displays the time interval as x coordinate interval. ***Scale range*** has a combobox to change the voltage range of the signal it collects. There are 13 options of voltage ranges (-1mv-1mv, -2mv-2mv, -5mv-5mv, -10mv-10mv, -20mv-20mv, -50mv-50mv, -100mv-100mv, -200mv-200mv, -500mv-500mv, -1v-1v, -2v-2v, -5v-5v,-10v-10v), and the figure displays the voltage range as y coordinate interval. ***Y shift*** has a trackbar to move up and down for y coordinate. There are 11 levels (-5000, -4000, -3000, -2000, -1000, 0, 1000, 2000, 3000, 4000, 5000) can be chosen to move the y coordinate. ***Amplify*** has a trackbar to amplify the signal. There are 11 levels (-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5) have been chosen to amplify the signal it collected. ***Trigger Threshold*** has a combobox to choose the threshold which signal can be triggered. There are 5 voltages (1v, 2v, 3v, 4v, 5v) can be chosen. ***Trigger Channel*** has a combobox to change the channel which can be triggered. There are 4 channels can be triggered. ***Sampling points*** has a combobox to change the points which can be sampled. There are 4 options of points (512, 1024, 2048, 4096), and the figure displays as the curve. ***Delay*** has a combobox to control the delay period of the signal, there are 5 percentages we can choose (0, 20%, 40%, 60%, 80%, 100%). ***Time Domain*** has two checkboxes to change the domain which the figure can display. There are two options of domain (time, frequency), which are calculated by fast Fourier transform (FFT). There is only ***open mode*** combobox to choose the mode between simulation and real time mode in settings part. In processing part, it has 5 checkboxes to control the properties, of the signal. ***De-noise*** can reduce the noise of the signal, and make the signal accuracy. ***Low Pass*** can make the signal pass the filter which signal is lower than cutoff frequency. ***High Pass*** can make the signal pass the filter which signal is higher than cutoff frequency. ***Band pass*** can make the signal pass the filter which signal is between lower cutoff frequency and higher

cutoff frequency. ***Notch pass*** can make the signal pass the filter except the cutoff frequency. When simulation data actives, the control option has the same functions, and all the components show below (Figure 4.21).

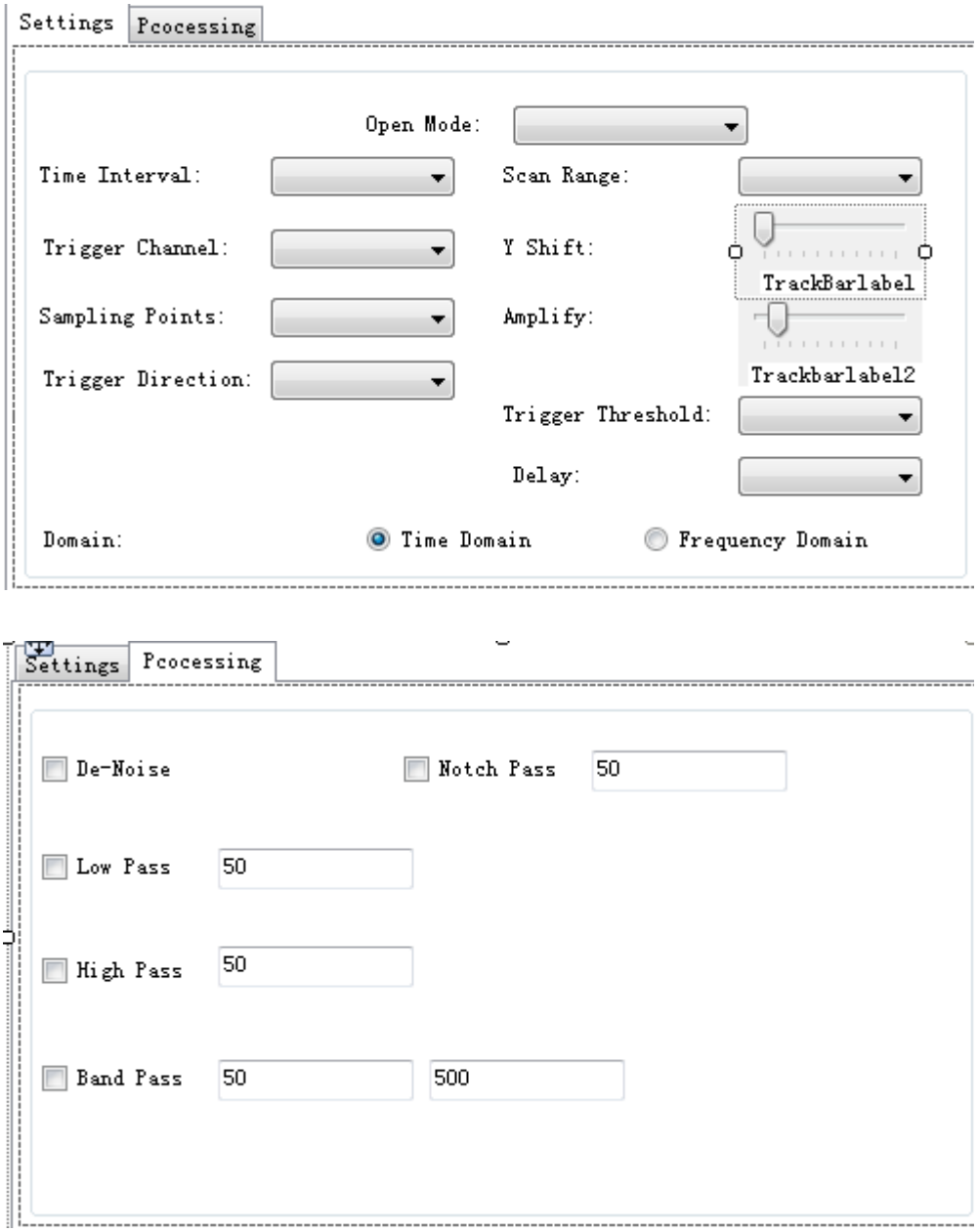


Figure 4.21 The "Settings" of "Realtime" View

4.6.2"Measurement" View

The "Measurement" View is for conducting measurements and analyzing data.

There are also two components in "Measurement" view, "Figure" and "Measurement control".

"Figure": There are three charts in this component, one of them can display the average result of the original curve by using blue color, and can also display the curve which is fitted by using least square fitting algorism by using yellow color when we click the ***"Measurement"*** button in "settings" component. In this chart, x coordinate means time interval and y coordinate means voltage (unit: mv). The second chart can display the hydration of its measurement every time. In this chart, x coordinate means the times it measures, and y coordinate means hydration (%), the last chart can show the depth of the skin which OTTER can measure. (Figure 4.23, measure 2 times).

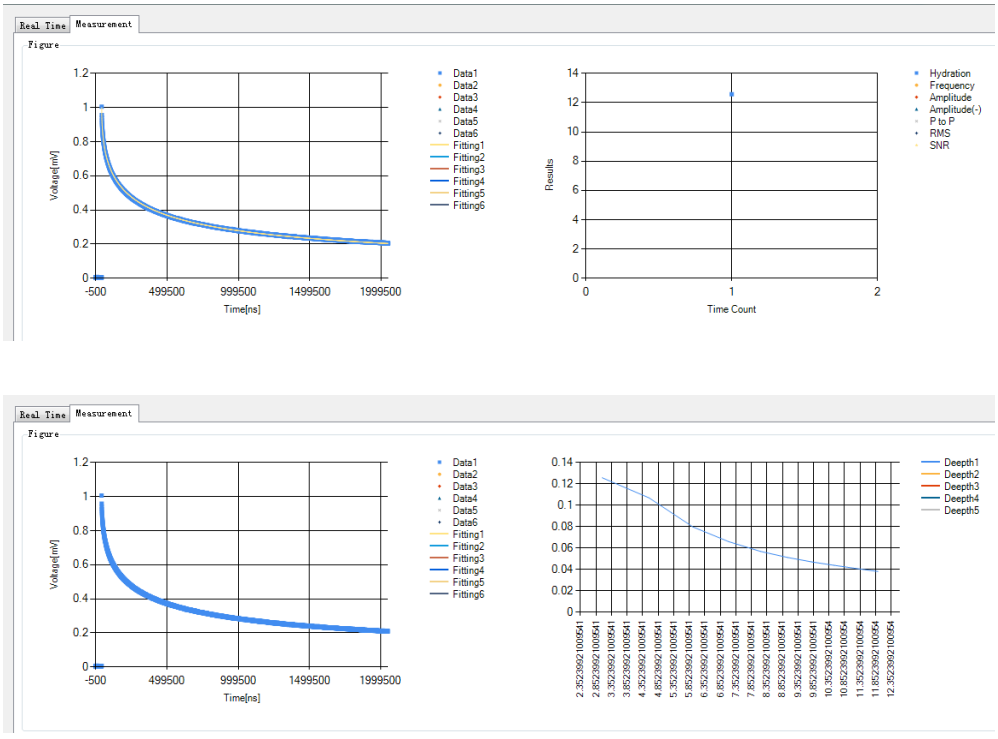


Figure 4.23 the "Display" of "Measurement" part

"Measurement Control": There are 6 control options in "measurement control" component. ***Average Times*** has a textbox to control the times it measures, and then display the average result in first chart of "Figure" component, we can input any

times we want, but more times it measures, slower display in chart. Close to average times textbox, it shows 6 parameters (frequency amplitude, minus amplitude, peak to peak value, RMS, SNR) to display the specification of signal we analyze. ***Saving (*xlsx)*** has two buttons and two textboxes, the buttons can scan the computer disk to save the results it receives, and the textboxes can show the location where the results save, it can save the results to different sheets in one word excel file depend on different channels. In the up word excel (xlsx) file, it saves the basic information of this measurement, and the values of each point it samples, in the below xlsx file, it saves the up file name, date, hydration gradient, and parameters which it receive after curve fitting (Figure 4.24). ***Comment*** has one textbox to input the comment into the file it saves. ***Open data file*** has a button to open the saved file, and then re-calculate again, and it can also re-calculate multiple data. ***Fitting Model*** has a combobox to choose 7 models it can fit: realtime1, realtime2 (A, Tau), realtime3 (A, Tau, W), realtime4, simtime1 (A, Tau), simtime2 (A, Tau, W), simtime3. For the details, it has been introduced in chapter 3.

Figure 4.24 the "Settings" of "Measurement" part

Figure 4.25 shows the working time of the program in measurement part.

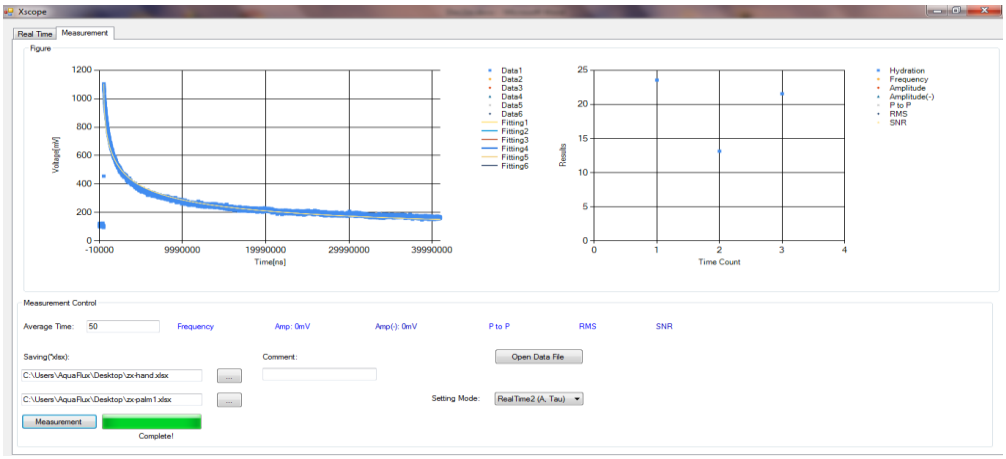


Figure 4.25 the working time of the program in "Measurement" part

In summary, the specification of Pico 4000 and 2000 series oscilloscopes is described, it also describes the overall structure of the new OTTER software program, the functions of its 8 modularized libraries and two main parts of the program for data acquisition and data analysis.

4.7 Opto-Thermal Multiple Wavelength and Depth Resolved Detection in vivo-skin measurements

4.7.1 Multiple Wavelength Detection

Although OTTER is fully spectroscopic measurement technology, only a few detection wavelengths (e.g. 13.1um and 9.5um) are used in OTTER measurements previously. All other detection wavelengths are ignored. In this work, we will study the effect of multiple detection wavelengths, i.e. 13.1um, 11.5um, 9.5um, 8.5um, 7.79um, 6.48um, 6.05um.

In this first experiment, four different test skin sites (A: upper arm, B: lower arm, C: back hand, D: palm) are chosen on a health volunteer (male, aged 30), and six different detection wavelengths (13.1um, 11.5um, 9.5um, 8.5um, 7.79um,

6.48um, 6.05um) were used. Figure 4.26 to 4.29 shows the raw OTTER signal at six different detection wavelengths.

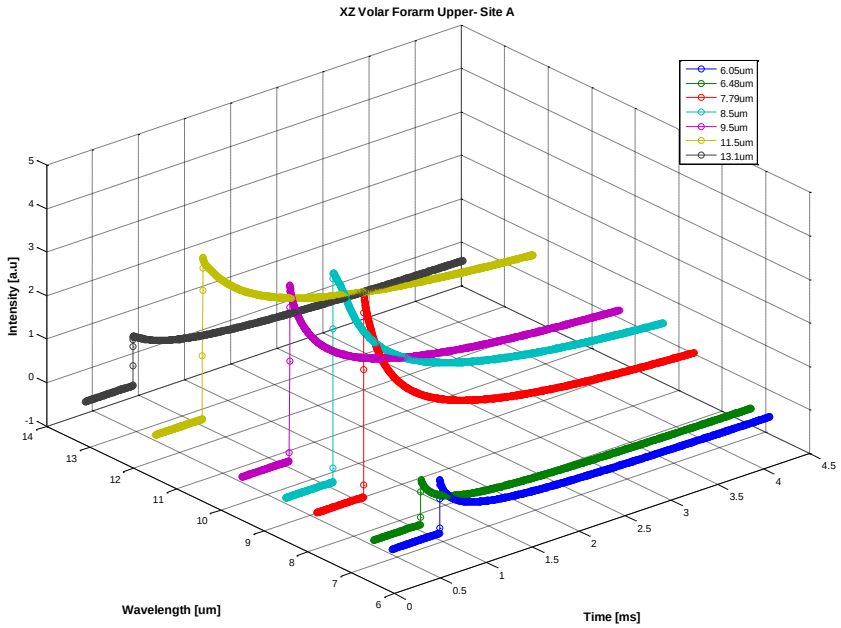


Figure 4.26 the OTTER signals at different detection wavelengths for skin site A.

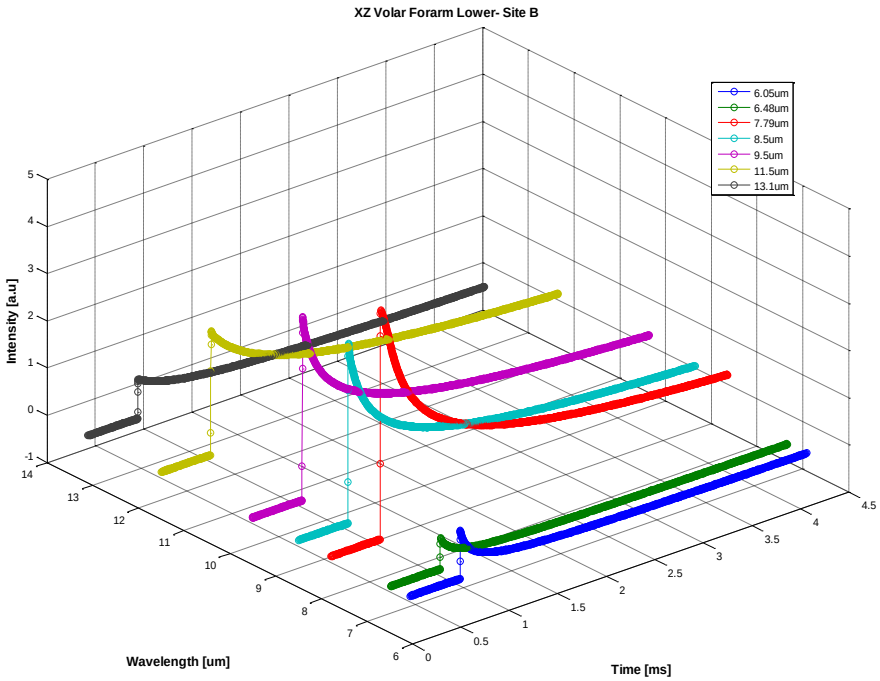


Figure 4.27 the OTTER signals at different detection wavelengths for skin site B

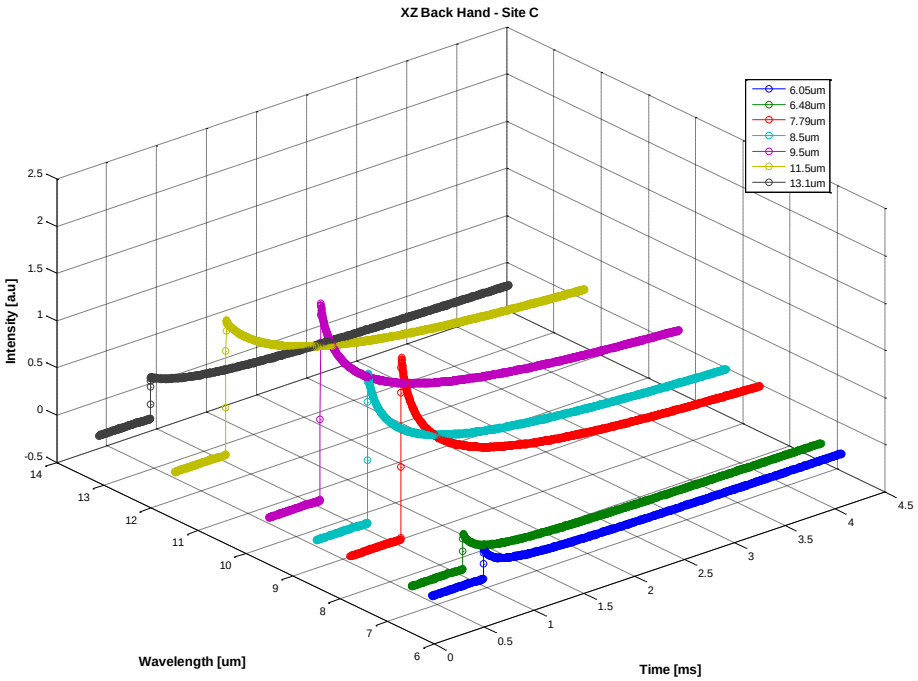


Figure 4.28 the OTTER signals at different detection wavelengths for skin site C

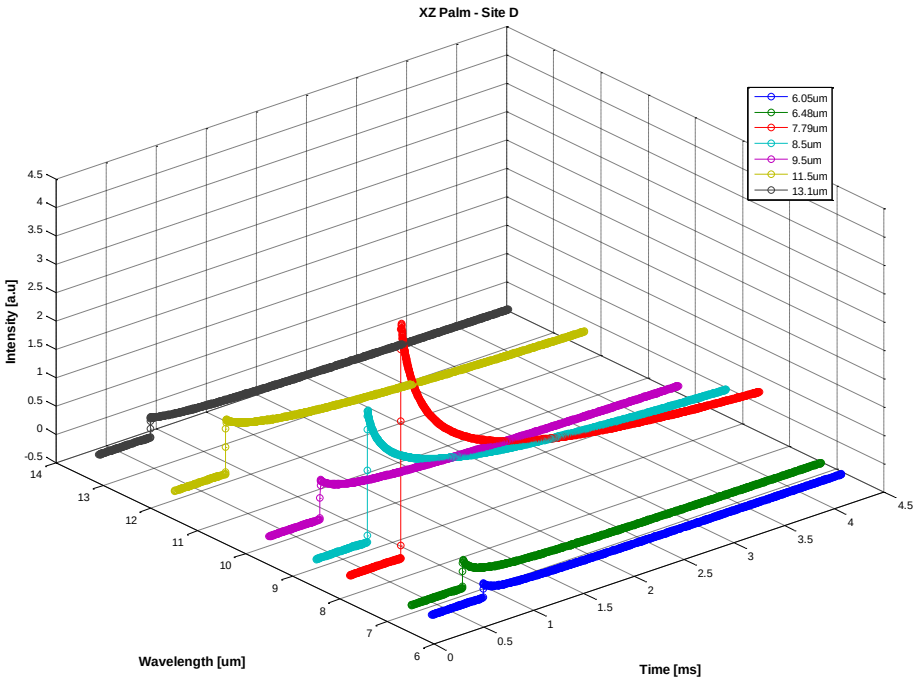


Figure 4.29 the OTTER signals at different detection wavelengths for skin site D

By analyzing the above raw OTTER signals using least squares (LS) fitting using Eq.(3-6), we can get the best fit decay lifetime, from which we can get detection absorption coefficient β, and using Eq.(3-10) we can work out the skin water hydration in percentage (%).

Figure 4.30 shows the skin hydration of four different test parts with six different wavelength laser pulses. All results are measured average results of two times in one measurement, the error bars in the figure are based on the standard deviation (SD) of OTTER, which is calculated by two times results of one measurement. Please note that although the results are presented as hydration percentage, only the results at wavelength 13.1um represents the true water content (Xiao P., 1997), the values at other wavelength are just calculated results.

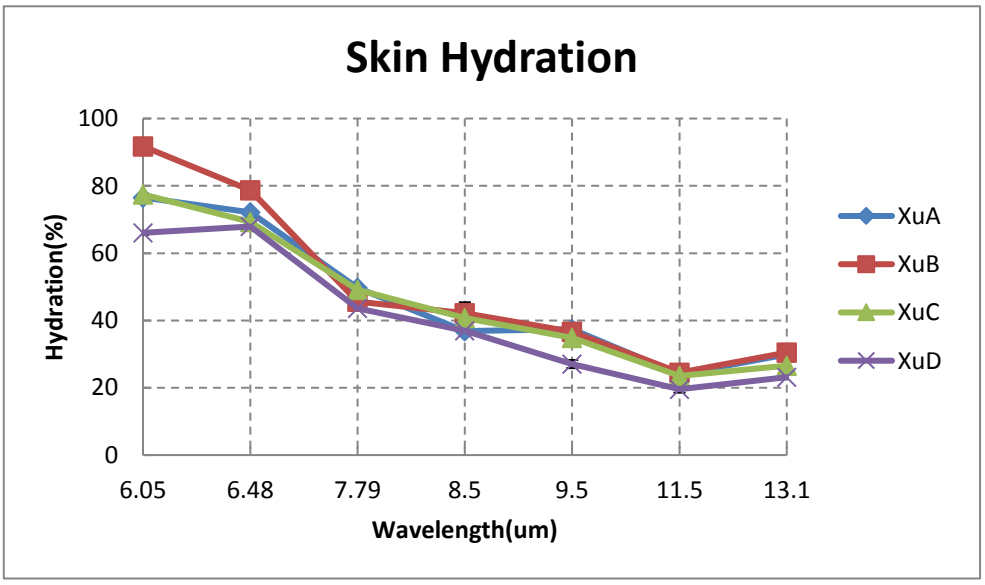


Figure 4.30 skin hydration of different parts and wavelength

The four skin sites were then treated differently in order to observe the changes. Skin site A was washed by a soap twice, for 3 minutes each time. Figure 4.31 shows the comparison of skin hydration between baseline and after soap washing in skin site A. The results show that the hydration values increased little at

wavelength 6.05um and decreased little at 6.48um, the other values keep nearly the same, which indicating soap washing cannot affect the skin in a short time.

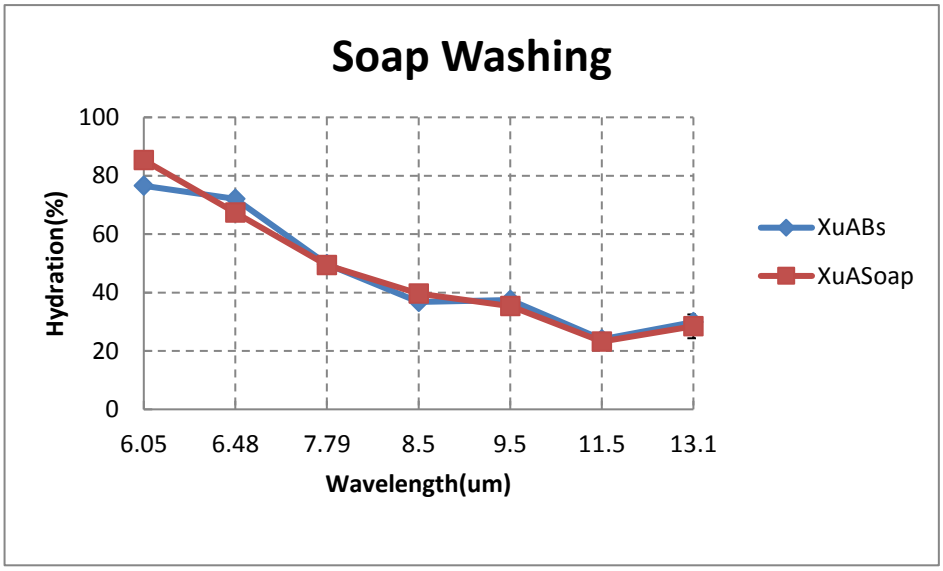


Figure 4.31 skin hydration between baseline and after soap washing

The skin site B is applied with pure ethanol for 5 minutes. Figure 4.32 shows the comparison of skin hydration before and after ethanol application on skin site B.

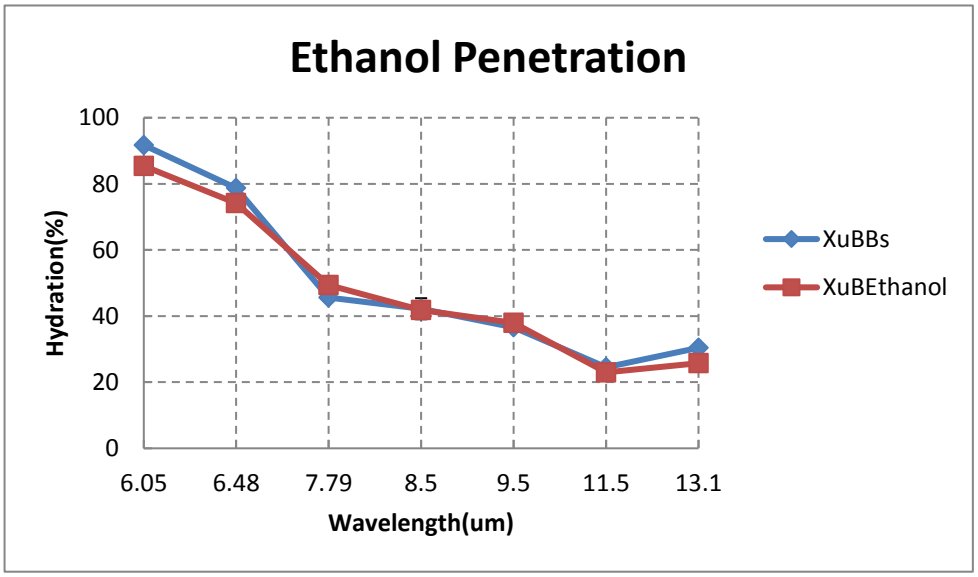


Figure 4.32 skin hydration before and after ethanol application.

The results show that ethanol changes the skin very little, although a small

decrease is observed, especially at 6.05, 6.48 and 13.1um, which indicating a very little drying effect due to the ethanol application.

Skin site C is applied with pure DMSO for 5 minutes. Figure 4.33 shows the comparison of skin hydration before and after DMSO application in skin site C, all results are measured results.

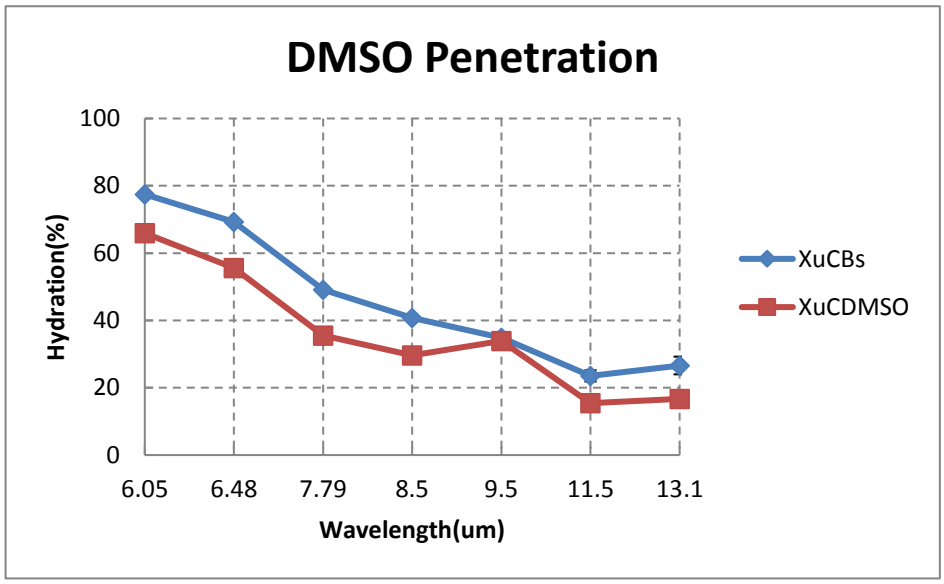


Figure 4.33 skin hydration before and after DMSO application.

The results show that the hydration values decreased nearly at all the wavelengths except 9.5um, which indicating a drying effect due to the DMSO application

Skin site D is applied with ethylene glycol 5 minutes. Figure 4.34 shows the comparison of skin hydration before and after ethylene glycol (EG) application, all results are measured results. The results show that ethylene glycol causes changes at almost all wavelengths, except 7.79um. Again, the biggest changes happen at 6.05, 6.48um, (decreased) and 9.5um (increased).

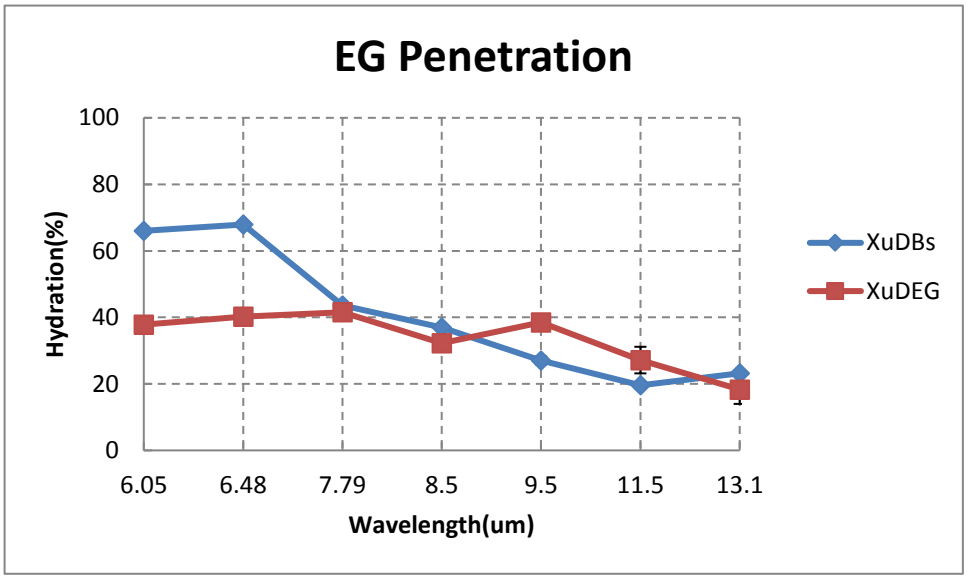


Figure 4.34 skin hydration before and after ethylene glycol application.

In summary, the overall results show that OTTER multiple detection wavelength signals can effectively differentiate the different types of skin changes, and therefore could potentially very useful for skin characterisations. It can also potentially identify different types of solvents applied on skin, even with mixed solvents.

4.7.2 Depth Resolved Profiles

By fitting the OTTER signal at 13.1um wavelength using enhanced SLS fitting, described in Chapter 3, we can also get the depth resolved hydration profiles of each skin sites. Figure 4.35 shows the hydration depth profiles of four different skin sites

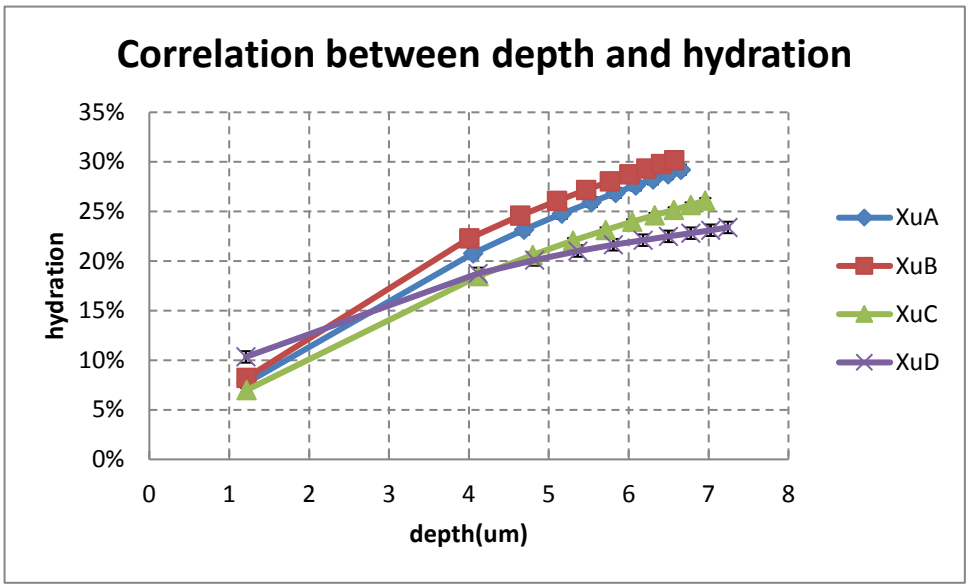


Figure 4.35 skin hydration in different depths of 4 different parts

The results show that all four skin site have similar hydration depth profiles, and the hydration increases with the depth. The skin sites A and B are from volar forearm, they have the highest hydration content and hydration gradient. The skin site D is from palm, which has the lowest hydration content and hydration gradient. The skin site C, from hand, is somewhere in the middle.

Figure 4.36 shows the comparison of skin hydration in different depths before and after soap washing in skin site A, all results are measured results. The results show that the hydration near surface increases after the wash, but the hydration at deeper skin hardly changes. This is very interesting, as it suggests that the soap washing affects more the skin layers near the surface, and affects less the skin deep down.

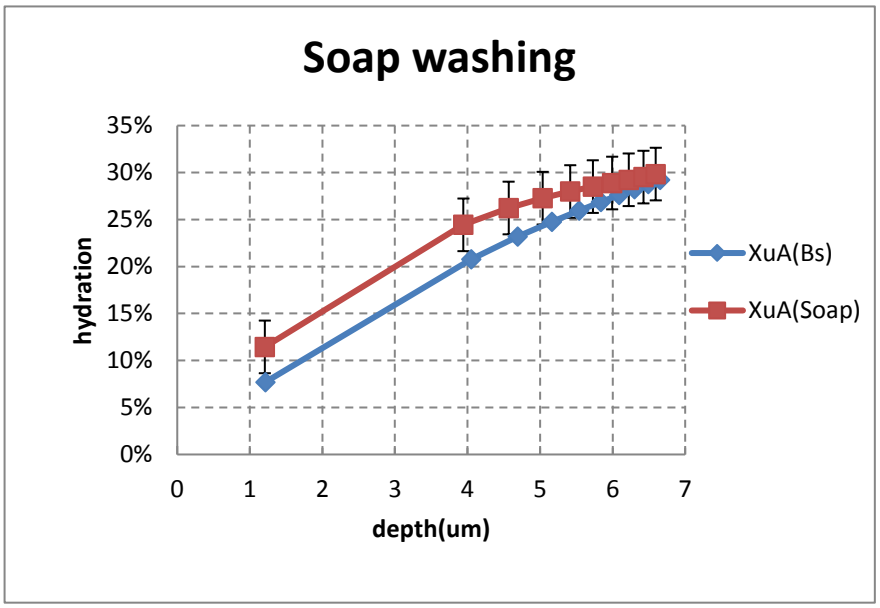


Figure 4.36 skin hydration in different depth before and after soap washing

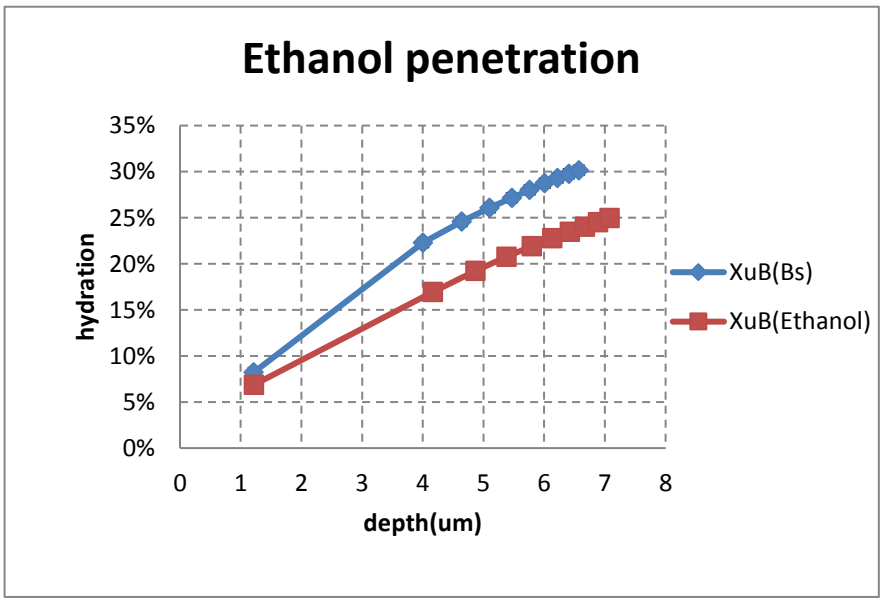


Figure 4.37 skin hydration in different depth before and after ethanol application.

Figure 4.37 shows the comparison of skin hydration in different depths before and after ethanol application in skin site B, all results are measured results. The results show that the hydration decreases after application, indicating a drying effect. It is interesting to point out that different from skin site A, where the changes are mainly happening at the surface, the changes here is mainly

happening at deeper, and surface only has small changes. This suggests that ethanol has penetrated into the skin and causes skin drying at all depths.

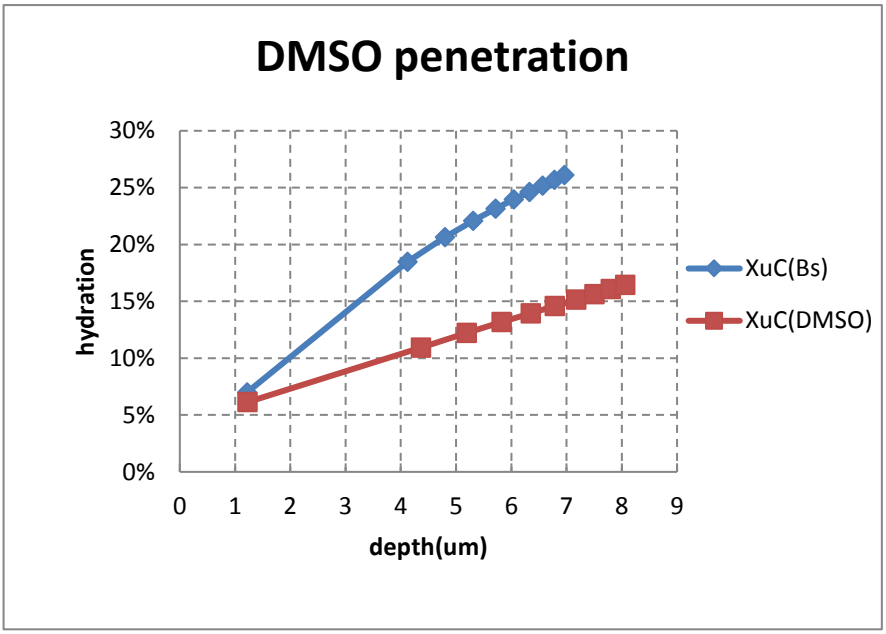


Figure 4.38 skin hydration in different depth before and after DMSO application.

Figure 4.38 shows the comparison of skin hydration in different depths before and after DMSO application in skin site C, all results are measured results. The results show that the hydration decreases after DMSO application, indicating a drying effect. Again, this decrease happens mainly at deeper skin, where skin surface has little changes. This also indicates the DMSO also penetrated into deep skin. From the scales of the changes, we can also conclude that DMSO damages skin more than ethanol.

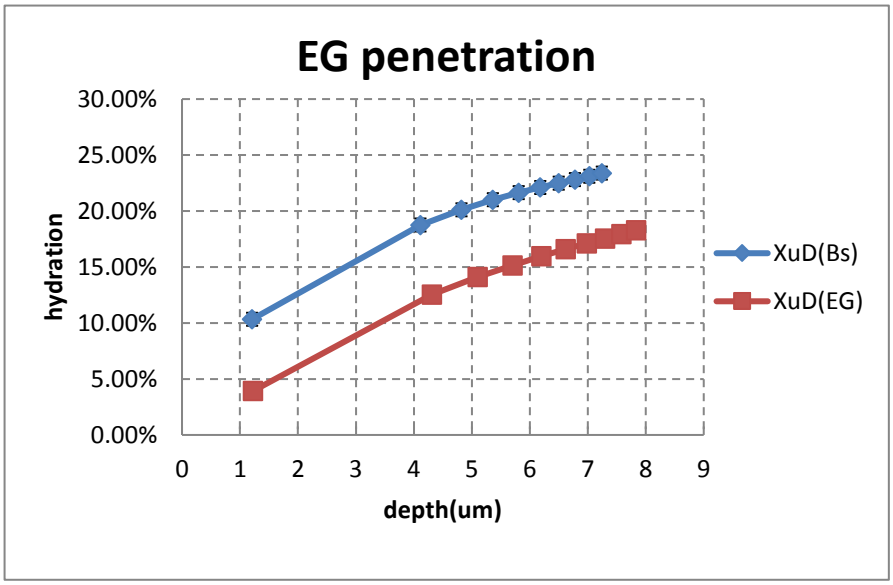


Figure 4.39 skin hydration in different depth before and after ethylene glycol application.

Figure 4.39 shows the comparison of skin hydration in different depths before and after ethylene glycol (EG) application in skin site D, all results are measured results. The results also show that the hydration decreases after EG application, indicating a drying effect. In this case, the decrease happens at all depths including surface, which indicates EG affects the skin at all depths.

In summary, the depth resolved hydration profiles shows very interesting results. It can not only show the differences of the different skin site, but also show the different types of skin changes. The depth hydration profiles show that washing only affect the skin surface layers, whilst the solvents, e.g. ethanol, DMSO, and EG can penetrate deep into skin, and affect skin at all depths. This makes OTTER a great tool to study skin damage assessments and skin solvent penetrations.

4.7.3 Skin Characterization by Soap Washing

This experiment presents the skin characterization by three different brand (A, B, and C) soaps. OTTER is used for this experiment, three different skin test sites on the forearm of two volunteers were chosen, and washed by using three brand soaps for 4 minutes, respectively. The measurements were repeated for five days, D1, D2, D3, D4, D5, with D1 is baseline, before the washes. OTTER is a single point contactless technology, hence by creating a 4 x 4 point matrix on the test sites (Figure 4.40), we can then measure the skin hydration of every point and calculate the average result as the test site hydration. We can also generate a skin image using the points.

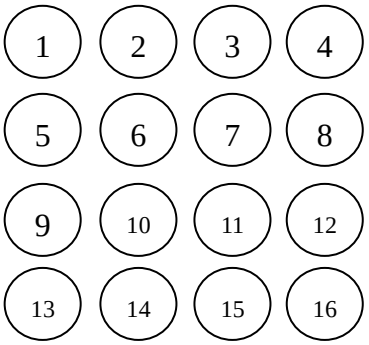
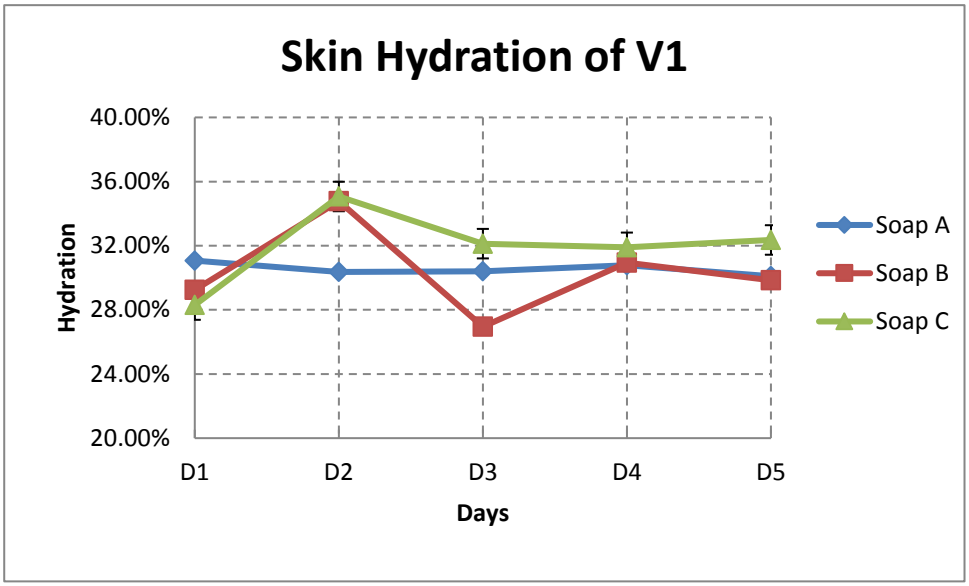


Figure 4.40 OTTER measurement matrix on skin site.

Figure 4.41 shows the 5-days skin hydration from OTTER of two volunteers by three different brand soap washing.



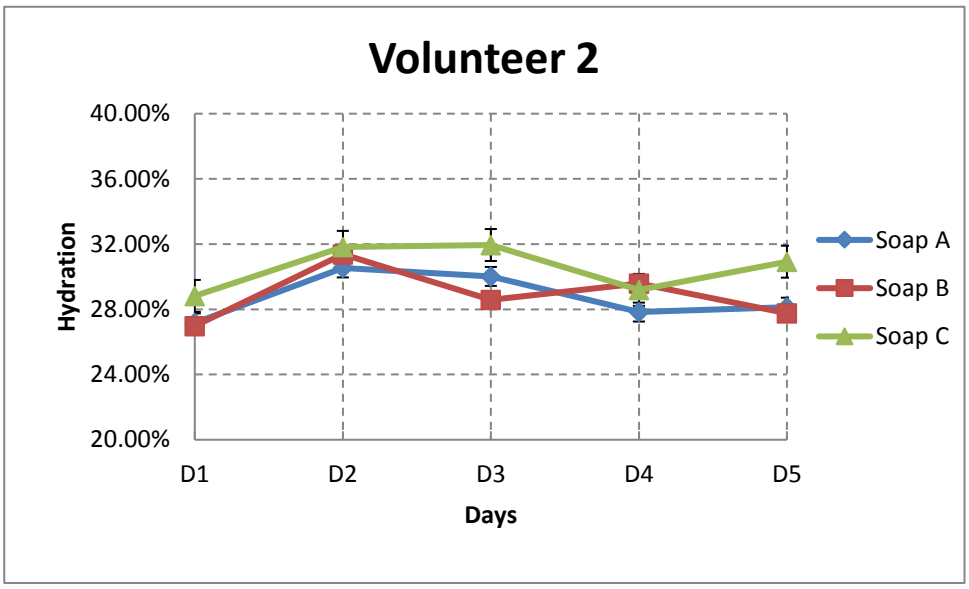


Figure 4.41 hydration of three different soaps on two volunteers

From curve charts, they show that for both two volunteers, the value of test skin site become higher than day 1 in day 2, and then decreased in other three days gradually except the Soap C of volunteer 2 in day 5, because the test skin site become so dry that the skin has been damaged.

4.7.4 3D Depth Profiling

By plotting all 16 measurement points data (Figure 4.40) together, analyzed by using enhanced SLS fitting, we can also create a 3D skin hydration depth profiles. As shown in Figure 4.42. The red color represents high water content and blue color represents low water content. The results show at Day 5 skin hydration depth profiles have significantly changed due to washes. These skin hydration depth profile changes are possibly due skin surface layer removal or skin irritations caused by washes.

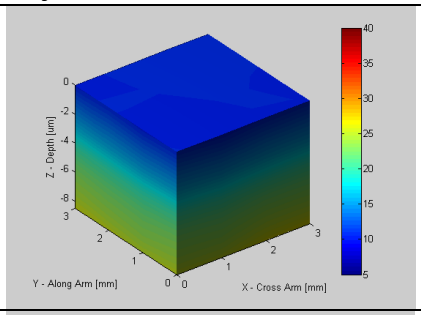
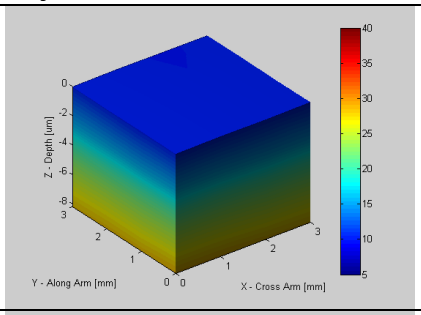
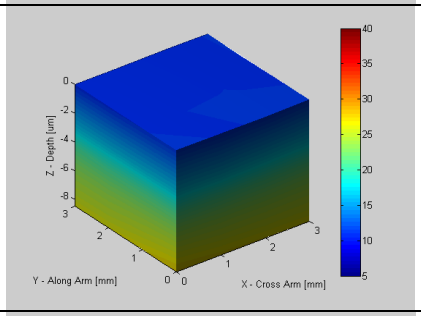
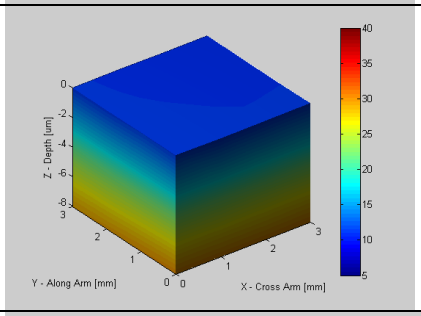
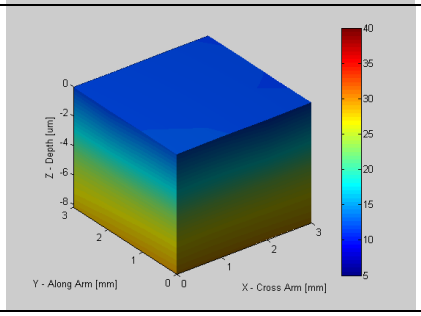
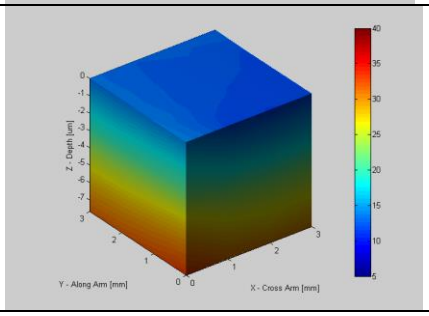
ZX	Day 1	Day 5
Soap A		
Soap B		
Soap C		

Figure 4.42. The 3D maps of skin hydration depth profiles of the skin sites of volunteer V1, on day 1 (before washes) and day 5 (after 4 days washes).

