การเติมหมู่ฟังก์ชันบนกราฟีนที่ได้จากการตกเคลือบด้วยไอเคมี

นายภพนิพิฐ ฉัตรมณีรุ่งเจริญ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาเคมี ภาควิชาเคมี
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2560
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

FUNCTIONALIZATION OF CHEMICAL VAPOR DEPOSITION-GROWN GRAPHENE

Mr. Bhobnibhit Chatmaneerungcharoen

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Chemistry

Department of Chemistry

Faculty of Science

Chulalongkorn University

Academic Year 2017

Thesis Title   FUNCTIONALIZATION OF CHEMICAL VAPOR DEPOSITION-GROWN GRAPHENE

By   Mr. Bhobnibhit Chatmaneerungcharoen

Field of Study   Chemistry

Thesis Advisor   Sakulsuk Unarunotai, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

---Dean of the Faculty of Science

(Associate Professor Polkit Sangvanich, Ph.D.)

THESIS COMMITTEE

---Chairman

(Associate Professor Vudhichai Parasuk, Ph.D.)

---Thesis Advisor

(Sakulsuk Unarunotai, Ph.D.)

---Examiner

(Professor Mongkol Sukwattanasinitt, Ph.D.)

---Examiner

(Assistant Professor Kanet Wongravee, Ph.D.)

---External Examiner

(Associate Professor Peerasak Paoprasert, Ph.D.)

ภพนิพิฐ ฉัตรมณีรุ่งเจริญ : การเติมหมู่ฟังก์ชันบนกราฟีนที่ได้จากการตกเคลือบด้วยไอเคมี (FUNCTIONALIZATION OF CHEMICAL VAPOR DEPOSITION-GROWN GRAPHENE) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ดร.สกุลสุข อุ่นอรุโณทัย, หน้า.