4.8 Summary

This chapter describes the development of the new OTTER data acquisition and data analysis software based on PicoScope Technology. It first introduces the basic background of PicoScope Technology, with a focus on PicoScope 4000 series and PicoScope 2204 oscilloscopes, then it describes the overall structure of the new OTTER software program, the modularized data analysis libraries and its functionalities.

Comparing with the existing OTTER software based on LabView, The new OTTER software offers many advantages: USB based, higher data sampling rate (80MS/s), better ADC (Analog-to-digital converter) resolutions (up to 16 bits), multiple programming language support (C/C++, C#, Visual Basic etc.), and multiple systems support, i.e. Windows, MAC, and Linux etc. It can also read data from multiple channels and multiple devices simultaneously. This is very useful when array detector is used. It also saves data in Excel format, which makes it easier for further analysis.

In OTTER experiments, the multiple wavelength detection shows very interesting results. In the past, with single wavelength detection, when signal changes, we don't know what causes the change. With multiple wavelength detection, it is possible identify what causes the changes. It can clearly differentiate different types of skin damage, and different types solvents applied.

The depth resolved hydration profiling using enhanced SLS fitting also shows very interesting results. It shows that washing only affects the skin surface layers, whilst the solvents, e.g. ethanol, DMSO, and EG can penetrate deep into skin, and affect skin at deeper depths.

The combination of multiple wavelength detection and depth profiling makes

OTTER a potential great tool to study skin damages and skin solvent penetrations. OTTER can measure not only the scale of the damages but also the types of damages, it can also differentiate different types of solvents on skin, and possible to detect multiple substances on skin surface.

By using 2D matrix points and enhanced SLS fitting, we can also create 3D skin hydration depth profile maps.

Chapter 5 Skin Hydration and Solvent Penetration Measurements

This chapter describes experimental investigation of skin hydration and skin solvent penetrations by using different instruments such as AquaFlux, Epsilon (Fingerprint sensor), Corneometer etc. these experiments are aimed to have a better understanding on *in-vivo* skin characterizations, skin hydration, trans-epidermal water loss (TEWL), solvent penetration through both *in-vivo* and *in-vitro* skin, the performance of each instrument and correlation between different instruments.

It is important to point out that during the course of this project work, Epsilon has gone through from a bare sensor, Fingerprint Sensor, which is uncalibrated and generates only grayscale images, to a fully calibrated permittivity imaging system, which generates full color RGB images. Experiments in section 5.1 - 5.4 are done by using the uncalibrated Fingerprint Sensor, while experiments in sections 5.5 – 5.7 use calibrated Epsilon permittivity imaging system. These experiments are vital for understanding the performance of Epsilon and its development.

5.1 Skin Instrumentation Evaluation Measurements

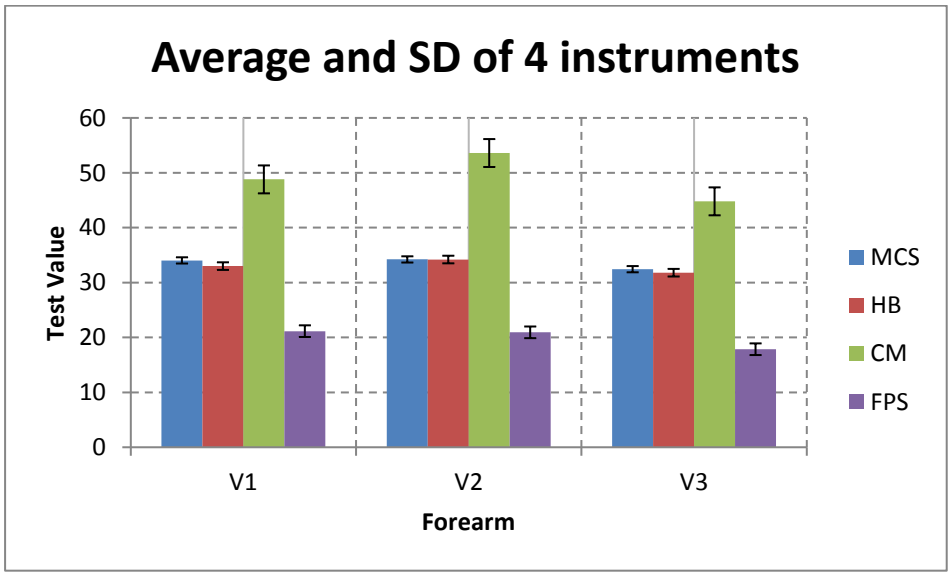
5.1.1 Repeatability of Different Instruments

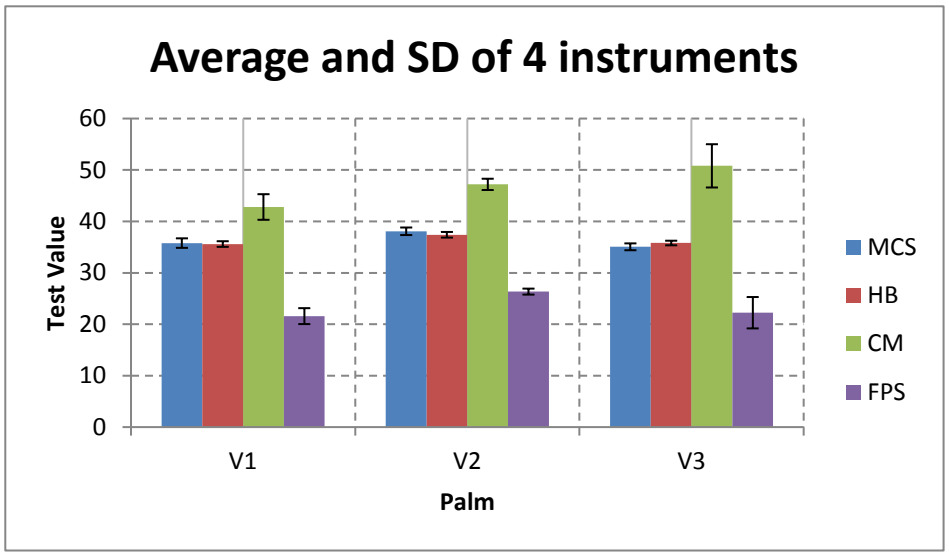
The aim of this experiment is to evaluate the repeatability of different skin measurement instruments. In this experiment, six different normal skin sites (volar forearm, palm, forehead, face, neck, lower leg) were chosen, four different instruments (Fingerprint Sensor, Corneometer, Hydrotest, and Moisture Checker)

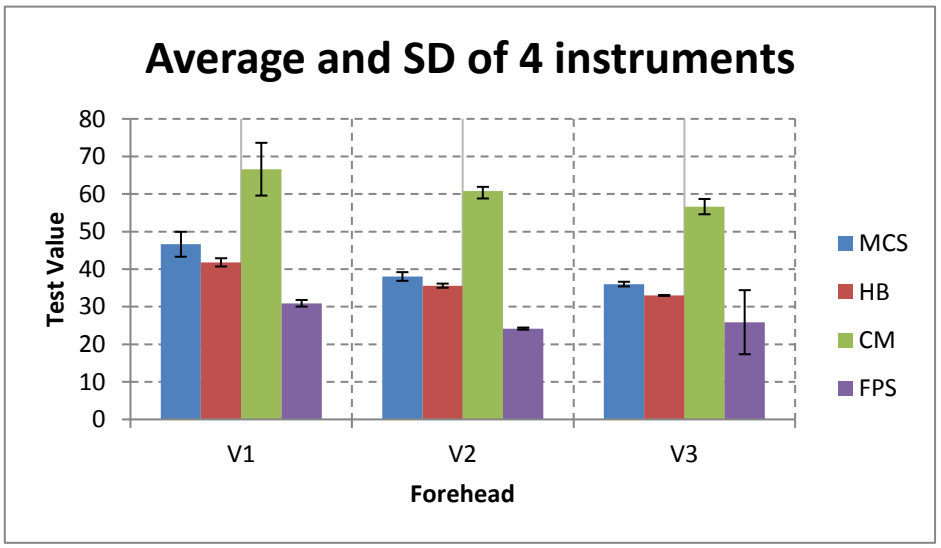
were used to measure each skin site repeatedly (five times), and the average values and standard deviation were calculated.

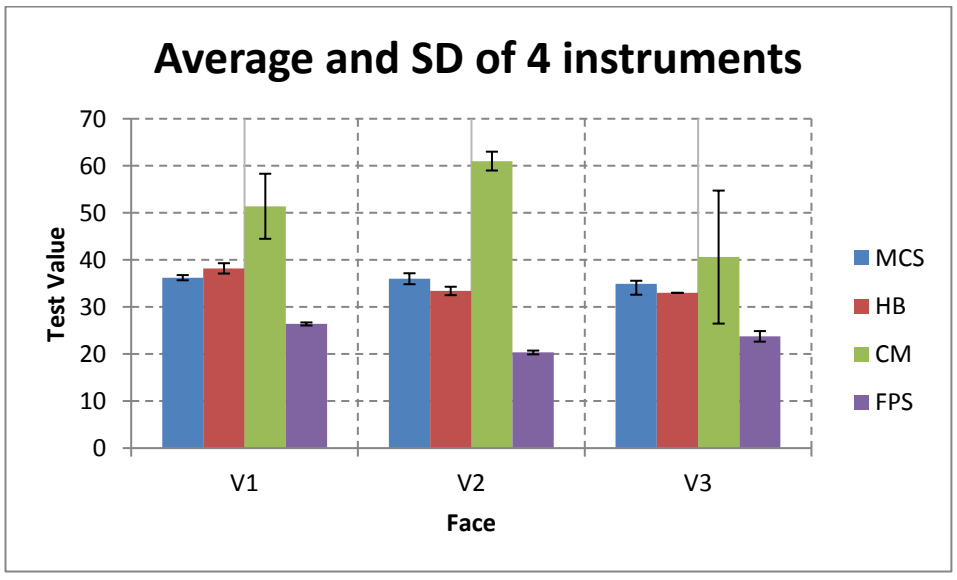
Figure 5.1 shows the average and standard deviation (SD) results of five measurements repeatedly on six different sites of three volunteers by using 4 different instruments, with FPC - Fingerprint Sensor, CM - Corneometer, HB - Hydrotest, and MCS - Moisture Checker.

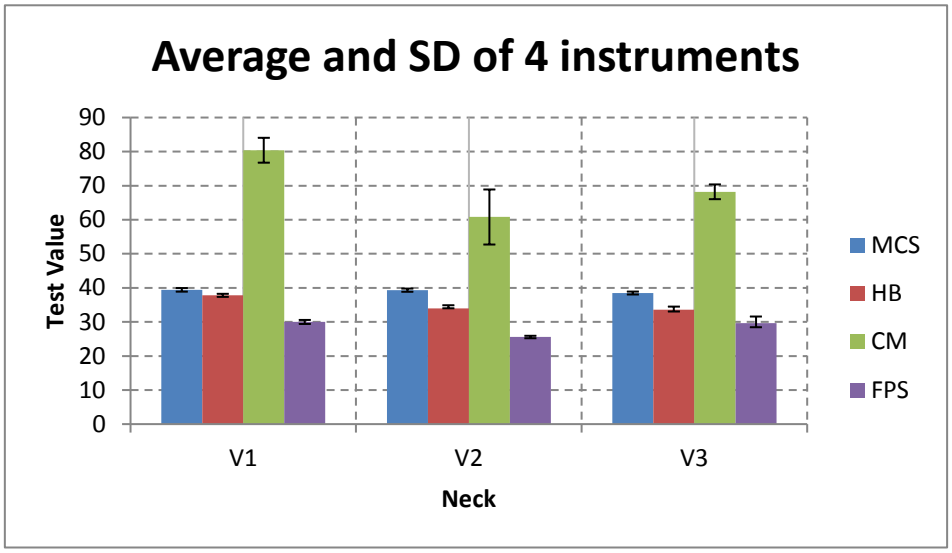
The coefficient of variation (CV) can reflect the repeatability of the instruments, and the CV of 4 instruments are 2% (MCS), 1% (HB), 7% (CM) and 3% (FPS) respectively. In general, all instruments show the good repeatability, and between different instruments, MCS, HB and FPS have the better repeatable than that of Corneometer.




Average and SD of 4 instruments
Test Value
60
50
40
30
20
10
0
V1
V2
V3
Palm
MCS
HB
CM
FPS


Average and SD of 4 instruments
Test Value
80
70
60
50
40
30
20
10
0
V1
V2
V3
Forehead
MCS
HB
CM
FPS


Average and SD of 4 instruments
Test Value
70
60
50
40
30
20
10
0
V1
V2
V3
Face
MCS
HB
CM
FPS



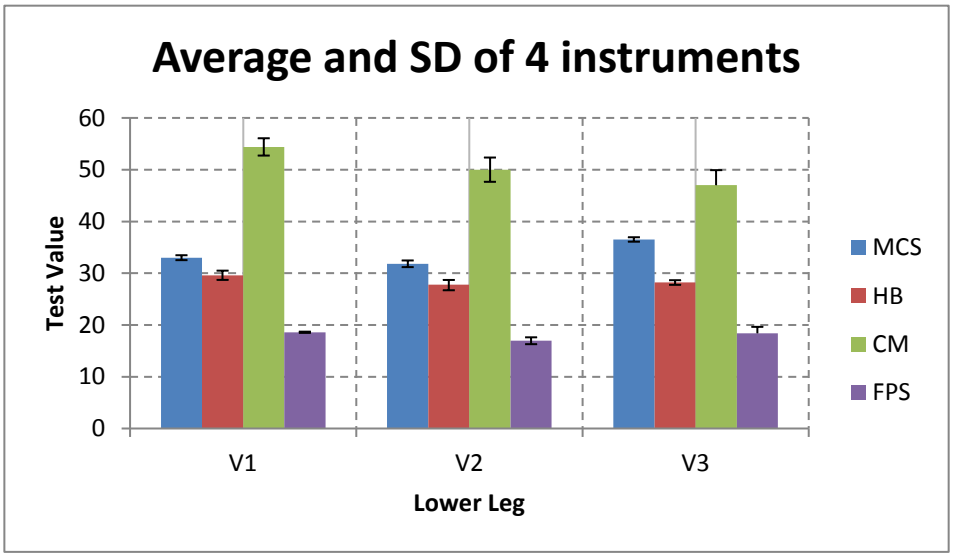
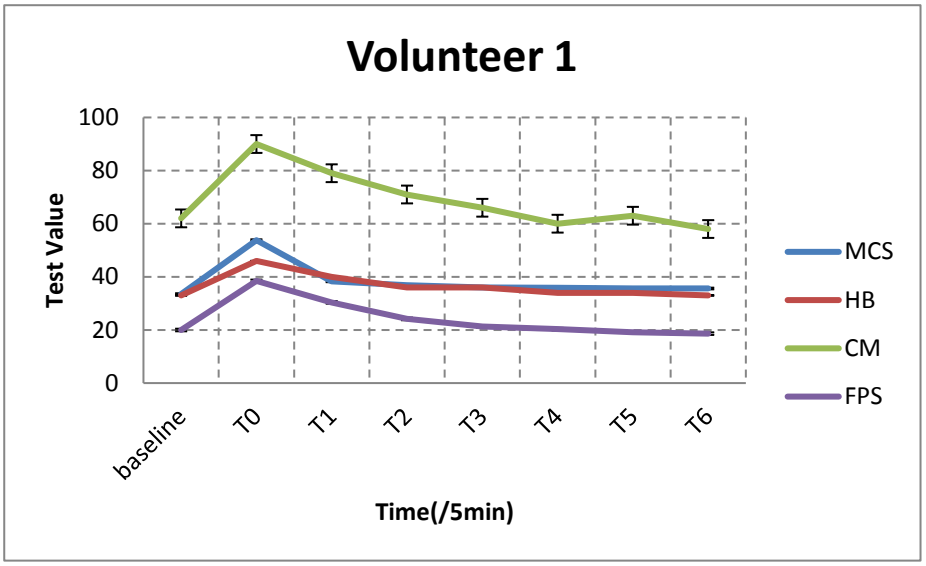


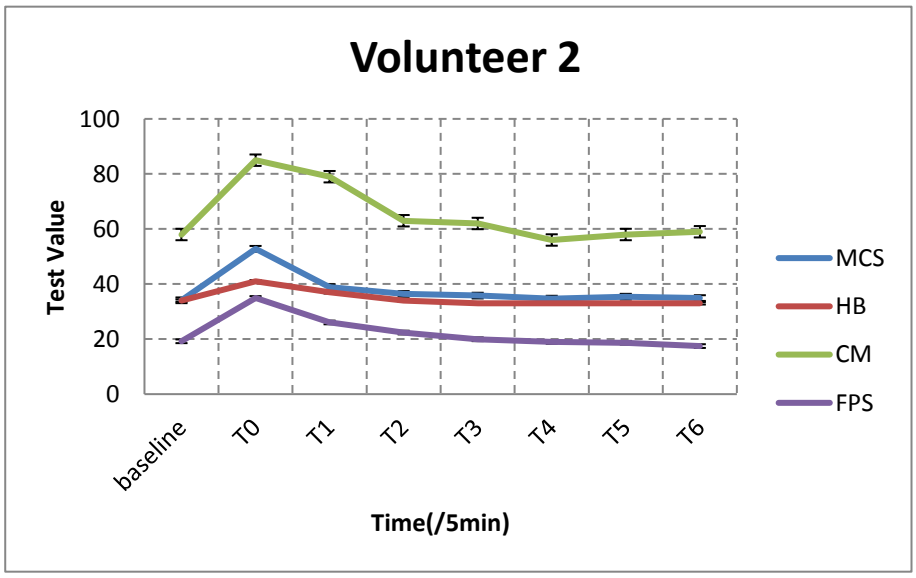
Figure5.1 The results of four different instruments from below to bottom are forearm, palm, forehead, face, neck and lower leg.

5.1.2 Correlations of Different Skin Instruments

The aim of this experiment is to evaluate the correlations of different skin instrument at the same skin site but different skin hydration levels. In this experiment, a skin site on the left volar forearm was chosen, soaking in very wet tissue for 45min. Four healthy volunteers (aged 20 -30) were selected. Four different instruments were used to measure before and after soaking (0, 5, 10, 15,

20, 25, 30 minutes).





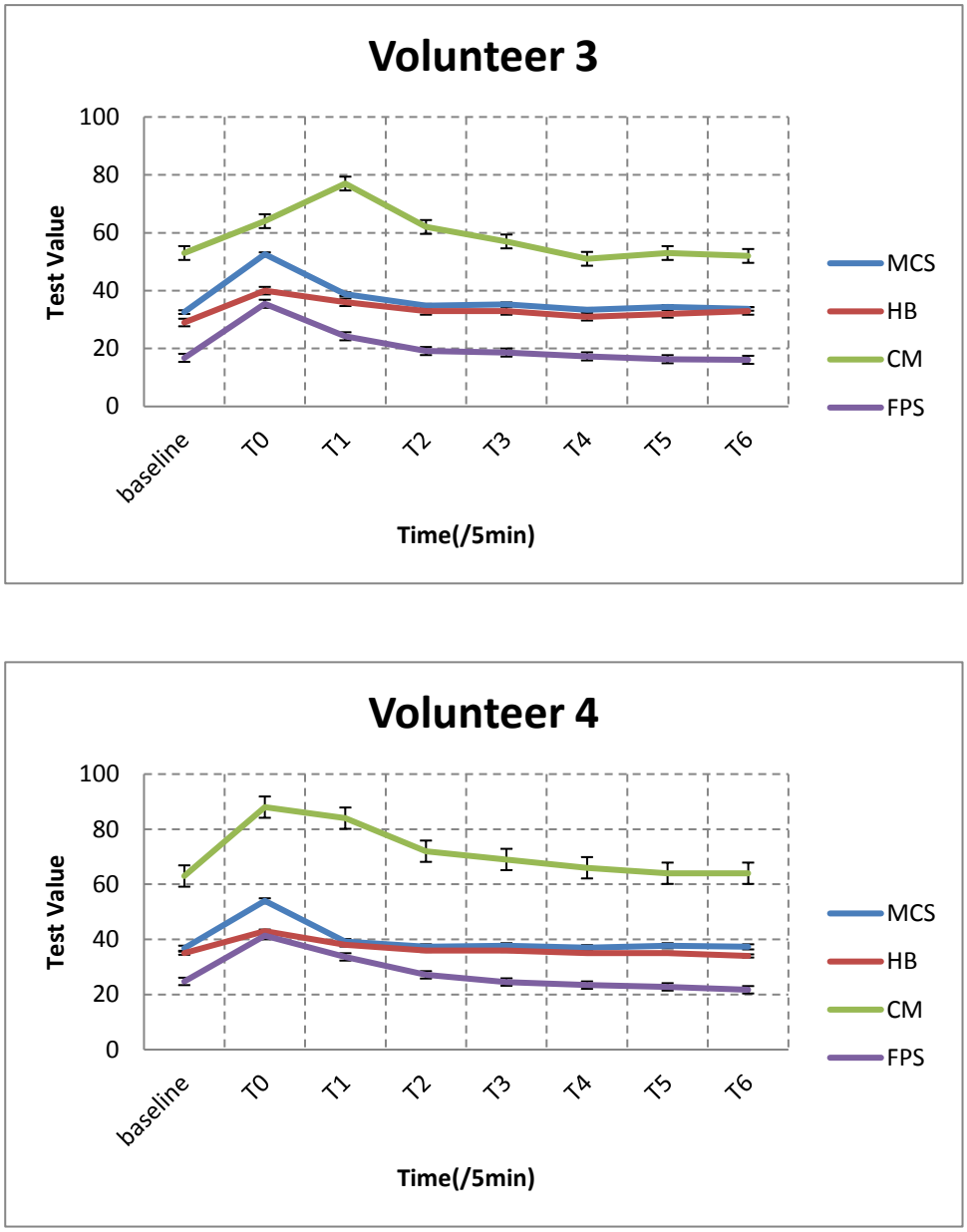
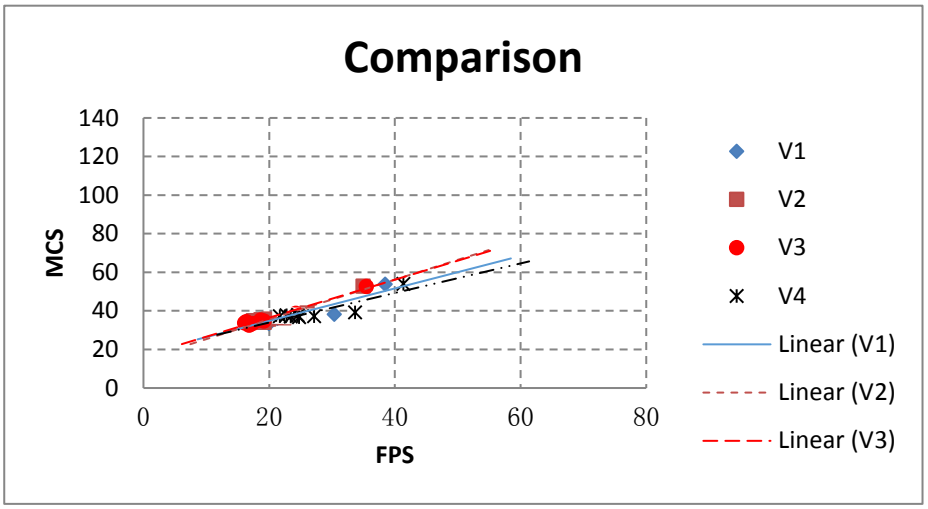
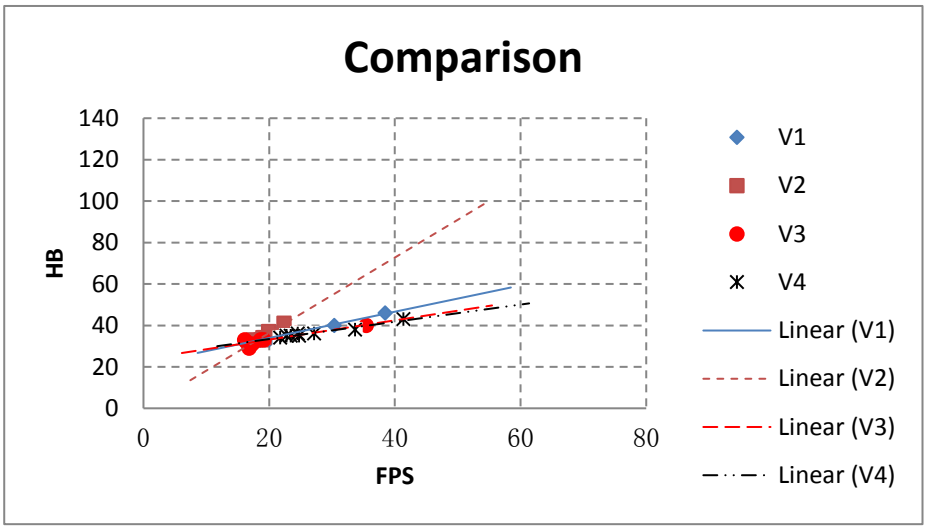


Figure 5.2 Skin hydration of four volunteers

Figure 5.2 shows the skin hydration of four different instruments on the test skin site of four volunteers. The error bars in the figure are based on the standard deviation (SD) of each instrument, as shown in Figure 5.1. The results show that the results change due to hydration changes are more significant than that of error bars. The results also show that, after 45 minutes soaking, the skin hydration increased to a higher level. Then, as skin recovers under ambient condition, skin hydration gradually reduced back to its normal level.


Comparison
MCS
FPS
0
20
40
60
80
100
120
140
V1
V2
V3
V4
Linear (V1)
Linear (V2)
Linear (V3)


Comparison
HB
FPS
0
20
40
60
80
100
120
140
V1
V2
V3
V4
Linear (V1)
Linear (V2)
Linear (V3)
Linear (V4)

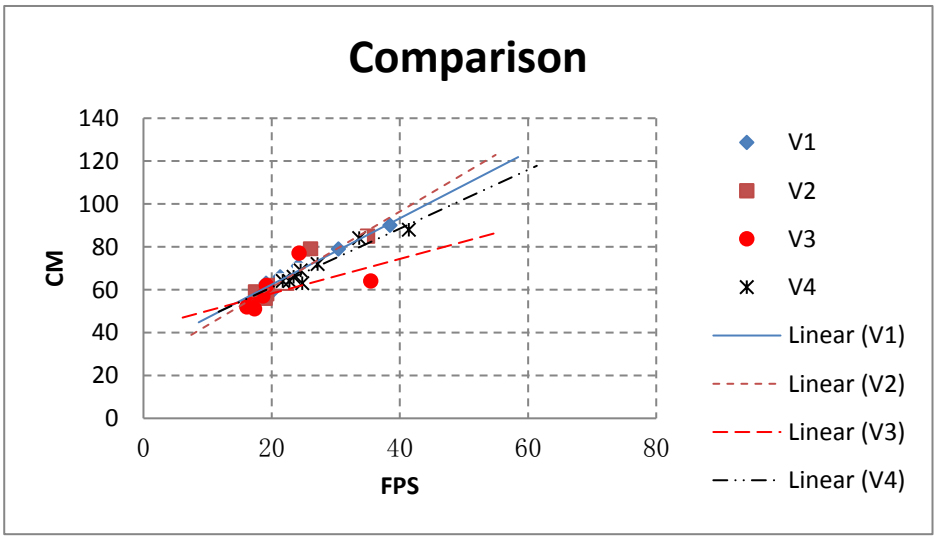

Comparison
CM
FPS
0
20
40
60
80
100
120
140
V1
V2
V3
V4
Linear (V1)
Linear (V2)
Linear (V3)
Linear (V4)

Figure 5.3 shows the comparison between Epsilon and other instruments with skin hydration measurement on the left forearm of four volunteers. The results show that Fingerprint Sensor has reasonable good correlations with other three instruments, Moisture Checker, Hydratest, and Corneometer. However, different volunteers seem to have slightly different correlations, this could due to the differences in skin texture, skin softness, and skin roughness of the different volunteers.

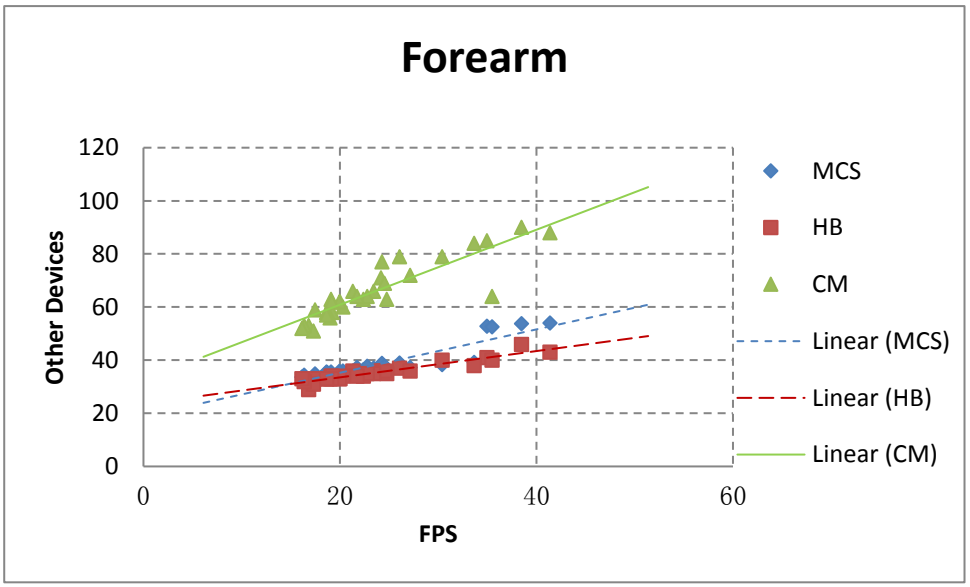


Figure 5.4 Comparison between Fingerprint Sensor and other instruments

Figure 5.4 shows the comparison between Epsilon and other instruments of skin hydration on forearm part (all volunteers' data together). The results show that Fingerprint Sensor has good correlations to all instruments, but has the best correlation with Corneometer, which is the industrial standard for skin hydration measurements.

5.2 Skin Damage Measurements by Multiple Devices

The aim of this experiment is to evaluate the effectiveness of different skin measurement instruments for measuring skin damages caused by intensive washes. In this experiment, two skin sites marked as control site and test site were chosen on the right volar forearm near the elbow area where there is less hair. The test skin site was damaged by intensive washes using Fairy original washing liquid, three minutes each time, and totally three times. Measurements were performed both before and after the washing, every 10 minutes, totally 60 minutes. The results were collected by Fingerprint sensor, AquaFlux and Corneometer. Before every measurement, skin visible pictures were taken by using ProScope HR2 digital microscope.

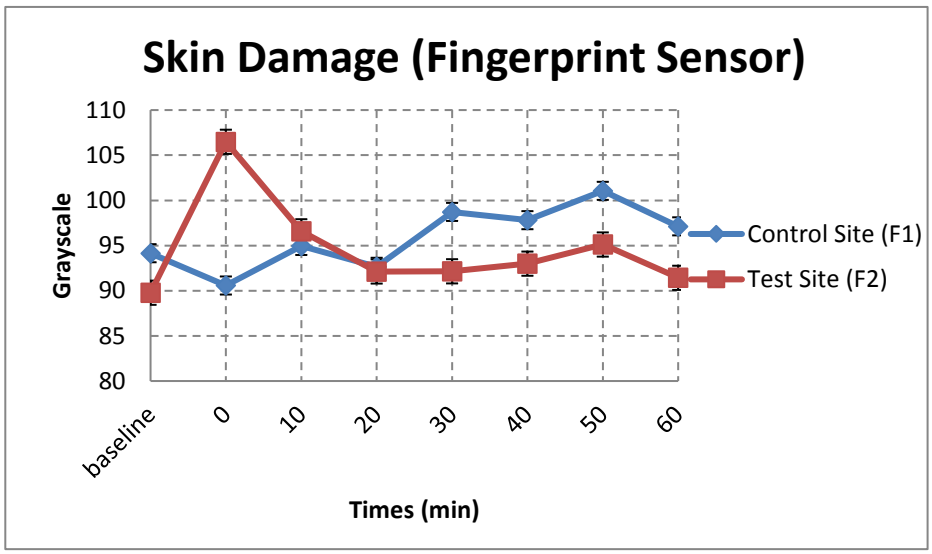


Figure5.5 (a) the results of Fingerprint Sensor for the experiment

Figure 5.5 (a) shows the Fingerprint Sensor measurement results. Baseline is before the washing, 0 is immediately after washing, and 10, 20, 30, 40, 50, and 60 minutes after washing. The error bars in the figure are based on the standard deviation (SD) of Fingerprint sensor, as shown in Figure 5.1, which shows that the changes in the figure, especially the fluctions of the control site, as well as

the late part of the test site, reflects more of the skin changes rather than the instrument errors.

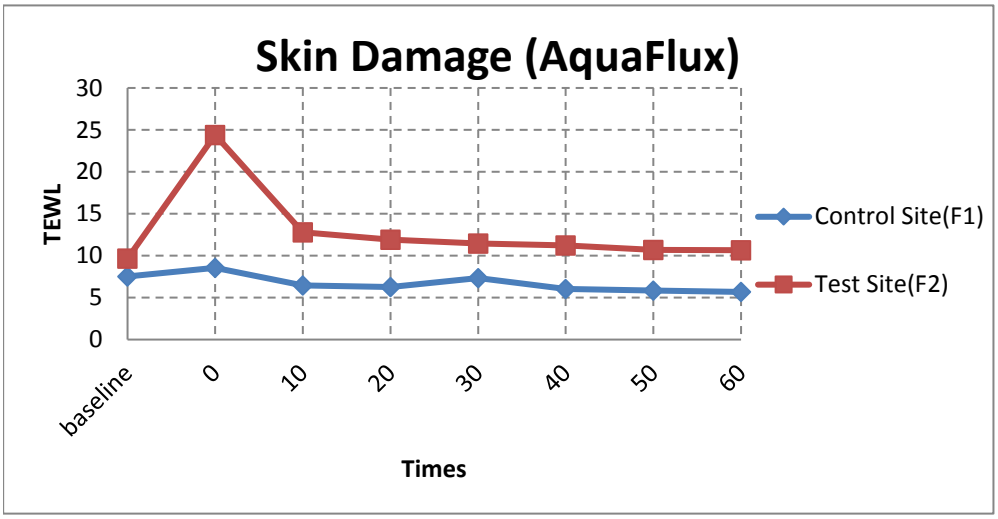


Figure5.5 (b) the results of AquaFlux for the experiment

Figure 5.5 (b) shows the AquaFlux measurement results. The error bars in the figure are based on the standard deviation (SD) of AquaFlux (Imhof R. E., et al, 2005). The results show that the control site remains relatively unchanged during the measurements, while the test skin site shows a huge hydration increase and TEWL increase immediately after washing. Then the test skin site recovers, they gradually return to normal level. It is interesting to point out that even after 60 minutes, test skin site is still not fully recovered, its hydration level is lower than the control site (Figure 5.1 (a)) and TEWL value is higher than the control site (Figure 5.1 (b)).

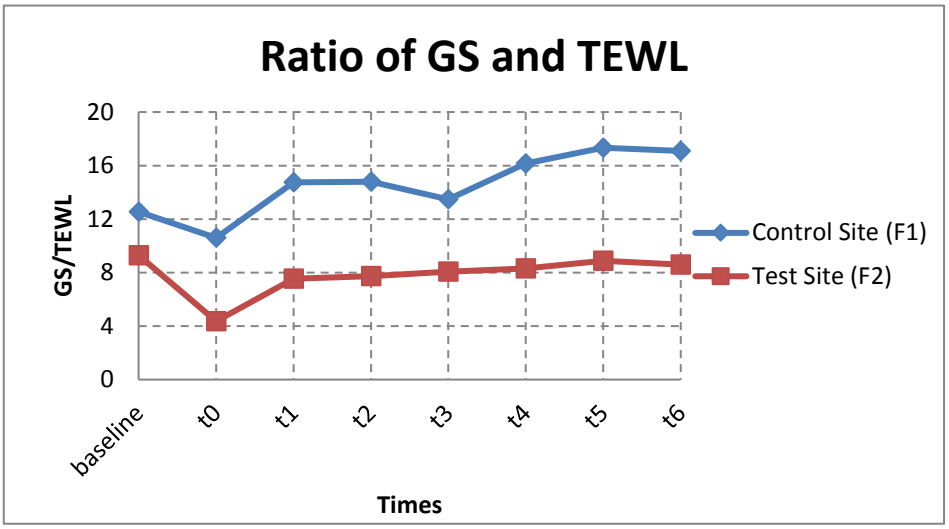
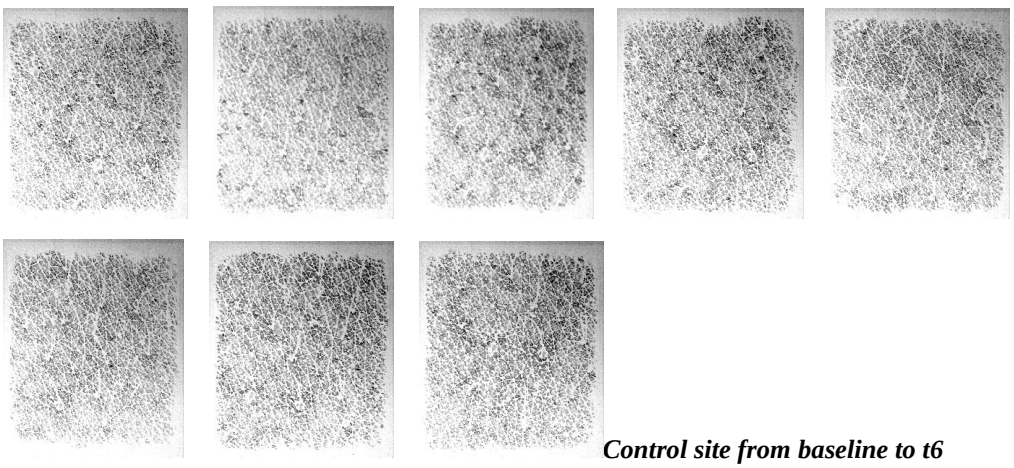


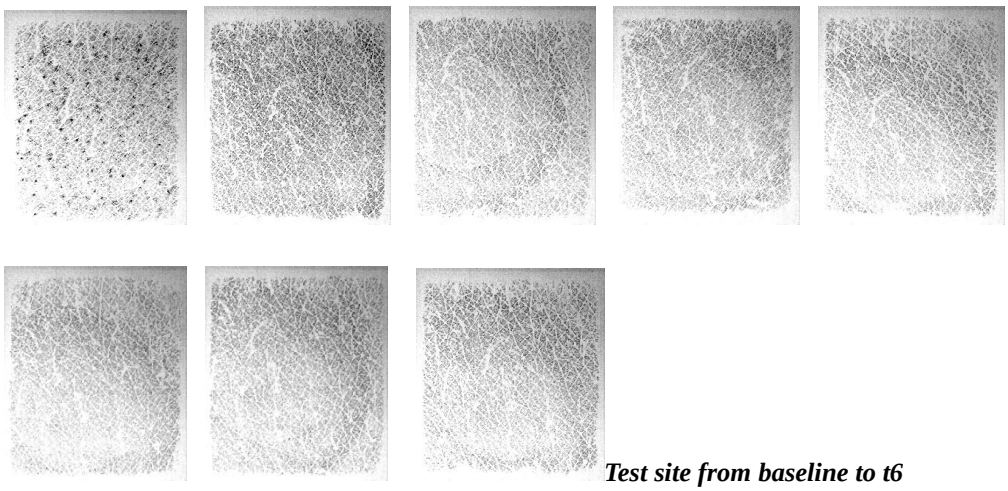
Figure 5.6 the ratio of grayscale and TEWL

By dividing the skin hydration (grayscale) by TEWL, we can get the ratio of hydration (grayscale) and TEWL. Figure 5.6 shows the calculation results of ratio of grayscale and TEWL, the results show that ratio of hydration (grayscale) and TEWL can separate the test skin site and control site better.

Figure 5.7 (a) shows the corresponding Fingerprint Sensor grayscale images. It clearly shows that on the test site, the grayscales reduced step by step. In another word, the hydration in skin is reduced.



Control site from baseline to t6

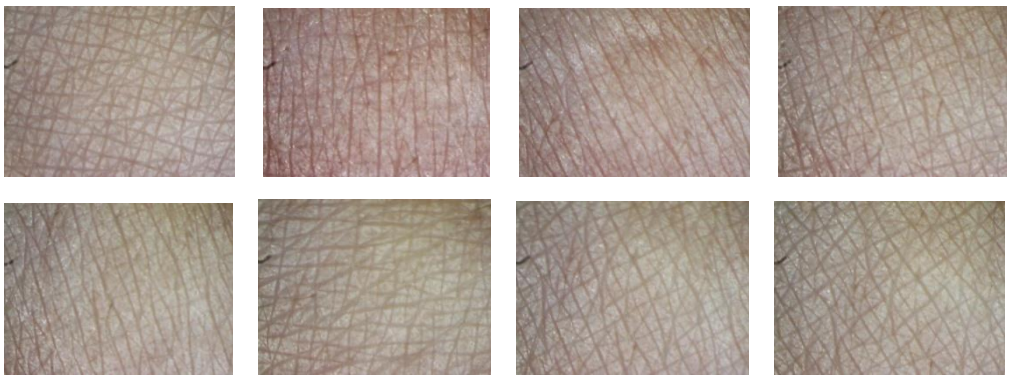


Test site from baseline to t6

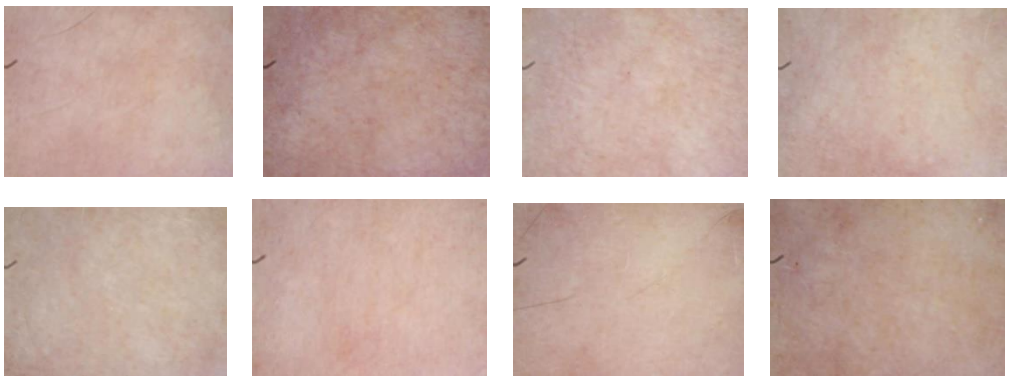
Figure 5.7 (a) R-Arm graphs from 0 to 60 minutes and baseline

Figure 5.7 (b) shows the visible color skin images taken by ProScope HR2 digital microscope. With its unique cross polarized LED lighting skin images, both from the surface and underneath surface are taken. The both skin surface and underneath surface images of the test site show increased redness immediately after washing, and then as skin recovers, the color gradually returns to normal. However, it is difficult to extract consistent values out of the images due to inconsistent lighting.

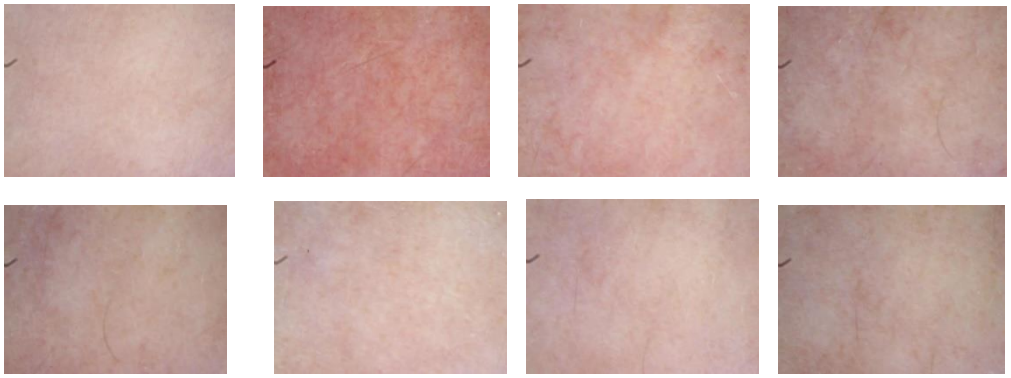
Outside shots of the control site on R-Arm skin from baseline to T6



Outside shots of the test site on R-Arm skin from baseline to T6



Inside shots of the control site on R-Arm skin from baseline to T6



Inside shots of the test site on R-Arm skin from baseline to T6

Figure 5.7 (b) Comparison of color images both from skin surface (outside) and underneath skin (inside) from ProScopeHR2 digital microscope.

Figure 5.8 shows the Corneometer measurement results. The error bar in the

figure is based on the standard deviation (SD) of Corneometer, as shown in Figure 5.1. Again, it shows the measurement value changes are more significant than that of error bars. The results also show the similar trends with the Fingerprint sensor results.

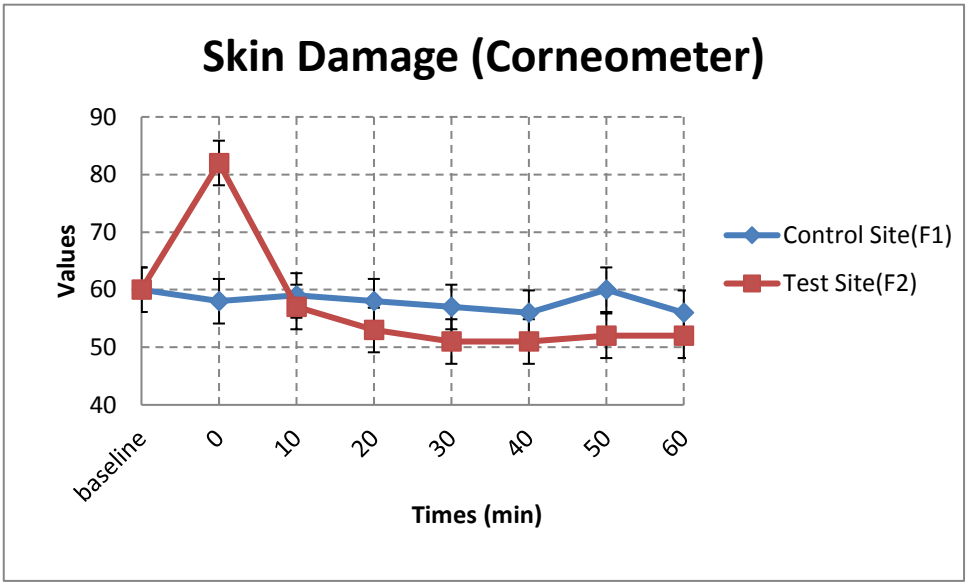


Figure 5.8 Comparison of the data from Corneometer.

In summary, the results show that Fingerprint Sensor, AquaFlux, Corneometer can effective measure the skin damages. Fingerprint Sensor and Corneometer results have similar trends. The ratio of skin hydration (grayscale) and TEWL can differentiate better between the test skin site and control site. The color RGB images can also show the changes during skin damages. But it is difficult to extract consistent values out of the images due to inconsistent lighting.

5.3 Skin Tape Stripping Measurements by Multiple Devices

The aim of this experiment is to evaluate the effectiveness of different skin instruments during skin tape stripping measurements. In this experiment, two skin sites marked as control site and tape stripping test site were chosen on the left volar

forearm. The measurements were performed both before tape stripping and after every two strips. Totally 14 strips were performed. Figure 5.9 shows the Fingerprint Sensor results and Figure 5.10 shows the TEWL results. The error bars in the figure are based on the standard deviation (SD) of Fingerprint sensor (as shown in Figure 5.1), and AquaFlux (Imhof R. E., et al, 2005). It clearly shows that as the tape stripping going one, the control site is more or less the same, but the test site is increased in both hydration and water loss. These increases are slow in the beginning and get really taking off after 12th tape strip.

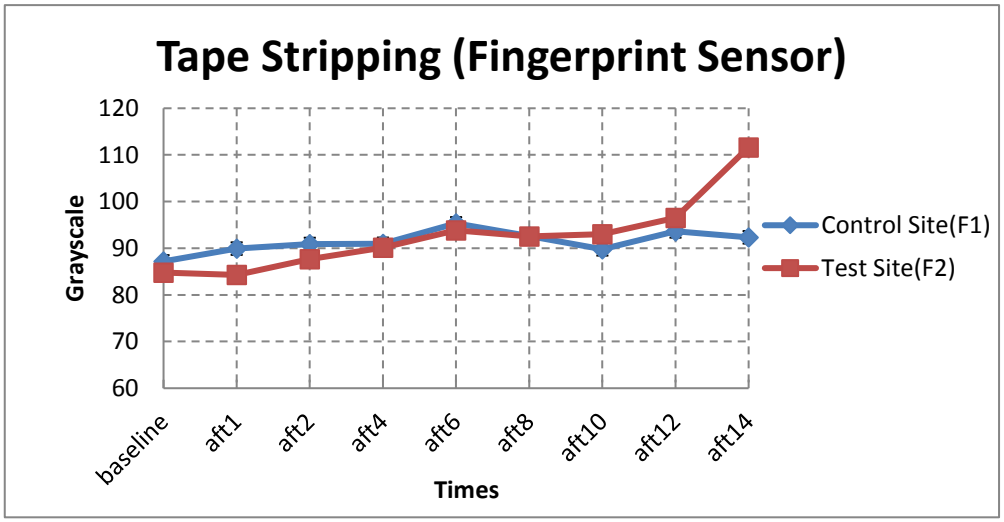


Figure 5.9 Grayscale data at different strips.

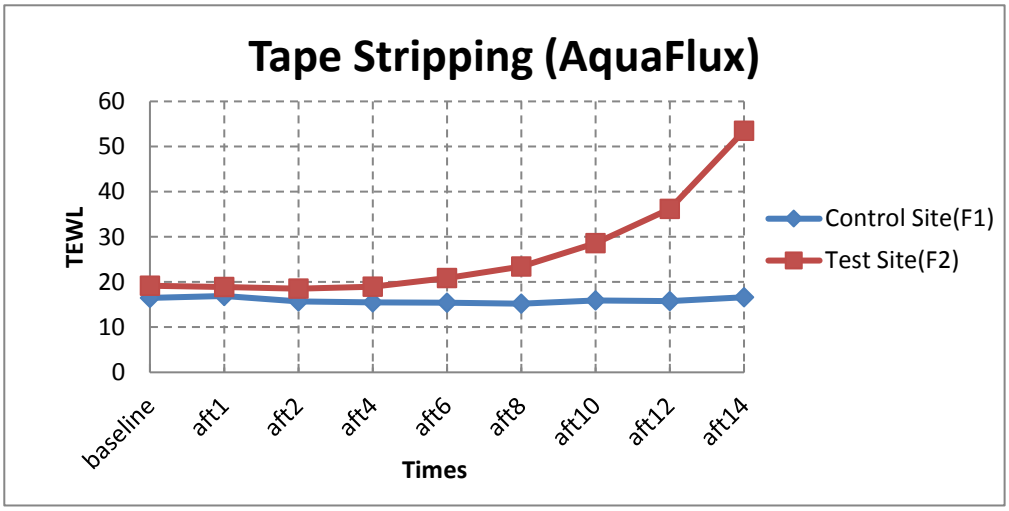


Figure5.10 TEWL data at different strips.

Figure 5.11 shows the corresponding grayscale images of Fingerprint sensor. The control site stays more or less the same, but the test site gets darker as tape stripping goes on. This indicates that skin surface is getting wetter and wetter.

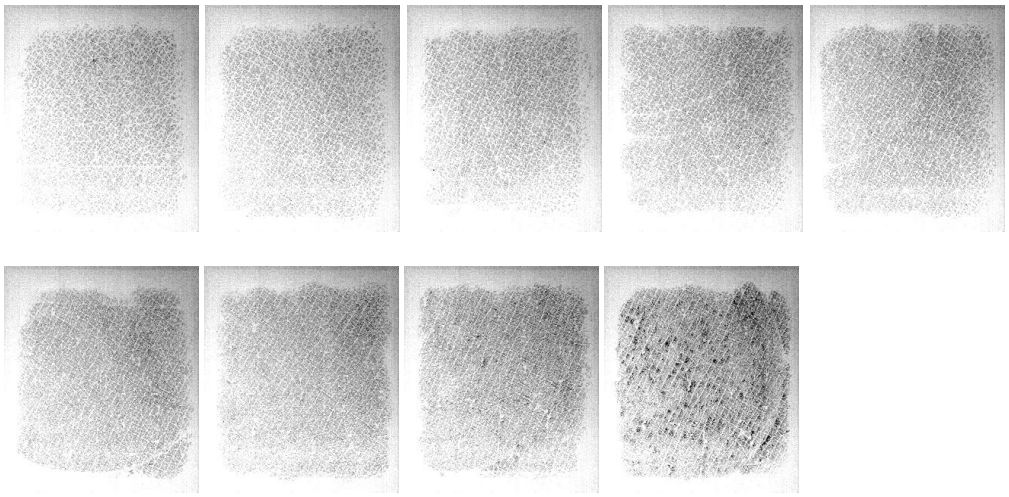


Figure 5.11 (a) Grayscales pictures from baseline to 14 times stripping on stripping site

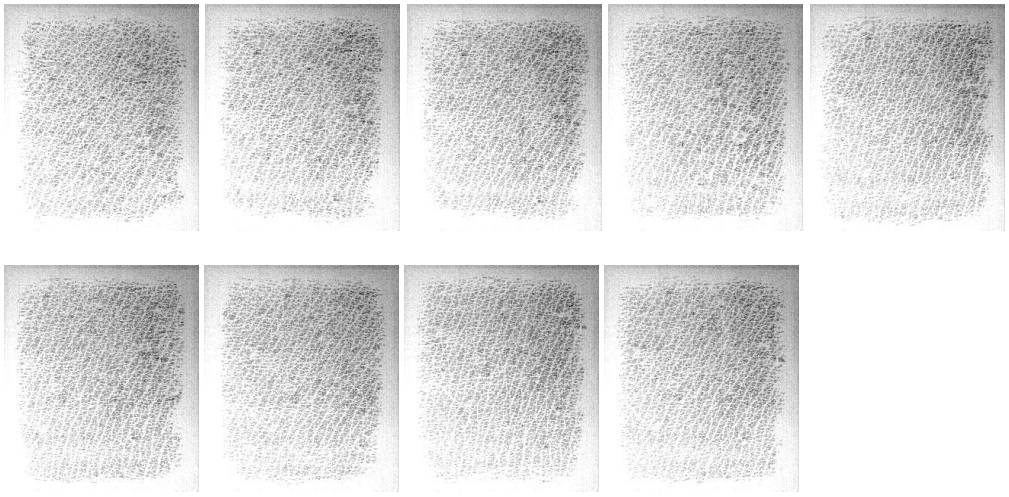


Figure 5.11 (b) Grayscales pictures from baseline to 14 times stripping on control site

Figure 5.11 Grayscales pictures from baseline to 14 times stripping on both sites

Figure 5.12 (a) outside pictures of control site

Figure 5.12 (b) outside picture of stripping site

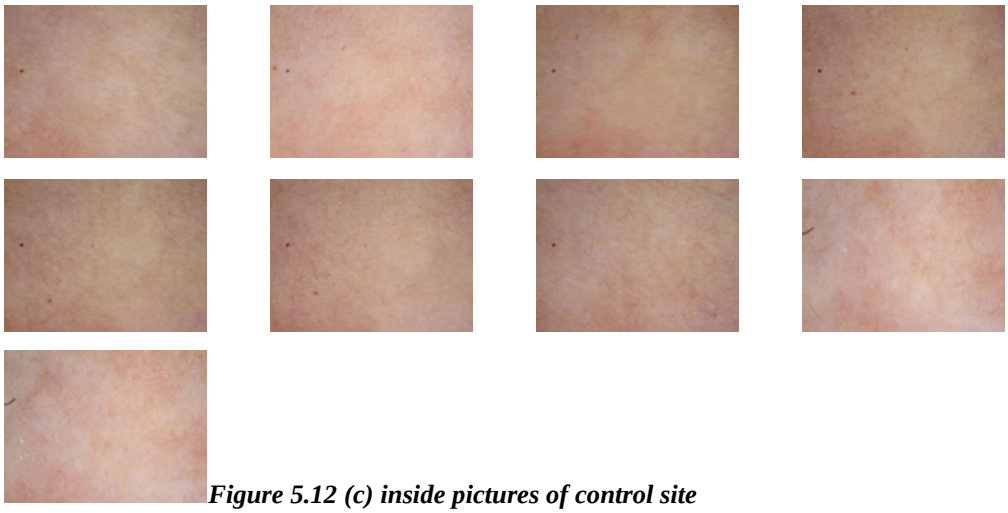


Figure 5.12 (c) inside pictures of control site

Figure 5.12 (d) inside pictures of stripping site

Figure5.12 skin pictures of both sites

Figure 5.12 (a) (b) (c) (d) shows the color skin images from ProScope HR2 digital microscope. Comparing with the control site, the stripping site seems becomes redder as tape stripping goes on. Again, it is difficult to extract consistent values out of the images due to inconsistent lighting.

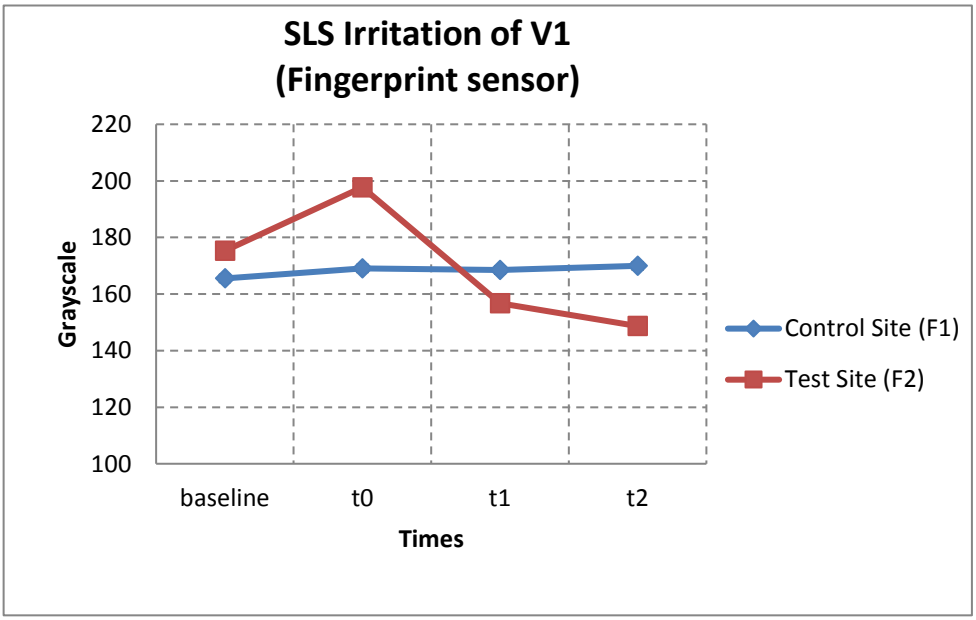
5.4 Skin Irritations by SLS

The aim of this experiment is to evaluate the efficiency of different skin instruments on measuring skin irritation caused by SLS (Sodium Lauryl Sulphate) solution. In this experiment, 2% SLS (Sodium Lauryl Sulphate) solution was created by mixing 1g SLS powder and 49ml water two volunteers are chosen (male, age 25-35). F1 is denoted as control site and F2 is denoted as test site. SLS irritation is done by using cotton buds to keep washing F2 site for a 1-minute period, three times on two volunteers.

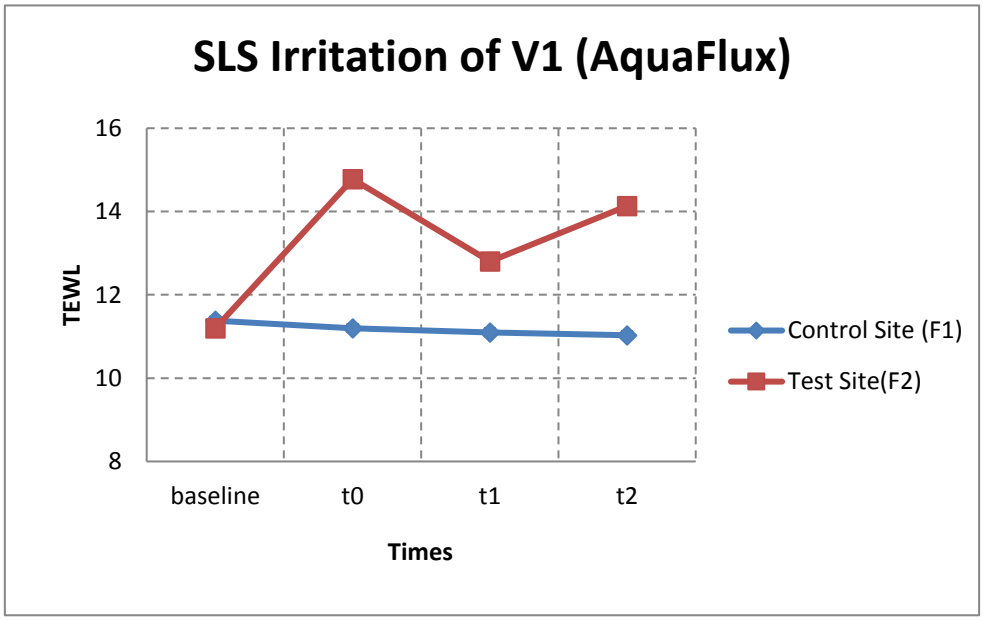
The data measured which are Baseline, T0 (immediately after washes), T1 (30min after washes), T2(wash again and 30min after washes). The experiment was then repeated for another two days, Day-2 and Day-3. In the following two days, F2 site only washed once, and measurements are taken at Baseline (before

washes) and T1 (30 minutes after washes).

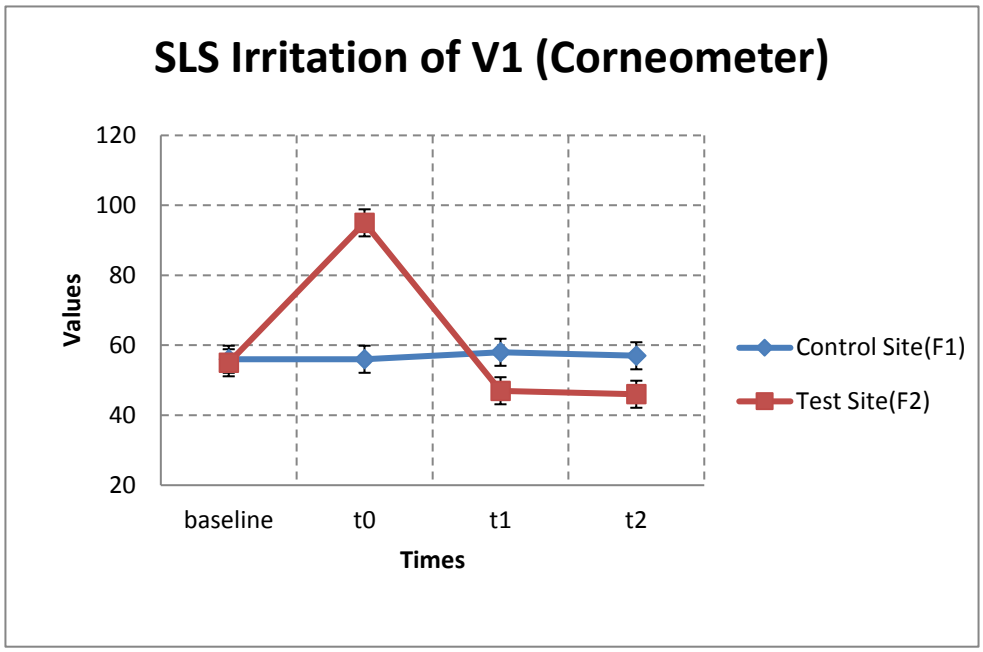
Figure 5.13 shows the measurement results on Day-1. The error bar in the figure, almost invisible, is based on the standard deviation (SD) of Fingerprint sensor and Corneometer, (as shown Figure 5.1), and AquaFlux (Imhof R. E., et al, 2005. The results show that the control site has almost the same value of both volunteers in three instruments, and for Fingerprint Sensor and Corneometer, the test site has a high value when immediately after the wash because of water, and then decreases afterward. It is interesting to point out there is an undershoot at time T2 for both Fingerprint Sensor and Corneometer, which indicate there is a significant drying effect due to SLS irritation. As for AquaFlux, the TEWL result at test site also increased immediately after wash, and then decreased afterward. It has a slightly higher value than the control site, which suggests skin is still damaged and not fully recovered.



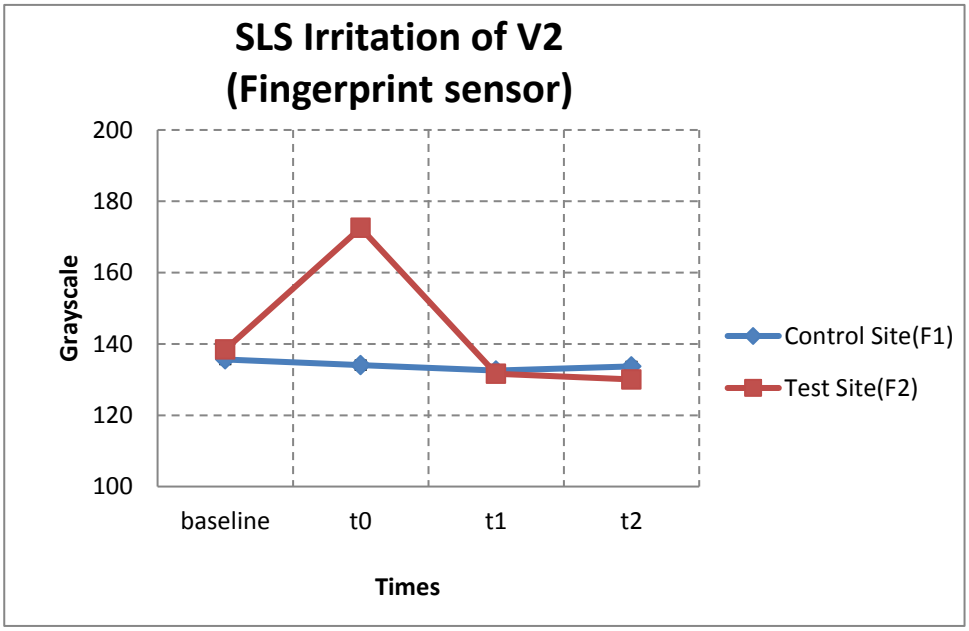
Grayscales data of V1's right forearm



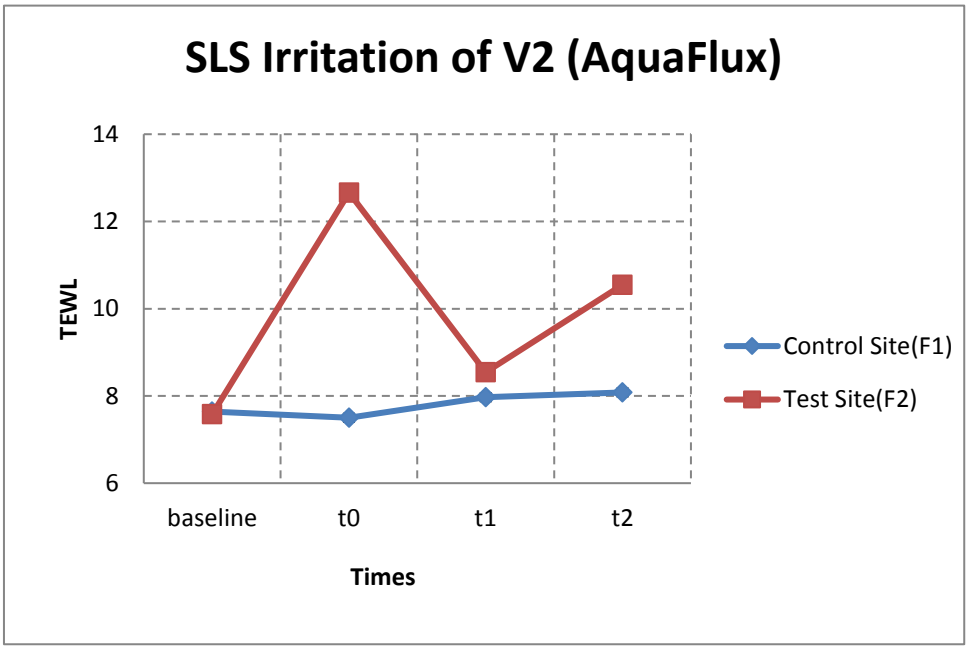
TEWL data of V1's right forearm



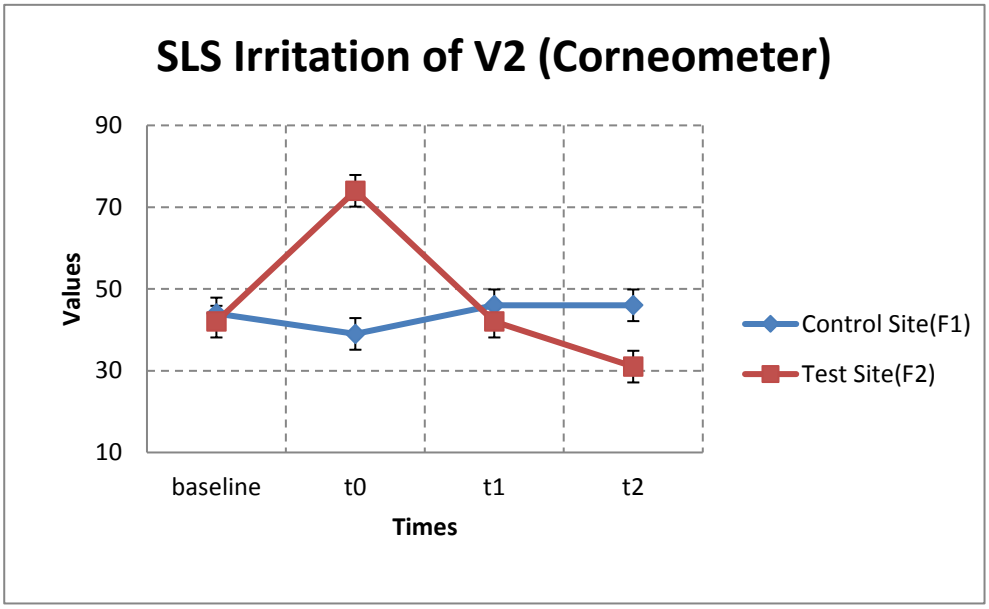
Corneometer data of V1's right forearm



Grayscales data of V2's right forearm



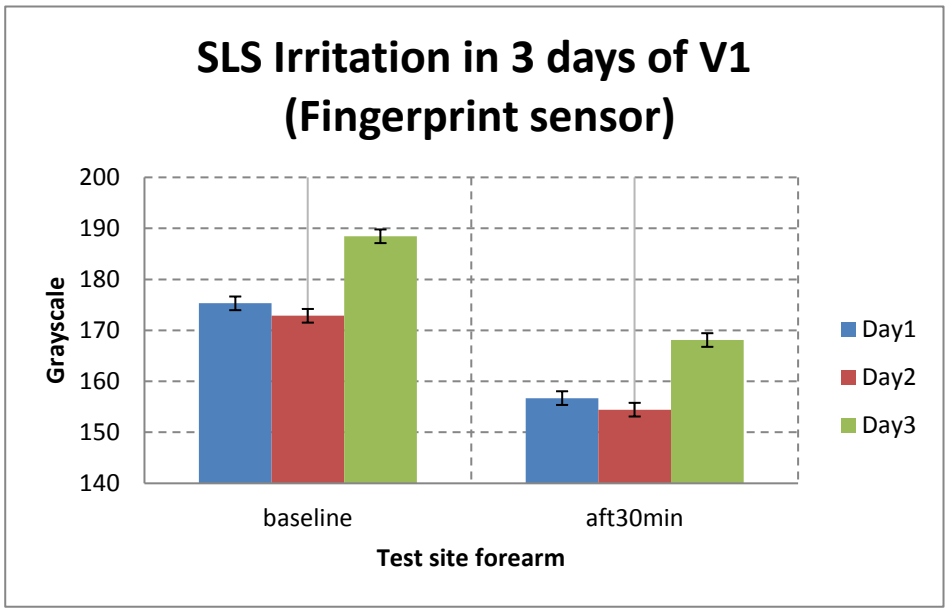
TEWL data of V2's right forearm



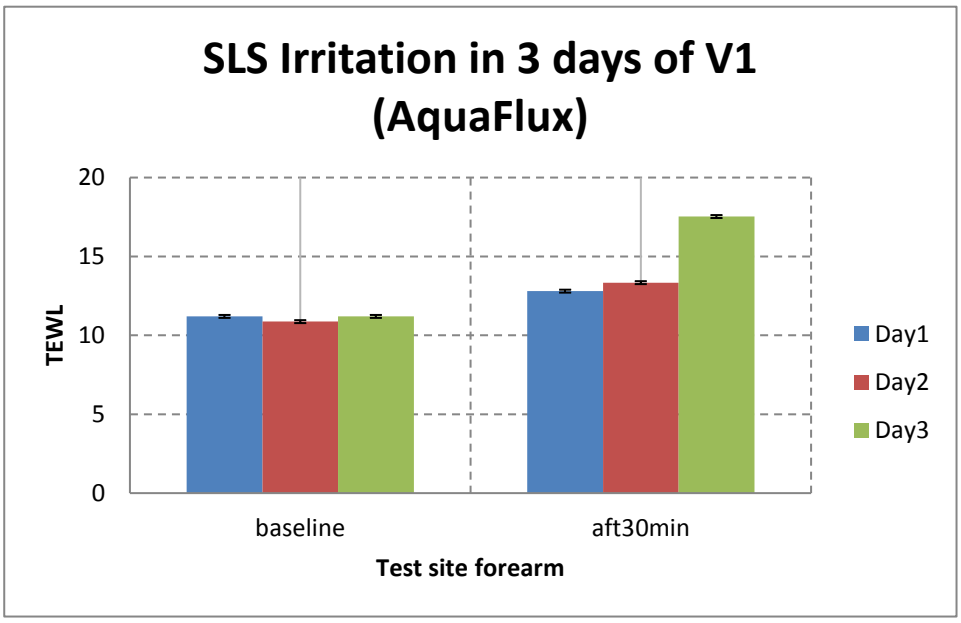
Corneometer data of V2's right forearm

Figure5.13 data of both two volunteers in 1 day

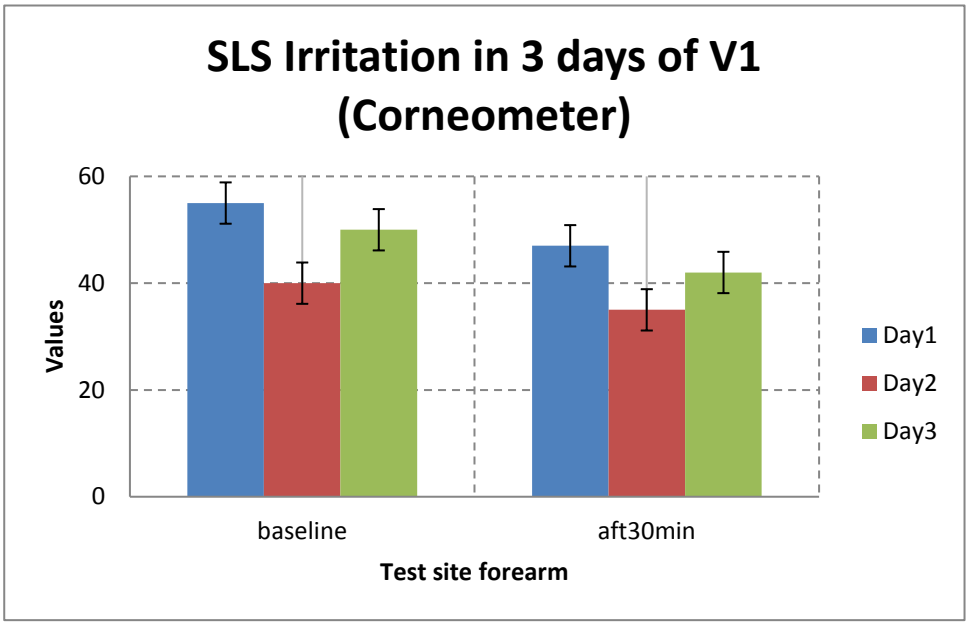
Figure 5.14 shows the three days data of the measurements from baseline to after 30mins. The error bar depends on the standard deviation (SD) of Fingerprint sensor and Corneometer (as shown in Figure 5.1), and AquaFlux (Imhof R. E., et al, 2005). The results are very interesting, as both Fingerprint Sensor and Corneometer failed to show cumulative effects, even though the test skin sites have been washed for three consecutive days. While AquaFlux results show a different story, there is a consistent day on day increase. This suggests TEWL results are more sensitive to skin SLS irritations than hydration results.



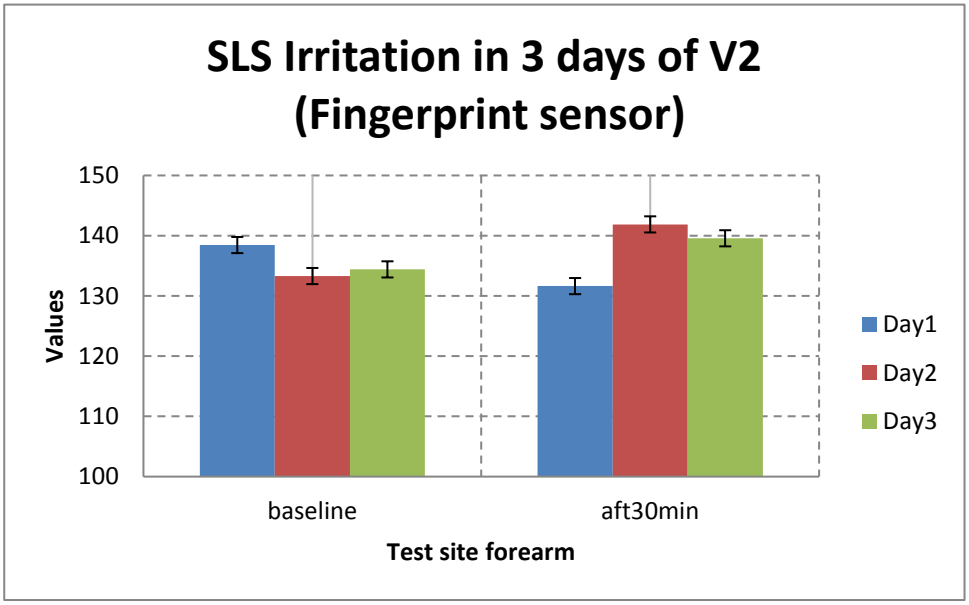
Grayscales data of V1's right forearm



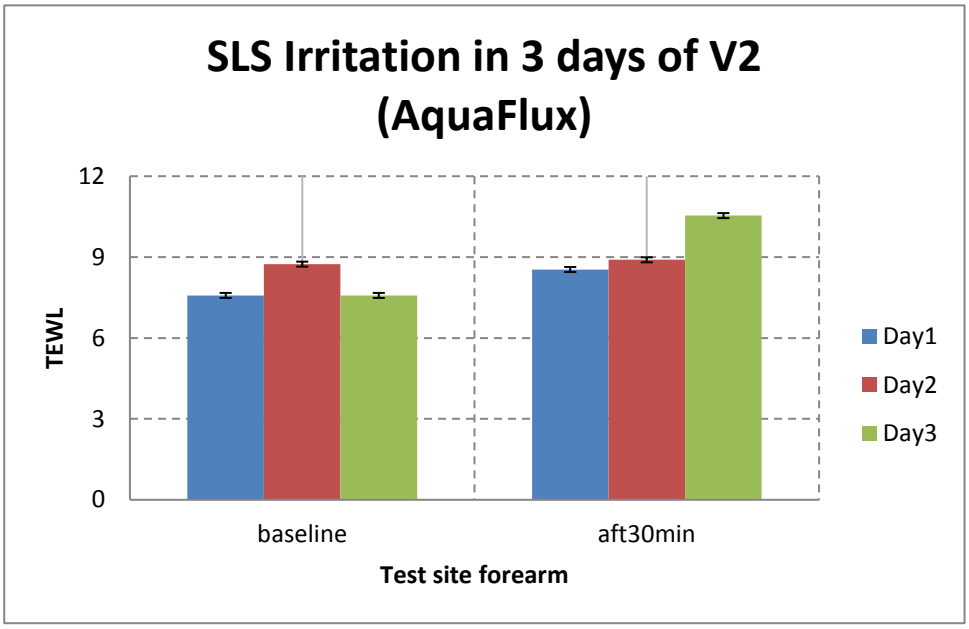
TEWL data of V1's right forearm



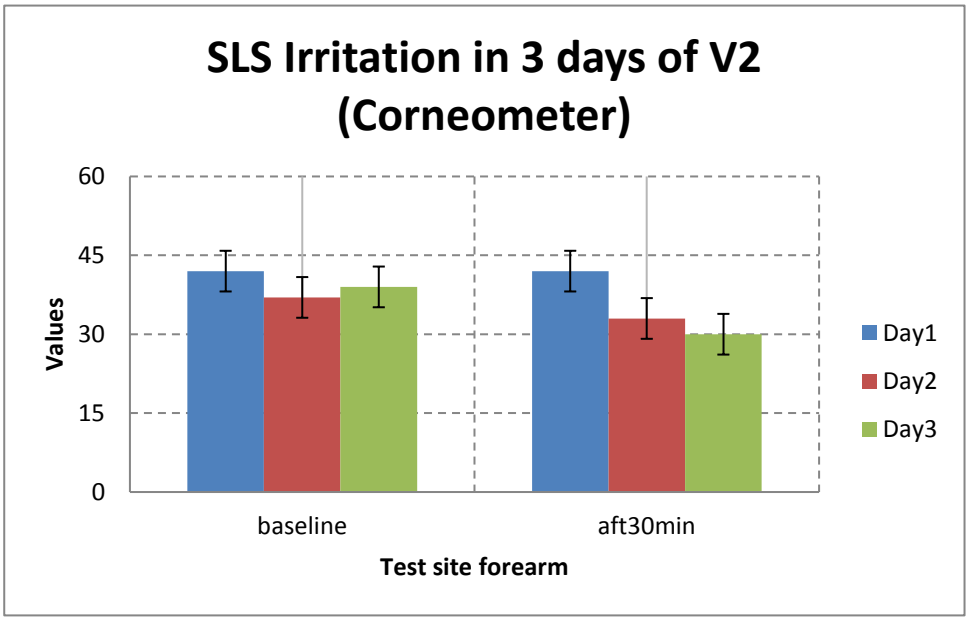
Corneometer data of V1's right forearm



Grayscales data of V2's right forearm



TEWL data of V2's right forearm



Corneometer data of V2's right forearm

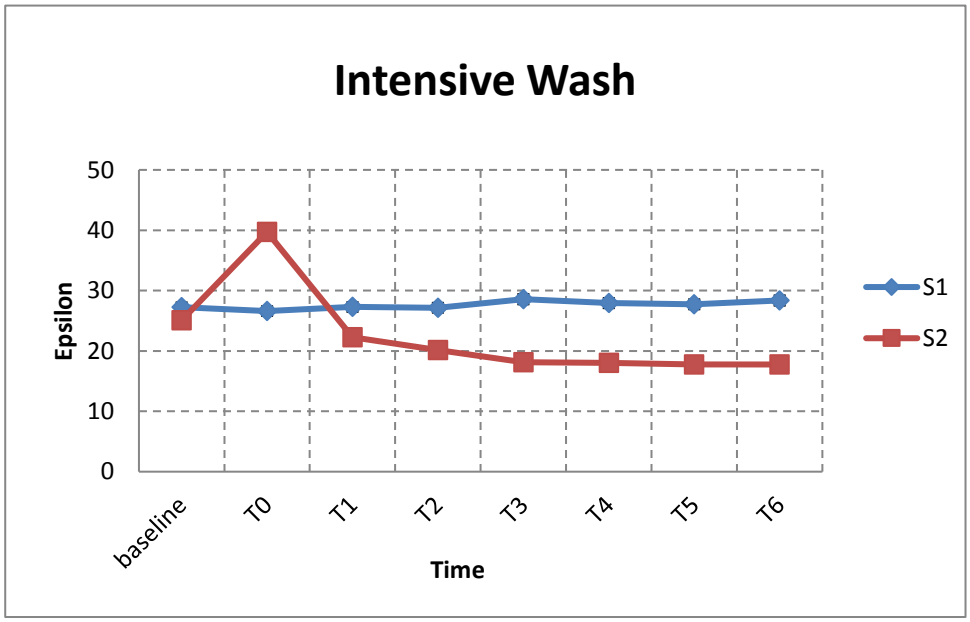
Figure5.14 data of both two volunteers in 3 days

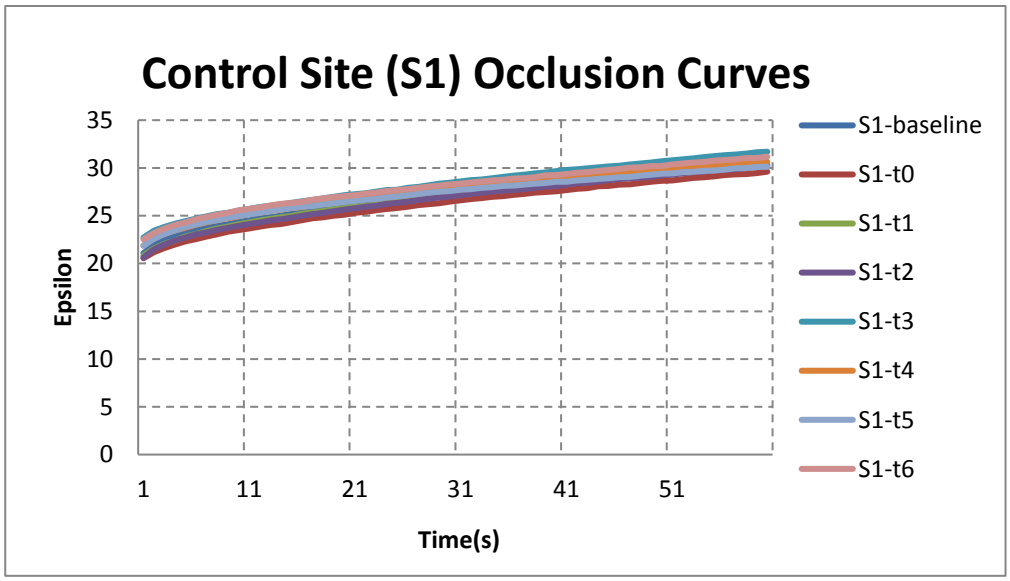
5.5 Capacitive Imaging Occlusion Effects

The aim of this experiment is to study the occlusion effect of new Epsilon permittivity imaging system, which is fully calibrated skin dielectric constant measurement system, this different from the original uncalibrated Fingerprint Sensor. The occlusion is achieved by occluding the skin test site with the sensor surface for a period of one minute, and recording the image average dielectric constant values periodically (every second). By plotting the image average dielectric constant values against time, we can get a time dependent Epsilon occlusion curves. Three skin conditions were studies, i.e. skin intensive wash, skin SLS irritation, and skin tape stripping.

In this experiment, control skin sites (S1) and test skin sites (S2) were chosen on the volar forearm of a healthy male volunteer (age 30). The skin intensive wash is achieved by washing skin test site using a Fairy original washing liquid for 3 minutes. This skin SLS irritation is achieved by applying SLS solution on skin test site for a few minutes, and skin tape stripping is achieved by using standard cellular tape to removed skin surface layers. The measurements were taken both before and periodically after the washes.

Figure 5.15 shows the intensive wash results. The error bars in the figure are based on the standard deviation (SD) of Epsilon (advanced version of Fingerprint sensor). The data measured which are Baseline (before wash), T0 (after wash immediately), T1 (after 5 min), T2 (after 10 min), T3 (after 15 min), T4 (after 20 min), T5 (after 25 min) T6 (after 30 min) for both S1 and S2 sites, and then, compare the results.


Intensive Wash
Epsilon
50
40
30
20
10
0
baseline
T0
T1
T2
T3
T4
T5
T6
Time
S1
S2


Control Site (S1) Occlusion Curves
Epsilon
35
30
25
20
15
10
5
0
1
11
21
31
41
51
Time(s)
S1-baseline
S1-t0
S1-t1
S1-t2
S1-t3
S1-t4
S1-t5
S1-t6

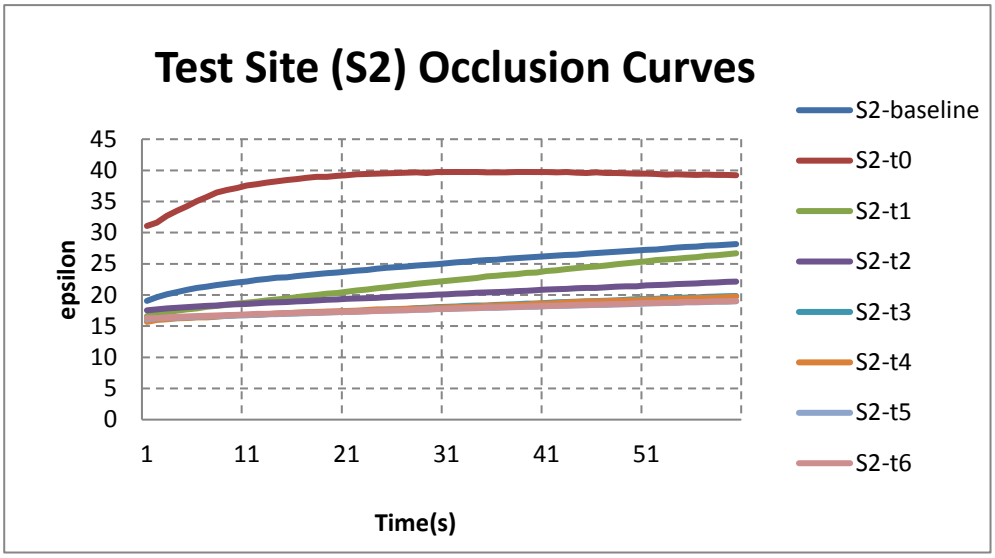
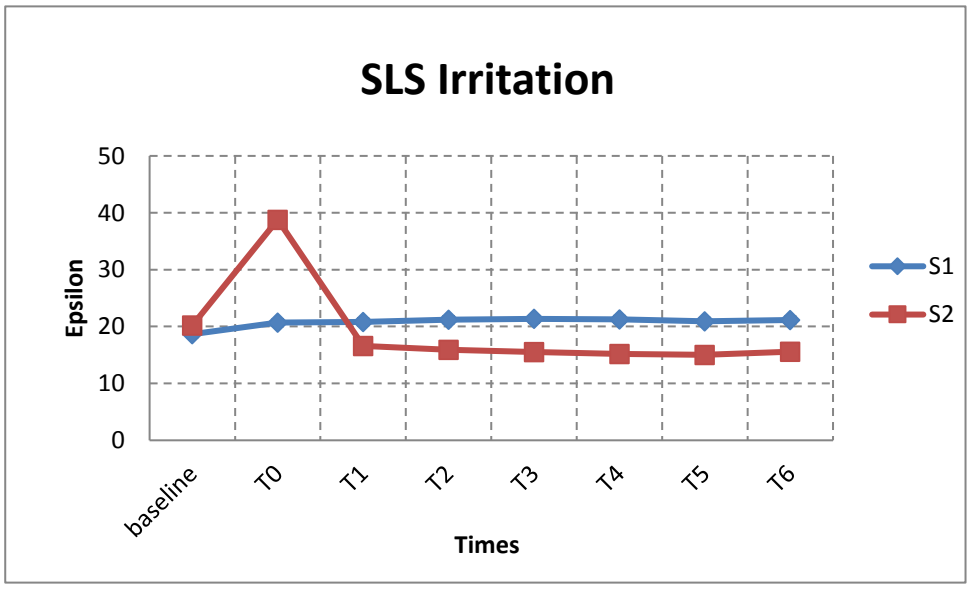
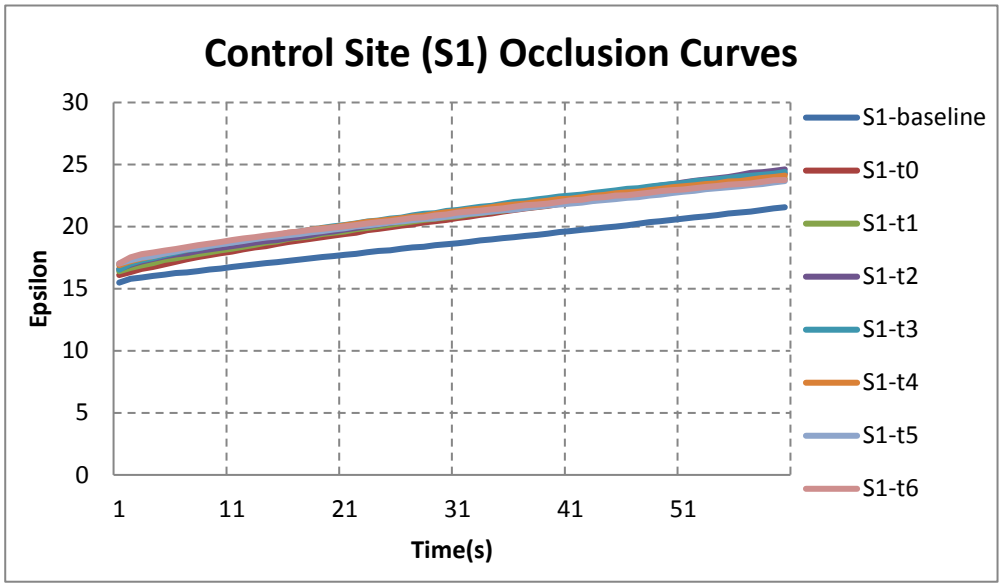


Figure 5.15 Epsilon values of intensive washes measurements, and corresponding S1 occlusion curves, as well as S2 occlusion curves.

Figure 5.16 shows the SLS irritation results. The data measured which are Baseline (before wash), T0 (after wash immediately), T1 (after 5 min), T2 (after 10 min), T3 (after 15 min), T4 (after 20 min), T5 (after 25 min) T6 (after 30 min) for both S1 and S2 sites, and then, compare the results.





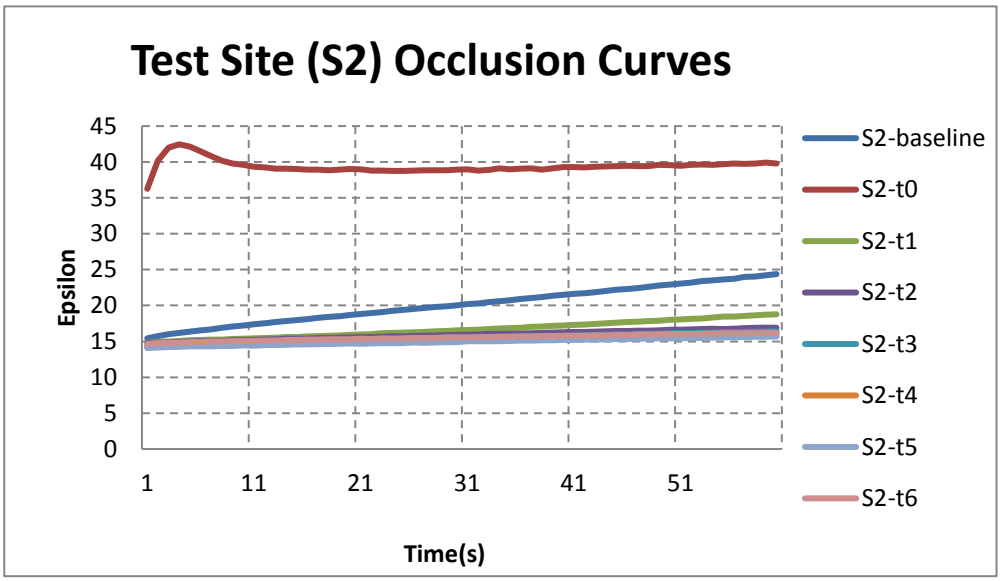
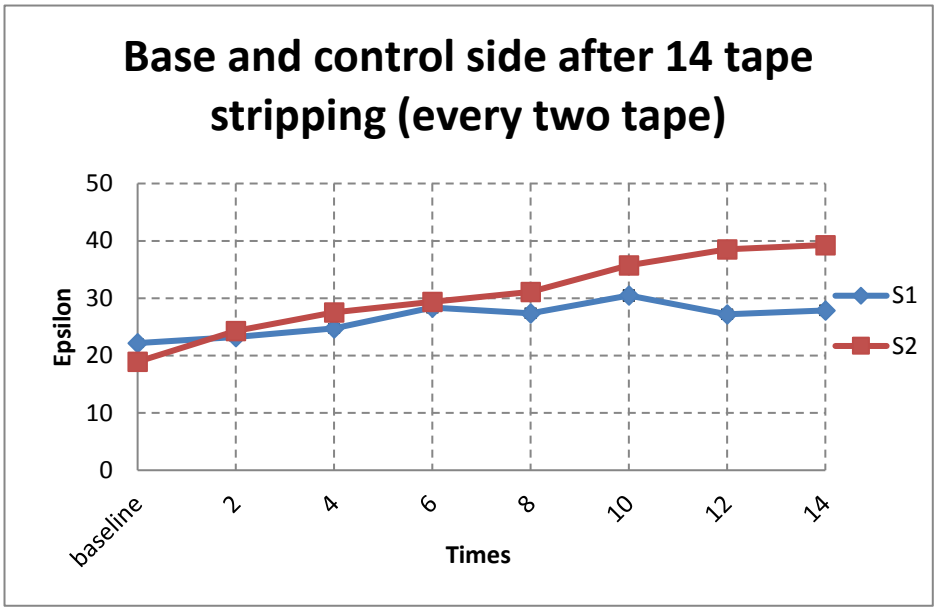
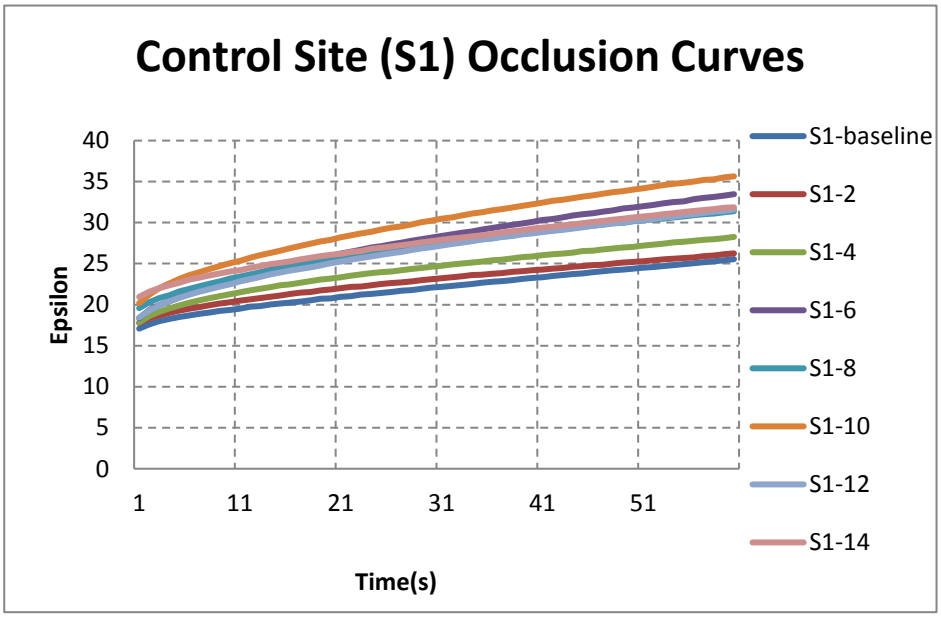


Figure 5.16 Epsilon values of SLS irritation measurements, and corresponding S1 occlusion curves, as well as S2 occlusion curves.

Figure 5.17 shows the tape stripping results. The data we measured which are baseline (before stripping), t2, t4, t6, t8, t10, t12, t14 for S2 every 2 time stripping and et2, t4, t6, t8, t10, t12, t14 for S1 equal to S2 every 2 time

measurement.





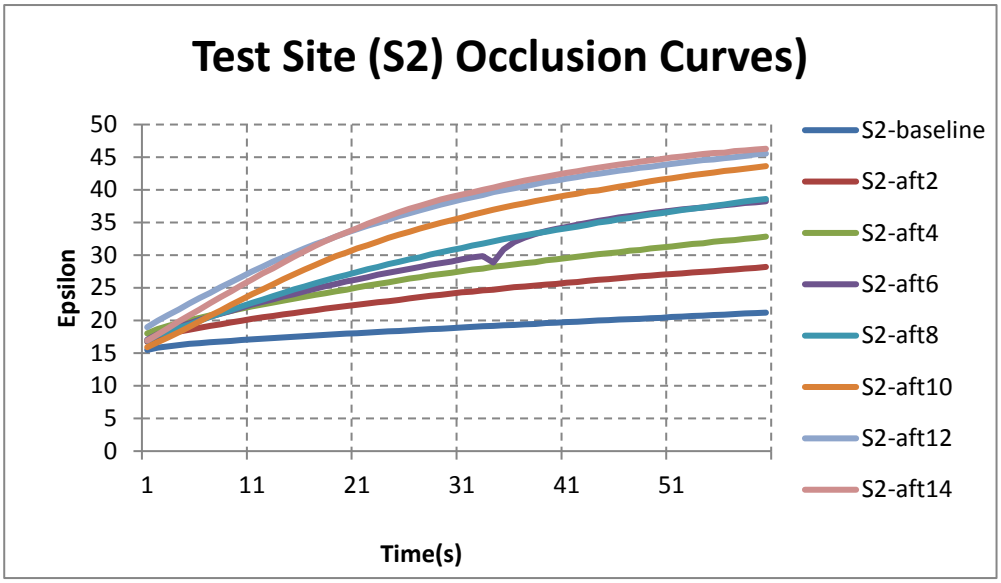


Figure 5.17 Epsilon values of Tape Stripping measurements, and corresponding S1 occlusion curves, as well as S2 occlusion curves.

By comparing the Epsilon occlusion curves, it is interesting to point out that the control sites stays more or less the same, whilst the occlusion curves of the test sites change significantly, not just the levels, but also the shapes. It shows that different types of skin damages have completely different occlusion curves, therefore, at least in theory, it is possible to use occlusion curves to assess the skin damages, as it is capable of not just assessing the scale of damages, but also the types of damages. This makes Epsilon a potential powerful tool for skin damage assessments.

Figure 5.18 shows the corresponding Epsilon skin images for both intensive washes and SLS irritation, the images immediately after in both two methods become very bright due to the high level hydration and then gradually become darker and darker which means the skin becomes dryer and dryer due to the drying effect of both two methods.

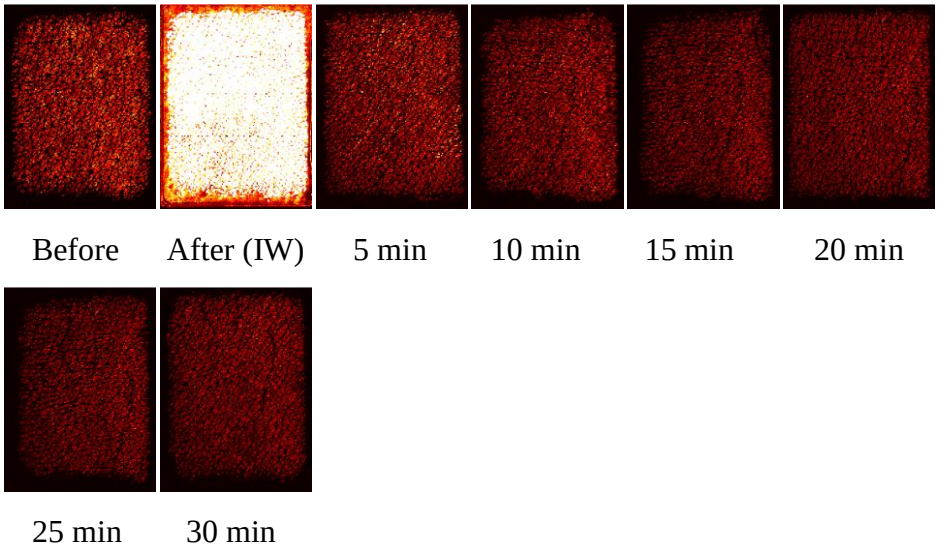


Figure 5.18Capacitive contact images of volar forearm test sites during intensive washes (up and) SLS (Sodium Lauryl Sulfate) irritation (bottom)

Figure 5.19 shows the corresponding Epsilon images of skin tape stripping, the test sites' (S2) images become brighter and brighter which means the skin hydration increases gradually.

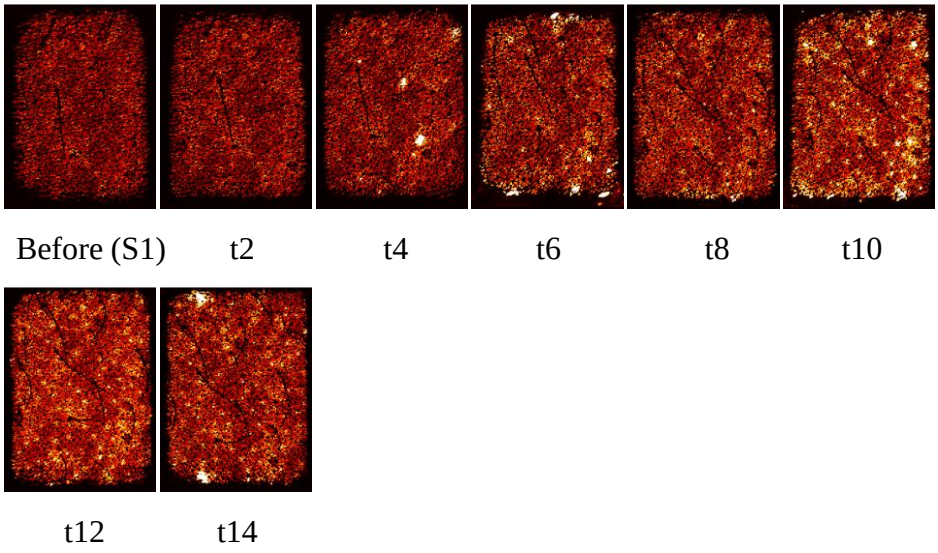


Figure 5.19 Capacitive contact images of volar forearm control site (S1) and test sites (S2) during tape stripping

By assuming measured Epsilon value (ε) is linearly dependent on that of dry skin ($\varepsilon_{dryskin}$=3) and water (ε_{water}=80), we can work out the water content using following equation (Xiao P. and Bontozoglou C., 2015):

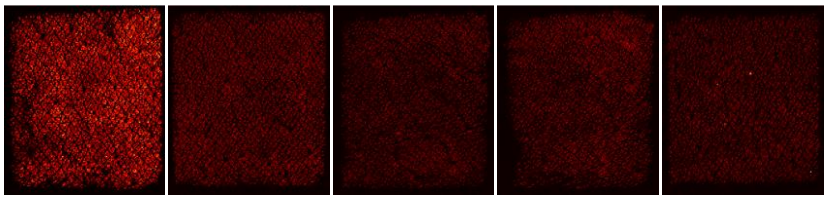
. $$H = (\varepsilon - \varepsilon_{dryskin}) / (\varepsilon_{water} - \varepsilon_{dryskin}) \quad (5.1)$$

For up experiments, it worked out that during tape stripping measurements skin

water content has increased from ~20% to ~47% after 14 tape strips. While in SLS irritation measurements, skin water content has decreased from initial ~22% down to ~16%.

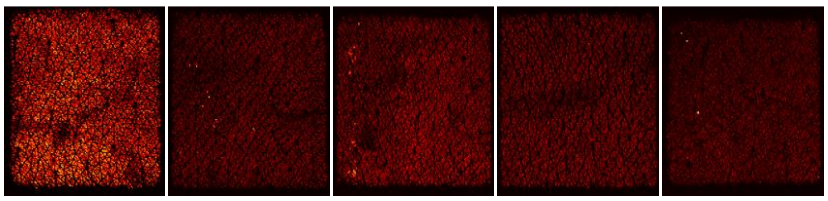
5.6 Effect of Soap Washing

The aim of this experiment is to evaluate the effect of different washing soap by using Epsilon and AquaFlux. The measurement procedure is the same as Chapter 4 section 4.6.3. AquaFlux is used to measure the TWEL, and Epsilon is used for skin hydration. Epsilon is used to measure the skin hydration in 60 seconds.



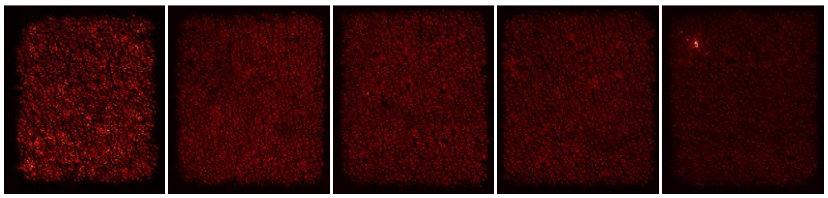
V1- Soap A

Day 1 Day 2 Day 3 Day 4 Day 5



V1 – Soap B

Day 1 Day 2 Day 3 Day 4 Day 5



V1 – Soap C

Day 1 Day 2 Day 3 Day 4 Day 5

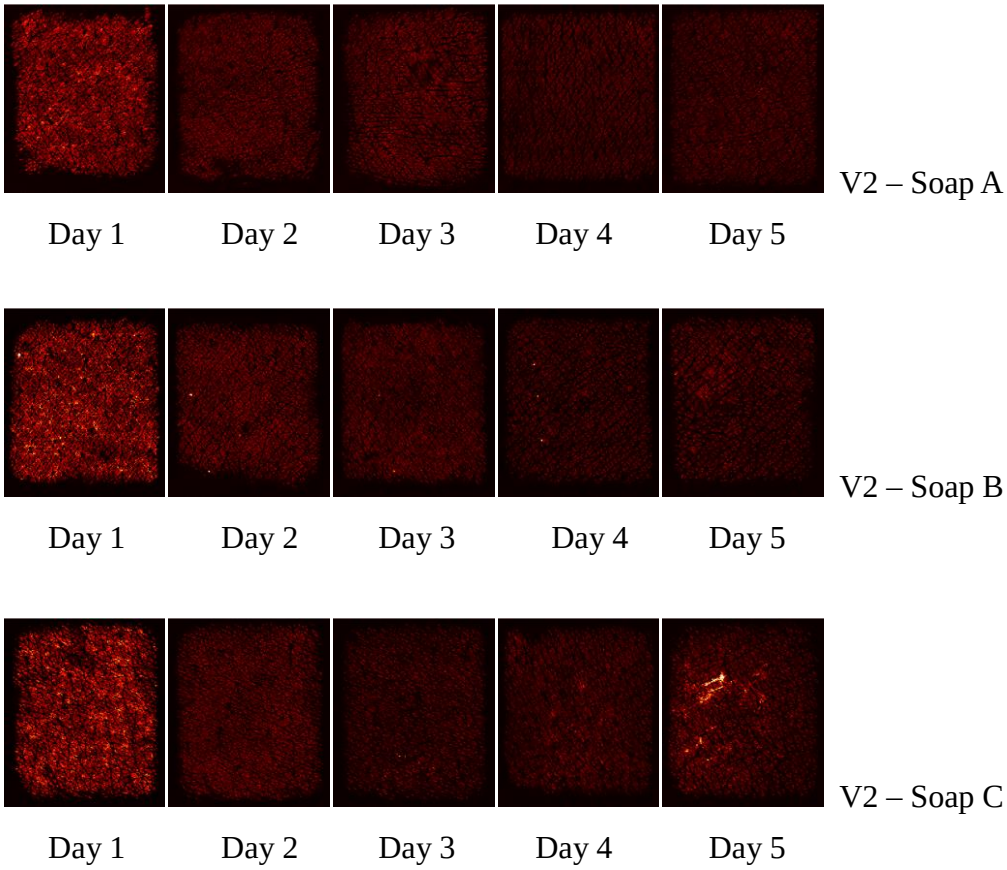
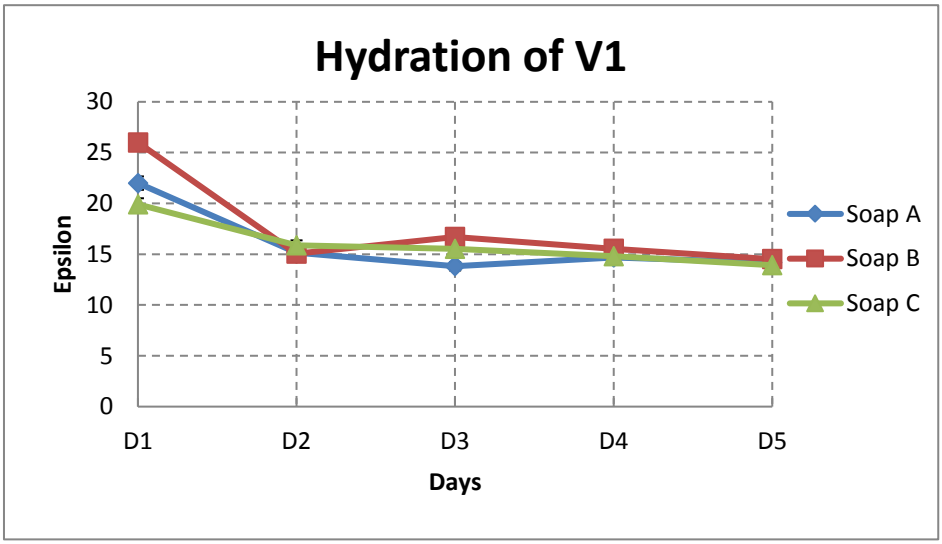


Figure 5.20 Hydration Images of three brands soaps on two volunteers.

Figure 5.20 shows the 5-days images from Epsilon of two volunteers by three different brand soap washing. Day1 is the baseline, before washes. The results show that as washes go on, the skin images are getting darker and darker, indicating the skin is getting dryer and dryer, due to the washes. It is possible to identify the subtle differences of different soap, as Soap C dries the skin most, and Soap B dries the skin least. The slight brighter spot on Soap C site at Day 5 of volunteer two, is because of skin broken down due to Soap C washing.

Figure 5.21 shows the trend of three different brand soap washing on two volunteers. The error bars in the figure are based on the standard deviation (SD) of Epsilon (advanced version of Fingerprint sensor).



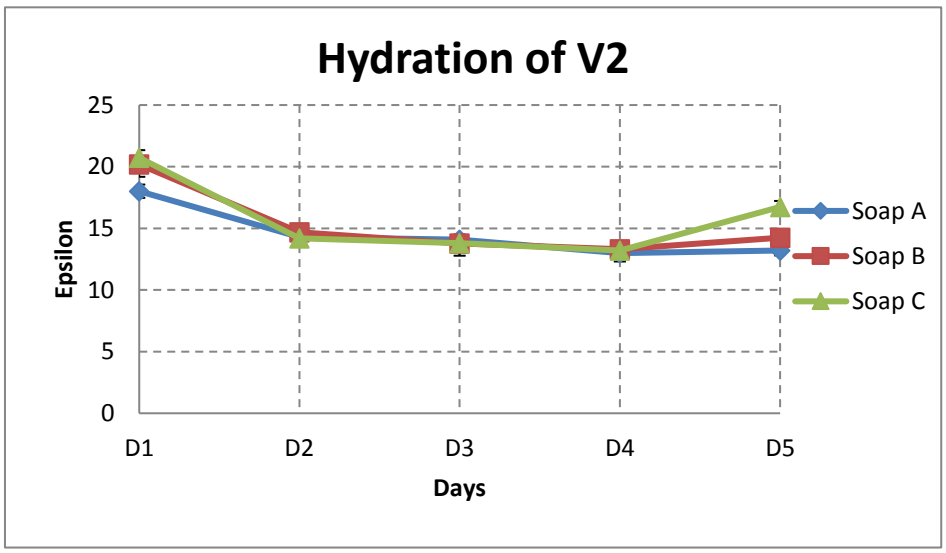
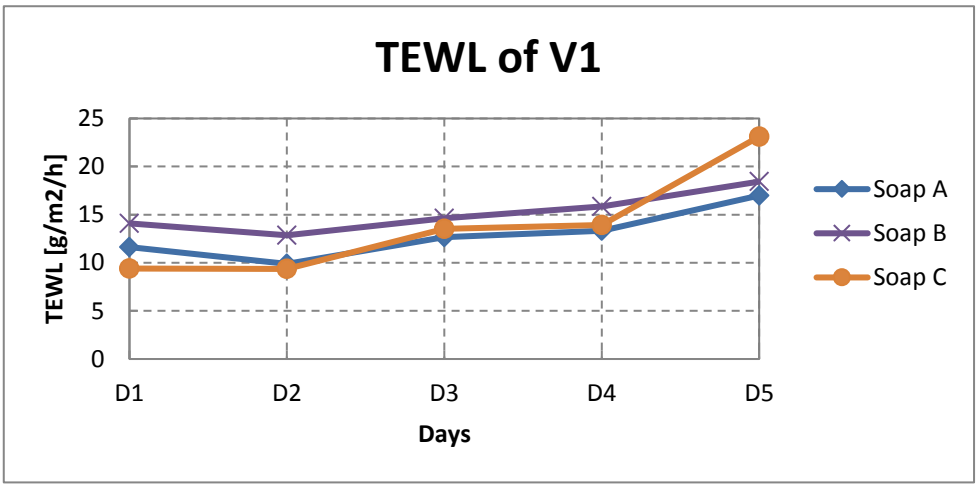


Figure 5.21 Epsilon values of 3 different soaps on 2 volunteers

Figure 5.22 shows the 5-days trans-epidermal water loss (TEWL) from AquaFlux of two volunteers by three different brand soap washing. The error

bars in the figure are based on the standard deviation (SD) of AquaFlux (Imhof R. E., et al, 2005).



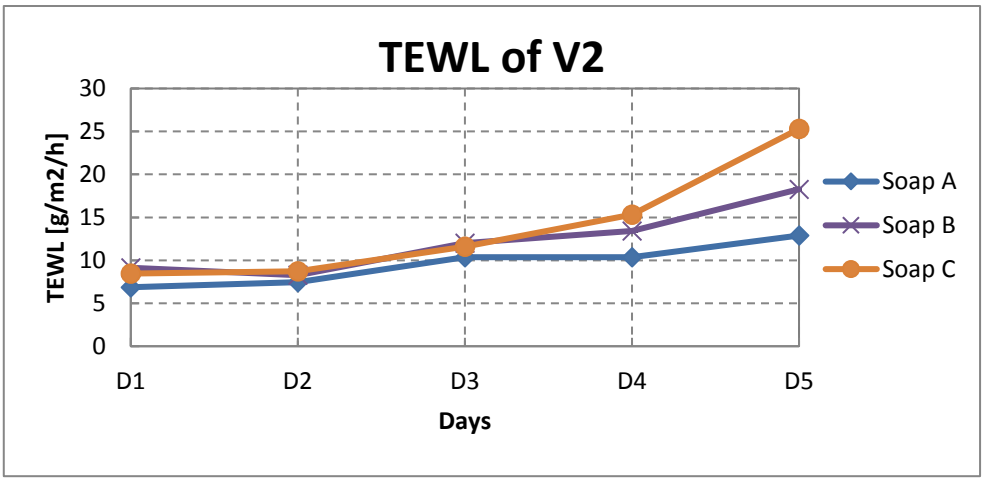


Figure 5.22 TEWL of 3 different soaps on 2 volunteers

From the curve charts, they show that with the soap washing, the TEWL increase because of the drying skin gradually from day 2 to day 5.

Figure 5.23 shows the calculation results of ratio of skin hydration (Epsilon) and TEWL.

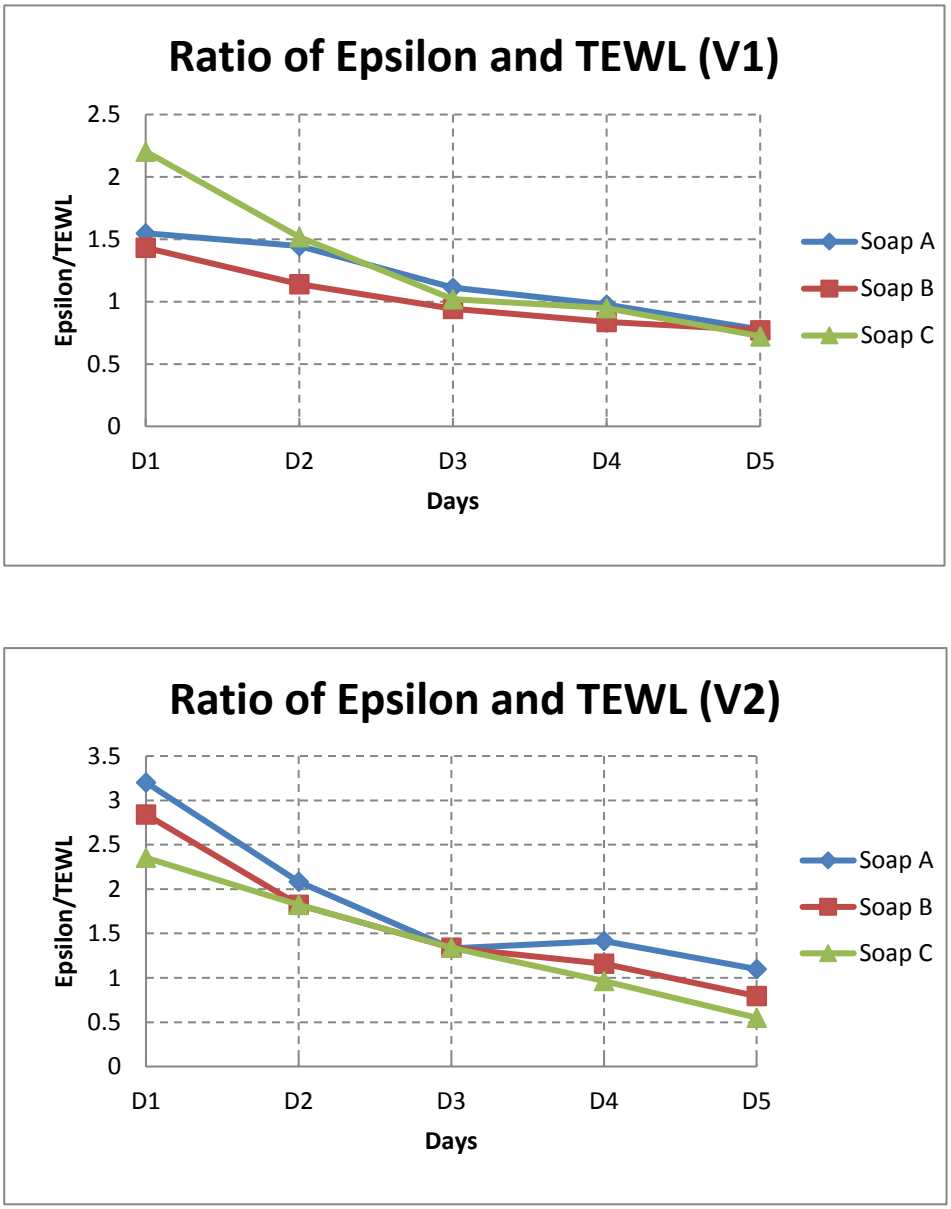


Figure 5.23 ratio of Epsilon and TEWL of 2 volunteers

The results show that ratio of skin hydration (Epsilon) and TEWL combines the Epsilon and TEWL results, and there has more consistent downward trends, indicating skin are getting dryer and dryer.

5.7 Solvent Penetration

5.7.1 *In-vitro* Solvents Penetration

The aim of this experiment is to investigate the performance of Epsilon permittivity system on *in-vitro* solvents penetration measurements. In this experiment, solvent penetration through an *in-vitro* pig skin sample (nearly 1mm thick) was studied. Four solvents (Ethanol, DMSO, ethylene glycol, and glycerol) were used and Epsilon video mode was used to record the penetration results during a period of 60mins.

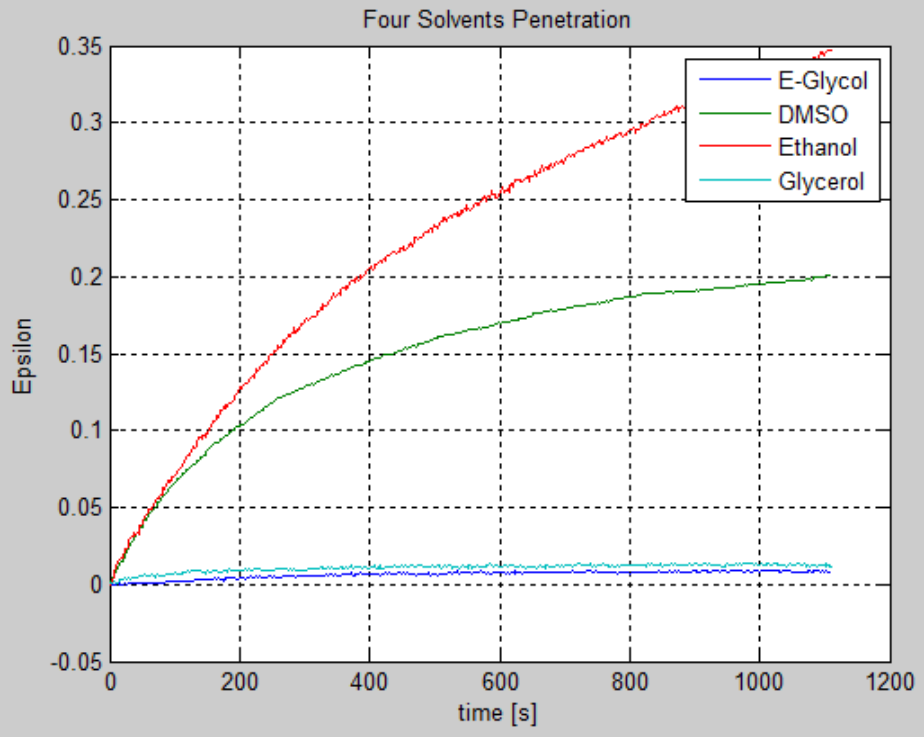
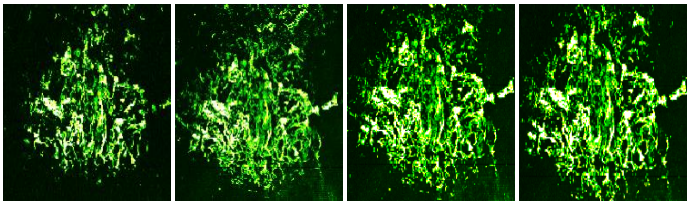


Figure 5.24 Time dependent penetration curves of four different solvents

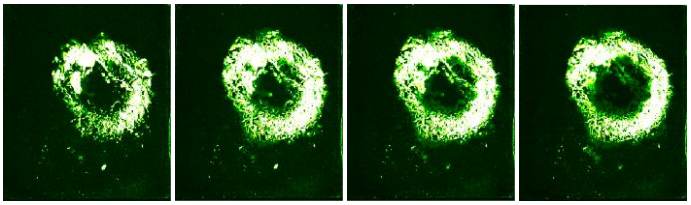
Figure 5.24 shows the time-dependent penetration curves of four different solvents in 20 mins, it shows that alcohol has the best penetration of these four

solvents, the second is DMSO, whilst ethylene glycol and glycerol do not penetrate through the skin very well.

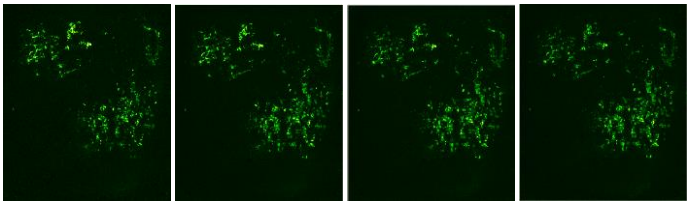
Figure 5.25 shows 4 solvents skin penetration images at 0, 20, 40, 60min. through the skin penetration, the corresponding skin part becomes brighter, means solvent can penetrate into the skin, and oppositely, it means solvent cannot penetrate into the skin.



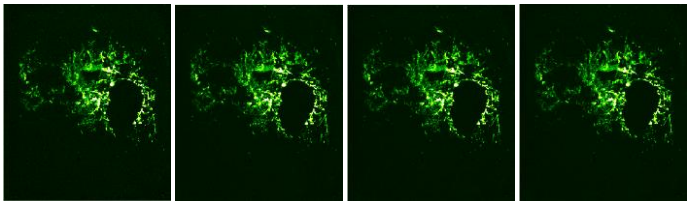
Ethanol: 0min 20min 40min 60min



DMSO: 0min 20min 40min 60min



E-Glycol: 0min 20min 40min 60min



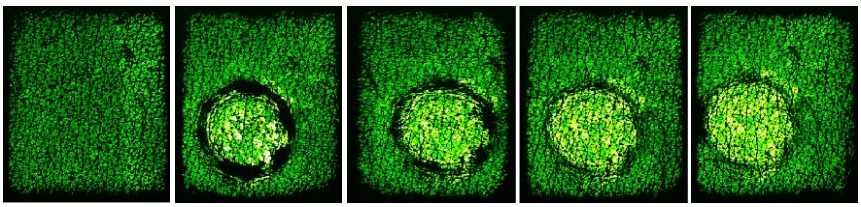
Glycerol: 0min 20min 40min 60min

Figure 5.25 4 solvents skin penetration images at 0, 20, 40, 60 min

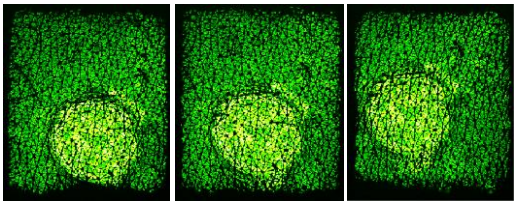
5.7.2 *In-vivo* Skin Solvent Penetration

The aim of this experiment is to evaluate the performance of Epsilon permittivity system on *in-vivo* solvents penetration measurements. In this experiment, three solvents are used, ethanol, glycerol, and ethylene glycol; and three skin sites on left forearm were chosen on one healthy male volunteer (age: 30) for three solvents respectively. For each skin site, a hollow donor chamber was mounted on top, solvents were placed within the chamber for a period of 10 minutes. Then, donor chamber was removed and skin surface was carefully wiped dry. Measurements were taken immediately and periodically after, every 10 minutes, totally for 60 minutes.

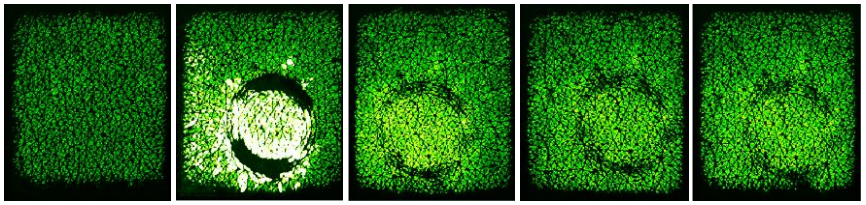
Figure 5.26 shows the Epsilon capacitive images before and after the three solvent applications. The darker rings in the images immediately after the solvent application are imprints from the plastic donor chamber, as a modest pressure was applied to ensure that solvent is not leaking out.



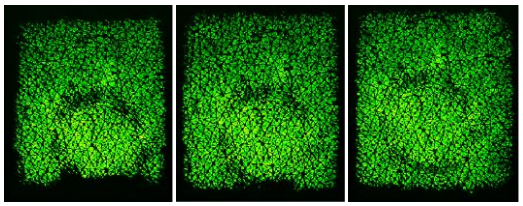
Glycerol base 0 min 10 min 20 min 30 min



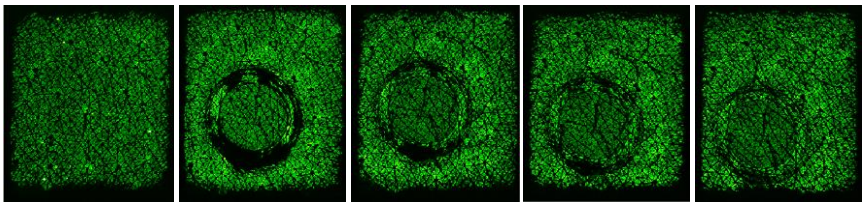
40 min 50 min 60 min



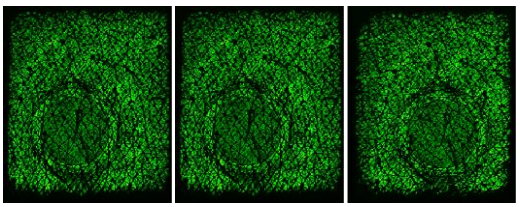
E-glycol base 0 min 10 min 20 min 30 min



40 min 50 min 60 min



Ethanol base 0 min 10 min 20 min 30 min



40 min 50 min 60 min

Figure.5.26 Capacitive contact images of skin site before and after solvent applications of 3 solvents

The image results show that after 10 minutes of solvent application, Ethylene Glycol penetrates most into the skin, while Ethanol penetrates the least. Glycerol is somewhere in the middle. This is probably because that Ethanol is very volatile and evaporated quickly after the application. Ethanol skin site also shows

some drying effect as the images are getting slightly darker. Ethylene Glycol seems to disappear quicker from the skin surface. As Ethylene Glycol is non-volatile, we assume it diffuses into deeper skin. Glycerol, however, tends to stuck within skin for a long period of time. It is interesting to point out that the skin area just outside the darker ring is also brighter than rest of the skin area, especially for Ethylene Glycol, which indicates that whence the solvent has penetrated into skin and it spread out underneath the skin.

By using an equation similar to Eq.(5.1) we can also work the solvent concentration (C).

$$C = (\varepsilon - \varepsilon_{dryskin}) / (\varepsilon_{solvent} - \varepsilon_{dryskin}) \quad (5.2)$$

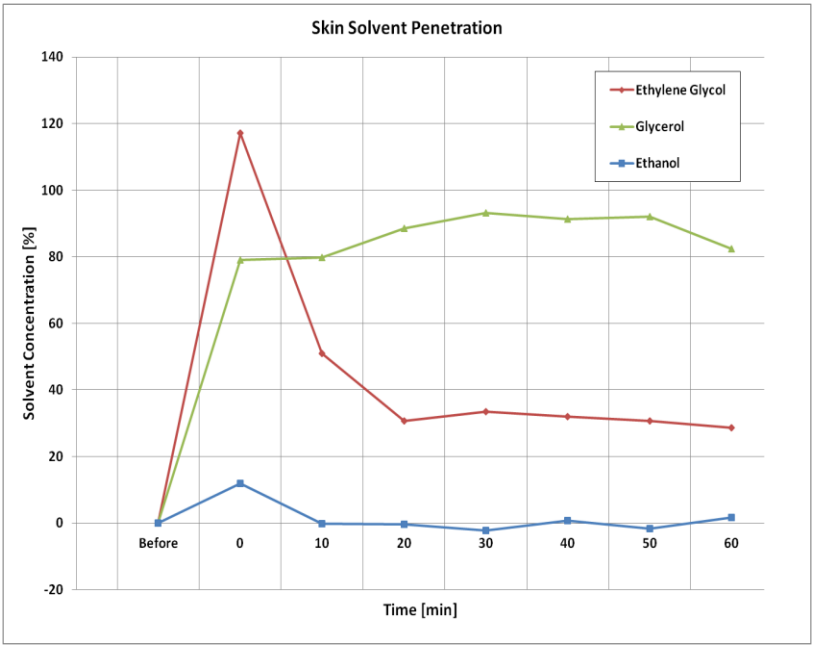


Figure.5.27 The calculated solvent concentration (volume ratio percentage) before after solvent applications

Figure 5.27 shows the corresponding calculated solvent concentration as volume ratio percentage before after solvent applications using the equation below, by using Eq.(5.2). The calculated results agree well with the image results.

The slight negative undershoots in Ethanol results are likely due to the skin drying effect because of Ethanol application. Similarly, the slight overshoot (>100%) just after the Ethylene Glycol application, is likely due to the water increase during 10 minutes application, as the solvent will block the trans-dermal water loss.

5.8 Summary

This chapter describes seven skin experimental investigations on, skin damage measurement, tape stripping measurement, instrument performance measurements, SLS washing measurement, capacitive imaging occlusion effects, solvent penetration measurement and vivo-skin solvent penetration measurement.

The RGB images from ProScope HR2 digital microscope also show interesting results. However, due to inconsistent lighting, it is still difficult to extract meaning results from the images. More works are needed.

The instrument performance measurements show that Epsilon (Fingerprint Sensor) has a good repeatability, and has a good correlation with other instruments, but correlates particularly well with the industrial standard Corneometer.

Skin SLS irritation measurements show that TEWL results are more sensitive than hydration measurements to reflect SLS irritations.

Epsilon time dependent occlusion curves results show that it is possible to use Epsilon occlusion curves for skin damage assessment, as it can assess not only

the scale of damages, but also the types of damages.

The ration of skin hydration and TEWL has been introduced, and results show that under certain conditions, such as soap washing, the hydration/TEWL ratio can generate more consistent results as it combines the effect of both results.

Epsilon solvent penetration results show that we can effectively show the differences of different solvents, and from the images, we can also calculate the solvent concentration.

Chapter 6 Conclusions and Future Work

6.1 Conclusions

This thesis describes the research work carried out by this study. It is divided into five chapters:

Chapter 1 describes the introduction and skin research review.

Chapter 2 describes the skin measurement instruments overview.

Chapter 3 describes the basic mathematical modeling for OTTER.

Chapter 4 describes the development of OTTER data acquisition and data analysis program.

Chapter 5 describes the skin hydration and solvent penetration measurements.

In chapter 1, a general introduction and review of skin characterization such as skin structure, functions and hydration are presented. Five latest skin measurement methods: skin conductance measurement, trans-epidermal water loss (TEWL) measurement, infrared measurement, skin penetration and photothermal measurement have been reviewed.

In chapter 2, several devices are also described, OTTER is a non-contact, non-destructive skin measurement technique and is quick and convenient to use. AquaFlux is good for measuring water vapor flux density from arbitrary surfaces. Epsilon can be applied in many areas, such as skin hydration and *in-vivo* and *in-vitro* solvent penetration through the skin and it has good repeatability, reproducibility, Corneometer is very similar with Epsilon, but it is not convenience and accuracy. Moisture Checker and Hydratest Beauty Pro are another two

instruments to measure the skin hydration. The ProScopeHR2 digital microscope can capture images from both skin surface (outside) and underneath the skin (inside), thanks to cross polarized LED lighting.

In chapter 3, three basic mathematics models: the semi-infinite homogeneous model, gradient model and segmental analysis model have been optimized for the OTTER measurement and analysis system. An enhanced SLS fitting has been proposed. Wavelet de-noising method is used for reducing the noise of collected signal, and fast Fourier transform can change the signal from time domain to frequency domain for analysis.

In chapter 4, it describes the OTTER software development based on PicoScope technology. It first introduces the basic background of the PicoScope 2000 and 4000 series oscilloscopes and the programming language used, and then it describes the functions of the program. In the real-time view, it can implement to display the collected signals, show the specifications (frequency, amplitude RMS, etc.) of the signal, reduce the noise, control the properties (time interval, scale range, whether triggered, etc.), and filtering in different modes. In the measurement view, it can display the curve of original data and fitted curve, it can also show the specifications of the signal, calculate the skin hydration using different models, and save the original data and analysis results in excel files respectively. The OTTER experimental results show that by combining multiple wavelength detection and enhanced SLS fitting, we can make OTTER a powerful tool for skin characterization.

In chapter 5, it concentrates on seven experimental investigations: skin damage measurement, tape stripping measurement, instrument performance measurements, SLS irritation measurement, capacitive imaging occlusion effect, *in-vivo* and *in-vitro* skin solvent penetration measurements. It can be concluded that Epsilon

has a good repeatability and good correlations with other skin measurement instruments, it can be effectively used for a range of different skin measurements. AquaFlux can more sensitive results for skin SLS irritation measurements. The ratio of skin hydration (Epsilon) over TEWL can be more useful for some measurements, such as soap washings. Epsilon time dependent occlusion curve results show that Epsilon can be a powerful tool for skin damage assessments. Epsilon has also shown good potential for *in-vivo* and *in-vitro* skin solvent measurements.

6.2 Future Work

For the future work, the following are suggested:

For the OTTER, it can be improved to use a multi pulsed lasers as the excitation source so that it can measure a matrix area not point by point, which can match the new multiple data acquisition. Otherwise, new mathematics algorithm can be designed to make the OTTER measurement more accuracy. There are also 24 bits ADC modules available, this can further improve the measurement accuracy.

The combination of multiple wavelength detection and depth profiling have shown promising results. More research work could be done to explore further of this approach for skin characterizations, skin disease/cancer detections, and skin contaminations detections etc.

For the AquaFlux, the recent software can only display the results which receive from AquaFlux, some new mathematics algorithms can be added into the software to help the result analysis, meanwhile, recently, AquaFlux is a little heavy and it needs the extra power supply, it can be improved that reduce the condenser so that it can be reduced the DC power supply instead of batteries, and then use the microcontroller technology to control the instrument for example put

a touch screen on the chamber to control the measurement and then use WIFI to send results to PC so that customer can not only control it on PC but also control it on itself.

For the Epsilon, it can be improved to connect to the computer by WIFI and analyze data on the Epsilon website directly and save the data into the cloud big database to share to all the Epsilon users. More image processing techniques, such as texture analysis, wrinkle analysis etc., can be developed. Epsilon is designed based on fingerprint sensor, which is widely used on mobile phone, a software can be designed on mobile phone by using the fingerprint sensor as the Epsilon, or use Bluetooth to connect Epsilon and mobile phone so that Epsilon can be controlled on the mobile phone.

The ratio of skin hydration (measured by Epsilon) and TEWL (measured by AquaFlux) has shown very interesting results. More research work could be done to further develop the method in the area of skin damage assessments.

The occlusion effect of Epsilon has shown potentials of not only capable of showing the scale of the skin damages, but also possible to show the types of the skin damages. More works are needed to systematically evaluate the efficiency and efficacy of this method. A potential commercial application could be developed out of this.

The current linear model to calculation the absolute water content/solvent content is a good first step, but more sophicated models are needed. As we know the water in skin can be free water, loosely bound water and tightly bound water. Their dielectronic constant will be different from free water on its own. More complicated mathematical/physical modeling works are needed in order to get more accurated absolute water content measurements.

Reference

Aforge.Net Framework. http://code.google.com/p/aforge/

Aminghafari M., Cheze N. and Poggi J-M., "Multivariate denoising using wavelets and principal component analysis," Computational Statistics & Data Analysis, 50, pp. 2381-2398, 2006.

An B., Research Multiwavelet on De-noising Method, 2011

Anderson R. L., Cassidy J. M., Hansen J. R. and Yellin W., "Hydration of Stratum Corneum". Biopolymers,**12**,2789-2802,1973.

Anthonissen M., Daly D., Fieuws S., et al, Measurement of elasticity and transepidermal water loss rate of burn scars with the Dermalab®, Burns, Volume 39, Issue 3, May 2013, Pages 420–428

AquaFlux,http://ecce1.lsbu.ac.uk/research/photo/hardwareaquaflux.html

Barry B. W., "Dermatological Formulations (Percutaneous Absorption)" .*New York: M Dekker,*1983.

Bates, D. M. and Watts, D. G. Nonlinear Regression and Its Applications. New York: Wiley, 1988.

Batisse D., Giron F. and Leveque J.L. Capacitance imaging of the skin surface. Skin Research and Technology 12, pp99-104, (2006)

Bevilacqua, A. Gherardi, A., Characterization of a capacitive imaging system for skin surface analysis, First International Workshops on Image Processing Theory, Tools & Applications, 23-26 Nov, 2008

HydraTest, http://www.beautypro.com/products/beautypro-hydratest.html

Benedek M., Kaernbach C., A continuous measure of phasic electrodermal activity. J

Neurosci Methods 2010;190:80–91.

Blank I. H., Moloney J., Emslie A. G., SimonI., Apt C, "The diffusion of water across the stratum corneum as a function of its water content", *J .Invert. Dermatol*.,**82**, 188-194,1984.

Chidambaram N., Burgess, D. J., AAPS PharmSci.1 (1999) 32.

Choi I. M. J., Zhai H., Loffler H., et al, Effect of tape stripping on percutaneous penetration and topical vaccination, Exog. Dermatol., 2 (2003), pp. 262–269

Coneometer,
http://www.courage-khazaka.de/index.php/en/faq-en/faq-scientific-devices/61-corneometer

Corneometer (CM 825), International Journal of Cosmetic Science, 2003, 25, 45-51

Cowen, J.A. In-vivo Opto-thermal Transdermal Diffusion Measurement. Ph.D. Thesis, London South Bank University, London, UK, 1999.

Donoho D., Johnstone I., Ideal spatial adaptation by wavelet shrinkage, Biometrika, 81 (3), pp. 425–455, 1994

Donoho, D.L., "Progress in wavelet analysis and WVD: a ten minute tour, "in Progress in wavelet analysis and applications, Y. Meyer, S. Roques, pp. 109–128. Frontières Ed, 1993

Donoho, D.L., "De-noising by soft-thresholding," IEEE Trans. on Inf. Theory, 42 3, pp. 613– 627, 1995

Donoho, D.L., Johnstone I., Kerkyacharian G., et al, "Wavelet shrinkage: asymptotia, "Jour. Roy. Stat. Soc., series B, Vol. 57, No. 2, pp.301–369, 1995

Dooren v M., VriesJ.J.G., Janssen H. J., Emotional sweating across the body: Comparing 16 different skin conductance measurement locations, Physiology & Behavior, Volume 106, Issue 2, 15 May 2012, Pages 298–304

"Dynamic Link Library", Wikipedia. http://en.wikipedia.org/wiki/Dynamic-link_library.

Fingerprint Sensor for Skin Hydration Imaging, http://ecce1.lsbu.ac.uk/research/photo/fingerprint.html

Godin B., Touitou E., Transdermal skin delivery: predictions for humans from in vivo, ex vivo and animal models, Adv. Drug Deliv. Rev. 59 1152-1161, 2007

Hamed H S., BilalAltrabsheh B., Assa'd T., et al, Construction, in-vitro and in-vivo evaluation of an in-house conductance meter for measurement of skin hydration, Medical Engineering & Physics, Volume 34, Issue 10, December 2012, Pages 1471–1476

Holper L., Scholkmann F., Wolf M., The relationship between sympathetic nervous activity and cerebral hemodynamics and oxygenation: A study using skin conductance measurement and functional near-infrared spectroscopy, Behavioural Brain Research, Volume 270, 15 August 2014, Pages 95–107

Human skin, Wikipedia, https://en.wikipedia.org/wiki/Human_skin, 2017

Imhof, R.E.; Birch, D.J.S.; Thornley, F.R. et al, Opto-thermal transient emission radiometry. J. Phys. E Sci. Instrum. 1984, 17, 521–525.

Imhof, R. E., McKendrick, A.D. and Xiao, P. Thermal emission decay Fourier

transform infrared spectroscopy. Rev. Sci. Instrum. 1995, 66, 5203–5213.

Imhof R. E., Xiao P., Berg E. P. & LI C.: Rapid measurement of TEWL with a condenser-chamber instrument. 15th International Meeting of the ISBS, Philadelphia, 2005

Imhof R. E. and Xiao P., Biox Epsilon - A New Permittivity Imaging System, Exhibitors' Workshop Presentation, ISBS World Congress, Copenhagen 2012.

Imhof R. E., Xiao P., and Angelova-Fischer I., TEWL, Closed-Chamber Methods: AquaFlux and VapoMeter. Non-Invasive Diagnostic Techniques in Clinical Dermatology (Editors: E Berardesca, HI. Maibach & K-P Wilhelm), 345-352, Springer Verlag 2014.

Imhof R. E., Zhang B. and Birch D. J. S., Progress in Photothermal and Photoacoustic Science and Technology (II). edMandelis A (Englewood Cli_s USA: PTR Prentice Hall), 1994: p. 185-236.

Jacques S. L., Nelson J. S., Wright W. H., et al, Applied Optics, 1993, 32, 2439-2446.

Susan C. H., Principles of dermatology, Donald P. Lookingbill, James G. Marks Jr. (Eds.). W. B. Saunders Co., Philadelphia (1986), Journal of the American Academy of Dermatology, Volume 15, Issue 2, Part 1,August 1986,Pages 323-324

Jee, T., Komvopoulos K., "*In vitro* measurement of the mechanical properties of skin by nano/micro indentation methods". Journal of Biomechanics, Volume 47, Issue 5, 21 March 2014, Pages 1186–1192.

Jesus de. M., PhD. Thesis, 1994;

Junqueira L. C., Carneiro J., *Basic Histology,* chapter 19, “Skin”,1977.

Kim Y. J., Kim B., Kim J. W., Combination of nanoparticles with photothermal effects and phase-change material enhances the non-invasive transdermal delivery of drugs. Colloids and Surfaces B: Biointerfaces, 2015, 135:324-311

Kimoto A., Aoki D., A layered sensor for simultaneous measurement of skin water content and softness, Sensors and Actuators A: Physical, Volume 217, 15 September 2014, Pages 111–115

Kusiaka A., Martanb J., Battaglia J. L., Using pulsed and modulated photothermal radiometry to measure the thermal conductivity of thin films, ThermochimicaActa, Volume 556, 20 March 2013, Pages 1–5

Leveque, J.L. and Querleux, B. Skin Chip, a new tool for investigating the skin surface in vivo. Skin Research and Technology 9, 343-347, (2003).

Long F. H., and Deutsch T. F., IEEE Journal of Quantum Electronics, 1987, QE-23, 1821-1826.

Machado M., Teresa M. Salgado T. M. ,Hadgraft J., Lane M. E., The relationship between transepidermal water loss and skin permeability, International Journal of Pharmaceutics, Volume 384, Issues 1–2, 15 January 2010, Pages 73–77

Makin A., Ellemann-Laursen S., Grand N., et al, Transepidermal water loss and tape stripping in mini pig skin, Toxicology Letters, Volume 238, Issue 2, Supplement, 16 October 2015, Page S271

MATLAB Automatic 1-D de-noising,
https://uk.mathworks.com/help/wavelet/ref/wden.html

MATLAB wavelet tool box, https://uk.mathworks.com/products/wavelet.html

Moisture Checker for the Skin, Taberna Pro Medicum, 1999

Ogorevc J., Geršak G., Novak D., et al, "Metrological evaluation of skin conductance measurements". Measurement Volume 46, Issue 9, November 2013, Pages 2993–3001.

Oakley. C, Development of a Fourier Transform in C#, 2011

Opto-Thermal Transient Radiometry (OTTER), http://ecce1.lsbu.ac.uk/research/photo/hardware.html

Ou, X., Pan, W., Xiao, P., In vivo skin capacitive imaging analysis by using grey level co-occurrence matrix (GLCM), International Journal of Pharmaceutics, Available online 2 November 2013, ISSN 0378-5173,http://dx.doi.org/10.1016/j.ijpharm.2013.10.024

Ou X., Pan W., Zhang X., et al, Skin image retrieval using Gabor wavelet texture feature, International Journal of Cosmetic Science, Volume 38, Issue 6, pages 607–614, December 2016, DOI:10.1111/ics.12332 (Impact Factor: 1.542，SCI)

Paleco R., Vucen S. R., Crean A. M., Enhancement of the in vitro penetration of quercetin through pig skin by combined microneedles and lipid microparticles. International Journal of Pharmaceutics, Volume 472, Issues 1–2,10 September 2014,Pages 206-213

Pan W., Zhang X., Chirikhina E., et al, Measurement of Skin Hydration with a Permittivity Contact Imaging System, IFSCC, Paris, 2014

Pan W., Zhang X., Lane M., et al, "The Occlusion Effects in Capacitive Contact Imaging for In-vivo Skin Damage Assessments", International Journal of Cosmetic Science, 2015, 37, 395–400, doi: 10.1111/ics.12209

Pan W., Zhang X., Chirikhina E., et al, "Skin Hydration Measurement Using Contact Imaging", SCC Showcase, New York, December 11 - 12, 2014.

Pan W., Ou X., Zhang X., et al, "Skin Damage Assessment by Using Capacitive Imaging and Condenser-TEWL Method", Skin Forum 14th Annual Meeting 4 to 5 September 2014, Prague, Czech Republic.

Pan W., Ou X., Zhang X., et al, "Skin Damage Assessment by Using Capacitive Imaging and Condenser-TEWL Method", Perspectives in Percutaneous Penetration Conference, 23 - 25 April 2014, La Grande Motte, France.

Pawlak M., Maliński M., Simultaneous measurement of thermal diffusivity and effective infrared absorption coefficient in IR semitransparent and semiconducting n-CdMgSe crystals using photothermal radiometry, ThermochimicaActaVolume 599, 10 January 2015, Pages 23–26

Picoscope® 4000 series, for detailed waveforms and accurate measurements.http://www.picotech.com/document/datasheets/PicoScope4000Series.pdf.

Picoscope® 4262 oscilloscope for detailed waveforms and accurate measurements.https://www.picotech.com/oscilloscope/4262/picoscope-4262-specifications

PicoScope 2200 Series Portable oscilloscopes

https://www.picotech.com/oscilloscope/2200/picoscope-2200-specifications

PicoScope 4000 Series pc oscilloscopes user's guide ps4000.en r6, copyright 2008-2014 Pico Technology Ltd. All rights reserved

PicoScope 4000 Series pc oscilloscopes programmer's guide ps4000.en r7,

copyright 2008-2012 Pico Technology Ltd. All rights reserved

PicoScope 2000 Series pc oscilloscopes user's guide ps20000044-2, copyright 2005-7 Pico Technology Ltd. All rights reserved

PicoScope 2000 Series pc oscilloscopes programmer's guide ps2000.en r10, copyright 2006-2013 Pico Technology Ltd. All rights reserved

Power J. F., Applied Spectroscopy, 1991, 45, 1240-1251.

ProScope HR2http://www.bodelin.com/proscope/proscope-hr2.

Prameela K. S., Formulation and evaluation of antiemetic patch comprising ondansetron hydrochloride, Master dissertation, KLE university, Belgium, 2010

Quesada J. I. P., Guillamón N. M., Anda R. M. D., et al, Effect of perspiration on skin temperature measurements by infrared thermography and contact thermometry during aerobic cycling, Infrared Physics & Technology, Volume 72, September 2015, Pages 68–76

Römgens A. M., Rem-Bronneberg D., Kassies R., Penetration and delivery characteristics of repetitive microjet injection into the skin. Journal of Controlled Release, Volume 234, 28 July 2016, Pages 98-103

Singh, H., Xiao, P., Berg, E.P., et al, Skin Capacitance Imaging for Surface Profiles and Dynamic Water Concentration Measurements, ISBS Conference, Seoul, Korea, May 7-10, 2008.

Soerensen D. D., ClausenS., Mercer J. B., et al, Determining the emissivity of pig skin for accurate infrared thermography, Computers and Electronics in Agriculture, Volume 109, November 2014, Pages 52–58

Spencer T. S., Linamen C. E., Akers W. A., et al, “Temperature dependence of

water content of stratum corneum." *Br J Dermatol*,**93** 159-164,1975.

Swarbrick J. and Boylan, "Permeation Enhancement through Skin" *Encyclopedia of Pharmaceutical Technology*,11,449-492, 1995.

Tachaprutinun A., Meinke M. C., Richter H., et al, Comparison of the skin penetration of Garcinia mangostana extract in particulate and non-particulate form. European Journal of Pharmaceutics and Biopharmaceutics, Volume 86, Issue 2,February 2014,Pages 307-313

Tsai J., Weiner N. D., Flynn G. L. and Ferry J., "Properties of adhesive tapes used for stratum corneum stripping", International Journal of Pharmaceutics,**72**,277-231, 1991.

Van De Graaff K. M., Fox S. I., "Concepts of Human Anatomy & physiology", *Wm C Brown Publishers,* 1995.

Warner, Myers and Taylor, "Water concentration profile across human skin". *The journal of investigative dermatology*,90,199-224,1988.

William A. C., Barry B. W., "Skin Absorption Enhancers", *Critical Reviews in Therapeutic Drug Carrier Systems*,9(3,4):305-353, 1992.

Wu Y. F., Yao J. G., Li T. B., et al, Application of Image Processing Techniques in Infrared detection of Faculty Insulator, 6th Chinese Conference, CCPR, 2014

Xiao P., "Photothermal Radiometry for Skin Research", Cosmetics, 3(1), 10, doi:10.3390/cosmetics3010010, 2016

Xiao, P. Opto-Thermal Mathematical Modeling and Data Analysis in Skin Measurements. Ph.D. Thesis, London South Bank University, London, UK, 1997.

Xiao P., Bontozoglou C., Capacitive contact imaging for in-vivo hair and nail

water content measurements, H&PC Today, Vol. 10(5), pp62-65, September/October 2015.

Xiao P., Cowen JA, and Imhof R. E., In-Vivo Transdermal Drug Diffusion Depth Profiling - A New Approach to Opto-Thermal Signal Analysis Analytical Sci.17 (Special) s349-s352, 2001

Xiao P. and Imhof R.E., UK Patent Application 0004374.5, 2000

Xiao P., Imhof R. E., Opto-thermal measurement of water distribution within the stratum corneum. Skin Bioengineering Techniques and Applications in Dermatology and Cosmetology, CurrProbl Dermatol, Basel: Karger, 1998. 26: p. 48-60.

Xiao, P., Lane, M.E., and Abdalghafor, H.M., Membrane Solvent Penetration Measurements using Contact Imaging, Stratum Corneum VII Conference, Cardiff, UK, Sep 10-12, 2012

Xiao P., Abdalghafor H. and Lane M. E., Membrane Solvent Penetration Measurements Using Contact Imaging, Advances in Dermatological Sciences 2013, DOI:10.1039/9781849734639-00355

Xiao, P., Singh, H., Zheng, X., Berg E.P. et al, In-vivo Skin Imaging For Hydration and Micro Relief Measurements, Stratum Corneum V conference, July 11-13, 2007, Cardiff, UK.

Xiao, P.; Ou, X.; Ciortea, L.I.; Berg, E.P.; Imhof, R.E. In Vivo Skin Solvent Penetration Measurements Using Opto-thermal Radiometry and Fingerprint Sensor. Int. J. Thermophys. 2012, 33, 1787–1794.

Xiao, P.; Zheng, X.; Imhof, R. E.; Hirata, K.; McAuley, W.J.; Mateus, R.; Hadgraft, J.; Lane, M.E. Opto-Thermal Transient Emission Radiometry (OTTER) to image

diffusion in nails in vivo. Int. J. Pharm. 2011, 406, 111–113.

Yoon G.; Welch A.J.; Motamedi, M.; et al, Development and application of Three-Dimensional Light Distribution Model for Laser Irradiated Tissue. IEEE J. Quan. Elec. 1987, 23, 1721–1733

Zamora-Rojas E., Aernouts B., Garrido-Varo A., et al, double integrating spheres measurements for estimating optical properties of pig subcutaneous adipose tissue. Innovative Food Science and Emerging Technologies, 19, 218 – 226 (2013a).

Zamora-Rojas E., Aernouts B., Garrido-Varo A., et al, Optical properties of pig skin epidermis and dermis estimated with double integrating spheres measurements. Innovative Food Science and Emerging Technologies, 20, 343 – 349 (2013b).

Zamora-Rojas E., Garrido-Varo A., Aernouts B., et al, Understanding near infrared radiation propagation in pig skin reflectance measurements, Innovative Food Science & Emerging Technologies, Volume 22, April 2014, Pages 137–146

Zhang X., Bontozoglou C., Xiao P., "Capacitive Imaging for Skin Characterization and Solvent Penetration", Skin Forum 16th Annual Meeting 21 - 22 June 2016, UCL, London, UK.

Zheng X, "Opto-Thermal Transient Emission Radiometry Development for Fundamental Skin Properties Study of The Human Underarm Area", PhD Thesis, 2012.

Publications

1. Zhang X., Bontozoglou C., Xiao P., "Photo-thermal radiometry multiple wavelength detection for skin characterization", The 14th International Workshop on Advanced Infrared Technology and Applications, September 27 to 29, 2017. Université Laval, in Quebec City, Canada

2. Zhang X., Bontozoglou C., Xiao P., "Opto-Thermal Depth Resolved Detection Spectra for In-vivo Skin Measurements", 19th International Conference on Photoacoustic and Photothermal Phenomena , Bilbao, Spain. July 16-20, 2017

3. Zhang X., Bontozoglou C., Xiao P., "Capacitive Imaging for Skin Characterization and Solvent Penetration", Skin Forum 16th Annual Meeting 21 - 22 June 2016, UCL, London, UK.

4. Ou X., Pan W., Zhang X. and Xiao P., Skin image retrieval using Gabor wavelet texture feature, International Journal of Cosmetic Science, Volume 38, Issue 6, pages 607–614, December 2016, DOI:10.1111/ics.12332

5. Pan W., Zhang X., Lane M. and Xiao P., "The Occlusion Effects in Capacitive Contact Imaging for In-vivo Skin Damage Assessments", International Journal of Cosmetic Science, 2015, 37, 395–400, doi: 10.1111/ics.12209

6. Pan W., Zhang X., Chirikhina E. and Xiao P., "Skin Hydration Measurement Using Contact Imaging", SCC Showcase, New York, December 11 - 12, 2014.

7. Pan W., Ou X., Zhang X. and Xiao P., "Skin Damage Assessment by Using

Capacitive Imaging and Condenser-TEWL Method", Skin Forum 14th Annual Meeting 4 to 5 September 2014, Prague, Czech Republic.

8. Pan W., Ou X., Zhang X. and Xiao P., "Skin Damage Assessment by Using Capacitive Imaging and Condenser-TEWL Method", Perspectives in Percutaneous Penetration Conference, 23 - 25 April 2014, La Grande Motte, France.

Appendix

Appendix I: OTTER C# main program

```
using AForge;
using AForge.Math;
using CalculateDLL;
using Frequency;
using LSF;
using PFilter;
using MathWorks.MATLAB.NET.Arrays;
using MathWorks.MATLAB.NET.Utility;
using Excel = Microsoft.Office.Interop.Excel;
using System.Reflection;
using picoDll;
using Pico2000Dll;
using DeN;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Pico4424
{
    public partial class Form1 : Form
    {
        //simulation model

        double dtsim;
        Random rtau = new Random();
        double D = 0.0000001;
        double beta;
        double betas;
        double betaw = 290000;
        double betad = 1000;

        //simulation model end

        //pico model

        picodll p1 = new picodll();
        pico2000dll p2 = new pico2000dll();

        int time_interval;
        int no_of_samples;
        short timebase;
        short channel_ext;
        short channel_enable;
        short threshold;
        int direction;
        short delay;
        short range;
        short offset;
        short timeoffset;
        short offset2000;
        short timeoffset2000;
        int trig2000CH;
        bool type;

        int Ps2000time_interval;
```

```
int Ps2000no_of_samples;
short Ps2000timebase;
short Ps2000channel_ext;
short Ps2000channel_enable;
short Ps2000threshold;
int Ps2000direction;
short Ps2000delay;
short Ps2000range;
int Ps2000max_samples;
short Ps2000time_units;

//pico model end

//writing file

string file;
string outputfile;
StreamWriter sw;
StreamWriter sw1;
int count = 0;
long counter;

Excel.Application xlApp;
Excel.Workbook xlWorkBook;
Excel.Worksheet xlWorkSheet;

//writing file end

//measurement model

string[] sf;
string[] sx;

double[] chaverage = new double[5000];
double[] chaverageA = new double[5000];
double[] chaverageB = new double[5000];
double[] chaverageC = new double[5000];
double[] chaverageD = new double[5000];
double[] chaverage2000A = new double[5000];
double[] chaverage2000B = new double[5000];
int n;

Calculatedll c1 = new Calculatedll();
LSF.LSF lsf = new LSF.LSF();
MWNumericArray mw = new MWNumericArray();

Frequency.Frequency fr = new Frequency.Frequency();
DeN.DeN den = new DeN.DeN();
PFilter.PFilter pf = new PFilter.PFilter();



MWNumericArray fr1 = new MWNumericArray();
MWNumericArray fr2 = new MWNumericArray();
MWNumericArray time = new MWNumericArray();

MWNumericArray mw1 = new MWNumericArray();
MWNumericArray mw2 = new MWNumericArray();
MWNumericArray mw3 = new MWNumericArray();
MWNumericArray mw4 = new MWNumericArray();

//measurement model end

//screen resize

private void Form1_Resize(object sender, EventArgs e)
{
    tabControl1.Height = this.Size.Height - statusStrip1.Height - 30;
    tabControl1.Width = this.Size.Width;

    tabControl3.Height = tabControl1.Height - statusStrip1.Height - 50;

    groupBox1.Left = tabControl1.Left;
```

```
        groupBox1.Height = tabControl1.Height - statusStrip1.Height - 40;
        groupBox1.Width = tabControl1.Width - tabControl3.Width - 50;

        chart1.Width = groupBox1.Width - 20;
        chart1.Height = groupBox1.Height - 50;

        groupBox2.Left = tabControl1.Left;
        groupBox2.Width = tabControl1.Width - 50;
        groupBox2.Height = tabControl1.Height - groupBox3.Height - 50;

        groupBox3.Width = tabControl1.Width - 50;

        chart2.Height = groupBox2.Height - 50;
        chart3.Height = groupBox2.Height - 50;
        chart4.Height = groupBox2.Height - 50;

        chart2.Width = groupBox2.Width - chart3.Width - 50;
        chart3.Width = groupBox2.Width - chart2.Width - 50;
    }

    //resize end

    public Form1()
    {
        InitializeComponent();
    }

    //open

    private void Form1_Load(object sender, EventArgs e)
    {
        openFileDialog1.Multiselect = true;

        string s = "";
        s = p1.open();

        p2.open();

        type = Environment.Is64BitOperatingSystem;

        if (s.Length != 0)
        {
            sf = s.Split('\t');
            sx = sf[3].Split(':');

            PicoScopetoolStripStatusLabel.Text = s;

        }

        else
        {

        }

        RealTimetimer.Enabled = true;
        PS2000RealTimetimer.Enabled = true;

        ChannelOnecheckBox.Checked = MyApp.Default.ChannelOne;
        ChannelTwocheckBox.Checked = MyApp.Default.ChannelTwo;
        ChannelThreecheckBox.Checked = MyApp.Default.ChannelThree;
        ChannelFourcheckBox.Checked = MyApp.Default.ChannelFour;
        TimeIntervalcomboBox.SelectedIndex = MyApp.Default.TimeInterval;
        ScanRangecomboBox.SelectedIndex = MyApp.Default.ScanRange;
        TriggerChannelcomboBox.SelectedIndex = MyApp.Default.TriggerChannel;
        TriggerThresholdcomboBox.SelectedIndex = MyApp.Default.TriggerThreshold;
        SamplingPointcomboBox.SelectedIndex = MyApp.Default.SamplingPoints;
        DelaycomboBox.SelectedIndex = MyApp.Default.Delay;
        TriggerDirectioncomboBox.SelectedIndex = MyApp.Default.TriggerDirection;
        OpenModecomboBox.SelectedIndex = MyApp.Default.OpenMode;
        SettingModecomboBox.SelectedIndex = MyApp.Default.SettingMode;
        SavingtextBox.Text = MyApp.Default.SavingText1;
        OutputFiletextBox.Text = MyApp.Default.SavingText2;
        Ps4000checkBox.Checked = MyApp.Default.Ps4000;
```

```
    Ps2000checkBox.Checked = MyApp.Default.Ps2000;
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    MyApp.Default.ChannelOne = ChannelOnecheckBox.Checked;
    MyApp.Default.ChannelTwo = ChannelTwocheckBox.Checked;
    MyApp.Default.ChannelThree = ChannelThreecheckBox.Checked;
    MyApp.Default.ChannelFour = ChannelFourcheckBox.Checked;
    MyApp.Default.TimeInterval = TimeIntervalcomboBox.SelectedIndex;
    MyApp.Default.ScanRange = ScanRangecomboBox.SelectedIndex;
    MyApp.Default.TriggerChannel = TriggerChannelcomboBox.SelectedIndex;
    MyApp.Default.TriggerThreshold = TriggerThresholdcomboBox.SelectedIndex;
    MyApp.Default.SamplingPoints = SamplingPointcomboBox.SelectedIndex;
    MyApp.Default.Delay = DelaycomboBox.SelectedIndex;
    MyApp.Default.TriggerDirection = TriggerDirectioncomboBox.SelectedIndex;
    MyApp.Default.OpenMode = OpenModecomboBox.SelectedIndex;
    MyApp.Default.SettingMode = SettingModecomboBox.SelectedIndex;
    MyApp.Default.SavingText1 = SavingtextBox.Text;
    MyApp.Default.SavingText2 = OutputFiletextBox.Text;
    MyApp.Default.Ps4000 = Ps4000checkBox.Checked;
    MyApp.Default.Ps2000 = Ps2000checkBox.Checked;

    MyApp.Default.Save();

    p1.close();
    p2.close();
}

//close

//real-time part

private void OpenModecomboBox_SelectedIndexChanged(object sender, EventArgs e)
{
    if (OpenModecomboBox.SelectedIndex == 0)
    {
        RealTimetimer.Enabled = true;
    }
}


private void Ps4000checkBox_Click(object sender, EventArgs e)
{
        if (Ps4000checkBox.Checked == true)
        {
            RealTimetimer.Enabled = true;
        }

        else
        {
            RealTimetimer.Enabled = false;
        }
}

private void Ps2000checkBox_Click(object sender, EventArgs e)
{
    if (Ps2000checkBox.Checked == true)
    {
        PS2000RealTimetimer.Enabled = true;
    }

    else
    {
        PS2000RealTimetimer.Enabled = false;
    }
}

private void RealTimetimer_Tick(object sender, EventArgs e)
{
    this.chart1.Series["Channel Simulation"].Points.Clear();
    this.chart1.Series["ChannelOne"].Points.Clear();
    this.chart1.Series["ChannelTwo"].Points.Clear();
```

```
            this.chart1.Series["ChannelThree"].Points.Clear();
            this.chart1.Series["ChannelFour"].Points.Clear();
            this.chart1.Series[5].Points.Clear();
            this.chart1.Series[6].Points.Clear();

            this.chart1.ChartAreas[0].AxisX.Title = "Time(ns)";

            chartconfigration();

            short[] chaA = new short[no_of_samples];
            short[] chaB = new short[no_of_samples];
            short[] chaC = new short[no_of_samples];
            short[] chaD = new short[no_of_samples];

            //simulation mode

            double[] chasim = new double[no_of_samples];
            double[] xsim = new double[no_of_samples];

            readSimulation(ref chasim, dtsim);

            double dfsim = 1 / (no_of_samples * (dtsim * 0.000000001));

            //end

            double[] x = new double[no_of_samples];
            double[] chafr = new double[no_of_samples];
            double[] chaDN1 = new double[no_of_samples];


            Complex[] y = new Complex[no_of_samples];
            AForge.Math.Complex[] input = new AForge.Math.Complex[no_of_samples];
            double df = 1 / (no_of_samples * (time_interval * 0.000000001));

            //simulation model

                if (OpenModecomboBox.SelectedIndex == 0 &&
TimeDomainradioButton.Checked==true)
                {
                    for (int i = 0; i < no_of_samples; i++)
                    {
                        xsim[i] = i * dtsim;
                        this.chart1.Series["Channel Simulation"].Points.AddXY(xsim[i],
chasim[i]);
                    }
                }

                //simulation model end

                else if (OpenModecomboBox.SelectedIndex == 1)
                {
                    p1.app(ref chaA, ref chaB, ref chaC, ref chaD, ref time_interval,
no_of_samples, timebase, channel_ext, channel_enable, threshold, direction, delay, 10, offset,
timeoffset, type);
                }

                if (TimeDomainradioButton.Checked == true)
                {
                    this.chart1.ChartAreas[0].AxisX.Title = "Time(ns)";

                    if (ChannelOnecheckBox.Checked == true)
                    {
                        for (int i = 0; i < no_of_samples; i++)
                        {
                            x[i] = i * time_interval;
                            this.chart1.Series["ChannelOne"].Points.AddXY(x[i], chaA[i]);
                        }

                        for (int i = 0; i < no_of_samples; i++)
                        {
                            chafr[i] = (double)chaA[i];
                        }
                        fr1 = chafr;
```

```
                    time = time_interval * 0.000000001;
                    mw1 = (MWNumericArray)(den.DeNtest(fr1));
                    mw2 = (MWNumericArray)(fr.ffttest(fr1, time));


                    double[] arrA =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);
                    AFrequencylabel.Text = "Frequency: " + "" + (int)arrA[0] + "Hz";
                    AAmplitudelabel.Text = "Amplitude: " + "" + (int)arrA[1] + "mV";
                    AAmplitudeMlabel.Text = "Amplitude(-): " + "" + (int)arrA[2] + "mV";
                    APtoPlabel.Text = "P to P Amp: " + "" + (int)arrA[3] + "mV";
                    ARMSlabel.Text = "RMS: " + "" + (int)arrA[4] + "mV";
                    ASNRlabel.Text = "SNR: " + "" + mw1.ToString();

                    if (DeNoisecheckBox.Checked == true)
                    {
                        this.chart1.Series["ChannelOne"].Points.Clear();

                        mw3 = (MWNumericArray)(den.DeNtestDN(fr1));

                        double[] chaDN =
(double[])((MWNumericArray)mw3).ToVector(MWArrayComponent.Real);

                        for (int i = 0; i < no_of_samples; i++)
                        {
                            x[i] = i * time_interval;
                            this.chart1.Series["ChannelOne"].Points.AddXY(x[i],
chaDN[i]);
                        }

                    }

                    if (LowFiltercheckBox.Checked == true)
                    {
                        this.chart1.Series["ChannelOne"].Points.Clear();

                        double Fc = Convert.ToDouble(LowFiltertextBox.Text);

                        mw4 = (MWNumericArray)(pf.LPfilter(fr1, time, Fc));

                        double[] chaLP =
(double[])((MWNumericArray)mw4).ToVector(MWArrayComponent.Real);

                        for (int i = 0; i < no_of_samples; i++)
                        {
                            x[i] = i * time_interval;
                            this.chart1.Series["ChannelOne"].Points.AddXY(x[i],
chaLP[i]);
                        }
                    }

                    if (HighFiltercheckBox.Checked == true)
                    {
                        this.chart1.Series["ChannelOne"].Points.Clear();

                        double Fc = Convert.ToDouble(HighFiltertextBox.Text);

                        mw4 = (MWNumericArray)(pf.HPfilter(fr1, time, Fc));

                        double[] chaHP =
(double[])((MWNumericArray)mw4).ToVector(MWArrayComponent.Real);

                        for (int i = 0; i < no_of_samples; i++)
                        {
                            x[i] = i * time_interval;
                            this.chart1.Series["ChannelOne"].Points.AddXY(x[i],
chaHP[i]);
                        }
                    }

                    if (BandFiltercheckBox.Checked == true)
                    {
                        this.chart1.Series["ChannelOne"].Points.Clear();
```

```
                        double Fc1 = Convert.ToDouble(BandFiltertextBox1.Text);
                        double Fc2 = Convert.ToDouble(BandFiltertextBox2.Text);

                        mw4 = (MWNumericArray)(pf.BPfilter(fr1, time, Fc1,Fc2));

                        double[] chaBP =
(double[])((MWNumericArray)mw4).ToVector(MWArrayComponent.Real);

                        for (int i = 0; i < no_of_samples; i++)
                        {
                            x[i] = i * time_interval;
                            this.chart1.Series["ChannelOne"].Points.AddXY(x[i],
chaBP[i]);
                        }
                    }

                    if (NotchFiltercheckBox.Checked == true)
                    {
                        this.chart1.Series["ChannelOne"].Points.Clear();

                        double Fc = Convert.ToDouble(NotchFiltertextBox1.Text);

                        mw4 = (MWNumericArray)(pf.NPfilter(fr1, time, Fc));

                        double[] chaNP =
(double[])((MWNumericArray)mw4).ToVector(MWArrayComponent.Real);

                        for (int i = 0; i < no_of_samples; i++)
                        {
                            x[i] = i * time_interval;
                            this.chart1.Series["ChannelOne"].Points.AddXY(x[i],
chaNP[i]);
                        }
                    }
                }

                if (ChannelTwocheckBox.Checked == true)
                {
                    for (int i = 0; i < no_of_samples; i++)
                    {
                        x[i] = i * time_interval;
                        this.chart1.Series["ChannelTwo"].Points.AddXY(x[i], chaB[i]);
                    }

                    for (int i = 0; i < no_of_samples; i++)
                    {
                        chafr[i] = (double)chaB[i];
                    }
                    fr1 = chafr;
                    time = time_interval * 0.000000001;
                    mw1 = (MWNumericArray)(den.DeNtest(fr1));
                    mw2 = (MWNumericArray)(fr.ffttest(fr1, time));
                    double[] arrB =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);
                    BFrequencylabel.Text = "Frequency: " + "" + (int)arrB[0] + "Hz";
                    BAmplitudelabel.Text = "Amplitude: " + "" + (int)arrB[1] + "mV";
                    BAmplitudeMlabel.Text = "Amplitude(-): " + "" + (int)arrB[2] + "mV";
                    BPtoPlabel.Text = "P to P Amp: " + "" + (int)arrB[3] + "mV";
                    BRMSlabel.Text = "RMS: " + "" + (int)arrB[4] + "mV";
                    BSNRlabel.Text = "SNR: " + "" + mw1.ToString();
                }

                if (ChannelThreecheckBox.Checked == true)
                {
                    for (int i = 0; i < no_of_samples; i++)
                    {
                        x[i] = i * time_interval;
                        this.chart1.Series["ChannelThree"].Points.AddXY(x[i], chaC[i]);
                    }

                    for (int i = 0; i < no_of_samples; i++)
                    {
```

```
                        chafr[i] = (double)chaC[i];
                    }
                    fr1 = chafr;
                    time = time_interval * 0.000000001;
                    mw1 = (MWNumericArray)(den.DeNtest(fr1));
                    mw2 = (MWNumericArray)(fr.ffttest(fr1, time));
                    double[] arrC =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);
                    CFrequencylabel.Text = "Frequency: " + "" + (int)arrC[0] + "Hz";
                    CAmplitudelabel.Text = "Amplitude: " + "" + (int)arrC[1] + "mV";
                    CAmplitudeMlabel.Text = "Amplitude(-): " + "" + (int)arrC[2] + "mV";
                    CPtoPlabel.Text = "P to P Amp: " + "" + (int)arrC[3] + "mV";
                    CRMSlabel.Text = "RMS: " + "" + (int)arrC[4] + "mV";
                    CSNRlabel.Text = "SNR: " + "" + mw1.ToString();
                }

                if (ChannelFourcheckBox.Checked == true)
                {
                    for (int i = 0; i < no_of_samples; i++)
                    {
                        x[i] = i * time_interval;
                        this.chart1.Series["ChannelFour"].Points.AddXY(x[i], chaD[i]);
                    }

                    for (int i = 0; i < no_of_samples; i++)
                    {
                        chafr[i] = (double)chaD[i];
                    }
                    fr1 = chafr;
                    time = time_interval * 0.000000001;
                    mw1 = (MWNumericArray)(den.DeNtest(fr1));
                    mw2 = (MWNumericArray)(fr.ffttest(fr1, time));
                    double[] arrD =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);
                    DFrequencylabel.Text = "Frequency: " + "" + (int)arrD[0] + "Hz";
                    DAmplitudelabel.Text = "Amplitude: " + "" + (int)arrD[1] + "mV";
                    DAmplitudeMlabel.Text = "Amplitude(-): " + "" + (int)arrD[2] + "mV";
                    DPtoPlabel.Text = "P to P Amp: " + "" + (int)arrD[3] + "mV";
                    DRMSlabel.Text = "RMS: " + "" + (int)arrD[4] + "mV";
                    DSNRlabel.Text = "SNR: " + "" + mw1.ToString();

                }
            }

            if (FrequencyDomainradioButton.Checked == true)
            {
                this.chart1.ChartAreas[0].AxisX.Title = "Frequency(Hz)";

                if (OpenModecomboBox.SelectedIndex == 0)
                {
                    for (int i = 0; i < no_of_samples; i++)
                    {
                        input[i].Re = chasim[i];
                        input[i].Im = 0.0;
                    }

                    c1.FFT(input, FourierTransform.Direction.Forward);

                    for (int i = 0; i < (no_of_samples / 2); i++)
                    {
                        xsim[i] = i * dfsim;
                        this.chart1.Series["Channel Simulation"].Points.AddXY(xsim[i],
Math.Abs(input[i].Re));

                    }
                }

                else if (OpenModecomboBox.SelectedIndex == 1)
                {

                    if (ChannelOnecheckBox.Checked == true)
                    {
                        for (int i = 0; i < no_of_samples; i++)
```

```
                        {
                            input[i].Re = chaA[i];
                            input[i].Im = 0.0;
                        }

                        c1.FFT(input, FourierTransform.Direction.Forward);

                        for (int i = 0; i < no_of_samples / 2; i++)
                        {
                            x[i] = i * df;
                            this.chart1.ChartAreas[0].AxisX.Maximum = 500000;
                            this.chart1.ChartAreas[0].AxisX.Interval = 50000;
                            this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
                            this.chart1.ChartAreas[0].AxisY.Interval = 100;

                            this.chart1.Series["ChannelOne"].Points.AddXY(x[i],
Math.Abs(input[i].Re));
                        }
                    }

                    if (ChannelTwocheckBox.Checked == true)
                    {
                        for (int i = 0; i < no_of_samples; i++)
                        {
                            input[i].Re = chaB[i];
                            input[i].Im = 0.0;
                        }

                        c1.FFT(input, FourierTransform.Direction.Forward);

                        for (int i = 0; i < no_of_samples / 2; i++)
                        {
                            x[i] = i * df;
                            this.chart1.Series["ChannelTwo"].Points.AddXY(x[i],
Math.Abs(input[i].Re));
                        }
                    }

                    if (ChannelThreecheckBox.Checked == true)
                    {
                        for (int i = 0; i < no_of_samples; i++)
                        {
                            input[i].Re = chaC[i];
                            input[i].Im = 0.0;
                        }

                        c1.FFT(input, FourierTransform.Direction.Forward);

                        for (int i = 0; i < no_of_samples / 2; i++)
                        {
                            x[i] = i * df;
                            this.chart1.Series["ChannelThree"].Points.AddXY(x[i],
Math.Abs(input[i].Re));
                        }
                    }

                    if (ChannelFourcheckBox.Checked == true)
                    {
                        for (int i = 0; i < no_of_samples; i++)
                        {
                            input[i].Re = chaD[i];
                            input[i].Im = 0.0;
                        }

                        c1.FFT(input, FourierTransform.Direction.Forward);

                        for (int i = 0; i < no_of_samples / 2; i++)
                        {
                            x[i] = i * df;
                            this.chart1.Series["ChannelFour"].Points.AddXY(x[i],
Math.Abs(input[i].Re));
                        }
                    }
```

```
                }
            }
        }

        private void PS2000RealTimetimer_Tick(object sender, EventArgs e)
        {
            this.chart1.Series[0].Points.Clear();
            this.chart1.Series[1].Points.Clear();
            this.chart1.Series[2].Points.Clear();
            this.chart1.Series[3].Points.Clear();
            this.chart1.Series[4].Points.Clear();
            this.chart1.Series[5].Points.Clear();
            this.chart1.Series[6].Points.Clear();

            Ps2000chartconfigration();

            short[] Ps2000ChA = new short[Ps2000no_of_samples];
            short[] Ps2000ChB = new short[Ps2000no_of_samples];
            short[] Ps2000ChC = new short[Ps2000no_of_samples];
            short[] Ps2000ChD = new short[Ps2000no_of_samples];

            double[] Ps2000chafr = new double[Ps2000no_of_samples];
            double[] x = new double[Ps2000no_of_samples];

            p2.app(ref Ps2000ChA, ref Ps2000ChB, ref Ps2000ChC, ref Ps2000ChD, Ps2000timebase,
Ps2000no_of_samples, ref Ps2000time_interval, ref Ps2000time_units, ref Ps2000max_samples,
Ps2000channel_ext, Ps2000threshold, (short)Ps2000direction, Ps2000delay, 6, offset2000,
timeoffset2000);

            if (Ps2000ChannelOnecheckBox.Checked == true)
            {
                for (int i = 0; i < Ps2000no_of_samples; i++)
                {
                    x[i] = i * Ps2000time_interval;
                    this.chart1.Series[5].Points.AddXY(x[i], Ps2000ChA[i]);
                }

                for (int i = 0; i < Ps2000no_of_samples; i++)
                {
                    Ps2000chafr[i] = (double)Ps2000ChA[i];
                }

                fr1 = Ps2000chafr;
                time = Ps2000time_interval * 0.000000001;
                mw1 = (MWNumericArray)(den.DeNtest(fr1));
                mw2 = (MWNumericArray)(fr.ffttest(fr1, time));
                double[] arrE =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);
                EFrequencylabel.Text = "Frequency: " + "" + (int)arrE[0] + "Hz";
                EAmplitudelabel.Text = "Amplitude: " + "" + (int)arrE[1] + "mV";
                EAmplitudeMlabel.Text = "Amplitude(-): " + "" + (int)arrE[2] + "mV";
                EPtoPlabel.Text = "P to P Amp: " + "" + (int)arrE[3] + "mV";
                ERMSlabel.Text = "RMS: " + "" + (int)arrE[4] + "mV";
                ESNRlabel.Text = "SNR: " + "" + mw1.ToString();
            }

            if (Ps2000ChannelTwocheckBox.Checked == true)
            {
                for (int i = 0; i < Ps2000no_of_samples; i++)
                {
                    x[i] = i * Ps2000time_interval;
                    this.chart1.Series[6].Points.AddXY(x[i], Ps2000ChB[i]);
                }

                for (int i = 0; i < Ps2000no_of_samples; i++)
                {
                    Ps2000chafr[i] = (double)Ps2000ChB[i];
                }

                fr1 = Ps2000chafr;
                time = Ps2000time_interval * 0.000000001;
                mw1 = (MWNumericArray)(den.DeNtest(fr1));
                mw2 = (MWNumericArray)(fr.ffttest(fr1, time));
```

```
                double[] arrF =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);
                FFrequencylabel.Text = "Frequency: " + "" + (int)arrF[0] + "Hz";
                FAmplitudelabel.Text = "Amplitude: " + "" + (int)arrF[1] + "mV";
                FAmplitudeMlabel.Text = "Amplitude(-): " + "" + (int)arrF[2] + "mV";
                FPtoPlabel.Text = "P to P Amp: " + "" + (int)arrF[3] + "mV";
                FRMSlabel.Text = "RMS: " + "" + (int)arrF[4] + "mV";
                FSNRlabel.Text = "SNR: " + "" + mw1.ToString();
            }
        }

        //real-time part end

        //measurement part

        private void Measurementbutton_Click(object sender, EventArgs e)
        {
            count++;

            file = SavingtextBox.Text;
            outputfile = OutputFiletextBox.Text;

            xlApp = new Microsoft.Office.Interop.Excel.Application();

            xlWorkBook = xlApp.Workbooks.Add(Missing.Value);
            xlApp.Visible = false;
            xlApp.DisplayAlerts = false;

            chaverageA = new double[5000];

            for (int i = 1; i < 8; i++)
            {
                try
                {
                    xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.get_Item(i);
                }
                catch (Exception ex)
                {
                    xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.Add(Missing.Value,
xlApp.Worksheets[xlApp.Sheets.Count], 1, Missing.Value);
                }
            }

            if (SettingModecomboBox.SelectedIndex == 0)
            {
                for (int i = 1; i < 7; i++)
                {
                    xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.get_Item(i);

                    if (OpenModecomboBox.SelectedIndex == 1)
                    {
                        xlWorkSheet.Cells[11, 1] = time_interval * 0.000000001;
                    }
                    // xlWorkSheet.Cells[9, 1] = time_interval * 0.000000001;

                    xlWorkSheet.Cells[12, 1] = no_of_samples;
                }
            }

            else
            {
                for (int i = 1; i < 8; i++)
                {
                    xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.get_Item(i);

                    xlWorkSheet.Cells[1, 1] = "OTTER 2.8.2 @" + "    " + "Time: " +
DateTime.Now.ToString() +
"--=3:Comment=4:Average=5:Excitation=6:Detection=7:LaserFreq=8:Voltage Calibration=9:Time
Interval=10:Buffer=11:Detection Time=12:Peak=--";
                    xlWorkSheet.Cells[2, 1] = "1";
                    xlWorkSheet.Cells[3, 1] = CommenttextBox.Text;
                    xlWorkSheet.Cells[4, 1] = AverageTimetextBox.Text;
                    xlWorkSheet.Cells[5, 1] = "2.94";
```

```
                    xlWorkSheet.Cells[6, 1] = "13.1";
                    xlWorkSheet.Cells[7, 1] = "10";
                    xlWorkSheet.Cells[8, 1] = "5";

                    if (OpenModecomboBox.SelectedIndex == 0)
                    {
                        xlWorkSheet.Cells[9, 1] = dtsim * 0.000000001;
                    }

                    if (OpenModecomboBox.SelectedIndex == 1)
                    {
                        xlWorkSheet.Cells[9, 1] = time_interval * 0.000000001;
                    }

                    xlWorkSheet.Cells[10, 1] = no_of_samples;
                    xlWorkSheet.Cells[11, 1] = "0.27";
                    xlWorkSheet.Cells[12, 1] = "518,92,183.199909E-3";
                }
            }

            xlWorkBook.SaveAs(file, Missing.Value, Missing.Value, Missing.Value,
Missing.Value, Missing.Value, Excel.XlSaveAsAccessMode.xlExclusive, Missing.Value,
Missing.Value, Missing.Value, Missing.Value, Missing.Value);
            xlWorkBook.Close();

            if (!File.Exists(outputfile))
            {
                xlWorkBook = xlApp.Workbooks.Add(Missing.Value);
                xlApp.Visible = false;
                xlApp.DisplayAlerts = false;

                for (int i = 1; i < 8; i++)
                {
                    try
                    {
                        xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.get_Item(i);
                    }
                    catch (Exception ex)
                    {
                        xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.Add(Missing.Value,
xlApp.Worksheets[xlApp.Sheets.Count], 1, Missing.Value);
                    }
                }

                if (SettingModecomboBox.SelectedIndex == 0)
                {
                    for (int i = 1; i < 7; i++)
                    {
                        xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.get_Item(i);

                        xlWorkSheet.Cells[1, 1] = "FileName";
                        xlWorkSheet.Cells[1, 2] = "Date";
                        xlWorkSheet.Cells[1, 3] = "Frequency";
                        xlWorkSheet.Cells[1, 4] = "Amplitude";
                        xlWorkSheet.Cells[1, 5] = "Amplitude(-)";
                        xlWorkSheet.Cells[1, 6] = "P-P";
                        xlWorkSheet.Cells[1, 7] = "RMS";
                        xlWorkSheet.Cells[1, 8] = "SNR";
                    }

                    xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.get_Item(7);

                    xlWorkSheet.Cells[1, 1] = "FileName";
                    xlWorkSheet.Cells[1, 2] = "Date";
                    xlWorkSheet.Cells[1, 3] = "Hydration";
                    xlWorkSheet.Cells[1, 4] = "Hydration Gradient";
                    xlWorkSheet.Cells[1, 5] = "Tau";
                    xlWorkSheet.Cells[1, 6] = "W";
                    xlWorkSheet.Cells[1, 7] = "A";
                    xlWorkSheet.Cells[1, 8] = "1";
                    xlWorkSheet.Cells[1, 9] = "Sampling points";
                    xlWorkSheet.Cells[1, 10] = "Hydration (D)";
                    xlWorkSheet.Cells[1, 11] = "Tau (D)";
```

```
                xlWorkSheet.Cells[1, 12] = "A (D)";
                xlWorkSheet.Cells[1, 13] = "Start point";
                xlWorkSheet.Cells[1, 14] = "Last point";
                xlWorkSheet.Cells[1, 15] = "Deep";
            }

            else
            {
                for (int i = 1; i < 8; i++)
                {
                    xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.get_Item(i);

                    xlWorkSheet.Cells[1, 1] = "FileName";
                    xlWorkSheet.Cells[1, 2] = "Date";
                    xlWorkSheet.Cells[1, 3] = "Hydration";
                    xlWorkSheet.Cells[1, 4] = "Hydration Gradient";
                    xlWorkSheet.Cells[1, 5] = "Tau";
                    xlWorkSheet.Cells[1, 6] = "W";
                    xlWorkSheet.Cells[1, 7] = "A";
                    xlWorkSheet.Cells[1, 8] = "1";
                    xlWorkSheet.Cells[1, 9] = "Sampling points";
                    xlWorkSheet.Cells[1, 10] = "Hydration (D)";
                    xlWorkSheet.Cells[1, 11] = "Tau (D)";
                    xlWorkSheet.Cells[1, 12] = "A (D)";
                    xlWorkSheet.Cells[1, 13] = "Start point";
                    xlWorkSheet.Cells[1, 14] = "Last point";
                    xlWorkSheet.Cells[1, 15] = "Deep";
                }
            }

            xlWorkBook.SaveAs(outputfile, Missing.Value, Missing.Value, Missing.Value,
Missing.Value, Missing.Value, Missing.Value, Excel.XlSaveAsAccessMode.xlExclusive, Missing.Value,
Missing.Value, Missing.Value, Missing.Value, Missing.Value);
            xlWorkBook.Close();
        }

        counter = 0;

        for (int i = 0; i < chaverage.Length; i++)
        {
            chaverage[i] = 0;
        }

        progressBar1.Value = 0;
        progressBar1.Maximum = Convert.ToInt16(AverageTimetextBox.Text);
        progressBar1.Step = 1;

        this.chart2.ChartAreas[0].AxisX.Title = "Time[ns]";
        this.chart2.ChartAreas[0].AxisY.Title = "Voltage[mV]";

        this.chart3.ChartAreas[0].AxisX.Title = "Time Count";
        this.chart3.ChartAreas[0].AxisY.Title = "Results";

        Measurementtimer.Enabled = true;
    }

    private void Measurementtimer_Tick(object sender, EventArgs e)
    {
        this.chart2.Series[0].Points.Clear();
        this.chart2.Series[1].Points.Clear();
        this.chart2.Series[2].Points.Clear();
        this.chart2.Series[3].Points.Clear();
        this.chart2.Series[4].Points.Clear();
        this.chart2.Series[5].Points.Clear();
        this.chart2.Series[6].Points.Clear();
        this.chart2.Series[7].Points.Clear();
        this.chart2.Series[8].Points.Clear();
        this.chart2.Series[9].Points.Clear();
        this.chart2.Series[10].Points.Clear();
        this.chart2.Series[11].Points.Clear();

        n = Convert.ToInt16(AverageTimetextBox.Text);
        file = SavingtextBox.Text;
```

```
            short[] chaA = new short[no_of_samples];
            short[] chaB = new short[no_of_samples];
            short[] chaC = new short[no_of_samples];
            short[] chaD = new short[no_of_samples];

            short[] Ps2000ChaA = new short[Ps2000no_of_samples];
            short[] Ps2000ChaB = new short[Ps2000no_of_samples];
            short[] Ps2000ChaC = new short[Ps2000no_of_samples];
            short[] Ps2000ChaD = new short[Ps2000no_of_samples];

            double[] x = new double[no_of_samples];
            double[] Ps2000x = new double[Ps2000no_of_samples];



            //simulation mode

            double[] chasim = new double[no_of_samples];
            double[] xsim = new double[no_of_samples ];

            //dtsim = 1000;

            readSimulation(ref chasim, dtsim);

            //end

            if (OpenModecomboBox.SelectedIndex == 0)
            {
                for (int i = 0; i < no_of_samples; i++)
                {
                    xsim[i] = i * dtsim;

                    chaverage[i] = chaverage[i] + (chasim[i]) / n;

                    this.chart2.Series[0].Points.AddXY(xsim[i], chasim[i]);
                }

            }

            else if (OpenModecomboBox.SelectedIndex == 1)
            {
                p1.app(ref chaA, ref chaB, ref chaC, ref chaD, ref time_interval, no_of_samples,
timebase, channel_ext, channel_enable, threshold, direction, delay, 10, offset, timeoffset,
type);
            }

            for (int i = 0; i < no_of_samples; i++)
            {
                x[i] = i * time_interval;

                if (ChannelOnecheckBox.Checked == true)
                {
                    chaverageA[i] = chaverageA[i] + (chaA[i] * 1.0) / n;

                    this.chart2.Series[0].Points.AddXY(x[i], chaA[i]);
                }

                if (ChannelTwocheckBox.Checked == true)
                {
                    chaverageB[i] = chaverageB[i] + (chaB[i] * 1.0) / n;

                    this.chart2.Series[1].Points.AddXY(x[i], chaB[i]);
                }

                if (ChannelThreecheckBox.Checked == true)
                {
                    chaverageC[i] = chaverageC[i] + (chaC[i] * 1.0) / n;

                    this.chart2.Series[2].Points.AddXY(x[i], chaC[i]);
                }

                if (ChannelFourcheckBox.Checked == true)
```

```
                {
                    chaverageD[i] = chaverageD[i] + (chaD[i] * 1.0) / n;

                    this.chart2.Series[3].Points.AddXY(x[i], chaD[i]);
                }

            }

            p2.app(ref Ps2000ChaA, ref Ps2000ChaB, ref Ps2000ChaC, ref Ps2000ChaD,
Ps2000timebase, Ps2000no_of_samples, ref Ps2000time_interval, ref Ps2000time_units, ref
Ps2000max_samples, Ps2000channel_ext, Ps2000threshold, (short)Ps2000direction, Ps2000delay,
6, offset2000, timeoffset2000);

            for (int i = 0; i < Ps2000no_of_samples; i++)
            {
                Ps2000x[i] = i * Ps2000time_interval;

                if (Ps2000ChannelOnecheckBox.Checked == true)
                {
                    chaverage2000A[i] = chaverage2000A[i] + (Ps2000ChaA[i] * 1.0) / n;

                    this.chart2.Series[5].Points.AddXY(Ps2000x[i], Ps2000ChaA[i]);
                }

                if (Ps2000ChannelTwocheckBox.Checked == true)
                {
                    chaverage2000B[i] = chaverage2000B[i] + (Ps2000ChaB[i] * 1.0) / n;

                    this.chart2.Series[6].Points.AddXY(Ps2000x[i], Ps2000ChaB[i]);
                }
            }

                progressBar1.PerformStep();
            Processinglabel.Text = "Processing..." + counter;

            counter++;

            if (counter < n)
            {

            }

            else
            {

                xlWorkBook = xlApp.Workbooks.Open(file);

                if (OpenModecomboBox.SelectedIndex == 0)
                {
                    this.chart2.Series[0].Points.Clear();

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(7);

                    for (int i = 0; i < no_of_samples; i++)
                    {
                        this.chart2.Series[0].Points.AddXY(xsim[i], chaverage[i]);
                    }

                    for (int i = 0; i < no_of_samples; i++)
                    {
                        xlWorkSheet.Cells[i+13, 1] = chaverage[i];
                    }
                }

                else if (OpenModecomboBox.SelectedIndex == 1)
                {
                    this.chart2.Series[0].Points.Clear();
                    this.chart2.Series[1].Points.Clear();
                    this.chart2.Series[2].Points.Clear();
                    this.chart2.Series[3].Points.Clear();
                    this.chart2.Series[4].Points.Clear();
                    this.chart2.Series[5].Points.Clear();
                    this.chart2.Series[6].Points.Clear();
```

```
        if (ChannelOnecheckBox.Checked == true)
        {
            xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

            for (int i = 0; i < no_of_samples; i++)
            {
                this.chart2.Series[0].Points.AddXY(x[i], chaverageA[i]);
            }

            for (int i = 0; i < no_of_samples; i++)
            {
                xlWorkSheet.Cells[i + 13, 1] = chaverageA[i];
            }
        }

        if (ChannelTwocheckBox.Checked == true)
        {
            xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(2);

            for (int i = 0; i < no_of_samples; i++)
            {
                this.chart2.Series[1].Points.AddXY(x[i], chaverageB[i]);
            }

            for (int i = 0; i < no_of_samples; i++)
            {
                xlWorkSheet.Cells[i + 13, 1] = chaverageB[i];
            }
        }

        if (ChannelThreecheckBox.Checked == true)
        {
            xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(3);

            for (int i = 0; i < no_of_samples; i++)
            {
                this.chart2.Series[2].Points.AddXY(x[i], chaverageC[i]);
            }

            for (int i = 0; i < no_of_samples; i++)
            {
                xlWorkSheet.Cells[i + 13, 1] = chaverageC[i];
            }
        }

        if (ChannelFourcheckBox.Checked == true)
        {
            xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(4);

            for (int i = 0; i < no_of_samples; i++)
            {
                this.chart2.Series[3].Points.AddXY(x[i], chaverageD[i]);
            }

            for (int i = 0; i < no_of_samples; i++)
            {
                xlWorkSheet.Cells[i + 13, 1] = chaverageD[i];
            }
        }
    }

    if (Ps2000ChannelOnecheckBox.Checked == true)
    {
        xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(5);

        for (int i = 0; i < Ps2000no_of_samples; i++)
        {
            this.chart2.Series[4].Points.AddXY(Ps2000x[i], chaverage2000A[i]);
        }

        for (int i = 0; i < Ps2000no_of_samples; i++)
        {
```

```
                    xlWorkSheet.Cells[i + 13, 1] = chaverage2000A[i];
                }
            }

            if (Ps2000ChannelTwocheckBox.Checked == true)
            {
                xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(6);

                for (int i = 0; i < Ps2000no_of_samples; i++)
                {
                    this.chart2.Series[5].Points.AddXY(Ps2000x[i], chaverage2000B[i]);
                }

                for (int i = 0; i < Ps2000no_of_samples; i++)
                {
                    xlWorkSheet.Cells[i + 13, 1] = chaverage2000B[i];
                }
            }

            xlWorkBook.Save();
            xlWorkBook.Close();
            xlApp.Quit();

            if (SettingModecomboBox.SelectedIndex == 0)
            {
                this.chart3.Visible = true;
                this.chart4.Visible = false;

                if (ChannelOnecheckBox.Checked == true)
                {
                    fr1 = chaverageA;
                    time = time_interval * 0.000000001;
                    mw1 = (MWNumericArray)(den.DeNtest(fr1));
                    mw2 = (MWNumericArray)(fr.ffttest(fr1, time));

                    double[] arrAvg = new double[5];
                    arrAvg =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);

                    MFrequencylabel.Text = "Fre: " + "" + (int)arrAvg[0] + "Hz";
                    MAmplitudelabel.Text = "Amp: " + "" + arrAvg[1] + "mV";
                    MAmplitudeMlabel.Text = "Amp(-): " + "" + (int)arrAvg[2] + "mV";
                    MPtoPlabel.Text = "P to P: " + "" + (int)arrAvg[3] + "mV";
                    MRMSlabel.Text = "RMS: " + "" + (int)arrAvg[4] + "mV";
                    MSNRlabel.Text = "SNR: " + "" + mw1.ToString();

                    this.chart3.Series[1].Points.AddXY(count, (int)arrAvg[0]);
                    this.chart3.Series[2].Points.AddXY(count, (int)arrAvg[1]);
                    this.chart3.Series[3].Points.AddXY(count, (int)arrAvg[2]);
                    this.chart3.Series[4].Points.AddXY(count, (int)arrAvg[3]);
                    this.chart3.Series[5].Points.AddXY(count, (int)arrAvg[4]);
                    this.chart3.Series[6].Points.AddXY(count, mw1.ToString());

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = arrAvg[0] + "Hz";
                    xlWorkSheet.Cells[_lastRow, 4] = arrAvg[1] + "Mv";
```

```
                    xlWorkSheet.Cells[_lastRow, 5] = arrAvg[2] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 6] = arrAvg[3] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 7] = arrAvg[4] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 8] = mw1.ToString();

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (ChannelTwocheckBox.Checked == true)
                {
                    fr1 = chaverageB;
                    time = time_interval * 0.000000001;
                    mw1 = (MWNumericArray)(den.DeNtest(fr1));
                    mw2 = (MWNumericArray)(fr.ffttest(fr1, time));

                    double[] arrAvg =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);
                    MFrequencylabel.Text = "Fre: " + "" + (int)arrAvg[0] + "Hz";
                    MAmplitudelabel.Text = "Amp: " + "" + (int)arrAvg[1] + "mV";
                    MAmplitudeMlabel.Text = "Amp(-): " + "" + (int)arrAvg[2] + "mV";
                    MPtoPlabel.Text = "P to P: " + "" + (int)arrAvg[3] + "mV";
                    MRMSlabel.Text = "RMS: " + "" + (int)arrAvg[4] + "mV";
                    MSNRlabel.Text = "SNR: " + "" + mw1.ToString();

                    this.chart3.Series[1].Points.AddXY(count, (int)arrAvg[0]);
                    this.chart3.Series[2].Points.AddXY(count, (int)arrAvg[1]);
                    this.chart3.Series[3].Points.AddXY(count, (int)arrAvg[2]);
                    this.chart3.Series[4].Points.AddXY(count, (int)arrAvg[3]);
                    this.chart3.Series[5].Points.AddXY(count, (int)arrAvg[4]);
                    this.chart3.Series[6].Points.AddXY(count, mw1.ToString());

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(2);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = arrAvg[0] + "Hz";
                    xlWorkSheet.Cells[_lastRow, 4] = arrAvg[1] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 5] = arrAvg[2] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 6] = arrAvg[3] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 7] = arrAvg[4] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 8] = mw1.ToString();

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (ChannelThreecheckBox.Checked == true)
                {
                    fr1 = chaverageC;
                    time = time_interval * 0.000000001;
                    mw1 = (MWNumericArray)(den.DeNtest(fr1));
                    mw2 = (MWNumericArray)(fr.ffttest(fr1, time));

                    double[] arrAvg =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);
                    MFrequencylabel.Text = "Fre: " + "" + (int)arrAvg[0] + "Hz";
```

```
                    MAmplitudelabel.Text = "Amp: " + "" + (int)arrAvg[1] + "mV";
                    MAmplitudeMlabel.Text = "Amp(-): " + "" + (int)arrAvg[2] + "mV";
                    MPtoPlabel.Text = "P to P: " + "" + (int)arrAvg[3] + "mV";
                    MRMSlabel.Text = "RMS: " + "" + (int)arrAvg[4] + "mV";
                    MSNRlabel.Text = "SNR: " + "" + mw1.ToString();

                    this.chart3.Series[1].Points.AddXY(count, (int)arrAvg[0]);
                    this.chart3.Series[2].Points.AddXY(count, (int)arrAvg[1]);
                    this.chart3.Series[3].Points.AddXY(count, (int)arrAvg[2]);
                    this.chart3.Series[4].Points.AddXY(count, (int)arrAvg[3]);
                    this.chart3.Series[5].Points.AddXY(count, (int)arrAvg[4]);
                    this.chart3.Series[6].Points.AddXY(count, mw1.ToString());

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(3);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = arrAvg[0] + "Hz";
                    xlWorkSheet.Cells[_lastRow, 4] = arrAvg[1] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 5] = arrAvg[2] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 6] = arrAvg[3] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 7] = arrAvg[4] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 8] = mw1.ToString();

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (ChannelFourcheckBox.Checked == true)
                {
                    fr1 = chaverageD;
                    time = time_interval * 0.000000001;
                    mw1 = (MWNumericArray)(den.DeNtest(fr1));
                    mw2 = (MWNumericArray)(fr.ffttest(fr1, time));

                    double[] arrAvg =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);
                    MFrequencylabel.Text = "Fre: " + "" + (int)arrAvg[0] + "Hz";
                    MAmplitudelabel.Text = "Amp: " + "" + (int)arrAvg[1] + "mV";
                    MAmplitudeMlabel.Text = "Amp(-): " + "" + (int)arrAvg[2] + "mV";
                    MPtoPlabel.Text = "P to P: " + "" + (int)arrAvg[3] + "mV";
                    MRMSlabel.Text = "RMS: " + "" + (int)arrAvg[4] + "mV";
                    MSNRlabel.Text = "SNR: " + "" + mw1.ToString();

                    this.chart3.Series[1].Points.AddXY(count, (int)arrAvg[0]);
                    this.chart3.Series[2].Points.AddXY(count, (int)arrAvg[1]);
                    this.chart3.Series[3].Points.AddXY(count, (int)arrAvg[2]);
                    this.chart3.Series[4].Points.AddXY(count, (int)arrAvg[3]);
                    this.chart3.Series[5].Points.AddXY(count, (int)arrAvg[4]);
                    this.chart3.Series[6].Points.AddXY(count, mw1.ToString());

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(4);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
```

```
                                Excel.XlFindLookIn.xlFormulas,
                                Excel.XlLookAt.xlPart,
                                Excel.XlSearchOrder.xlByRows,
                                Excel.XlSearchDirection.xlPrevious,
                                Missing.Value,
                                Missing.Value,
                                Missing.Value
                                ).Row + 1;

                        xlWorkSheet.Cells[_lastRow, 1] = file;
                        xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                        xlWorkSheet.Cells[_lastRow, 3] = arrAvg[0] + "Hz";
                        xlWorkSheet.Cells[_lastRow, 4] = arrAvg[1] + "Mv";
                        xlWorkSheet.Cells[_lastRow, 5] = arrAvg[2] + "Mv";
                        xlWorkSheet.Cells[_lastRow, 6] = arrAvg[3] + "Mv";
                        xlWorkSheet.Cells[_lastRow, 7] = arrAvg[4] + "Mv";
                        xlWorkSheet.Cells[_lastRow, 8] = mw1.ToString();

                        xlWorkBook.Save();
                        xlWorkBook.Close();
                        xlApp.Quit();
                    }

                    if (Ps2000ChannelOnecheckBox.Checked == true)
                    {
                        fr1 = chaverage2000A;
                        time = Ps2000time_interval * 0.000000001;
                        mw1 = (MWNumericArray)(den.DeNtest(fr1));
                        mw2 = (MWNumericArray)(fr.ffttest(fr1, time));

                        double[] arrAvg =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);
                        MFrequencylabel.Text = "Fre: " + "" + (int)arrAvg[0] + "Hz";
                        MAmplitudelabel.Text = "Amp: " + "" + (int)arrAvg[1] + "mV";
                        MAmplitudeMlabel.Text = "Amp(-): " + "" + (int)arrAvg[2] + "mV";
                        MPtoPlabel.Text = "P to P: " + "" + (int)arrAvg[3] + "mV";
                        MRMSlabel.Text = "RMS: " + "" + (int)arrAvg[4] + "mV";
                        MSNRlabel.Text = "SNR: " + "" + mw1.ToString();

                        this.chart3.Series[1].Points.AddXY(count, (int)arrAvg[0]);
                        this.chart3.Series[2].Points.AddXY(count, (int)arrAvg[1]);
                        this.chart3.Series[3].Points.AddXY(count, (int)arrAvg[2]);
                        this.chart3.Series[4].Points.AddXY(count, (int)arrAvg[3]);
                        this.chart3.Series[5].Points.AddXY(count, (int)arrAvg[4]);
                        this.chart3.Series[6].Points.AddXY(count, mw1.ToString());

                        xlWorkBook = xlApp.Workbooks.Open(outputfile);

                        xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(5);

                        int _lastRow = xlWorkSheet.Cells.Find(
                                "*",
                                xlWorkSheet.Cells[1, 1],
                                Excel.XlFindLookIn.xlFormulas,
                                Excel.XlLookAt.xlPart,
                                Excel.XlSearchOrder.xlByRows,
                                Excel.XlSearchDirection.xlPrevious,
                                Missing.Value,
                                Missing.Value,
                                Missing.Value
                                ).Row + 1;

                        xlWorkSheet.Cells[_lastRow, 1] = file;
                        xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                        xlWorkSheet.Cells[_lastRow, 3] = arrAvg[0] + "Hz";
                        xlWorkSheet.Cells[_lastRow, 4] = arrAvg[1] + "Mv";
                        xlWorkSheet.Cells[_lastRow, 5] = arrAvg[2] + "Mv";
                        xlWorkSheet.Cells[_lastRow, 6] = arrAvg[3] + "Mv";
                        xlWorkSheet.Cells[_lastRow, 7] = arrAvg[4] + "Mv";
                        xlWorkSheet.Cells[_lastRow, 8] = mw1.ToString();

                        xlWorkBook.Save();
                        xlWorkBook.Close();
```

```
                    xlApp.Quit();
                }

                if (Ps2000ChannelTwocheckBox.Checked == true)
                {
                    fr1 = chaverage2000B;
                    time = Ps2000time_interval * 0.000000001;
                    mw1 = (MWNumericArray)(den.DeNtest(fr1));
                    mw2 = (MWNumericArray)(fr.ffttest(fr1, time));

                    double[] arrAvg =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);
                    MFrequencylabel.Text = "Fre: " + "" + (int)arrAvg[0] + "Hz";
                    MAmplitudelabel.Text = "Amp: " + "" + (int)arrAvg[1] + "mV";
                    MAmplitudeMlabel.Text = "Amp(-): " + "" + (int)arrAvg[2] + "mV";
                    MPtoPlabel.Text = "P to P: " + "" + (int)arrAvg[3] + "mV";
                    MRMSlabel.Text = "RMS: " + "" + (int)arrAvg[4] + "mV";
                    MSNRlabel.Text = "SNR: " + "" + mw1.ToString();

                    this.chart3.Series[1].Points.AddXY(count, (int)arrAvg[0]);
                    this.chart3.Series[2].Points.AddXY(count, (int)arrAvg[1]);
                    this.chart3.Series[3].Points.AddXY(count, (int)arrAvg[2]);
                    this.chart3.Series[4].Points.AddXY(count, (int)arrAvg[3]);
                    this.chart3.Series[5].Points.AddXY(count, (int)arrAvg[4]);
                    this.chart3.Series[6].Points.AddXY(count, mw1.ToString());

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(5);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = arrAvg[0] + "Hz";
                    xlWorkSheet.Cells[_lastRow, 4] = arrAvg[1] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 5] = arrAvg[2] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 6] = arrAvg[3] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 7] = arrAvg[4] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 8] = mw1.ToString();

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }
            }

            if (SettingModecomboBox.SelectedIndex == 1)
            {
                this.chart3.Visible = true;
                this.chart4.Visible = false;

                if (ChannelOnecheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 1, no_of_samples,
time_interval));

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    A = arrFit1[0];
```

```
                    tau = arrFit1[1];

                    double[] yf = new double[no_of_samples];

                    int po = (int)(c1.maxnumber(chaverageA, no_of_samples) - 1);

                    for (int i = 0; i < no_of_samples; i++)
                    {

                        if (i > po)
                        {
                            yf[i] = chaverageA[po + 1] * Math.Exp((x[i] - x[po + 1]) *
0.000000001 / tau) * Erfc(Math.Sqrt((x[i] - x[po + 1]) * 0.000000001 / tau));

                            this.chart2.Series[6].Points.AddXY(x[i], yf[i]);
                        }
                    }

                    double H = calculatehydration(tau);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (ChannelTwocheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 2, no_of_samples,
time_interval));

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    A = arrFit1[0];
                    tau = arrFit1[1];

                    double[] yf = new double[no_of_samples];

                    int po = (int)(c1.maxnumber(chaverageB, no_of_samples) - 1);

                    for (int i = 0; i < no_of_samples; i++)
                    {

                        if (i > po)
                        {
                            yf[i] = chaverageB[po + 1] * Math.Exp((x[i] - x[po + 1]) *
```

```
0.000000001 / tau) * Erfc(Math.Sqrt((x[i] - x[po + 1]) * 0.000000001 / tau));

                            this.chart2.Series[7].Points.AddXY(x[i], yf[i]);
                        }
                    }

                    double H = calculatehydration(tau);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(2);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (ChannelThreecheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 3, no_of_samples,
time_interval));

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    A = arrFit1[0];
                    tau = arrFit1[1];

                    double[] yf = new double[no_of_samples];

                    int po = (int)(c1.maxnumber(chaverageC, no_of_samples) - 1);

                    for (int i = 0; i < no_of_samples; i++)
                    {

                        if (i > po)
                        {
                            yf[i] = chaverageC[po + 1] * Math.Exp((x[i] - x[po + 1]) *
0.000000001 / tau) * Erfc(Math.Sqrt((x[i] - x[po + 1]) * 0.000000001 / tau));

                            this.chart2.Series[8].Points.AddXY(x[i], yf[i]);
                        }
                    }

                    double H = calculatehydration(tau);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(3);
```

```
                    int _lastRow = xlWorkSheet.Cells.Find(
                            "*",
                            xlWorkSheet.Cells[1, 1],
                            Excel.XlFindLookIn.xlFormulas,
                            Excel.XlLookAt.xlPart,
                            Excel.XlSearchOrder.xlByRows,
                            Excel.XlSearchDirection.xlPrevious,
                            Missing.Value,
                            Missing.Value,
                            Missing.Value
                            ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (ChannelFourcheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 4, no_of_samples,
time_interval));

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    A = arrFit1[0];
                    tau = arrFit1[1];

                    double[] yf = new double[no_of_samples];

                    int po = (int)(c1.maxnumber(chaverageD, no_of_samples) - 1);

                    for (int i = 0; i < no_of_samples; i++)
                    {

                        if (i > po)
                        {
                            yf[i] = chaverageD[po + 1] * Math.Exp((x[i] - x[po + 1]) *
0.000000001 / tau) * Erfc(Math.Sqrt((x[i] - x[po + 1]) * 0.000000001 / tau));

                            this.chart2.Series[9].Points.AddXY(x[i], yf[i]);
                        }
                    }

                    double H = calculatehydration(tau);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(4);

                    int _lastRow = xlWorkSheet.Cells.Find(
                            "*",
                            xlWorkSheet.Cells[1, 1],
                            Excel.XlFindLookIn.xlFormulas,
                            Excel.XlLookAt.xlPart,
                            Excel.XlSearchOrder.xlByRows,
                            Excel.XlSearchDirection.xlPrevious,
                            Missing.Value,
                            Missing.Value,
                            Missing.Value
                            ).Row + 1;
```

```
                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (Ps2000ChannelOnecheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 5,
Ps2000no_of_samples, Ps2000time_interval));

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    A = arrFit1[0];
                    tau = arrFit1[1];

                    double[] yf = new double[Ps2000no_of_samples];

                    int po = (int)(c1.maxnumber(chaverage2000A, Ps2000no_of_samples) - 1);

                    for (int i = 0; i < Ps2000no_of_samples; i++)
                    {

                        if (i > po)
                        {
                            yf[i] = chaverage2000A[po + 1] * Math.Exp((Ps2000x[i] -
Ps2000x[po + 1]) * 0.000000001 / tau) * Erfc(Math.Sqrt((Ps2000x[i] - Ps2000x[po + 1]) *
0.000000001 / tau));

                            this.chart2.Series[10].Points.AddXY(Ps2000x[i], yf[i]);
                        }
                    }

                    double H = calculatehydration(tau);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(5);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;
                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
```

```
                }

                if (Ps2000ChannelTwocheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 6,
Ps2000no_of_samples, Ps2000time_interval));

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    A = arrFit1[0];
                    tau = arrFit1[1];

                    double[] yf = new double[Ps2000no_of_samples];

                    int po = (int)(c1.maxnumber(chaverage2000B, Ps2000no_of_samples) - 1);

                    for (int i = 0; i < Ps2000no_of_samples; i++)
                    {

                        if (i > po)
                        {
                            yf[i] = chaverage2000B[po + 1] * Math.Exp((Ps2000x[i] -
Ps2000x[po + 1]) * 0.000000001 / tau) * Erfc(Math.Sqrt((Ps2000x[i] - Ps2000x[po + 1]) *
0.000000001 / tau));

                            this.chart2.Series[11].Points.AddXY(Ps2000x[i], yf[i]);
                        }
                    }

                    double H = calculatehydration(tau);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(5);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }
            }

            if (SettingModecomboBox.SelectedIndex == 2)
            {
                this.chart3.Visible = true;
                this.chart4.Visible = false;

                if (ChannelOnecheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit2(SavingtextBox.Text, 1, no_of_samples,
```

```
time_interval));

                        double[] arrFit2 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                        double A;
                        double tau;
                        double w;
                        A = arrFit2[0];
                        tau = arrFit2[1];
                        w = arrFit2[2] * 0.00000000001;

                        double[] yf = new double[no_of_samples];

                        int po = (int)(c1.maxnumber(chaverageA, no_of_samples) - 1);

                        for (int i = 0; i < no_of_samples; i++)
                        {
                            if (i > po)
                            {
                                yf[i] = chaverageA[po + 1] * (2 * w * Math.Sqrt((x[i] - x[po +
1]) * 0.000000001 * tau) / (Math.Sqrt(Math.PI) * (2 * w * (x[i] - x[po + 1]) * 0.000000001
+ 1))) + (1 / (Math.Sqrt(Math.Pow((2 * w * (x[i] - x[po + 1]) * 0.000000001 + 1), 3)))) *
Math.Exp((x[i] - x[po + 1]) * 0.000000001 / tau / (2 * w * (x[i] - x[po + 1]) * 0.000000001
+ 1)) * Erfc(Math.Sqrt((x[i] - x[po + 1]) * 0.000000001 / tau / (2 * w * (x[i] - x[po + 1])
* 0.000000001 + 1)));
                                this.chart2.Series[6].Points.AddXY(x[i], yf[i]);
                            }
                        }

                        double H = calculatehydration(tau);
                        double HW = w / D / (betaw - betad);
                        this.chart3.Series[0].Points.AddXY(count, H * 100);

                        xlWorkBook = xlApp.Workbooks.Open(outputfile);

                        xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

                        int _lastRow = xlWorkSheet.Cells.Find(
                                 "*",
                                 xlWorkSheet.Cells[1, 1],
                                 Excel.XlFindLookIn.xlFormulas,
                                 Excel.XlLookAt.xlPart,
                                 Excel.XlSearchOrder.xlByRows,
                                 Excel.XlSearchDirection.xlPrevious,
                                 Missing.Value,
                                 Missing.Value,
                                 Missing.Value
                                 ).Row + 1;

                        xlWorkSheet.Cells[_lastRow, 1] = file;
                        xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                        xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                        xlWorkSheet.Cells[_lastRow, 4] = HW;
                        xlWorkSheet.Cells[_lastRow, 5] = tau;
                        xlWorkSheet.Cells[_lastRow, 6] = w;
                        xlWorkSheet.Cells[_lastRow, 7] = A;

                        xlWorkBook.Save();
                        xlWorkBook.Close();
                        xlApp.Quit();
                    }

                    if (ChannelTwocheckBox.Checked == true)
                    {
                        mw = (MWNumericArray)(lsf.LSFit2(SavingtextBox.Text, 2, no_of_samples,
time_interval));

                        double[] arrFit2 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                        double A;
                        double tau;
```

```
                    double w;
                    A = arrFit2[0];
                    tau = arrFit2[1];
                    w = arrFit2[2] * 0.00000000001;

                    double[] yf = new double[no_of_samples];

                    int po = (int)(c1.maxnumber(chaverageB, no_of_samples) - 1);

                    for (int i = 0; i < no_of_samples; i++)
                    {
                        if (i > po)
                        {
                            yf[i] = chaverageB[po + 1] * (2 * w * Math.Sqrt((x[i] - x[po +
1]) * 0.000000001 * tau) / (Math.Sqrt(Math.PI) * (2 * w * (x[i] - x[po + 1]) * 0.000000001
+ 1))) + (1 / (Math.Sqrt(Math.Pow((2 * w * (x[i] - x[po + 1]) * 0.000000001 + 1), 3)))) *
Math.Exp((x[i] - x[po + 1]) * 0.000000001 / tau / (2 * w * (x[i] - x[po + 1]) * 0.000000001
+ 1)) * Erfc(Math.Sqrt((x[i] - x[po + 1]) * 0.000000001 / tau / (2 * w * (x[i] - x[po + 1])
* 0.000000001 + 1)));
                            this.chart2.Series[7].Points.AddXY(x[i], yf[i]);
                        }
                    }

                    double H = calculatehydration(tau);
                    double HW = w / D / (betaw - betad);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(2);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = HW;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = w;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (ChannelThreecheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit2(SavingtextBox.Text, 3, no_of_samples,
time_interval));

                    double[] arrFit2 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    double w;
                    A = arrFit2[0];
                    tau = arrFit2[1];
                    w = arrFit2[2] * 0.00000000001;

                    double[] yf = new double[no_of_samples];
```

```
                    int po = (int)(c1.maxnumber(chaverageC, no_of_samples) - 1);

                    for (int i = 0; i < no_of_samples; i++)
                    {
                        if (i > po)
                        {
                            yf[i] = chaverageC[po + 1] * (2 * w * Math.Sqrt((x[i] - x[po +
1]) * 0.000000001 * tau) / (Math.Sqrt(Math.PI) * (2 * w * (x[i] - x[po + 1]) * 0.000000001
+ 1))) + (1 / (Math.Sqrt(Math.Pow((2 * w * (x[i] - x[po + 1]) * 0.000000001 + 1), 3)))) *
Math.Exp((x[i] - x[po + 1]) * 0.000000001 / tau / (2 * w * (x[i] - x[po + 1]) * 0.000000001
+ 1)) * Erfc(Math.Sqrt((x[i] - x[po + 1]) * 0.000000001 / tau / (2 * w * (x[i] - x[po + 1])
* 0.000000001 + 1)));
                            this.chart2.Series[8].Points.AddXY(x[i], yf[i]);
                        }
                    }

                    double H = calculatehydration(tau);
                    double HW = w / D / (betaw - betad);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(3);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = HW;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = w;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (ChannelFourcheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit2(SavingtextBox.Text, 4, no_of_samples,
time_interval));

                    double[] arrFit2 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    double w;
                    A = arrFit2[0];
                    tau = arrFit2[1];
                    w = arrFit2[2] * 0.00000000001;

                    double[] yf = new double[no_of_samples];

                    int po = (int)(c1.maxnumber(chaverageD, no_of_samples) - 1);

                    for (int i = 0; i < no_of_samples; i++)
                    {
                        if (i > po)
                        {
                            yf[i] = chaverageD[po + 1] * (2 * w * Math.Sqrt((x[i] - x[po +
```

```
1]) * 0.000000001 * tau) / (Math.Sqrt(Math.PI) * (2 * w * (x[i] - x[po + 1]) * 0.000000001
+ 1))) + (1 / (Math.Sqrt(Math.Pow((2 * w * (x[i] - x[po + 1]) * 0.000000001 + 1), 3)))) *
Math.Exp((x[i] - xsim[po + 1]) * 0.000000001 / tau / (2 * w * (x[i] - x[po + 1]) * 0.000000001
+ 1)) * Erfc(Math.Sqrt((x[i] - x[po + 1]) * 0.000000001 / tau / (2 * w * (x[i] - x[po + 1])
* 0.000000001 + 1)));
                            this.chart2.Series[9].Points.AddXY(x[i], yf[i]);
                        }
                    }

                    double H = calculatehydration(tau);
                    double HW = w / D / (betaw - betad);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(4);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = HW;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = w;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (Ps2000ChannelOnecheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit2(SavingtextBox.Text, 5,
Ps2000no_of_samples, Ps2000time_interval));

                    double[] arrFit2 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    double w;
                    A = arrFit2[0];
                    tau = arrFit2[1];
                    w = arrFit2[2] * 0.00000000001;

                    double[] yf = new double[Ps2000no_of_samples];

                    int po = (int)(c1.maxnumber(chaverage2000A, Ps2000no_of_samples) - 1);

                    for (int i = 0; i < Ps2000no_of_samples; i++)
                    {
                        if (i > po)
                        {
                            yf[i] = chaverage2000A[po + 1] * (2 * w * Math.Sqrt((Ps2000x[i]
- Ps2000x[po + 1]) * 0.000000001 * tau) / (Math.Sqrt(Math.PI) * (2 * w * (Ps2000x[i] - Ps2000x[po
+ 1]) * 0.000000001 + 1))) + (1 / (Math.Sqrt(Math.Pow((2 * w * (Ps2000x[i] - Ps2000x[po + 1])
* 0.000000001 + 1), 3)))) * Math.Exp((Ps2000x[i] - Ps2000x[po + 1]) * 0.000000001 / tau / (2
* w * (Ps2000x[i] - Ps2000x[po + 1]) * 0.000000001 + 1)) * Erfc(Math.Sqrt((Ps2000x[i] -
Ps2000x[po + 1]) * 0.000000001 / tau / (2 * w * (Ps2000x[i] - Ps2000x[po + 1]) * 0.000000001
+ 1)));
                            this.chart2.Series[10].Points.AddXY(x[i], yf[i]);
```

```
                        }
                    }

                    double H = calculatehydration(tau);
                    double HW = w / D / (betaw - betad);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(5);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = HW;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = w;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (Ps2000ChannelTwocheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit2(SavingtextBox.Text, 6,
Ps2000no_of_samples, Ps2000time_interval));

                    double[] arrFit2 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    double w;
                    A = arrFit2[0];
                    tau = arrFit2[1];
                    w = arrFit2[2] * 0.00000000001;

                    double[] yf = new double[Ps2000no_of_samples];

                    int po = (int)(c1.maxnumber(chaverage2000B, Ps2000no_of_samples) - 1);

                    for (int i = 0; i < Ps2000no_of_samples; i++)
                    {
                        if (i > po)
                        {
                            yf[i] = chaverage2000B[po + 1] * (2 * w * Math.Sqrt((Ps2000x[i]
- Ps2000x[po + 1]) * 0.000000001 * tau) / (Math.Sqrt(Math.PI) * (2 * w * (Ps2000x[i] - Ps2000x[po
+ 1]) * 0.000000001 + 1))) + (1 / (Math.Sqrt(Math.Pow((2 * w * (Ps2000x[i] - Ps2000x[po + 1])
* 0.000000001 + 1), 3)))) * Math.Exp((Ps2000x[i] - Ps2000x[po + 1]) * 0.000000001 / tau / (2
* w * (Ps2000x[i] - Ps2000x[po + 1]) * 0.000000001 + 1)) * Erfc(Math.Sqrt((Ps2000x[i] -
Ps2000x[po + 1]) * 0.000000001 / tau / (2 * w * (Ps2000x[i] - Ps2000x[po + 1]) * 0.000000001
+ 1)));
                            this.chart2.Series[11].Points.AddXY(x[i], yf[i]);
                        }
                    }

                    double H = calculatehydration(tau);
                    double HW = w / D / (betaw - betad);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);
```

```
                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(6);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = HW;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = w;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }
            }

            if (SettingModecomboBox.SelectedIndex == 3)
            {
                this.chart3.Visible = false;
                this.chart4.Visible = true;

                if (ChannelOnecheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 1, no_of_samples,
time_interval));

                    double A;
                    double tau;
                    int t0, t1;
                    int step;
                    double t;

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    A = arrFit1[0];
                    tau = arrFit1[1];

                    beta = Math.Sqrt(1 / (tau * D));

                    double H = calculatehydration(tau);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;
```

```
                        xlWorkSheet.Cells[_lastRow, 1] = file;
                        xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                        xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                        xlWorkSheet.Cells[_lastRow, 4] = 0;
                        xlWorkSheet.Cells[_lastRow, 5] = tau;
                        xlWorkSheet.Cells[_lastRow, 6] = 0;
                        xlWorkSheet.Cells[_lastRow, 7] = A;
                        xlWorkSheet.Cells[_lastRow, 8] = 1;
                        xlWorkSheet.Cells[_lastRow, 9] = no_of_samples;

                        xlWorkBook.Save();
                        xlWorkBook.Close();
                        xlApp.Quit();

                        int po = (int)(c1.maxnumber(chaverageA, no_of_samples) - 1);

                        step = (no_of_samples - po) / 10;

                        double[] yd = new double[10];

                        for (int k = 0; k < 10; k++)
                        {
                             t0 = 0;
                             t1 = (k + 1) * step - 1;

                            mw1 = (MWNumericArray)(lsf.LSFitdeep(SavingtextBox.Text, 1, t0, t1,
no_of_samples, time_interval));

                            double[] arrFit2 =
(double[])((MWNumericArray)mw1).ToVector(MWArrayComponent.Real);

                            A = arrFit2[0];
                            tau = arrFit2[1];

                            H = calculatehydration(tau);

                            t = ((t1 - t0) / 2 + t0) * time_interval;

                            yd[k] = (2 / Math.Sqrt(Math.PI)) * (Math.Sqrt(t * 0.000000001 / tau)
/ (beta * Math.Exp(t * 0.000000001 / tau) * Erfc((Math.Sqrt(t * 0.000000001 / tau))))) - 2
* (t * 0.000000001 / tau) / beta;

                            this.chart4.Series[0].Points.AddXY(yd[k] * 1000000, H);

                            xlWorkBook = xlApp.Workbooks.Open(outputfile);

                            xlWorkSheet =
(Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

                            _lastRow = xlWorkSheet.Cells.Find(
                                    "*",
                                    xlWorkSheet.Cells[1, 1],
                                    Excel.XlFindLookIn.xlFormulas,
                                    Excel.XlLookAt.xlPart,
                                    Excel.XlSearchOrder.xlByRows,
                                    Excel.XlSearchDirection.xlPrevious,
                                    Missing.Value,
                                    Missing.Value,
                                    Missing.Value
                                    ).Row + 1;

                            xlWorkSheet.Cells[_lastRow, 10] = (H * 100) + "%";
                            xlWorkSheet.Cells[_lastRow, 11] = tau;
                            xlWorkSheet.Cells[_lastRow, 12] = A;
                            xlWorkSheet.Cells[_lastRow, 13] = t0;
                            xlWorkSheet.Cells[_lastRow, 14] = t1;
                            xlWorkSheet.Cells[_lastRow, 15] = yd[k];

                            xlWorkBook.Save();
                            xlWorkBook.Close();
                            xlApp.Quit();

                        }
```

```
                }

                if (ChannelTwocheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 2, no_of_samples,
time_interval));

                    double A;
                    double tau;
                    int t0, t1;
                    int step;
                    double t;

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    A = arrFit1[0];
                    tau = arrFit1[1];

                    beta = Math.Sqrt(1 / (tau * D));

                    double H = calculatehydration(tau);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(2);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;
                    xlWorkSheet.Cells[_lastRow, 8] = 1;
                    xlWorkSheet.Cells[_lastRow, 9] = no_of_samples;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();

                    int po = (int)(c1.maxnumber(chaverageB, no_of_samples) - 1);

                    step = (no_of_samples - po) / 10;

                    double[] yd = new double[10];

                    for (int k = 0; k < 10; k++)
                    {
                        t0 = 0;
                        t1 = (k + 1) * step - 1;

                        mw1 = (MWNumericArray)(lsf.LSFitdeep(SavingtextBox.Text, 2, t0, t1,
no_of_samples, time_interval));

                        double[] arrFit2 =
(double[])((MWNumericArray)mw1).ToVector(MWArrayComponent.Real);

                        A = arrFit2[0];
                        tau = arrFit2[1];
```

```
                        H = calculatehydration(tau);

                        t = ((t1 - t0) / 2 + t0) * time_interval;

                        yd[k] = (2 / Math.Sqrt(Math.PI)) * (Math.Sqrt(t * 0.000000001 / tau)
/ (beta * Math.Exp(t * 0.000000001 / tau) * Erfc((Math.Sqrt(t * 0.000000001 / tau))))) - 2
* (t * 0.000000001 / tau) / beta;

                        this.chart4.Series[0].Points.AddXY(yd[k] * 1000000, H);

                        xlWorkBook = xlApp.Workbooks.Open(outputfile);

                        xlWorkSheet =
(Excel.Worksheet)xlWorkBook.Worksheets.get_Item(2);

                        _lastRow = xlWorkSheet.Cells.Find(
                                "*",
                                xlWorkSheet.Cells[1, 1],
                                Excel.XlFindLookIn.xlFormulas,
                                Excel.XlLookAt.xlPart,
                                Excel.XlSearchOrder.xlByRows,
                                Excel.XlSearchDirection.xlPrevious,
                                Missing.Value,
                                Missing.Value,
                                Missing.Value
                                ).Row + 1;

                        xlWorkSheet.Cells[_lastRow, 10] = (H * 100) + "%";
                        xlWorkSheet.Cells[_lastRow, 11] = tau;
                        xlWorkSheet.Cells[_lastRow, 12] = A;
                        xlWorkSheet.Cells[_lastRow, 13] = t0;
                        xlWorkSheet.Cells[_lastRow, 14] = t1;
                        xlWorkSheet.Cells[_lastRow, 15] = yd[k];

                        xlWorkBook.Save();
                        xlWorkBook.Close();
                        xlApp.Quit();

                    }
                }

                if (ChannelThreecheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 3, no_of_samples,
time_interval));

                    double A;
                    double tau;
                    int t0, t1;
                    int step;
                    double t;

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    A = arrFit1[0];
                    tau = arrFit1[1];

                    beta = Math.Sqrt(1 / (tau * D));

                    double H = calculatehydration(tau);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(3);

                    int _lastRow = xlWorkSheet.Cells.Find(
                            "*",
                            xlWorkSheet.Cells[1, 1],
                            Excel.XlFindLookIn.xlFormulas,
                            Excel.XlLookAt.xlPart,
                            Excel.XlSearchOrder.xlByRows,
```

```
                                Excel.XlSearchDirection.xlPrevious,
                                Missing.Value,
                                Missing.Value,
                                Missing.Value
                                ).Row + 1;

                        xlWorkSheet.Cells[_lastRow, 1] = file;
                        xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                        xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                        xlWorkSheet.Cells[_lastRow, 4] = 0;
                        xlWorkSheet.Cells[_lastRow, 5] = tau;
                        xlWorkSheet.Cells[_lastRow, 6] = 0;
                        xlWorkSheet.Cells[_lastRow, 7] = A;
                        xlWorkSheet.Cells[_lastRow, 8] = 1;
                        xlWorkSheet.Cells[_lastRow, 9] = no_of_samples;

                        xlWorkBook.Save();
                        xlWorkBook.Close();
                        xlApp.Quit();

                        int po = (int)(c1.maxnumber(chaverageC, no_of_samples) - 1);

                        step = (no_of_samples - po) / 10;

                        double[] yd = new double[10];

                        for (int k = 0; k < 10; k++)
                        {
                            t0 = 0;
                            t1 = (k + 1) * step - 1;

                            mw1 = (MWNumericArray)(lsf.LSFitdeep(SavingtextBox.Text, 3, t0, t1,
no_of_samples, time_interval));

                            double[] arrFit2 =
(double[])((MWNumericArray)mw1).ToVector(MWArrayComponent.Real);

                            A = arrFit2[0];
                            tau = arrFit2[1];

                            H = calculatehydration(tau);

                            t = ((t1 - t0) / 2 + t0) * time_interval;

                            yd[k] = (2 / Math.Sqrt(Math.PI)) * (Math.Sqrt(t * 0.000000001 / tau)
/ (beta * Math.Exp(t * 0.000000001 / tau) * Erfc((Math.Sqrt(t * 0.000000001 / tau))))) - 2
* (t * 0.000000001 / tau) / beta;

                            this.chart4.Series[0].Points.AddXY(yd[k] * 1000000, H);

                            xlWorkBook = xlApp.Workbooks.Open(outputfile);

                            xlWorkSheet =
(Excel.Worksheet)xlWorkBook.Worksheets.get_Item(3);

                            _lastRow = xlWorkSheet.Cells.Find(
                                    "*",
                                    xlWorkSheet.Cells[1, 1],
                                    Excel.XlFindLookIn.xlFormulas,
                                    Excel.XlLookAt.xlPart,
                                    Excel.XlSearchOrder.xlByRows,
                                    Excel.XlSearchDirection.xlPrevious,
                                    Missing.Value,
                                    Missing.Value,
                                    Missing.Value
                                    ).Row + 1;

                            xlWorkSheet.Cells[_lastRow, 10] = (H * 100) + "%";
                            xlWorkSheet.Cells[_lastRow, 11] = tau;
                            xlWorkSheet.Cells[_lastRow, 12] = A;
                            xlWorkSheet.Cells[_lastRow, 13] = t0;
                            xlWorkSheet.Cells[_lastRow, 14] = t1;
                            xlWorkSheet.Cells[_lastRow, 15] = yd[k];
```

```
                        xlWorkBook.Save();
                        xlWorkBook.Close();
                        xlApp.Quit();
                    }
                }

                if (ChannelFourcheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 4, no_of_samples,
time_interval));

                    double A;
                    double tau;
                    int t0, t1;
                    int step;
                    double t;

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    A = arrFit1[0];
                    tau = arrFit1[1];

                    beta = Math.Sqrt(1 / (tau * D));

                    double H = calculatehydration(tau);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(4);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;
                    xlWorkSheet.Cells[_lastRow, 8] = 1;
                    xlWorkSheet.Cells[_lastRow, 9] = no_of_samples;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();

                    int po = (int)(c1.maxnumber(chaverageD, no_of_samples) - 1);

                    step = (no_of_samples - po) / 10;

                    double[] yd = new double[10];

                    for (int k = 0; k < 10; k++)
                    {
                        t0 = 0;
                        t1 = (k + 1) * step - 1;

                        mw1 = (MWNumericArray)(lsf.LSFitdeep(SavingtextBox.Text, 4, t0, t1,
no_of_samples, time_interval));

                        double[] arrFit2 =
(double[])((MWNumericArray)mw1).ToVector(MWArrayComponent.Real);
```

```
                        A = arrFit2[0];
                        tau = arrFit2[1];

                        H = calculatehydration(tau);

                        t = ((t1 - t0) / 2 + t0) * time_interval;

                        yd[k] = (2 / Math.Sqrt(Math.PI)) * (Math.Sqrt(t * 0.000000001 / tau)
/ (beta * Math.Exp(t * 0.000000001 / tau) * Erfc((Math.Sqrt(t * 0.000000001 / tau))))) - 2
* (t * 0.000000001 / tau) / beta;

                        this.chart4.Series[0].Points.AddXY(yd[k] * 1000000, H);

                        xlWorkBook = xlApp.Workbooks.Open(outputfile);

                        xlWorkSheet =
(Excel.Worksheet)xlWorkBook.Worksheets.get_Item(4);

                        _lastRow = xlWorkSheet.Cells.Find(
                                "*",
                                xlWorkSheet.Cells[1, 1],
                                Excel.XlFindLookIn.xlFormulas,
                                Excel.XlLookAt.xlPart,
                                Excel.XlSearchOrder.xlByRows,
                                Excel.XlSearchDirection.xlPrevious,
                                Missing.Value,
                                Missing.Value,
                                Missing.Value
                                ).Row + 1;

                        xlWorkSheet.Cells[_lastRow, 10] = (H * 100) + "%";
                        xlWorkSheet.Cells[_lastRow, 11] = tau;
                        xlWorkSheet.Cells[_lastRow, 12] = A;
                        xlWorkSheet.Cells[_lastRow, 13] = t0;
                        xlWorkSheet.Cells[_lastRow, 14] = t1;
                        xlWorkSheet.Cells[_lastRow, 15] = yd[k];

                        xlWorkBook.Save();
                        xlWorkBook.Close();
                        xlApp.Quit();
                    }
                }

                if (Ps2000ChannelOnecheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 5,
Ps2000no_of_samples, Ps2000time_interval));

                    double A;
                    double tau;
                    int t0, t1;
                    int step;
                    double t;

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    A = arrFit1[0];
                    tau = arrFit1[1];

                    beta = Math.Sqrt(1 / (tau * D));

                    double H = calculatehydration(tau);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(5);

                    int _lastRow = xlWorkSheet.Cells.Find(
                            "*",
                            xlWorkSheet.Cells[1, 1],
                            Excel.XlFindLookIn.xlFormulas,
```

```
                                Excel.XlLookAt.xlPart,
                                Excel.XlSearchOrder.xlByRows,
                                Excel.XlSearchDirection.xlPrevious,
                                Missing.Value,
                                Missing.Value,
                                Missing.Value
                                ).Row + 1;

                        xlWorkSheet.Cells[_lastRow, 1] = file;
                        xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                        xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                        xlWorkSheet.Cells[_lastRow, 4] = 0;
                        xlWorkSheet.Cells[_lastRow, 5] = tau;
                        xlWorkSheet.Cells[_lastRow, 6] = 0;
                        xlWorkSheet.Cells[_lastRow, 7] = A;
                        xlWorkSheet.Cells[_lastRow, 8] = 1;
                        xlWorkSheet.Cells[_lastRow, 9] = Ps2000no_of_samples;

                        xlWorkBook.Save();
                        xlWorkBook.Close();
                        xlApp.Quit();

                        int po = (int)(c1.maxnumber(chaverage2000A, Ps2000no_of_samples) - 1);

                        step = (Ps2000no_of_samples - po) / 10;

                        double[] yd = new double[10];

                        for (int k = 0; k < 10; k++)
                        {
                            t0 = 0;
                            t1 = (k + 1) * step - 1;

                            mw1 = (MWNumericArray)(lsf.LSFitdeep(SavingtextBox.Text, 5, t0, t1,
Ps2000no_of_samples, Ps2000time_interval));

                            double[] arrFit2 =
(double[])((MWNumericArray)mw1).ToVector(MWArrayComponent.Real);

                            A = arrFit2[0];
                            tau = arrFit2[1];

                            H = calculatehydration(tau);

                            t = ((t1 - t0) / 2 + t0) * Ps2000time_interval;

                            yd[k] = (2 / Math.Sqrt(Math.PI)) * (Math.Sqrt(t * 0.000000001 / tau)
/ (beta * Math.Exp(t * 0.000000001 / tau) * Erfc((Math.Sqrt(t * 0.000000001 / tau))))) - 2
* (t * 0.000000001 / tau) / beta;

                            this.chart4.Series[0].Points.AddXY(yd[k] * 1000000, H);

                            xlWorkBook = xlApp.Workbooks.Open(outputfile);

                            xlWorkSheet =
(Excel.Worksheet)xlWorkBook.Worksheets.get_Item(5);

                            _lastRow = xlWorkSheet.Cells.Find(
                                    "*",
                                    xlWorkSheet.Cells[1, 1],
                                    Excel.XlFindLookIn.xlFormulas,
                                    Excel.XlLookAt.xlPart,
                                    Excel.XlSearchOrder.xlByRows,
                                    Excel.XlSearchDirection.xlPrevious,
                                    Missing.Value,
                                    Missing.Value,
                                    Missing.Value
                                    ).Row + 1;

                            xlWorkSheet.Cells[_lastRow, 10] = (H * 100) + "%";
                            xlWorkSheet.Cells[_lastRow, 11] = tau;
                            xlWorkSheet.Cells[_lastRow, 12] = A;
                            xlWorkSheet.Cells[_lastRow, 13] = t0;
```

```
                        xlWorkSheet.Cells[_lastRow, 14] = t1;
                        xlWorkSheet.Cells[_lastRow, 15] = yd[k];

                        xlWorkBook.Save();
                        xlWorkBook.Close();
                        xlApp.Quit();

                    }
                }

                if (Ps2000ChannelTwocheckBox.Checked == true)
                {
                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 6,
Ps2000no_of_samples, Ps2000time_interval));

                    double A;
                    double tau;
                    int t0, t1;
                    int step;
                    double t;

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    A = arrFit1[0];
                    tau = arrFit1[1];

                    beta = Math.Sqrt(1 / (tau * D));

                    double H = calculatehydration(tau);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(6);

                    int _lastRow = xlWorkSheet.Cells.Find(
                            "*",
                            xlWorkSheet.Cells[1, 1],
                            Excel.XlFindLookIn.xlFormulas,
                            Excel.XlLookAt.xlPart,
                            Excel.XlSearchOrder.xlByRows,
                            Excel.XlSearchDirection.xlPrevious,
                            Missing.Value,
                            Missing.Value,
                            Missing.Value
                            ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;
                    xlWorkSheet.Cells[_lastRow, 8] = 1;
                    xlWorkSheet.Cells[_lastRow, 9] = Ps2000no_of_samples;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();

                    int po = (int)(c1.maxnumber(chaverage2000B, Ps2000no_of_samples) - 1);

                    step = (Ps2000no_of_samples - po) / 10;

                    double[] yd = new double[10];

                    for (int k = 0; k < 10; k++)
                    {
                        t0 = 0;
                        t1 = (k + 1) * step - 1;

                       // t0 = k * step;
```

```
                        mw1 = (MWNumericArray)(lsf.LSFitdeep(SavingtextBox.Text, 6, t0, t1,
Ps2000no_of_samples, Ps2000time_interval));

                        double[] arrFit2 =
(double[])((MWNumericArray)mw1).ToVector(MWArrayComponent.Real);

                        A = arrFit2[0];
                        tau = arrFit2[1];

                        H = calculatehydration(tau);

                        t = ((t1 - t0) / 2 + t0) * Ps2000time_interval;

                        yd[k] = (2 / Math.Sqrt(Math.PI)) * (Math.Sqrt(t * 0.000000001 / tau)
/ (beta * Math.Exp(t * 0.000000001 / tau) * Erfc((Math.Sqrt(t * 0.000000001 / tau))))) - 2
* (t * 0.000000001 / tau) / beta;

                        this.chart4.Series[0].Points.AddXY(yd[k] * 1000000, H);

                        xlWorkBook = xlApp.Workbooks.Open(outputfile);

                        xlWorkSheet =
(Excel.Worksheet)xlWorkBook.Worksheets.get_Item(6);

                        _lastRow = xlWorkSheet.Cells.Find(
                                "*",
                                xlWorkSheet.Cells[1, 1],
                                Excel.XlFindLookIn.xlFormulas,
                                Excel.XlLookAt.xlPart,
                                Excel.XlSearchOrder.xlByRows,
                                Excel.XlSearchDirection.xlPrevious,
                                Missing.Value,
                                Missing.Value,
                                Missing.Value
                                ).Row + 1;

                        xlWorkSheet.Cells[_lastRow, 10] = (H * 100) + "%";
                        xlWorkSheet.Cells[_lastRow, 11] = tau;
                        xlWorkSheet.Cells[_lastRow, 12] = A;
                        xlWorkSheet.Cells[_lastRow, 13] = t0;
                        xlWorkSheet.Cells[_lastRow, 14] = t1;
                        xlWorkSheet.Cells[_lastRow, 15] = yd[k];

                        xlWorkBook.Save();
                        xlWorkBook.Close();
                        xlApp.Quit();
                    }
                }
            }

            if (SettingModecomboBox.SelectedIndex == 4)
            {
                this.chart3.Visible = true;
                this.chart4.Visible = false;

                mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 7, no_of_samples,
dtsim));

                double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                double A;
                double tau;
                A = arrFit1[0];
                tau = arrFit1[1];

                double[] yf = new double[no_of_samples];

                int po = (int)(c1.maxnumber(chaverage, no_of_samples) - 1);

                for (int i = 0; i < no_of_samples; i++)
                {
                    if (i > po)
```

```
                    {
                        yf[i] = chaverage[po + 1] * Math.Exp((xsim[i] - xsim[po + 1]) *
0.000000001 / tau) * Erfc(Math.Sqrt((xsim[i] - xsim[po + 1]) * 0.000000001 / tau));

                        this.chart2.Series[6].Points.AddXY(xsim[i], yf[i]);
                    }
                }

                double H = calculatehydration(tau);
                this.chart3.Series[0].Points.AddXY(count, H * 100);

                xlWorkBook = xlApp.Workbooks.Open(outputfile);

                xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(7);

                int _lastRow = xlWorkSheet.Cells.Find(
                         "*",
                         xlWorkSheet.Cells[1, 1],
                         Excel.XlFindLookIn.xlFormulas,
                         Excel.XlLookAt.xlPart,
                         Excel.XlSearchOrder.xlByRows,
                         Excel.XlSearchDirection.xlPrevious,
                         Missing.Value,
                         Missing.Value,
                         Missing.Value
                         ).Row + 1;

                xlWorkSheet.Cells[_lastRow, 1] = file;
                xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                xlWorkSheet.Cells[_lastRow, 3] = (H*100)+"%";
                xlWorkSheet.Cells[_lastRow, 4] = 0;
                xlWorkSheet.Cells[_lastRow, 5] = tau;
                xlWorkSheet.Cells[_lastRow, 6] = 0;
                xlWorkSheet.Cells[_lastRow, 7] = A;
                xlWorkBook.Save();
                xlWorkBook.Close();
                xlApp.Quit();
            }

            if (SettingModecomboBox.SelectedIndex == 5)
            {
                this.chart3.Visible = true;
                this.chart4.Visible = false;

                mw = (MWNumericArray)(lsf.LSFit2(SavingtextBox.Text, 7, no_of_samples,
dtsim));
                double[] arrFit2 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                double A;
                double tau;
                double w;
                A = arrFit2[0];
                tau = arrFit2[1];
                w = arrFit2[2]*0.00000000001;

                double[] yf = new double[no_of_samples];

                int po = (int)(c1.maxnumber(chaverage, no_of_samples) - 1);

                for (int i = 0; i < no_of_samples; i++)
                {
                    if (i > po)
                    {
                        yf[i] = chaverage[po + 1] * (2 * w * Math.Sqrt((xsim[i] - xsim[po
+ 1]) * 0.000000001 * tau) / (Math.Sqrt(Math.PI) * (2 * w * (xsim[i] - xsim[po + 1]) * 0.000000001
+ 1))) + (1 / (Math.Sqrt(Math.Pow((2 * w * (xsim[i] - xsim[po + 1]) * 0.000000001 + 1), 3))))
* Math.Exp((xsim[i] - xsim[po + 1]) * 0.000000001 / tau / (2 * w * (xsim[i] - xsim[po + 1])
* 0.000000001 + 1)) * Erfc(Math.Sqrt((xsim[i] - xsim[po + 1]) * 0.000000001 / tau / (2 * w
* (xsim[i] - xsim[po + 1]) * 0.000000001 + 1)));
                        this.chart2.Series[6].Points.AddXY(xsim[i], yf[i]);
                    }
                }
```

```
                    double H = calculatehydration(tau);
                    double HW = w / D / (betaw-betad);
                    this.chart3.Series[0].Points.AddXY(count, H * 100);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(7);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = HW;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = w;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }

                if (SettingModecomboBox.SelectedIndex == 6)
                {
                    this.chart3.Visible = false;
                    this.chart4.Visible = true;

                    mw = (MWNumericArray)(lsf.LSFit1(SavingtextBox.Text, 7, no_of_samples,
dtsim));

                    double A;
                    double tau;
                    int t0, t1;
                    int step;
                    double t;

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    A = arrFit1[0];
                    tau = arrFit1[1];

                    beta = Math.Sqrt(1 / (tau * D));

                    double H = calculatehydration(tau);

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(7);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;
```

```
                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;
                    xlWorkSheet.Cells[_lastRow, 8] = 1;
                    xlWorkSheet.Cells[_lastRow, 9] = no_of_samples;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();

                    int po = (int)(c1.maxnumber(chaverage, no_of_samples) - 1);

                    step = (no_of_samples - po) / 10;

                    double[] yd = new double[10];

                    for (int k = 0; k < 10; k++)
                    {
                        t0 = 0;
                        t1 = (k + 1) * step - 1;

                        mw1 = (MWNumericArray)(lsf.LSFitdeep(SavingtextBox.Text, 7, t0, t1,
no_of_samples, dtsim));

                        double[] arrFit2 =
(double[])((MWNumericArray)mw1).ToVector(MWArrayComponent.Real);

                        A = arrFit2[0];
                        tau = arrFit2[1];

                        H = calculatehydration(tau);

                        t = ((t1 - t0) / 2 + t0) * dtsim;

                        yd[k] = (2 / Math.Sqrt(Math.PI)) * (Math.Sqrt(t*0.000000001 / tau) /
(beta * Math.Exp(t*0.000000001 / tau) * Erfc((Math.Sqrt(t*0.000000001 / tau))))) - 2 *
(t*0.000000001 / tau) / beta;

                        this.chart4.Series[0].Points.AddXY(yd[k]*1000000, H);

                        xlWorkBook = xlApp.Workbooks.Open(outputfile);

                        xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(7);

                         _lastRow = xlWorkSheet.Cells.Find(
                                 "*",
                                 xlWorkSheet.Cells[1, 1],
                                 Excel.XlFindLookIn.xlFormulas,
                                 Excel.XlLookAt.xlPart,
                                 Excel.XlSearchOrder.xlByRows,
                                 Excel.XlSearchDirection.xlPrevious,
                                 Missing.Value,
                                 Missing.Value,
                                 Missing.Value
                                 ).Row + 1;

                        xlWorkSheet.Cells[_lastRow, 10] = (H * 100) + "%";
                        xlWorkSheet.Cells[_lastRow, 11] = tau;
                        xlWorkSheet.Cells[_lastRow, 12] = A;
                        xlWorkSheet.Cells[_lastRow, 13] = t0;
                        xlWorkSheet.Cells[_lastRow, 14] = t1;
                        xlWorkSheet.Cells[_lastRow, 15] = yd[k];

                        xlWorkBook.Save();
                        xlWorkBook.Close();
                        xlApp.Quit();

                    }
```

```
                }

                double max = c1.maximum(chaverage, chaverage.Length - 1);
                double min = c1.minimum(chaverage, chaverage.Length - 1);

                MAmplitudelabel.Text = "Amp: " + "" + max + "mV";
                MAmplitudeMlabel.Text = "Amp(-): " + "" + min + "mV";

                for (int i = 0; i < n; i++)
                {

                }

                Processinglabel.Text = "Complete!";
                Measurementtimer.Enabled = false;
            }
        }

        //measurement part end

        //write file part

        private void Choosebutton1_Click(object sender, EventArgs e)
        {
            SaveFileDialog ofd = new SaveFileDialog();

            ofd.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            ofd.Filter = "ExcelFile(*.xlsx)|*.xlsx|AllFiles(*.*)|*.*";
            if (ofd.ShowDialog(this) == DialogResult.OK)
            {
                SavingtextBox.Text = ofd.FileName;

            }
        }

        private void Choosebutton2_Click(object sender, EventArgs e)
        {
            SaveFileDialog sfd = new SaveFileDialog();

            sfd.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            sfd.Filter = "ExcelFile(*.xlsx)|*.xlsx|AllFiles(*.*)|*.*";
            if (sfd.ShowDialog(this) == DialogResult.OK)
            {
                OutputFiletextBox.Text = sfd.FileName;

            }
        }

        private void OpenDataFilebutton_Click(object sender, EventArgs e)
        {
            this.chart2.ChartAreas[0].AxisX.Title = "Time[ns]";
            this.chart2.ChartAreas[0].AxisY.Title = "Voltage[mV]";

            this.chart3.ChartAreas[0].AxisX.Title = "Time Count";
            this.chart3.ChartAreas[0].AxisY.Title = "Results";

            this.chart4.ChartAreas[0].AxisX.Title = "Depth[um]";
            this.chart4.ChartAreas[0].AxisY.Title = "Hydration[%]";

            this.chart2.Series[0].IsVisibleInLegend = false;
            this.chart2.Series[1].IsVisibleInLegend = false;
            this.chart2.Series[2].IsVisibleInLegend = false;
            this.chart2.Series[3].IsVisibleInLegend = false;
            this.chart2.Series[4].IsVisibleInLegend = false;
            this.chart2.Series[5].IsVisibleInLegend = false;
            this.chart2.Series[6].IsVisibleInLegend = false;
            this.chart2.Series[7].IsVisibleInLegend = false;
            this.chart2.Series[8].IsVisibleInLegend = false;
            this.chart2.Series[9].IsVisibleInLegend = false;
            this.chart2.Series[10].IsVisibleInLegend = false;
```

```
            this.chart2.Series[11].IsVisibleInLegend = false;

            this.chart3.Series[0].IsVisibleInLegend = false;
            this.chart3.Series[1].IsVisibleInLegend = false;
            this.chart3.Series[2].IsVisibleInLegend = false;
            this.chart3.Series[3].IsVisibleInLegend = false;
            this.chart3.Series[4].IsVisibleInLegend = false;
            this.chart3.Series[5].IsVisibleInLegend = false;
            this.chart3.Series[6].IsVisibleInLegend = false;

            this.chart4.Series[0].IsVisibleInLegend = false;
            this.chart4.Series[1].IsVisibleInLegend = false;
            this.chart4.Series[2].IsVisibleInLegend = false;
            this.chart4.Series[3].IsVisibleInLegend = false;
            this.chart4.Series[4].IsVisibleInLegend = false;


            this.chart2.Series[0].Points.Clear();
            this.chart2.Series[1].Points.Clear();
            this.chart2.Series[2].Points.Clear();
            this.chart2.Series[3].Points.Clear();
            this.chart2.Series[4].Points.Clear();
            this.chart2.Series[5].Points.Clear();
            this.chart2.Series[6].Points.Clear();
            this.chart2.Series[7].Points.Clear();
            this.chart2.Series[8].Points.Clear();
            this.chart2.Series[9].Points.Clear();
            this.chart2.Series[10].Points.Clear();
            this.chart2.Series[11].Points.Clear();

            this.chart3.Series[0].Points.Clear();
            this.chart3.Series[1].Points.Clear();
            this.chart3.Series[2].Points.Clear();
            this.chart3.Series[3].Points.Clear();
            this.chart3.Series[4].Points.Clear();
            this.chart3.Series[5].Points.Clear();
            this.chart3.Series[6].Points.Clear();

            this.chart4.Series[0].Points.Clear();
            this.chart4.Series[1].Points.Clear();
            this.chart4.Series[2].Points.Clear();
            this.chart4.Series[3].Points.Clear();
            this.chart4.Series[4].Points.Clear();

            string[] filename = new string[12];

            double[] ya = new double[6000];

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                filename = openFileDialog1.FileNames;

               if (SettingModecomboBox.SelectedIndex == 0)
               {
                    this.chart3.Visible = true;
                    this.chart4.Visible = false;

                   for (int i = 0; i < openFileDialog1.FileNames.Length; i++)
                   {
                       this.chart2.Series[i].IsVisibleInLegend = true;
                       this.chart2.Series[i+6].IsVisibleInLegend = true;

                       this.chart3.Series[i+1].IsVisibleInLegend = true;

                       xlApp = new Microsoft.Office.Interop.Excel.Application();
                       xlWorkBook = xlApp.Workbooks.Open(filename[i]);

                       xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.get_Item(1);
                       int rowsint =
xlWorkSheet.Range["A65535"].End[Excel.XlDirection.xlUp].Row;

                       Microsoft.Office.Interop.Excel.Range rng =
xlWorkSheet.Cells.get_Range("A11", "A" + rowsint);
```

```
                    object[,] values = (object[,])rng.Cells.Value;

                    string[,] s1 = new string[rowsint - 11, 1];

                    for (int j = 1; j <= rowsint - 11; j++)
                    {
                        s1[j - 1, 0] = values[j, 1].ToString();
                    }

                        count++;

                    double[] x = new double[Convert.ToInt32(s1[1 , 0])];

                    for (int p = 2; p < Convert.ToDouble(s1[1, 0]); p++)
                    {
                        ya[p - 2] = Convert.ToDouble(s1[p, 0]);
                    }

                    fr1 = ya;
                    time = Convert.ToDouble(s1[0 , 0]);
                    mw1 = (MWNumericArray)(den.DeNtest(fr1));
                    mw2 = (MWNumericArray)(fr.ffttest(fr1, time));

                    double[] arrAvg =
(double[])((MWNumericArray)mw2).ToVector(MWArrayComponent.Real);

                    this.chart3.Series[1].Points.AddXY(count, (int)arrAvg[0]);
                    this.chart3.Series[2].Points.AddXY(count, (int)arrAvg[1]);
                    this.chart3.Series[3].Points.AddXY(count, (int)arrAvg[2]);
                    this.chart3.Series[4].Points.AddXY(count, (int)arrAvg[3]);
                    this.chart3.Series[5].Points.AddXY(count, (int)arrAvg[4]);
                    this.chart3.Series[6].Points.AddXY(count, mw1.ToString());

                    for (int k = 2; k < Convert.ToInt32(s1[1, 0]); k++)
                    {
                        x[k] = (k - 2) * Convert.ToDouble(s1[0, 0]);
                        this.chart2.Series[i].Points.AddXY(x[k], s1[k, 0]);
                    }

                    outputfile = OutputFiletextBox.Text;

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = file;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = arrAvg[0] + "Hz";
                    xlWorkSheet.Cells[_lastRow, 4] = arrAvg[1] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 5] = arrAvg[2] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 6] = arrAvg[3] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 7] = arrAvg[4] + "Mv";
                    xlWorkSheet.Cells[_lastRow, 8] = mw1.ToString();

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();

                }
            }
```

```
            if (SettingModecomboBox.SelectedIndex == 4)
            {
                this.chart3.Visible = true;
                this.chart4.Visible = false;

                for (int i = 0; i < openFileDialog1.FileNames.Length; i++)
                {
                    this.chart2.Series[i].IsVisibleInLegend = true;
                    this.chart2.Series[i + 6].IsVisibleInLegend = true;

                    this.chart3.Series[0].IsVisibleInLegend = true;

                    xlApp = new Microsoft.Office.Interop.Excel.Application();
                    xlWorkBook = xlApp.Workbooks.Open(filename[i]);

                    xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.get_Item(1);

                    int rowsint =
xlWorkSheet.Range["A65535"].End[Excel.XlDirection.xlUp].Row;

                    Microsoft.Office.Interop.Excel.Range rng =
xlWorkSheet.Cells.get_Range("A9", "A" + rowsint);
                    object[,] values = (object[,])rng.Cells.Value;

                    string[,] s1 = new string[rowsint - 9 , 1];

                    for (int j = 1; j <= rowsint - 9; j++)
                    {
                        s1[j - 1 , 0] = values[j , 1].ToString();
                    }


                    count++;

                    mw = (MWNumericArray)(lsf.LSFit1(filename[i], 1,
Convert.ToDouble(s1[1 , 0]), (Convert.ToDouble(s1[0 , 0])*1000000000)));

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    A = arrFit1[0];
                    tau = arrFit1[1];

                    double[] yf = new double[Convert.ToInt32(s1[1 , 0])];
                    double[] x = new double[Convert.ToInt32(s1[1 , 0])];

                    double[] Hy = new double[openFileDialog1.FileNames.Length];

                    for (int p = 4; p < Convert.ToDouble(s1[1 , 0]); p++)
                    {
                        ya[p - 4] = Convert.ToDouble(s1[p , 0]);
                    }

                    int po = (int)(c1.maxnumber(ya, s1.Length - 1) - 1);

                    for (int z = 0; z < Convert.ToInt32(s1[1 , 0]); z++)
                    {
                        x[z] = z * Convert.ToDouble(s1[0 , 0]);
                        if (z > po)
                        {
                            yf[z] = Convert.ToDouble(s1[po+5, 0]) * Math.Exp((x[z] - x[po
+ 1]) / tau) * Erfc(Math.Sqrt((x[z] - x[po + 1]) / tau));

                            this.chart2.Series[i+6].Points.AddXY(x[z], yf[z]);
                        }

                    }

                    for (int k = 4; k < Convert.ToInt32(s1[1 , 0]); k++)
                    {
                        x[k] = (k - 4 ) * Convert.ToDouble(s1[0 , 0]);
```

```
                        this.chart2.Series[i].Points.AddXY(x[k], Convert.ToDouble(s1[k ,
0]));
                    }

                    double H = calculatehydration(tau);
                    Hy[i] = H;

                    this.chart3.Series[0].Points.AddXY(count, Hy[i]);

                    outputfile = OutputFiletextBox.Text;

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(7);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = filename;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }
            }

            if (SettingModecomboBox.SelectedIndex == 5)
            {
                this.chart3.Visible = true;
                this.chart4.Visible = false;

                for (int i = 0; i < openFileDialog1.FileNames.Length; i++)
                {
                    this.chart2.Series[i].IsVisibleInLegend = true;
                    this.chart2.Series[i + 5].IsVisibleInLegend = true;

                    this.chart3.Series[0].IsVisibleInLegend = true;

                    xlApp = new Microsoft.Office.Interop.Excel.Application();
                    xlWorkBook = xlApp.Workbooks.Open(filename[i]);

                    xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.get_Item(1);

                    int rowsint =
xlWorkSheet.Range["A65535"].End[Excel.XlDirection.xlUp].Row;

                    Microsoft.Office.Interop.Excel.Range rng =
xlWorkSheet.Cells.get_Range("A9", "A" + rowsint);
                    object[,] values = (object[,])rng.Cells.Value;

                    string[,] s1 = new string[rowsint - 9, 1];

                    for (int j = 1; j <= rowsint - 9; j++)
                    {
                        s1[j - 1, 0] = values[j, 1].ToString();
                    }

                    count++;
```

```
                    mw = (MWNumericArray)(lsf.LSFit2(filename[i], 1,
Convert.ToDouble(s1[1, 0]), (Convert.ToDouble(s1[0, 0]) * 1000000000)));

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    double w;
                    A = arrFit1[0];
                    tau = arrFit1[1];
                    w = arrFit1[2] / 1000000000;

                    double[] yf = new double[Convert.ToInt32(s1[1, 0])];
                    double[] x = new double[Convert.ToInt32(s1[1, 0])];

                    double[] Hy = new double[openFileDialog1.FileNames.Length];

                    for (int p = 4; p < Convert.ToDouble(s1[1, 0]); p++)
                    {
                        ya[p - 4] = Convert.ToDouble(s1[p, 0]);
                    }

                    int po = (int)(c1.maxnumber(ya, s1.Length - 1) - 1);

                    for (int j = 0; j < Convert.ToInt32(s1[1, 0]); j++)
                    {
                        x[j] = j * Convert.ToDouble(s1[0, 0]);
                        if (j > po)
                        {
                            yf[j] = Convert.ToDouble(s1[po + 5, 0]) * (2 * w * Math.Sqrt((x[j]
- x[po + 1]) * tau) / (Math.Sqrt(Math.PI) * (2 * w * (x[j] - x[po + 1]) + 1)) + (1 /
(Math.Sqrt(Math.Pow((2 * w * (x[j] - x[po + 1]) + 1), 3)))) * Math.Exp((x[j] - x[po + 1]) /
tau / (2 * w * (x[j] - x[po + 1]) + 1)) * Erfc(Math.Sqrt((x[j] - x[po + 1]) / tau / (2 * w
* (x[j] - x[po + 1]) + 1))));

                            this.chart2.Series[i+5].Points.AddXY(x[j], yf[j]);
                        }

                    }

                    for (int k = 4; k < Convert.ToInt32(s1[1, 0]); k++)
                    {
                        x[k] = (k - 4) * Convert.ToDouble(s1[0, 0]);
                        this.chart2.Series[i].Points.AddXY(x[k], Convert.ToDouble(s1[k,
0]));
                    }

                    double H = calculatehydration(tau);
                    Hy[i] = H;
                    double HW = w / D / (betaw - betad);

                    this.chart3.Series[0].Points.AddXY(count, Hy[i]);

                    outputfile = OutputFiletextBox.Text;

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(7);

                    int _lastRow = xlWorkSheet.Cells.Find(
                            "*",
                            xlWorkSheet.Cells[1, 1],
                            Excel.XlFindLookIn.xlFormulas,
                            Excel.XlLookAt.xlPart,
                            Excel.XlSearchOrder.xlByRows,
                            Excel.XlSearchDirection.xlPrevious,
                            Missing.Value,
                            Missing.Value,
                            Missing.Value
                            ).Row + 1;
```

```
                    xlWorkSheet.Cells[_lastRow, 1] = filename;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 3] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 4] = HW;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = w;
                    xlWorkSheet.Cells[_lastRow, 7] = A;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }
            }

            if (SettingModecomboBox.SelectedIndex == 6)
            {
                this.chart3.Visible = false;
                this.chart4.Visible = true;

                double zmax = 0;

                for (int i = 0; i < openFileDialog1.FileNames.Length; i++)
                {
                    this.chart4.Series[i].IsVisibleInLegend = true;

                    xlApp = new Microsoft.Office.Interop.Excel.Application();
                    xlWorkBook = xlApp.Workbooks.Open(filename[i]);

                    xlWorkSheet = (Excel.Worksheet)xlApp.Worksheets.get_Item(1);

                    int rowsint =
xlWorkSheet.Range["A65535"].End[Excel.XlDirection.xlUp].Row;

                    Microsoft.Office.Interop.Excel.Range rng =
xlWorkSheet.Cells.get_Range("A9", "A" + rowsint);
                    object[,] values = (object[,])rng.Cells.Value;

                    string[,] s1 = new string[rowsint - 9, 1];

                    for (int j = 1; j <= rowsint - 9; j++)
                    {
                        s1[j - 1, 0] = values[j, 1].ToString();
                    }

                    count++;

                    mw = (MWNumericArray)(lsf.LSFit1(filename[i], 1,
Convert.ToDouble(s1[1, 0]), (Convert.ToDouble(s1[0, 0]) * 1000000000)));

                    double[] arrFit1 =
(double[])((MWNumericArray)mw).ToVector(MWArrayComponent.Real);

                    double A;
                    double tau;
                    double t;
                    A = arrFit1[0];
                    tau = arrFit1[1];

                    double[] yf = new double[Convert.ToInt32(s1[1, 0])];
                    double[] x = new double[Convert.ToInt32(s1[1, 0])];

                    double[] Hy = new double[openFileDialog1.FileNames.Length];

                    for (int p = 4; p < Convert.ToDouble(s1[1, 0]); p++)
                    {
                        ya[p - 4] = Convert.ToDouble(s1[p, 0]);
                    }

                    int po = (int)(c1.maxnumber(ya, s1.Length - 1) - 1);

                    for (int j = 0; j < Convert.ToInt32(s1[1, 0]); j++)
                    {
                        x[j] = j * Convert.ToDouble(s1[0, 0]);
```

```
                        if (j > po)
                        {
                            yf[j] = Convert.ToDouble(s1[po + 5, 0]) * Math.Exp((x[j] - x[po
+ 1]) / tau) * Erfc(Math.Sqrt((x[j] - x[po + 1]) / tau));

                            this.chart2.Series[i + 5].Points.AddXY(x[j], yf[j]);
                        }

                    }

                    for (int k = 4; k < Convert.ToInt32(s1[1, 0]); k++)
                    {
                        x[k] = (k - 4) * Convert.ToDouble(s1[0, 0]);
                        this.chart2.Series[i].Points.AddXY(x[k], Convert.ToDouble(s1[k,
0]));
                    }

                    outputfile = OutputFiletextBox.Text;

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(7);

                    int _lastRow = xlWorkSheet.Cells.Find(
                             "*",
                             xlWorkSheet.Cells[1, 1],
                             Excel.XlFindLookIn.xlFormulas,
                             Excel.XlLookAt.xlPart,
                             Excel.XlSearchOrder.xlByRows,
                             Excel.XlSearchDirection.xlPrevious,
                             Missing.Value,
                             Missing.Value,
                             Missing.Value
                             ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 1] = filename;
                    xlWorkSheet.Cells[_lastRow, 2] = DateTime.Now.ToString();
                    xlWorkSheet.Cells[_lastRow, 4] = 0;
                    xlWorkSheet.Cells[_lastRow, 5] = tau;
                    xlWorkSheet.Cells[_lastRow, 6] = 0;
                    xlWorkSheet.Cells[_lastRow, 7] = A;
                    xlWorkSheet.Cells[_lastRow, 8] = 1;
                    xlWorkSheet.Cells[_lastRow, 9] = no_of_samples;

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();

                    beta = Math.Sqrt(1 / (tau * D));

                int step = (no_of_samples - po) / 10;

                double[] yd = new double[10];

                for (int l = 0; l < 10; l++)
                {
                    int t0 = 0;
                    int t1;

                    t0 = l * step;
                    t1 = t0 + step - 1;

                    mw1 = (MWNumericArray)(lsf.LSFitdeep(filename[i], 1, t0, t1,
Convert.ToDouble(s1[1, 0]), (Convert.ToDouble(s1[0, 0]) * 1000000000)));

                    double[] arrFit2 =
(double[])((MWNumericArray)mw1).ToVector(MWArrayComponent.Real);

                    A = arrFit2[0];
                    tau = arrFit2[1];

                    betas = Math.Sqrt(1 / (tau * D));
```

```
                    double H = calculatehydration(tau);

                    t = ((t1 - t0) / 4 + t0) * (Convert.ToDouble(s1[0, 0]) * 1000000000);

                    yd[l] = (2 / Math.Sqrt(Math.PI)) * (Math.Sqrt(t * 0.000000001 / tau)
/ (beta * Math.Exp(t * 0.000000001 / tau) * Erfc((Math.Sqrt(t * 0.000000001 / tau))))) - 2
* (t * 0.000000001 / tau) / beta;

                    this.chart4.Series[i].Points.AddXY(yd[l] * 1000000, H*100);

                    if (yd[l] > zmax)
                    {
                        zmax = yd[l];
                    }

                    outputfile = OutputFiletextBox.Text;

                    xlWorkBook = xlApp.Workbooks.Open(outputfile);

                    xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(7);

                    _lastRow = xlWorkSheet.Cells.Find(
                            "*",
                            xlWorkSheet.Cells[1, 1],
                            Excel.XlFindLookIn.xlFormulas,
                            Excel.XlLookAt.xlPart,
                            Excel.XlSearchOrder.xlByRows,
                            Excel.XlSearchDirection.xlPrevious,
                            Missing.Value,
                            Missing.Value,
                            Missing.Value
                            ).Row + 1;

                    xlWorkSheet.Cells[_lastRow, 10] = (H * 100) + "%";
                    xlWorkSheet.Cells[_lastRow, 11] = tau;
                    xlWorkSheet.Cells[_lastRow, 12] = A;
                    xlWorkSheet.Cells[_lastRow, 13] = t0;
                    xlWorkSheet.Cells[_lastRow, 14] = t1;
                    xlWorkSheet.Cells[_lastRow, 15] = yd[l];

                    xlWorkBook.Save();
                    xlWorkBook.Close();
                    xlApp.Quit();
                }
              }

                this.chart4.ChartAreas[0].AxisX.Minimum = 0;
                this.chart4.ChartAreas[0].AxisX.Maximum = zmax*1000000;
                this.chart4.ChartAreas[0].AxisX.Interval = 2;

            }
        }
    }

    private string readfile(string filename)
    {
        string s = "";
        string[] lines = System.IO.File.ReadAllLines(@filename);
        int counter = 0;

        // Display the file contents by using a foreach loop.

        foreach (string line in lines)
        {
           string line0 = line.Trim();

            if (counter == 8)
            {
                s = s + line0 + " ";
            }
            else if (counter == 9)
            {
                s = s + line0 + " ";
```

```
                }
                else if (counter > 11)
                {
                    s = s + " " + line0;
                }
                counter++;
            }
            return s;
        }

        //write file part end

        //simulation part

        private void readSimulation(ref double[] data, double dtsim)
        {
            double A, t, tau;
            A = 1;
            dtsim = dtsim * 0.000000001;
            tau = 0.0002;
            data = new double[no_of_samples];
            Random random = new Random();

            tau = 0.0001 + rtau.Next(0, 100) / 200000.0;
            for (int i = 0; i < no_of_samples; i++)
            {
                t = i * dtsim;
                if (i < 85)
                {
                    data[i] = 0;

                }
                else
                {
                    data[i] = A * Math.Exp((t - 85 * dtsim) / tau) * Erfc(Math.Sqrt((t - 85
* dtsim) / tau));
                }

                double randomNumber = random.Next(0, 100) / 10000.0;
                data[i] = data[i] + randomNumber;
            }
        }

        private double Erfc(double x)
        {
            return 1 - Erf(x);
        }

        private double Erf(double x)
        {
            // constants
            double a1 = 0.254829592;
            double a2 = -0.284496736;
            double a3 = 1.421413741;
            double a4 = -1.453152027;
            double a5 = 1.061405429;
            double p = 0.3275911;

            // Save the sign of x
            int sign = 1;
            if (x < 0)
                sign = -1;
            x = Math.Abs(x);

            // A&S formula 7.1.26
            double t = 1.0 / (1.0 + p * x);
            double y = 1.0 - (((((a5 * t + a4) * t) + a3) * t + a2) * t + a1) * t * Math.Exp(-x
* x);

            return sign * y;
        }

        private double calculatehydration(double tau)
```

```
        {
            double H;
            beta = 1 / Math.Sqrt(tau * D);
            H = (beta - betad) / (betaw - betad);
            return H;
        }

        //simulation part end

        //configuration part

        private void chartconfigration()
        {
            this.chart1.ChartAreas[0].AxisY.Title = "Voltage(mv)";
            this.chart1.ChartAreas[0].AxisX.Minimum = 0;

            if (TimeIntervalcomboBox.SelectedIndex == 0)
            {
                timebase = 2;

                if (sx[1].Trim() == "4424")
                {
                    timebase = 2;
                }

                else if(sx[1].Trim() == "4262")
                {
                    timebase = 2;
                }

                dtsim = 50;

                this.chart1.ChartAreas[0].AxisX.Interval = 20000;
                this.chart1.ChartAreas[0].AxisX.Maximum =
this.chart1.ChartAreas[0].AxisX.Interval * 2;

                if (ScanRangecomboBox.SelectedIndex == 0)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -1;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 1;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
                }

                else if (ScanRangecomboBox.SelectedIndex == 1)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -2;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 2;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
                }

                else if (ScanRangecomboBox.SelectedIndex == 2)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -5;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 5;
                    this.chart1.ChartAreas[0].AxisY.Interval = 1;
                }

                else if (ScanRangecomboBox.SelectedIndex == 3)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
                    this.chart1.ChartAreas[0].AxisY.Interval = 2;
                }

                else if (ScanRangecomboBox.SelectedIndex == 4)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
                    this.chart1.ChartAreas[0].AxisY.Interval = 4;
                }

                else if (ScanRangecomboBox.SelectedIndex == 5)
                {
```

```
            this.chart1.ChartAreas[0].AxisY.Minimum = -50;
            this.chart1.ChartAreas[0].AxisY.Maximum = 50;
            this.chart1.ChartAreas[0].AxisY.Interval = 10;
        }

        else if (ScanRangecomboBox.SelectedIndex == 6)
        {
            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }

        else if (ScanRangecomboBox.SelectedIndex == 7)
        {
            this.chart1.ChartAreas[0].AxisY.Minimum = -200;
            this.chart1.ChartAreas[0].AxisY.Maximum = 200;
            this.chart1.ChartAreas[0].AxisY.Interval = 40;
        }

        else if (ScanRangecomboBox.SelectedIndex == 8)
        {
            this.chart1.ChartAreas[0].AxisY.Minimum = -500;
            this.chart1.ChartAreas[0].AxisY.Maximum = 500;
            this.chart1.ChartAreas[0].AxisY.Interval = 100;
        }

        else if (ScanRangecomboBox.SelectedIndex == 9)
        {
            this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
            this.chart1.ChartAreas[0].AxisY.Interval = 200;
        }

        else if (ScanRangecomboBox.SelectedIndex == 10)
        {
            this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
            this.chart1.ChartAreas[0].AxisY.Interval = 400;
        }

        else if (ScanRangecomboBox.SelectedIndex == 11)
        {
            this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
            this.chart1.ChartAreas[0].AxisY.Interval = 1000;
        }

        else if (ScanRangecomboBox.SelectedIndex == 12)
        {
            this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
            this.chart1.ChartAreas[0].AxisY.Interval = 2000;
        }

        else
        {
            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }
    }

    else if (TimeIntervalcomboBox.SelectedIndex == 1)
    {

        if (sx[1].Trim() == "4424")
        {
            timebase = 3;
        }

        else if (sx[1].Trim() == "4262")
        {
            timebase = 0;
```

```
                }

                dtsim = 100;

                this.chart1.ChartAreas[0].AxisX.Interval = 40000;
                this.chart1.ChartAreas[0].AxisX.Maximum =
this.chart1.ChartAreas[0].AxisX.Interval * 2;

                if (ScanRangecomboBox.SelectedIndex == 0)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -1;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 1;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
                }

                else if (ScanRangecomboBox.SelectedIndex == 1)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -2;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 2;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
                }

                else if (ScanRangecomboBox.SelectedIndex == 2)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -5;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 5;
                    this.chart1.ChartAreas[0].AxisY.Interval = 1;
                }

                else if (ScanRangecomboBox.SelectedIndex == 3)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
                    this.chart1.ChartAreas[0].AxisY.Interval = 2;
                }

                else if (ScanRangecomboBox.SelectedIndex == 4)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
                    this.chart1.ChartAreas[0].AxisY.Interval = 4;
                }

                else if (ScanRangecomboBox.SelectedIndex == 5)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
                    this.chart1.ChartAreas[0].AxisY.Interval = 10;
                }

                else if (ScanRangecomboBox.SelectedIndex == 6)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                    this.chart1.ChartAreas[0].AxisY.Interval = 20;
                }

                else if (ScanRangecomboBox.SelectedIndex == 7)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
                    this.chart1.ChartAreas[0].AxisY.Interval = 40;
                }

                else if (ScanRangecomboBox.SelectedIndex == 8)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
                    this.chart1.ChartAreas[0].AxisY.Interval = 100;
                }

                else if (ScanRangecomboBox.SelectedIndex == 9)
                {
```

```
                this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
                this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
                this.chart1.ChartAreas[0].AxisY.Interval = 200;
            }

            else if (ScanRangecomboBox.SelectedIndex == 10)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
                this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
                this.chart1.ChartAreas[0].AxisY.Interval = 400;
            }

            else if (ScanRangecomboBox.SelectedIndex == 11)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
                this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
                this.chart1.ChartAreas[0].AxisY.Interval = 1000;
            }

            else if (ScanRangecomboBox.SelectedIndex == 12)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
                this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
                this.chart1.ChartAreas[0].AxisY.Interval = 2000;
            }

            else
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -100;
                this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                this.chart1.ChartAreas[0].AxisY.Interval = 20;
            }
        }

        else if (TimeIntervalcomboBox.SelectedIndex == 2)
        {

            if (sx[1].Trim() == "4424")
            {
                timebase = 11;
            }

            else if (sx[1].Trim() == "4262")
            {
                timebase = 4;
            }
            dtsim = 500;

            this.chart1.ChartAreas[0].AxisX.Interval = 200000;
            this.chart1.ChartAreas[0].AxisX.Maximum =
this.chart1.ChartAreas[0].AxisX.Interval * 2;

            if (ScanRangecomboBox.SelectedIndex == 0)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -1;
                this.chart1.ChartAreas[0].AxisY.Maximum = 1;
                this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
            }

            else if (ScanRangecomboBox.SelectedIndex == 1)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -2;
                this.chart1.ChartAreas[0].AxisY.Maximum = 2;
                this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
            }

            else if (ScanRangecomboBox.SelectedIndex == 2)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -5;
                this.chart1.ChartAreas[0].AxisY.Maximum = 5;
                this.chart1.ChartAreas[0].AxisY.Interval = 1;
            }
```

```
else if (ScanRangecomboBox.SelectedIndex == 3)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
    this.chart1.ChartAreas[0].AxisY.Interval = 2;
}

else if (ScanRangecomboBox.SelectedIndex == 4)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
    this.chart1.ChartAreas[0].AxisY.Interval = 4;
}

else if (ScanRangecomboBox.SelectedIndex == 5)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
    this.chart1.ChartAreas[0].AxisY.Interval = 10;
}

else if (ScanRangecomboBox.SelectedIndex == 6)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
    this.chart1.ChartAreas[0].AxisY.Interval = 20;
}

else if (ScanRangecomboBox.SelectedIndex == 7)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
    this.chart1.ChartAreas[0].AxisY.Interval = 40;
}

else if (ScanRangecomboBox.SelectedIndex == 8)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
    this.chart1.ChartAreas[0].AxisY.Interval = 100;
}

else if (ScanRangecomboBox.SelectedIndex == 9)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
    this.chart1.ChartAreas[0].AxisY.Interval = 200;
}

else if (ScanRangecomboBox.SelectedIndex == 10)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
    this.chart1.ChartAreas[0].AxisY.Interval = 400;
}

else if (ScanRangecomboBox.SelectedIndex == 11)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
    this.chart1.ChartAreas[0].AxisY.Interval = 1000;
}

else if (ScanRangecomboBox.SelectedIndex == 12)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
    this.chart1.ChartAreas[0].AxisY.Interval = 2000;
}

else
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
```

```
                this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                this.chart1.ChartAreas[0].AxisY.Interval = 20;
            }
        }

        else if (TimeIntervalcomboBox.SelectedIndex == 3)
        {

            if (sx[1].Trim() == "4424")
            {
                timebase = 21;
            }

            else if (sx[1].Trim() == "4262")
            {
                timebase = 9;
            }
            dtsim = 1000;

            this.chart1.ChartAreas[0].AxisX.Interval = 400000;
            this.chart1.ChartAreas[0].AxisX.Maximum =
this.chart1.ChartAreas[0].AxisX.Interval * 2;

            if (ScanRangecomboBox.SelectedIndex == 0)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -1;
                this.chart1.ChartAreas[0].AxisY.Maximum = 1;
                this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
            }

            else if (ScanRangecomboBox.SelectedIndex == 1)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -2;
                this.chart1.ChartAreas[0].AxisY.Maximum = 2;
                this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
            }

            else if (ScanRangecomboBox.SelectedIndex == 2)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -5;
                this.chart1.ChartAreas[0].AxisY.Maximum = 5;
                this.chart1.ChartAreas[0].AxisY.Interval = 1;
            }

            else if (ScanRangecomboBox.SelectedIndex == 3)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -10;
                this.chart1.ChartAreas[0].AxisY.Maximum = 10;
                this.chart1.ChartAreas[0].AxisY.Interval = 2;
            }

            else if (ScanRangecomboBox.SelectedIndex == 4)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -20;
                this.chart1.ChartAreas[0].AxisY.Maximum = 20;
                this.chart1.ChartAreas[0].AxisY.Interval = 4;
            }

            else if (ScanRangecomboBox.SelectedIndex == 5)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -50;
                this.chart1.ChartAreas[0].AxisY.Maximum = 50;
                this.chart1.ChartAreas[0].AxisY.Interval = 10;
            }

            else if (ScanRangecomboBox.SelectedIndex == 6)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -100;
                this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                this.chart1.ChartAreas[0].AxisY.Interval = 20;
            }

            else if (ScanRangecomboBox.SelectedIndex == 7)
```

```
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
                    this.chart1.ChartAreas[0].AxisY.Interval = 40;
                }

                else if (ScanRangecomboBox.SelectedIndex == 8)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
                    this.chart1.ChartAreas[0].AxisY.Interval = 100;
                }

                else if (ScanRangecomboBox.SelectedIndex == 9)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 200;
                }

                else if (ScanRangecomboBox.SelectedIndex == 10)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 400;
                }

                else if (ScanRangecomboBox.SelectedIndex == 11)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 1000;
                }

                else if (ScanRangecomboBox.SelectedIndex == 12)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 2000;
                }

                else
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                    this.chart1.ChartAreas[0].AxisY.Interval = 20;
                }
            }

            else if (TimeIntervalcomboBox.SelectedIndex == 4)
            {

                if (sx[1].Trim() == "4424")
                {
                    timebase = 101;
                }

                else if (sx[1].Trim() == "4262")
                {
                    timebase = 49;
                }
                dtsim = 5000;

                this.chart1.ChartAreas[0].AxisX.Interval = 2000000;
                this.chart1.ChartAreas[0].AxisX.Maximum =
this.chart1.ChartAreas[0].AxisX.Interval * 2;

                if (ScanRangecomboBox.SelectedIndex == 0)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -1;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 1;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
                }
```

```
else if (ScanRangecomboBox.SelectedIndex == 1)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -2;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2;
    this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
}

else if (ScanRangecomboBox.SelectedIndex == 2)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -5;
    this.chart1.ChartAreas[0].AxisY.Maximum = 5;
    this.chart1.ChartAreas[0].AxisY.Interval = 1;
}

else if (ScanRangecomboBox.SelectedIndex == 3)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
    this.chart1.ChartAreas[0].AxisY.Interval = 2;
}

else if (ScanRangecomboBox.SelectedIndex == 4)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
    this.chart1.ChartAreas[0].AxisY.Interval = 4;
}

else if (ScanRangecomboBox.SelectedIndex == 5)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
    this.chart1.ChartAreas[0].AxisY.Interval = 10;
}

else if (ScanRangecomboBox.SelectedIndex == 6)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
    this.chart1.ChartAreas[0].AxisY.Interval = 20;
}

else if (ScanRangecomboBox.SelectedIndex == 7)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
    this.chart1.ChartAreas[0].AxisY.Interval = 40;
}

else if (ScanRangecomboBox.SelectedIndex == 8)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
    this.chart1.ChartAreas[0].AxisY.Interval = 100;
}

else if (ScanRangecomboBox.SelectedIndex == 9)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
    this.chart1.ChartAreas[0].AxisY.Interval = 200;
}

else if (ScanRangecomboBox.SelectedIndex == 10)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
    this.chart1.ChartAreas[0].AxisY.Interval = 400;
}

else if (ScanRangecomboBox.SelectedIndex == 11)
{
```

```
                this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
                this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
                this.chart1.ChartAreas[0].AxisY.Interval = 1000;
            }

            else if (ScanRangecomboBox.SelectedIndex == 12)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
                this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
                this.chart1.ChartAreas[0].AxisY.Interval = 2000;
            }

            else
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -100;
                this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                this.chart1.ChartAreas[0].AxisY.Interval = 20;
            }
        }

        else if (TimeIntervalcomboBox.SelectedIndex == 5)
        {

            if (sx[1].Trim() == "4424")
            {
                timebase = 201;
            }

            else if (sx[1].Trim() == "4262")
            {
                timebase = 99;
            }
            dtsim = 10000;

            this.chart1.ChartAreas[0].AxisX.Interval = 4000000;
            this.chart1.ChartAreas[0].AxisX.Maximum =
this.chart1.ChartAreas[0].AxisX.Interval * 2;

            if (ScanRangecomboBox.SelectedIndex == 0)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -1;
                this.chart1.ChartAreas[0].AxisY.Maximum = 1;
                this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
            }

            else if (ScanRangecomboBox.SelectedIndex == 1)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -2;
                this.chart1.ChartAreas[0].AxisY.Maximum = 2;
                this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
            }

            else if (ScanRangecomboBox.SelectedIndex == 2)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -5;
                this.chart1.ChartAreas[0].AxisY.Maximum = 5;
                this.chart1.ChartAreas[0].AxisY.Interval = 1;
            }

            else if (ScanRangecomboBox.SelectedIndex == 3)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -10;
                this.chart1.ChartAreas[0].AxisY.Maximum = 10;
                this.chart1.ChartAreas[0].AxisY.Interval = 2;
            }

            else if (ScanRangecomboBox.SelectedIndex == 4)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -20;
                this.chart1.ChartAreas[0].AxisY.Maximum = 20;
                this.chart1.ChartAreas[0].AxisY.Interval = 4;
            }
```

```
    else if (ScanRangecomboBox.SelectedIndex == 5)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -50;
        this.chart1.ChartAreas[0].AxisY.Maximum = 50;
        this.chart1.ChartAreas[0].AxisY.Interval = 10;
    }

    else if (ScanRangecomboBox.SelectedIndex == 6)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -100;
        this.chart1.ChartAreas[0].AxisY.Maximum = 100;
        this.chart1.ChartAreas[0].AxisY.Interval = 20;
    }

    else if (ScanRangecomboBox.SelectedIndex == 7)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -200;
        this.chart1.ChartAreas[0].AxisY.Maximum = 200;
        this.chart1.ChartAreas[0].AxisY.Interval = 40;
    }

    else if (ScanRangecomboBox.SelectedIndex == 8)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -500;
        this.chart1.ChartAreas[0].AxisY.Maximum = 500;
        this.chart1.ChartAreas[0].AxisY.Interval = 100;
    }

    else if (ScanRangecomboBox.SelectedIndex == 9)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
        this.chart1.ChartAreas[0].AxisY.Interval = 200;
    }

    else if (ScanRangecomboBox.SelectedIndex == 10)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
        this.chart1.ChartAreas[0].AxisY.Interval = 400;
    }

    else if (ScanRangecomboBox.SelectedIndex == 11)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
        this.chart1.ChartAreas[0].AxisY.Interval = 1000;
    }

    else if (ScanRangecomboBox.SelectedIndex == 12)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
        this.chart1.ChartAreas[0].AxisY.Interval = 2000;
    }

    else
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -100;
        this.chart1.ChartAreas[0].AxisY.Maximum = 100;
        this.chart1.ChartAreas[0].AxisY.Interval = 20;
    }
}

else if (TimeIntervalcomboBox.SelectedIndex == 6)
{

    if (sx[1].Trim() == "4424")
    {
        timebase = 1001;
    }

    else if (sx[1].Trim() == "4262")
```

```
                {
                    timebase = 499;
                }
                dtsim = 50000;

                this.chart1.ChartAreas[0].AxisX.Interval = 20000000;
                this.chart1.ChartAreas[0].AxisX.Maximum =
this.chart1.ChartAreas[0].AxisX.Interval * 2;

                if (ScanRangecomboBox.SelectedIndex == 0)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -1;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 1;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
                }

                else if (ScanRangecomboBox.SelectedIndex == 1)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -2;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 2;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
                }

                else if (ScanRangecomboBox.SelectedIndex == 2)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -5;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 5;
                    this.chart1.ChartAreas[0].AxisY.Interval = 1;
                }

                else if (ScanRangecomboBox.SelectedIndex == 3)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
                    this.chart1.ChartAreas[0].AxisY.Interval = 2;
                }

                else if (ScanRangecomboBox.SelectedIndex == 4)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
                    this.chart1.ChartAreas[0].AxisY.Interval = 4;
                }

                else if (ScanRangecomboBox.SelectedIndex == 5)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
                    this.chart1.ChartAreas[0].AxisY.Interval = 10;
                }

                else if (ScanRangecomboBox.SelectedIndex == 6)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                    this.chart1.ChartAreas[0].AxisY.Interval = 20;
                }

                else if (ScanRangecomboBox.SelectedIndex == 7)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
                    this.chart1.ChartAreas[0].AxisY.Interval = 40;
                }

                else if (ScanRangecomboBox.SelectedIndex == 8)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
                    this.chart1.ChartAreas[0].AxisY.Interval = 100;
                }

                else if (ScanRangecomboBox.SelectedIndex == 9)
```

```
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 200;
                }

                else if (ScanRangecomboBox.SelectedIndex == 10)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 400;
                }

                else if (ScanRangecomboBox.SelectedIndex == 11)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 1000;
                }

                else if (ScanRangecomboBox.SelectedIndex == 12)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 2000;
                }

                else
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                    this.chart1.ChartAreas[0].AxisY.Interval = 20;
                }
            }

            else if (TimeIntervalcomboBox.SelectedIndex == 7)
            {

                if (sx[1].Trim() == "4424")
                {
                    timebase = 2001;
                }

                else if (sx[1].Trim() == "4262")
                {
                    timebase = 999;
                }
                dtsim = 100000;

                this.chart1.ChartAreas[0].AxisX.Interval = 40000000;
                this.chart1.ChartAreas[0].AxisX.Maximum =
this.chart1.ChartAreas[0].AxisX.Interval * 2;

                if (ScanRangecomboBox.SelectedIndex == 0)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -1;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 1;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
                }

                else if (ScanRangecomboBox.SelectedIndex == 1)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -2;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 2;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
                }

                else if (ScanRangecomboBox.SelectedIndex == 2)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -5;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 5;
                    this.chart1.ChartAreas[0].AxisY.Interval = 1;
                }
```

```
else if (ScanRangecomboBox.SelectedIndex == 3)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
    this.chart1.ChartAreas[0].AxisY.Interval = 2;
}

else if (ScanRangecomboBox.SelectedIndex == 4)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
    this.chart1.ChartAreas[0].AxisY.Interval = 4;
}

else if (ScanRangecomboBox.SelectedIndex == 5)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
    this.chart1.ChartAreas[0].AxisY.Interval = 10;
}

else if (ScanRangecomboBox.SelectedIndex == 6)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
    this.chart1.ChartAreas[0].AxisY.Interval = 20;
}

else if (ScanRangecomboBox.SelectedIndex == 7)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
    this.chart1.ChartAreas[0].AxisY.Interval = 40;
}

else if (ScanRangecomboBox.SelectedIndex == 8)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
    this.chart1.ChartAreas[0].AxisY.Interval = 100;
}

else if (ScanRangecomboBox.SelectedIndex == 9)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
    this.chart1.ChartAreas[0].AxisY.Interval = 200;
}

else if (ScanRangecomboBox.SelectedIndex == 10)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
    this.chart1.ChartAreas[0].AxisY.Interval = 400;
}

else if (ScanRangecomboBox.SelectedIndex == 11)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
    this.chart1.ChartAreas[0].AxisY.Interval = 1000;
}

else if (ScanRangecomboBox.SelectedIndex == 12)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
    this.chart1.ChartAreas[0].AxisY.Interval = 2000;
}

else
{
```

```
                this.chart1.ChartAreas[0].AxisY.Minimum = -100;
                this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                this.chart1.ChartAreas[0].AxisY.Interval = 20;
            }
        }

        else if (TimeIntervalcomboBox.SelectedIndex == 8)
        {

            if (sx[1].Trim() == "4424")
            {
                timebase = 10001;
            }

            else if (sx[1].Trim() == "4262")
            {
                timebase = 4999;
            }
            dtsim = 500000;

            this.chart1.ChartAreas[0].AxisX.Interval = 200000000;
            this.chart1.ChartAreas[0].AxisX.Maximum =
this.chart1.ChartAreas[0].AxisX.Interval * 2;

            if (ScanRangecomboBox.SelectedIndex == 0)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -1;
                this.chart1.ChartAreas[0].AxisY.Maximum = 1;
                this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
            }

            else if (ScanRangecomboBox.SelectedIndex == 1)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -2;
                this.chart1.ChartAreas[0].AxisY.Maximum = 2;
                this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
            }

            else if (ScanRangecomboBox.SelectedIndex == 2)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -5;
                this.chart1.ChartAreas[0].AxisY.Maximum = 5;
                this.chart1.ChartAreas[0].AxisY.Interval = 1;
            }

            else if (ScanRangecomboBox.SelectedIndex == 3)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -10;
                this.chart1.ChartAreas[0].AxisY.Maximum = 10;
                this.chart1.ChartAreas[0].AxisY.Interval = 2;
            }

            else if (ScanRangecomboBox.SelectedIndex == 4)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -20;
                this.chart1.ChartAreas[0].AxisY.Maximum = 20;
                this.chart1.ChartAreas[0].AxisY.Interval = 4;
            }

            else if (ScanRangecomboBox.SelectedIndex == 5)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -50;
                this.chart1.ChartAreas[0].AxisY.Maximum = 50;
                this.chart1.ChartAreas[0].AxisY.Interval = 10;
            }

            else if (ScanRangecomboBox.SelectedIndex == 6)
            {
                this.chart1.ChartAreas[0].AxisY.Minimum = -100;
                this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                this.chart1.ChartAreas[0].AxisY.Interval = 20;
            }
```

```
                else if (ScanRangecomboBox.SelectedIndex == 7)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
                    this.chart1.ChartAreas[0].AxisY.Interval = 40;
                }

                else if (ScanRangecomboBox.SelectedIndex == 8)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
                    this.chart1.ChartAreas[0].AxisY.Interval = 100;
                }

                else if (ScanRangecomboBox.SelectedIndex == 9)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 200;
                }

                else if (ScanRangecomboBox.SelectedIndex == 10)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 400;
                }

                else if (ScanRangecomboBox.SelectedIndex == 11)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 1000;
                }

                else if (ScanRangecomboBox.SelectedIndex == 12)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 2000;
                }

                else
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                    this.chart1.ChartAreas[0].AxisY.Interval = 20;
                }
            }

            else if (TimeIntervalcomboBox.SelectedIndex == 9)
            {

                if (sx[1].Trim() == "4424")
                {
                    timebase = 20001;
                }

                else if (sx[1].Trim() == "4262")
                {
                    timebase = 9999;
                }
                dtsim = 1000000;

                this.chart1.ChartAreas[0].AxisX.Interval = 400000000;
                this.chart1.ChartAreas[0].AxisX.Maximum =
this.chart1.ChartAreas[0].AxisX.Interval * 2;

                if (ScanRangecomboBox.SelectedIndex == 0)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -1;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 1;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
```

```
}

else if (ScanRangecomboBox.SelectedIndex == 1)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -2;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2;
    this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
}

else if (ScanRangecomboBox.SelectedIndex == 2)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -5;
    this.chart1.ChartAreas[0].AxisY.Maximum = 5;
    this.chart1.ChartAreas[0].AxisY.Interval = 1;
}

else if (ScanRangecomboBox.SelectedIndex == 3)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
    this.chart1.ChartAreas[0].AxisY.Interval = 2;
}

else if (ScanRangecomboBox.SelectedIndex == 4)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
    this.chart1.ChartAreas[0].AxisY.Interval = 4;
}

else if (ScanRangecomboBox.SelectedIndex == 5)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
    this.chart1.ChartAreas[0].AxisY.Interval = 10;
}

else if (ScanRangecomboBox.SelectedIndex == 6)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
    this.chart1.ChartAreas[0].AxisY.Interval = 20;
}

else if (ScanRangecomboBox.SelectedIndex == 7)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
    this.chart1.ChartAreas[0].AxisY.Interval = 40;
}

else if (ScanRangecomboBox.SelectedIndex == 8)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
    this.chart1.ChartAreas[0].AxisY.Interval = 100;
}

else if (ScanRangecomboBox.SelectedIndex == 9)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
    this.chart1.ChartAreas[0].AxisY.Interval = 200;
}

else if (ScanRangecomboBox.SelectedIndex == 10)
{
    this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
    this.chart1.ChartAreas[0].AxisY.Interval = 400;
}

else if (ScanRangecomboBox.SelectedIndex == 11)
```

```
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 1000;
                }

                else if (ScanRangecomboBox.SelectedIndex == 12)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
                    this.chart1.ChartAreas[0].AxisY.Interval = 2000;
                }

                else
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
                    this.chart1.ChartAreas[0].AxisY.Interval = 20;
                }
            }

            else
            {

                if (sx[1].Trim() == "4424")
                {
                    timebase = 3;
                }

                else if (sx[1].Trim() == "4262")
                {
                    timebase = 0;
                }
                dtsim = 100;

                this.chart1.ChartAreas[0].AxisX.Interval = 40000;
                this.chart1.ChartAreas[0].AxisX.Maximum =
this.chart1.ChartAreas[0].AxisX.Interval * 2;

                if (ScanRangecomboBox.SelectedIndex == 0)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -1;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 1;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
                }

                else if (ScanRangecomboBox.SelectedIndex == 1)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -2;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 2;
                    this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
                }

                else if (ScanRangecomboBox.SelectedIndex == 2)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -5;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 5;
                    this.chart1.ChartAreas[0].AxisY.Interval = 1;
                }

                else if (ScanRangecomboBox.SelectedIndex == 3)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
                    this.chart1.ChartAreas[0].AxisY.Interval = 2;
                }

                else if (ScanRangecomboBox.SelectedIndex == 4)
                {
                    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
                    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
                    this.chart1.ChartAreas[0].AxisY.Interval = 4;
                }
```

```
    else if (ScanRangecomboBox.SelectedIndex == 5)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -50;
        this.chart1.ChartAreas[0].AxisY.Maximum = 50;
        this.chart1.ChartAreas[0].AxisY.Interval = 10;
    }

    else if (ScanRangecomboBox.SelectedIndex == 6)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -100;
        this.chart1.ChartAreas[0].AxisY.Maximum = 100;
        this.chart1.ChartAreas[0].AxisY.Interval = 20;
    }

    else if (ScanRangecomboBox.SelectedIndex == 7)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -200;
        this.chart1.ChartAreas[0].AxisY.Maximum = 200;
        this.chart1.ChartAreas[0].AxisY.Interval = 40;
    }

    else if (ScanRangecomboBox.SelectedIndex == 8)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -500;
        this.chart1.ChartAreas[0].AxisY.Maximum = 500;
        this.chart1.ChartAreas[0].AxisY.Interval = 100;
    }

    else if (ScanRangecomboBox.SelectedIndex == 9)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
        this.chart1.ChartAreas[0].AxisY.Interval = 200;
    }

    else if (ScanRangecomboBox.SelectedIndex == 10)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
        this.chart1.ChartAreas[0].AxisY.Interval = 400;
    }

    else if (ScanRangecomboBox.SelectedIndex == 11)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
        this.chart1.ChartAreas[0].AxisY.Interval = 1000;
    }

    else if (ScanRangecomboBox.SelectedIndex == 12)
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
        this.chart1.ChartAreas[0].AxisY.Interval = 2000;
    }

    else
    {
        this.chart1.ChartAreas[0].AxisY.Minimum = -100;
        this.chart1.ChartAreas[0].AxisY.Maximum = 100;
        this.chart1.ChartAreas[0].AxisY.Interval = 20;
    }
}

//trigger channel

if (TriggerChannelcomboBox.SelectedIndex == 0)
{
    channel_enable = 0;
}

else if (TriggerChannelcomboBox.SelectedIndex == 1)
```

```
            {
                channel_enable = 1;
                channel_ext = 0;
            }

            else if (TriggerChannelcomboBox.SelectedIndex == 2)
            {
                channel_enable = 1;
                channel_ext = 1;
            }

            else if (TriggerChannelcomboBox.SelectedIndex == 3)
            {
                channel_enable = 1;
                channel_ext = 2;
            }

            else if (TriggerChannelcomboBox.SelectedIndex == 4)
            {
                channel_enable = 1;
                channel_ext = 3;
            }

            else
            {
                channel_enable = 1;
                channel_ext = 5;
            }

            //trigger threshold

            if (TriggerThresholdcomboBox.SelectedIndex == 0)
            {
                threshold = 1000;
            }

            else if (TriggerThresholdcomboBox.SelectedIndex == 1)
            {
                threshold = 2000;
            }

            else if (TriggerThresholdcomboBox.SelectedIndex == 2)
            {
                threshold = 3000;
            }

            else if (TriggerThresholdcomboBox.SelectedIndex == 3)
            {
                threshold = 4000;
            }

            else if (TriggerThresholdcomboBox.SelectedIndex == 4)
            {
                threshold = 5000;
            }

            else
            {
                threshold = 1000;
            }

            //sampling points

            if (SamplingPointcomboBox.SelectedIndex >= 0)
            {
                no_of_samples =
Convert.ToInt16(SamplingPointcomboBox.Items[SamplingPointcomboBox.SelectedIndex].ToString
());
            }
            else
            {
                no_of_samples = 1024;
            }
```

```
        //delay

        if (DelaycomboBox.SelectedIndex == 0)
        {
            delay = 0;
        }

        else if (DelaycomboBox.SelectedIndex == 1)
        {
            delay = 20;
        }

        else if (DelaycomboBox.SelectedIndex == 2)
        {
            delay = 40;
        }

        else if (DelaycomboBox.SelectedIndex == 3)
        {
            delay = 60;
        }

        else if (DelaycomboBox.SelectedIndex == 4)
        {
            delay = 80;
        }

        else if (DelaycomboBox.SelectedIndex == 5)
        {
            delay = 100;
        }



        else
        {
            delay = 0;
        }

        //trigger direction

        if (TriggerDirectioncomboBox.SelectedIndex >= 0)
        {
            direction = TriggerDirectioncomboBox.SelectedIndex;
        }

        else
        {
            direction = 2;
        }

        trackBar1.Maximum = 5000;
        trackBar1.Minimum = -5000;
        trackBar1.SmallChange = 100;
        trackBar1.LargeChange = 1000;
        TrackBarlabel.Text =  trackBar1.Value.ToString();
        offset = (short)trackBar1.Value;

        trackBar2.Maximum = 5;
        trackBar2.Minimum = -5;
        trackBar2.SmallChange = 1;
        trackBar2.LargeChange = 1;
        Trackbarlabel2.Text = trackBar2.Value.ToString();
        timeoffset = (short)trackBar2.Value;

    }

    //Ps2000 Configration

    private void Ps2000chartconfigration()
    {
        this.chart1.ChartAreas[0].AxisY.Title = "Voltage(mv)";
```

```
this.chart1.ChartAreas[0].AxisX.Minimum = 0;

//x,y scan range

if (Ps2000TimeIntervalcomboBox.SelectedIndex == 0)
{
    Ps2000timebase = 2;

    this.chart1.ChartAreas[0].AxisX.Interval = 16000;

    if (Ps2000ScanRangecomboBox.SelectedIndex == 0)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -1;
        this.chart1.ChartAreas[0].AxisY.Maximum = 1;
        this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 1)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -2;
        this.chart1.ChartAreas[0].AxisY.Maximum = 2;
        this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 2)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -5;
        this.chart1.ChartAreas[0].AxisY.Maximum = 5;
        this.chart1.ChartAreas[0].AxisY.Interval = 1;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 3)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -10;
        this.chart1.ChartAreas[0].AxisY.Maximum = 10;
        this.chart1.ChartAreas[0].AxisY.Interval = 2;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 4)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -20;
        this.chart1.ChartAreas[0].AxisY.Maximum = 20;
        this.chart1.ChartAreas[0].AxisY.Interval = 4;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 5)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -50;
        this.chart1.ChartAreas[0].AxisY.Maximum = 50;
        this.chart1.ChartAreas[0].AxisY.Interval = 10;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 6)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -100;
        this.chart1.ChartAreas[0].AxisY.Maximum = 100;
        this.chart1.ChartAreas[0].AxisY.Interval = 20;
    }
```

```
    else if (Ps2000ScanRangecomboBox.SelectedIndex == 7)
    {
        range = 1;

        this.chart1.ChartAreas[0].AxisY.Minimum = -200;
        this.chart1.ChartAreas[0].AxisY.Maximum = 200;
        this.chart1.ChartAreas[0].AxisY.Interval = 40;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 8)
    {
        range = 2;

        this.chart1.ChartAreas[0].AxisY.Minimum = -500;
        this.chart1.ChartAreas[0].AxisY.Maximum = 500;
        this.chart1.ChartAreas[0].AxisY.Interval = 100;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 9)
    {
        range = 3;

        this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
        this.chart1.ChartAreas[0].AxisY.Interval = 200;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 10)
    {
        range = 4;

        this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
        this.chart1.ChartAreas[0].AxisY.Interval = 400;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 11)
    {
        range = 5;

        this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
        this.chart1.ChartAreas[0].AxisY.Interval = 1000;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 12)
    {
        range = 6;

        this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
        this.chart1.ChartAreas[0].AxisY.Interval = 2000;
    }

    else
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -100;
        this.chart1.ChartAreas[0].AxisY.Maximum = 100;
        this.chart1.ChartAreas[0].AxisY.Interval = 20;
    }
}

else if (Ps2000TimeIntervalcomboBox.SelectedIndex == 1)
{
    Ps2000timebase = 3;

    this.chart1.ChartAreas[0].AxisX.Interval = 32000;

    if (Ps2000ScanRangecomboBox.SelectedIndex == 0)
    {
        range = 0;
```

```
    this.chart1.ChartAreas[0].AxisY.Minimum = -1;
    this.chart1.ChartAreas[0].AxisY.Maximum = 1;
    this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 1)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -2;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2;
    this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 2)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -5;
    this.chart1.ChartAreas[0].AxisY.Maximum = 5;
    this.chart1.ChartAreas[0].AxisY.Interval = 1;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 3)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
    this.chart1.ChartAreas[0].AxisY.Interval = 2;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 4)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
    this.chart1.ChartAreas[0].AxisY.Interval = 4;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 5)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
    this.chart1.ChartAreas[0].AxisY.Interval = 10;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 6)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
    this.chart1.ChartAreas[0].AxisY.Interval = 20;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 7)
{
    range = 1;

    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
    this.chart1.ChartAreas[0].AxisY.Interval = 40;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 8)
{
    range = 2;
```

```
            this.chart1.ChartAreas[0].AxisY.Minimum = -500;
            this.chart1.ChartAreas[0].AxisY.Maximum = 500;
            this.chart1.ChartAreas[0].AxisY.Interval = 100;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 9)
        {
            range = 3;

            this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
            this.chart1.ChartAreas[0].AxisY.Interval = 200;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 10)
        {
            range = 4;

            this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
            this.chart1.ChartAreas[0].AxisY.Interval = 400;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 11)
        {
            range = 5;

            this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
            this.chart1.ChartAreas[0].AxisY.Interval = 1000;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 12)
        {
            range = 6;

            this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
            this.chart1.ChartAreas[0].AxisY.Interval = 2000;
        }

        else
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }
    }

    else if (Ps2000TimeIntervalcomboBox.SelectedIndex == 2)
    {
        Ps2000timebase = 4;

        this.chart1.ChartAreas[0].AxisX.Interval = 64000;

        if (Ps2000ScanRangecomboBox.SelectedIndex == 0)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -1;
            this.chart1.ChartAreas[0].AxisY.Maximum = 1;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 1)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -2;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
```

```
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 2)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -5;
    this.chart1.ChartAreas[0].AxisY.Maximum = 5;
    this.chart1.ChartAreas[0].AxisY.Interval = 1;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 3)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
    this.chart1.ChartAreas[0].AxisY.Interval = 2;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 4)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
    this.chart1.ChartAreas[0].AxisY.Interval = 4;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 5)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
    this.chart1.ChartAreas[0].AxisY.Interval = 10;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 6)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
    this.chart1.ChartAreas[0].AxisY.Interval = 20;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 7)
{
    range = 1;

    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
    this.chart1.ChartAreas[0].AxisY.Interval = 40;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 8)
{
    range = 2;

    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
    this.chart1.ChartAreas[0].AxisY.Interval = 100;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 9)
{
    range = 3;

    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
    this.chart1.ChartAreas[0].AxisY.Interval = 200;
}
```

```
        else if (Ps2000ScanRangecomboBox.SelectedIndex == 10)
        {
            range = 4;

            this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
            this.chart1.ChartAreas[0].AxisY.Interval = 400;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 11)
        {
            range = 5;

            this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
            this.chart1.ChartAreas[0].AxisY.Interval = 1000;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 12)
        {
            range = 6;

            this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
            this.chart1.ChartAreas[0].AxisY.Interval = 2000;
        }

        else
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }
    }

    else if (Ps2000TimeIntervalcomboBox.SelectedIndex == 3)
    {
        Ps2000timebase = 6;

        this.chart1.ChartAreas[0].AxisX.Interval = 256000;

        if (Ps2000ScanRangecomboBox.SelectedIndex == 0)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -1;
            this.chart1.ChartAreas[0].AxisY.Maximum = 1;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 1)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -2;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 2)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -5;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5;
            this.chart1.ChartAreas[0].AxisY.Interval = 1;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 3)
        {
```

```
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
    this.chart1.ChartAreas[0].AxisY.Interval = 2;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 4)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
    this.chart1.ChartAreas[0].AxisY.Interval = 4;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 5)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
    this.chart1.ChartAreas[0].AxisY.Interval = 10;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 6)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
    this.chart1.ChartAreas[0].AxisY.Interval = 20;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 7)
{
    range = 1;

    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
    this.chart1.ChartAreas[0].AxisY.Interval = 40;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 8)
{
    range = 2;

    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
    this.chart1.ChartAreas[0].AxisY.Interval = 100;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 9)
{
    range = 3;

    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
    this.chart1.ChartAreas[0].AxisY.Interval = 200;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 10)
{
    range = 4;

    this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
    this.chart1.ChartAreas[0].AxisY.Interval = 400;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 11)
{
    range = 5;
```

```
            this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
            this.chart1.ChartAreas[0].AxisY.Interval = 1000;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 12)
        {
            range = 6;

            this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
            this.chart1.ChartAreas[0].AxisY.Interval = 2000;
        }

        else
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }
    }

    else if (Ps2000TimeIntervalcomboBox.SelectedIndex == 4)
    {
        Ps2000timebase = 7;

        this.chart1.ChartAreas[0].AxisX.Interval = 512000;

        if (Ps2000ScanRangecomboBox.SelectedIndex == 0)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -1;
            this.chart1.ChartAreas[0].AxisY.Maximum = 1;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 1)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -2;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 2)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -5;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5;
            this.chart1.ChartAreas[0].AxisY.Interval = 1;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 3)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -10;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10;
            this.chart1.ChartAreas[0].AxisY.Interval = 2;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 4)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -20;
            this.chart1.ChartAreas[0].AxisY.Maximum = 20;
```

```
    this.chart1.ChartAreas[0].AxisY.Interval = 4;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 5)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
    this.chart1.ChartAreas[0].AxisY.Interval = 10;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 6)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
    this.chart1.ChartAreas[0].AxisY.Interval = 20;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 7)
{
    range = 1;

    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
    this.chart1.ChartAreas[0].AxisY.Interval = 40;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 8)
{
    range = 2;

    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
    this.chart1.ChartAreas[0].AxisY.Interval = 100;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 9)
{
    range = 3;

    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
    this.chart1.ChartAreas[0].AxisY.Interval = 200;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 10)
{
    range = 4;

    this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
    this.chart1.ChartAreas[0].AxisY.Interval = 400;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 11)
{
    range = 5;

    this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
    this.chart1.ChartAreas[0].AxisY.Interval = 1000;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 12)
{
    range = 6;

    this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
    this.chart1.ChartAreas[0].AxisY.Interval = 2000;
```

```
        }

        else
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }
    }

    else if (Ps2000TimeIntervalcomboBox.SelectedIndex == 5)
    {
        Ps2000timebase = 9;

        this.chart1.ChartAreas[0].AxisX.Interval = 2048000;

        if (Ps2000ScanRangecomboBox.SelectedIndex == 0)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -1;
            this.chart1.ChartAreas[0].AxisY.Maximum = 1;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 1)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -2;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 2)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -5;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5;
            this.chart1.ChartAreas[0].AxisY.Interval = 1;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 3)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -10;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10;
            this.chart1.ChartAreas[0].AxisY.Interval = 2;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 4)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -20;
            this.chart1.ChartAreas[0].AxisY.Maximum = 20;
            this.chart1.ChartAreas[0].AxisY.Interval = 4;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 5)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -50;
            this.chart1.ChartAreas[0].AxisY.Maximum = 50;
            this.chart1.ChartAreas[0].AxisY.Interval = 10;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 6)
```

```
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 7)
        {
            range = 1;

            this.chart1.ChartAreas[0].AxisY.Minimum = -200;
            this.chart1.ChartAreas[0].AxisY.Maximum = 200;
            this.chart1.ChartAreas[0].AxisY.Interval = 40;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 8)
        {
            range = 2;

            this.chart1.ChartAreas[0].AxisY.Minimum = -500;
            this.chart1.ChartAreas[0].AxisY.Maximum = 500;
            this.chart1.ChartAreas[0].AxisY.Interval = 100;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 9)
        {
            range = 3;

            this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
            this.chart1.ChartAreas[0].AxisY.Interval = 200;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 10)
        {
            range = 4;

            this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
            this.chart1.ChartAreas[0].AxisY.Interval = 400;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 11)
        {
            range = 5;

            this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
            this.chart1.ChartAreas[0].AxisY.Interval = 1000;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 12)
        {
            range = 6;

            this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
            this.chart1.ChartAreas[0].AxisY.Interval = 2000;
        }

        else
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }
    }

    else if (Ps2000TimeIntervalcomboBox.SelectedIndex == 6)
```

```
{
    Ps2000timebase = 10;

    this.chart1.ChartAreas[0].AxisX.Interval = 4096000;

    if (Ps2000ScanRangecomboBox.SelectedIndex == 0)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -1;
        this.chart1.ChartAreas[0].AxisY.Maximum = 1;
        this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 1)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -2;
        this.chart1.ChartAreas[0].AxisY.Maximum = 2;
        this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 2)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -5;
        this.chart1.ChartAreas[0].AxisY.Maximum = 5;
        this.chart1.ChartAreas[0].AxisY.Interval = 1;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 3)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -10;
        this.chart1.ChartAreas[0].AxisY.Maximum = 10;
        this.chart1.ChartAreas[0].AxisY.Interval = 2;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 4)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -20;
        this.chart1.ChartAreas[0].AxisY.Maximum = 20;
        this.chart1.ChartAreas[0].AxisY.Interval = 4;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 5)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -50;
        this.chart1.ChartAreas[0].AxisY.Maximum = 50;
        this.chart1.ChartAreas[0].AxisY.Interval = 10;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 6)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -100;
        this.chart1.ChartAreas[0].AxisY.Maximum = 100;
        this.chart1.ChartAreas[0].AxisY.Interval = 20;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 7)
    {
        range = 1;

        this.chart1.ChartAreas[0].AxisY.Minimum = -200;
```

```
            this.chart1.ChartAreas[0].AxisY.Maximum = 200;
            this.chart1.ChartAreas[0].AxisY.Interval = 40;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 8)
        {
            range = 2;

            this.chart1.ChartAreas[0].AxisY.Minimum = -500;
            this.chart1.ChartAreas[0].AxisY.Maximum = 500;
            this.chart1.ChartAreas[0].AxisY.Interval = 100;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 9)
        {
            range = 3;

            this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
            this.chart1.ChartAreas[0].AxisY.Interval = 200;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 10)
        {
            range = 4;

            this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
            this.chart1.ChartAreas[0].AxisY.Interval = 400;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 11)
        {
            range = 5;

            this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
            this.chart1.ChartAreas[0].AxisY.Interval = 1000;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 12)
        {
            range = 6;

            this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
            this.chart1.ChartAreas[0].AxisY.Interval = 2000;
        }

        else
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }
    }

    else if (Ps2000TimeIntervalcomboBox.SelectedIndex == 7)
    {
        Ps2000timebase = 12;

        this.chart1.ChartAreas[0].AxisX.Interval = 16384000;

        if (Ps2000ScanRangecomboBox.SelectedIndex == 0)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -1;
            this.chart1.ChartAreas[0].AxisY.Maximum = 1;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
        }
```

```
else if (Ps2000ScanRangecomboBox.SelectedIndex == 1)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -2;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2;
    this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 2)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -5;
    this.chart1.ChartAreas[0].AxisY.Maximum = 5;
    this.chart1.ChartAreas[0].AxisY.Interval = 1;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 3)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -10;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10;
    this.chart1.ChartAreas[0].AxisY.Interval = 2;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 4)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
    this.chart1.ChartAreas[0].AxisY.Interval = 4;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 5)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
    this.chart1.ChartAreas[0].AxisY.Interval = 10;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 6)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
    this.chart1.ChartAreas[0].AxisY.Interval = 20;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 7)
{
    range = 1;

    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
    this.chart1.ChartAreas[0].AxisY.Interval = 40;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 8)
{
    range = 2;

    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
    this.chart1.ChartAreas[0].AxisY.Interval = 100;
}
```

```
    else if (Ps2000ScanRangecomboBox.SelectedIndex == 9)
    {
        range = 3;

        this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
        this.chart1.ChartAreas[0].AxisY.Interval = 200;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 10)
    {
        range = 4;

        this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
        this.chart1.ChartAreas[0].AxisY.Interval = 400;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 11)
    {
        range = 5;

        this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
        this.chart1.ChartAreas[0].AxisY.Interval = 1000;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 12)
    {
        range = 6;

        this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
        this.chart1.ChartAreas[0].AxisY.Interval = 2000;
    }

    else
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -100;
        this.chart1.ChartAreas[0].AxisY.Maximum = 100;
        this.chart1.ChartAreas[0].AxisY.Interval = 20;
    }
}

else if (Ps2000TimeIntervalcomboBox.SelectedIndex == 8)
{
    Ps2000timebase = 14;

    this.chart1.ChartAreas[0].AxisX.Interval = 65536000;

    if (Ps2000ScanRangecomboBox.SelectedIndex == 0)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -1;
        this.chart1.ChartAreas[0].AxisY.Maximum = 1;
        this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 1)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -2;
        this.chart1.ChartAreas[0].AxisY.Maximum = 2;
        this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 2)
    {
        range = 0;
```

```
        this.chart1.ChartAreas[0].AxisY.Minimum = -5;
        this.chart1.ChartAreas[0].AxisY.Maximum = 5;
        this.chart1.ChartAreas[0].AxisY.Interval = 1;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 3)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -10;
        this.chart1.ChartAreas[0].AxisY.Maximum = 10;
        this.chart1.ChartAreas[0].AxisY.Interval = 2;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 4)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -20;
        this.chart1.ChartAreas[0].AxisY.Maximum = 20;
        this.chart1.ChartAreas[0].AxisY.Interval = 4;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 5)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -50;
        this.chart1.ChartAreas[0].AxisY.Maximum = 50;
        this.chart1.ChartAreas[0].AxisY.Interval = 10;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 6)
    {
        range = 0;

        this.chart1.ChartAreas[0].AxisY.Minimum = -100;
        this.chart1.ChartAreas[0].AxisY.Maximum = 100;
        this.chart1.ChartAreas[0].AxisY.Interval = 20;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 7)
    {
        range = 1;

        this.chart1.ChartAreas[0].AxisY.Minimum = -200;
        this.chart1.ChartAreas[0].AxisY.Maximum = 200;
        this.chart1.ChartAreas[0].AxisY.Interval = 40;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 8)
    {
        range = 2;

        this.chart1.ChartAreas[0].AxisY.Minimum = -500;
        this.chart1.ChartAreas[0].AxisY.Maximum = 500;
        this.chart1.ChartAreas[0].AxisY.Interval = 100;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 9)
    {
        range = 3;

        this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
        this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
        this.chart1.ChartAreas[0].AxisY.Interval = 200;
    }

    else if (Ps2000ScanRangecomboBox.SelectedIndex == 10)
    {
        range = 4;
```

```
            this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
            this.chart1.ChartAreas[0].AxisY.Interval = 400;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 11)
        {
            range = 5;

            this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
            this.chart1.ChartAreas[0].AxisY.Interval = 1000;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 12)
        {
            range = 6;

            this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
            this.chart1.ChartAreas[0].AxisY.Interval = 2000;
        }

        else
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }
    }

    else if (Ps2000TimeIntervalcomboBox.SelectedIndex == 9)
    {
        Ps2000timebase = 16;

        this.chart1.ChartAreas[0].AxisX.Interval = 261144000;

        if (Ps2000ScanRangecomboBox.SelectedIndex == 0)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -1;
            this.chart1.ChartAreas[0].AxisY.Maximum = 1;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 1)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -2;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 2)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -5;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5;
            this.chart1.ChartAreas[0].AxisY.Interval = 1;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 3)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -10;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10;
            this.chart1.ChartAreas[0].AxisY.Interval = 2;
```

```
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 4)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -20;
    this.chart1.ChartAreas[0].AxisY.Maximum = 20;
    this.chart1.ChartAreas[0].AxisY.Interval = 4;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 5)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
    this.chart1.ChartAreas[0].AxisY.Interval = 10;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 6)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
    this.chart1.ChartAreas[0].AxisY.Interval = 20;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 7)
{
    range = 1;

    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
    this.chart1.ChartAreas[0].AxisY.Interval = 40;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 8)
{
    range = 2;

    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
    this.chart1.ChartAreas[0].AxisY.Interval = 100;
}

else if (ScanRangecomboBox.SelectedIndex == 9)
{
    range = 3;

    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
    this.chart1.ChartAreas[0].AxisY.Interval = 200;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 10)
{
    range = 4;

    this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
    this.chart1.ChartAreas[0].AxisY.Interval = 400;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 11)
{
    range = 5;

    this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
    this.chart1.ChartAreas[0].AxisY.Interval = 1000;
}
```

```
        else if (Ps2000ScanRangecomboBox.SelectedIndex == 12)
        {
            range = 6;

            this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
            this.chart1.ChartAreas[0].AxisY.Interval = 2000;
        }

        else
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }
    }

    else
    {
        Ps2000timebase = 3;

        this.chart1.ChartAreas[0].AxisX.Interval = 40000;

        if (Ps2000ScanRangecomboBox.SelectedIndex == 0)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -1;
            this.chart1.ChartAreas[0].AxisY.Maximum = 1;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.2;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 1)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -2;
            this.chart1.ChartAreas[0].AxisY.Maximum = 2;
            this.chart1.ChartAreas[0].AxisY.Interval = 0.4;
        }

        else if (ScanRangecomboBox.SelectedIndex == 2)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -5;
            this.chart1.ChartAreas[0].AxisY.Maximum = 5;
            this.chart1.ChartAreas[0].AxisY.Interval = 1;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 3)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -10;
            this.chart1.ChartAreas[0].AxisY.Maximum = 10;
            this.chart1.ChartAreas[0].AxisY.Interval = 2;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 4)
        {
            range = 0;

            this.chart1.ChartAreas[0].AxisY.Minimum = -20;
            this.chart1.ChartAreas[0].AxisY.Maximum = 20;
            this.chart1.ChartAreas[0].AxisY.Interval = 4;
        }

        else if (Ps2000ScanRangecomboBox.SelectedIndex == 5)
        {
```

```
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -50;
    this.chart1.ChartAreas[0].AxisY.Maximum = 50;
    this.chart1.ChartAreas[0].AxisY.Interval = 10;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 6)
{
    range = 0;

    this.chart1.ChartAreas[0].AxisY.Minimum = -100;
    this.chart1.ChartAreas[0].AxisY.Maximum = 100;
    this.chart1.ChartAreas[0].AxisY.Interval = 20;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 7)
{
    range = 1;

    this.chart1.ChartAreas[0].AxisY.Minimum = -200;
    this.chart1.ChartAreas[0].AxisY.Maximum = 200;
    this.chart1.ChartAreas[0].AxisY.Interval = 40;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 8)
{
    range = 2;

    this.chart1.ChartAreas[0].AxisY.Minimum = -500;
    this.chart1.ChartAreas[0].AxisY.Maximum = 500;
    this.chart1.ChartAreas[0].AxisY.Interval = 100;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 9)
{
    range = 3;

    this.chart1.ChartAreas[0].AxisY.Minimum = -1000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 1000;
    this.chart1.ChartAreas[0].AxisY.Interval = 200;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 10)
{
    range = 4;

    this.chart1.ChartAreas[0].AxisY.Minimum = -2000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 2000;
    this.chart1.ChartAreas[0].AxisY.Interval = 400;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 11)
{
    range = 5;

    this.chart1.ChartAreas[0].AxisY.Minimum = -5000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 5000;
    this.chart1.ChartAreas[0].AxisY.Interval = 1000;
}

else if (Ps2000ScanRangecomboBox.SelectedIndex == 12)
{
    range = 6;

    this.chart1.ChartAreas[0].AxisY.Minimum = -10000;
    this.chart1.ChartAreas[0].AxisY.Maximum = 10000;
    this.chart1.ChartAreas[0].AxisY.Interval = 2000;
}

else
{
    range = 0;
```

```
            this.chart1.ChartAreas[0].AxisY.Minimum = -100;
            this.chart1.ChartAreas[0].AxisY.Maximum = 100;
            this.chart1.ChartAreas[0].AxisY.Interval = 20;
        }
    }

    //trigger channel

    if (trig2000CH == 0)
    {
        Ps2000channel_enable = 0;
    }

    else if (trig2000CH == 1)
    {
        Ps2000channel_enable = 1;
        Ps2000channel_ext = 0;
    }

    else if (trig2000CH == 2)
    {
        Ps2000channel_enable = 1;
        Ps2000channel_ext = 1;
    }

    else if (trig2000CH == 3)
    {
        Ps2000channel_enable = 1;
        Ps2000channel_ext = 2;
    }

    else if (trig2000CH == 4)
    {
        Ps2000channel_enable = 1;
        Ps2000channel_ext = 3;
    }

    else
    {
        Ps2000channel_enable = 0;
    }

    //trigger threshold

    if (Ps2000TriggerThresholdcomboBox.SelectedIndex == 0)
    {
        Ps2000threshold = 1000;
    }

    else if (Ps2000TriggerThresholdcomboBox.SelectedIndex == 1)
    {
        Ps2000threshold = 2000;
    }

    else if (Ps2000TriggerThresholdcomboBox.SelectedIndex == 2)
    {
        Ps2000threshold = 3000;
    }

    else if (Ps2000TriggerThresholdcomboBox.SelectedIndex == 3)
    {
        Ps2000threshold = 4000;
    }

    else if (Ps2000TriggerThresholdcomboBox.SelectedIndex == 4)
    {
        Ps2000threshold = 5000;
    }

    else
    {
        Ps2000threshold = 1000;
```

```
            }

            //sampling points

            if (Ps2000SamplingPointcomboBox.SelectedIndex >= 0)
            {
                Ps2000no_of_samples =
Convert.ToInt16(Ps2000SamplingPointcomboBox.Items[Ps2000SamplingPointcomboBox.SelectedInd
ex].ToString());
            }
            else
            {
                Ps2000no_of_samples = 1024;
            }

            //delay

            if (Ps2000DelaycomboBox.SelectedIndex == 0)
            {
                Ps2000delay = 0;
            }

            else if (Ps2000DelaycomboBox.SelectedIndex == 1)
            {
                Ps2000delay = 20;
            }

            else if (Ps2000DelaycomboBox.SelectedIndex == 2)
            {
                Ps2000delay = 40;
            }

            else if (Ps2000DelaycomboBox.SelectedIndex == 3)
            {
                Ps2000delay = 60;
            }

            else if (Ps2000DelaycomboBox.SelectedIndex == 4)
            {
                Ps2000delay = 80;
            }

            else if (Ps2000DelaycomboBox.SelectedIndex == 5)
            {
                Ps2000delay = 100;
            }



            else
            {
                Ps2000delay = 0;
            }

            //trigger direction

            if (Ps2000TriggerDirectioncomboBox.SelectedIndex >= 0)
            {
                Ps2000direction = TriggerDirectioncomboBox.SelectedIndex;
            }

            else
            {
                Ps2000direction = 2;
            }

            trackBarPs2000shift.Maximum = 5000;
            trackBarPs2000shift.Minimum = -5000;
            trackBarPs2000shift.SmallChange = 100;
            trackBarPs2000shift.LargeChange = 1000;
            Ps2000trackbarlabel1.Text = trackBarPs2000shift.Value.ToString();
            offset2000 = (short)trackBarPs2000shift.Value;
```

```
            trackBarPs2000time.Maximum = 5;
            trackBarPs2000time.Minimum = -5;
            trackBarPs2000time.SmallChange = 1;
            trackBarPs2000time.LargeChange = 1;
            Ps2000trackbarlabel2.Text = trackBarPs2000time.Value.ToString();
            timeoffset2000 = (short)trackBarPs2000time.Value;
        }


        private void Ps2000TriggerChannelcomboBox_Click(object sender, EventArgs e)
        {
            trig2000CH = Ps2000TriggerChannelcomboBox.SelectedIndex;
        }

        //configuration part end

    }
}
```

Appendix II: DLL Libraries

C# PicoDLL:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ImportsDll;

namespace picoDll
{
    public class picodll
    {
        public const int QUAD_SCOPE = 4;
        public const int DUAL_SCOPE = 2;
        private int _channelCount;
        private Imports.Range _firstRange;
        private Imports.Range _lastRange;

        short overflow;
        int max_samples;
        short required;
        int[] triggerChannelPropertiesArray = new int[6];
        short threshold;
        short ps4000_Ext_Max_Value;
        object j = null;
        string S = "";
        short i;
        short status;
        short handle_Renamed = 0;
        picodll pico;

        private picodll(short handle_Renamed)
        {
            // TODO: Complete member initialization
            this.handle_Renamed = handle_Renamed;
        }

        public picodll()
        {

        }


        //Channel A

        public void app( ref short[] values_a, ref short[] values_b, ref short[] values_c,
ref short[] values_d, ref int time_interval, int no_of_samples, short timebase, short
channel_ext, short channel_enable, short thresholdV, int direction, short delay, short range,
```

```
short offset, short timeoffset, bool type)
        {


            for (i = 0; i <= 5; i += 1)
            {
                j = picosetting.ps4000GetUnitInfo(handle_Renamed, S, 255, ref required, i);
                //Cells(18 + i, "G").Value = S
            }

            picosetting.ps4000SetChannel(handle_Renamed, 0, 1, 0, range);
            picosetting.ps4000SetChannel(handle_Renamed, 1, 1, 0, range);
            picosetting.ps4000SetChannel(handle_Renamed, 2, 1, 0, range);
            picosetting.ps4000SetChannel(handle_Renamed, 3, 1, 0, range);


            status = picosetting.ps4000GetTimebase(handle_Renamed, timebase, no_of_samples,
ref time_interval, 1, ref max_samples, 0);

            picosetting.ps4000SetDataBuffer(handle_Renamed, 0, ref values_a[0],
no_of_samples);
            picosetting.ps4000SetDataBuffer(handle_Renamed, 1, ref values_b[0],
no_of_samples);
            picosetting.ps4000SetDataBuffer(handle_Renamed, 2, ref values_c[0],
no_of_samples);
            picosetting.ps4000SetDataBuffer(handle_Renamed, 3, ref values_d[0],
no_of_samples);

            //ABOVE, BELOW, RISING, FALLING and RISING_OR_FALLING.
           // delay = 0;
            //Should be in the range of -100% to +100% of the period.

            ps4000_Ext_Max_Value = 32767;
            //ADC count

            threshold = (short)((thresholdV*1.0 / 20000/5) * ps4000_Ext_Max_Value);
            // Scale using +/- 5V range on Ext input

            status = picosetting.ps4000SetSimpleTrigger(handle_Renamed, channel_enable,
channel_ext, threshold, direction, delay, 200);

            status = picosetting.RunBlock(handle_Renamed, 0, no_of_samples, timebase, 0, 0);

            while (picosetting.IsReady(handle_Renamed) == 0)
            {
                picosetting.Sleep((0));
            }

            picosetting.ps4000Stop(handle_Renamed);

            short x = 0;
            status = picosetting.ps4000GetValues(handle_Renamed, 0, ref no_of_samples, 1, 0,
0, ref x);

            if (type == true)
            {
                for (i = 0; i <= (no_of_samples - 1); i++)
                {
                    values_a[i] = (short)(((values_a[i] * 1.0 / 32767) * 20000 * 5) * timeoffset
+ offset);
                    values_b[i] = (short)(((values_b[i] * 1.0 / 32767) * 20000 * 5) * timeoffset
+ offset);
                    values_c[i] = (short)(((values_c[i] * 1.0 / 32767) * 20000 * 5) * timeoffset
+ offset);
                    values_d[i] = (short)(((values_d[i] * 1.0 / 32767) * 20000 * 5) * timeoffset
+ offset);
                }
            }

            else
            {
                values_a[i] = (short)(((values_a[i] * 1.0 / 32767) * 20000) * timeoffset +
offset);
```

```
                values_b[i] = (short)(((values_b[i] * 1.0 / 32767) * 20000) * timeoffset +
offset);
                values_c[i] = (short)(((values_c[i] * 1.0 / 32767) * 20000) * timeoffset +
offset);
                values_d[i] = (short)(((values_d[i] * 1.0 / 32767) * 20000) * timeoffset +
offset);
            }


        }


        public string open()
        {
            string s = "";
            status = picosetting.ps4000OpenUnit(out handle_Renamed);

            if (status != 0)
            {
                s = "" + 0;
            }
            else
            {

                pico = new picodll(handle_Renamed);
                s = pico.GetDeviceInfo();
            }
            return s;
        }

        public short close()
        {
            short status = picosetting.ps4000CloseUnit((handle_Renamed));
            return status;
        }

        public string GetDeviceInfo()
        {
            string s = "";
            int variant = 0;
            string[] description = {
                          "Driver Version: ",
                          "    USB Version: ",
                          "    Hardware Version: ",
                          "    Variant Info: ",
                          "    Serial: ",
                          "    Cal Date: ",
                          "    Kernel Ver: "
                        };
            System.Text.StringBuilder line = new System.Text.StringBuilder(80);

            if (handle_Renamed >= 0)
            {
                for (int i = 0; i < 7; i++)
                {
                    short requiredSize;
                    Imports.GetUnitInfo(handle_Renamed, line, 80, out requiredSize, i);
                    if (i == 3)
                    {
                        line.Length = 4;
                        variant = Convert.ToInt16(line.ToString());
                    }
                    s = s + description[i] + line + "\t";
                }

                switch (variant)
                {
                    case (int)Imports.Model.PS4223:
                        _firstRange = Imports.Range.Range_50MV;
                        _lastRange = Imports.Range.Range_100V;
                        _channelCount = DUAL_SCOPE;
                        break;
```

```
                case (int)Imports.Model.PS4224:
                    _firstRange = Imports.Range.Range_50MV;
                    _lastRange = Imports.Range.Range_20V;
                    _channelCount = DUAL_SCOPE;
                    break;

                case (int)Imports.Model.PS4423:
                    _firstRange = Imports.Range.Range_50MV;
                    _lastRange = Imports.Range.Range_100V;
                    _channelCount = QUAD_SCOPE;
                    break;

                case (int)Imports.Model.PS4424:
                    _firstRange = Imports.Range.Range_50MV;
                    _lastRange = Imports.Range.Range_20V;
                    _channelCount = QUAD_SCOPE;
                    break;

                case (int)Imports.Model.PS4226:
                    _firstRange = Imports.Range.Range_50MV;
                    _lastRange = Imports.Range.Range_20V;
                    _channelCount = DUAL_SCOPE;
                    break;

                case (int)Imports.Model.PS4227:
                    _firstRange = Imports.Range.Range_50MV;
                    _lastRange = Imports.Range.Range_20V;
                    _channelCount = DUAL_SCOPE;
                    break;

                case (int)Imports.Model.PS4262:
                    _firstRange = Imports.Range.Range_10MV;
                    _lastRange = Imports.Range.Range_20V;
                    _channelCount = DUAL_SCOPE;
                    break;
            }
        }

        return s;
    }
  }
}
```

C# Pico2000Dll:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Pico2000Dll
{
    public class pico2000dll
    {
        short ps2000_handle;
        int ok;
        short overflow;
        int time_indisposed;
        short mv_a;
        short[] voltage_range = new short[9];

        public void app(ref short[] Values_a, ref short[] Values_b, ref short[] Values_c, ref
short[] Values_d, short timebase, int no_of_values, ref int time_interval, ref short time_units,
ref int max_samples, short channel_ext, short threshold, short direction, short delay, short
range, short offset, short timeoffset)
        {
            voltage_range[0] = 100;
            voltage_range[1] = 200;
            voltage_range[2] = 500;
            voltage_range[3] = 1000;
```

```
            voltage_range[4] = 2000;
            voltage_range[5] = 5000;
            voltage_range[6] = 10000;
            voltage_range[7] = 20000;

            if (ps2000_handle == 1)
            {
                Pico2000settings.ps2000_set_channel(ps2000_handle, 0, 1, 0, (short)(range +
3));
                Pico2000settings.ps2000_set_channel(ps2000_handle, 1, 1, 0, (short)(range +
3));

                Pico2000settings.ps2000_set_ets(ps2000_handle, 0, 0, 0);

                ok = Pico2000settings.ps2000_run_block(ps2000_handle, no_of_values, timebase,
0, ref time_indisposed);

                ok = Pico2000settings.ps2000_get_timebase(ps2000_handle, timebase,
no_of_values, ref time_interval, ref time_units, 1, ref max_samples);
                ok = Pico2000settings.ps2000_set_trigger(ps2000_handle, channel_ext,
threshold, direction, delay, 20000);

                if (ok == 0)
                {

                }

                if (ok > 0)
                {
                    ok = Pico2000settings.ps2000_get_values(ps2000_handle, ref Values_a[0],
ref Values_b[0], ref Values_c[0], ref Values_d[0], ref overflow, no_of_values);

                    mv_a = voltage_range[range];

                    for (int i = 0; i <= (no_of_values - 1); i++)
                    {
                        Values_a[i] = (short)((((Values_a[i] / 32767.0) * (mv_a)) * timeoffset
+ offset));

                        Values_b[i] = (short)((((Values_b[i] / 32767.0) * (mv_a)) * timeoffset
+ offset));
                    }
                }
            }
        }

        public void open()
        {
            ps2000_handle = Pico2000settings.ps2000_open_unit();

            if (ps2000_handle > 0)
            {

            }

            else
            {

            }
        }

        public void close()
        {
            if ((ps2000_handle > 0))
            {
                Pico2000settings.ps2000_stop(ps2000_handle);
                Pico2000settings.ps2000_close_unit(ps2000_handle);
                ps2000_handle = 0;
            }
            else
            {
                Pico2000settings.ps2000_close_unit(ps2000_handle);
            }
        }
```

```
    }
}
```

C# CalculateDll:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
    using AForge.Math;

namespace CalculateDLL
{
    public class Calculatedll
    {

       public double maximum(double[] y, int length)
        {
            double r;
            r = y[0];
            for (int i = 0; i < length; i++)
            {
                if (r < y[i])
                {
                    r = y[i];
                }
            }
            return r;
        }

       public double maxnumber(double[] y, int length)
       {
           double r;
           double n=0;
           r = y[0];
           for (int i = 0; i < length; i++)
           {
               if (r < y[i])
               {
                   r = y[i];
                   n = i;
               }
           }

           return n;
       }

       public double minimum(double[] y, int length)
       {
           double r;
           r = y[0];
           for (int i = 0; i < length; i++)
           {

               if (r > y[i])
               {
                   r = y[i];
               }
           }
           return r;
       }

       public Complex[] FFT(Complex[] y, FourierTransform.Direction n)
       {

          FourierTransform.FFT(y, n);
          return y;
```

```
      }
   }
}
```

MATLAB LSFDll:

LSFit1.m:

```
function y=LSFit1(k,sheet,p,dt)
xlRange='A13:A10000';
s=xlsread(k,sheet,xlRange);
sm=max(s);
sl=length(s);
pos=0;
for i=1 : p
   if (s(i)==sm)
      pos=i;
      break
   end
end
s=s(pos:sl);
ymax=s(1);
s=s/ymax;
sl=length(s);
t=(0:sl-1)';
t=t*dt*0.000000001;



f=fittype('exp(x/tau)*erfc(sqrt(x/tau))','independent',{'x'},'coefficients',{'tau
'});
fit1=fit(t,s,f,'StartPoint',[0.002],'Lower',[0],'Upper',[1]);
a=1;
tau=fit1.tau;
y=[a,tau];
```

LSFit2.m:

```
function y=LSFit2(k,sheet,p,dt)
xlRange='A13:A10000';
s=xlsread(k,sheet,xlRange);
sm=max(s);
sl=length(s);
pos=0;
for i=1 : p
   if (s(i)==sm)
      pos=i;
      break
   end
end
s=s(pos:sl);
ymax=s(1);
s=s/ymax;
sl=length(s);
t=(0:sl-1)';
t=t*dt*0.000000001;

f=fittype('(2*w*sqrt(x*tau)/(sqrt(3.1415926)*(2*w*x+1))+(1/(sqrt(2*w*x+1)^3))*exp
(x/tau/(2*w*x+1))*erfc(sqrt(x/tau)/(2*w*x+1)))','independent',{'x'},'coefficients
',{'tau','w'});
fit1=fit(t,s,f,'StartPoint',[0.002, 9000000000],'Lower',[0, 0],'Upper',[1,
10000000000]);
a=1;
tau=fit1.tau;
w=fit1.w;
y=[a,tau,w];
```

LSFitdeep.m:

```
function y=LSFitdeep(k,sheet,p0,p1,p,dt)
xlRange='A13:A10000';
s=xlsread(k,sheet,xlRange);
```

```
sm=max(s);
sl=length(s);
pos=0;
for i=1 : p
   if (s(i)==sm)
      pos=i;
      break
   end
end
if (p1+pos)>sl
   s=s(p0+pos:sl);
else
   s=s(p0+pos:p1+pos);
end
ymax=s(1);
s=s/ymax;
sl=length(s);
t=(p0:p0+sl-1)';
t=t*dt*0.000000001;

f=fittype('exp(x/tau)*erfc(sqrt(x/tau))','independent',{'x'},'coefficients',{'tau
'});
fit1=fit(t,s,f,'StartPoint',[0.002],'Lower',[0],'Upper',[1]);
a=1;
tau=fit1.tau;
y=[a,tau];
```

MATLAB DeNDll:

DeNtest.m:

```
function y=DeNtest(s)
lev = 5;

s_den = wden(s,'sqtwolog','s','one',lev,'sym6');
s_noise=s-s_den;
smax=max(s_den);
smin=min(s_den);
snmax=max(s_noise);
snmin=min(s_noise);
snr=(smax-smin)/(snmax-snmin);
y=snr;
```

DeNtestDN.m:

```
function y=DeNtestDN(s)
lev = 5;

s_den = wden(s,'sqtwolog','s','one',lev,'sym6');
y=s_den;
```

MATLAB FrequencyDll:

ffttest.m:

```
function y=ffttest(s,t)
s=s';
f=abs(fft(s-mean(s)));
[maxvalue,indexmax]=max(f);
frequency=indexmax*(1/t)/length(f);
amp=max(s);
amp1=min(s);
ptp=max(s)-min(s);
ssum=sum(s.^2);
rms1=sqrt(ssum/length(s));
y=[frequency,amp,amp1,ptp,rms1];
```

MATLAB PFilterDll:

LPfilter.m:

```
%Low Pass Filter design
function y=LPfilter(f,t,Fc)
```

```
 N = 5;   % Order
% Calculate the zpk values using the BUTTER function.
Fs=1/t;
Wn=Fc/(Fs/2);
[b,a] = butter(N, Wn,'low');

%Fs  sampling frequency
%Fc  cut-off frequency

f2 = filter(b,a,f);
y=f2;
```

HPfilter.m:

```
%High Pass Filter design
function y=HPfilter(f,t,Fc)
 N = 5;   % Order
% Calculate the zpk values using the BUTTER function.
Fs=1/t;
Wn=Fc/(Fs/2);
[b,a] = butter(N, Wn,'high');

%Fs  sampling frequency
%Fc  cut-off frequency

f2 = filter(b,a,f);
y=f2;
```

BPfilter.m:

```
%Band Pass Filter design
function y=BPfilter(f,t,Fc1,Fc2)
 N = 5;   % Order
% Calculate the zpk values using the BUTTER function.
Fs=1/t;
Wn=[Fc1 Fc2]/(Fs/2);

[b,a] = butter(N, Wn,'bandpass');

%Fs  sampling frequency
%Fc  cut-off frequency

f2 = filter(b,a,f);
y=f2;
```

NPfilter.m:

```
%Notch Pass Filter design
function y=NPfilter(f,t,Fc)

Fs=1/t;
wo=Fc/(Fs/2);
bw=wo/35;
[b,a] = iirnotch(wo,bw);
f1=filter(b,a,f);
y=f1;
```