กราฟีนได้กลายเป็นวัสดุที่น่าอัศจรรย์ตั้งแต่ที่มีการค้นพบในปี ค.ศ. 2004 เนื่องจากมีสมบัติที่น่าสนใจ เช่น มีค่าการนำความร้อนและการนำไฟฟ้าที่เป็นเลิศ มีความยืดหยุ่น และมีความแข็งแรงสูง นักวิจัยทั่วโลกได้หันมาสนใจศึกษาอัญรูปคาร์บอนนี้กันมากขึ้น ในการสังเคราะห์กราฟีนให้มีคุณภาพสูง การตกเคลือบด้วยไอเคมีเป็นวิธีที่มีความเป็นไปได้ที่จะทำการผลิตในเชิงอุตสาหกรรม โดยทั่วไป แก๊สมีเทนความบริสุทธิ์สูงซึ่งนิยมใช้เป็นสารตั้งต้นในการสังเคราะห์กราฟีนมีราคาสูง ผู้วิจัยจึงมีแนวคิดที่จะใช้แก๊สอะเซทิลีนแทนแก๊สมีเทน นอกจากนั้น การตกเคลือบด้วยไอเคมีที่ความดันบรรยากาศน่าจะสามารถลดต้นทุนการผลิตได้อย่างมาก อย่างไรก็ตาม การสังเคราะห์จะต้องทำการหาสภาวะที่เหมาะสมใหม่สำหรับแหล่งคาร์บอนชนิดใหม่โดยอาศัยการออกแบบการทดลอง ขั้นแรก ทำการตรวจกรองปัจจัยที่มีผลต่อการสังเคราะห์กราฟีนด้วยเทคนิคการตกเคลือบด้วยไอเคมีด้วยการออกแบบการทดลองแบบแพล็กเค็ต–เบอร์มาน โดยพิจารณาจากพารามิเตอร์ 6 ชนิดที่ได้จากสเปกตรัมรามานของกราฟีน พบว่าอุณหภูมิที่ทำการตกเคลือบและผลรวมอัตราการไหลของแก๊สอาร์กอนและแก๊สไฮโดรเจนมีนัยสำคัญในการสังเคราะห์กราฟีนด้วยระดับความเชื่อมั่น 70 ถึง 80 เปอร์เซ็นต์ จากนั้นทำการศึกษาปัจจัยทั้งสองประกอบกับอัตราการไหลของแก๊สอะเซทิลีนโดยการสร้างแบบจำลองพื้นผิวตอบสนองด้วยการออกแบบการทดลองแบบบ็อกซ์–เบ็ห์นเคน หลังจากการวิเคราะห์การถดถอยพหุคูณจึงได้สมการแสดงความสัมพันธ์ระหว่างพารามิเตอร์ทั้งหมด 4 ชนิดที่ได้จากสเปกตรัมรามานของกราฟีนกับปัจจัยทั้งสามชนิด ซึ่งมีสัมประสิทธิ์การกำหนดระหว่าง 0.987 – 0.942 และทำให้ได้สภาวะการสังเคราะห์ที่เหมาะสมที่สุด ที่อุณหภูมิ 1050 องศาเซลเซียส อัตราการไหลของแก๊สอาร์กอน แก๊สไฮโดรเจน และแก๊สอะเซทิลีนเป็น 900, 100 และ 0.4 ลูกบาศก์เซนติเมตรต่อนาที ตามลำดับ จนถึงตอนนี้ กราฟีนมักจะถูกนำไปใช้เป็นตัวนำไฟฟ้าโปร่งใส ในขณะที่การนำกราฟีนไปใช้เป็นสารกึ่งตัวนำถูกจำกัดเนื่องด้วยกราฟีนไม่มีช่องว่างพลังงาน การดัดแปรกราฟีนด้วยพันธะโคเวเลนต์เป็นวิธีหนึ่งซึ่งสามารถสร้างช่องว่างพลังงานให้กราฟีนได้ ถึงแม้ว่าการเติมหมู่ฟังก์ชันบนกราฟีนด้วยเกลือไดแอโซเนียม โดยการแช่กราฟีนในสารละลายเกลือไดแอโซเนียมที่มีสารลดแรงตึงผิวจะทำได้ง่ายสำหรับกราฟีนชั้นเดียว จำนวนโมเลกุลที่เติมลงมีจำนวนไม่มากสำหรับกราฟีนที่หนามากกว่าหนึ่งชั้น ในที่นี้ ผู้วิจัยทำการศึกษาผลของวัสดุรองรับและแสง พบว่ากราฟีนที่ทำปฏิกิริยาเติมหมู่ฟังก์ชันบนวัสดุรองรับที่มีขั้วอย่างซิลิกอนไดออกไซด์บนซิลิกอนถูกปนเปื้อนด้วยเกลือไดแอโซเนียมที่เหลือจากปฏิกิริยา ในขณะที่การเติมหมู่ฟังก์ชันให้กับกราฟีนบนวัสดุรองรับที่ไม่มีขั้วอย่างทองแดงได้ผลิตภัณฑ์ที่สะอาดกว่า ผู้วิจัยยังพบว่าแสงในช่วงยูวี-เอจากหลอดแบล็กไลต์หรือดวงอาทิตย์มีบทบาทสำคัญในการเติมหมู่ฟังก์ชัน โดยสันนิษฐานว่าแสงในช่วงดังกล่าวทำให้เกิดอิเล็กตรอนและฟีนิลแคตไอออนจากกราฟีนและเกลือไดแอโซเนียม ตามลำดับ กระนั้น ปฏิกิริยาการเติมหมู่ฟังก์ชันสามารถเกิดได้ในที่มืดอย่างช้า ๆ โดยสรุป แสงสามารถเร่งปฏิกิริยาการเติมหมู่ไดแอโซเนียมบนกราฟีนได้

ภาควิชา	เคมี	ลายมือชื่อนิสิต	....
สาขาวิชา	เคมี	ลายมือชื่อ อ.ที่ปรึกษาหลัก	....
ปีการศึกษา	2560		

5772094923 : MAJOR CHEMISTRY

KEYWORDS: GRAPHENE / CHEMICAL VAPOR DEPOSITION / FUNCTIONALIZATION / DIAZONIUM / ACETYLENE / ATMOSPHERIC PRESSURE

BHOBNIBHIT CHATMANEERUNGCHAROEN: FUNCTIONALIZATION OF CHEMICAL VAPOR DEPOSITION-GROWN GRAPHENE. ADVISOR: SAKULSUK UNARUNOTAI, Ph.D., pp.

Graphene has been a wonder material since its discovery in 2004. Due to its interesting properties such as excellent thermal and electrical conductivities, flexibility and high strength, many researchers around the globe have turned their attention to this carbon allotrope. To synthesize graphene with high quality, chemical vapor deposition (CVD) is an industrially feasible method. Generally, methane is the most widely used carbon precursor. However, high purity grade is expensive in Thailand. Instead, acetylene should be a possible replacement. Moreover, operating CVD at atmospheric pressure would also dramatically reduce the cost. Still, growth conditions need to be optimized for new carbon sources using experimental design. First, seven factors in CVD process were screened by Plackett–Burman design and six Raman parameters of graphene were dependent variables. The results suggested that growth temperature and Ar-H_2 total flow rates were significant at confidence intervals of 70-80%. The two significant factors together with acetylene flow rate were then subject to constructing response surface model with Box–Behnken design. Four relationships were developed by multiple linear regression (MLR) with $R^2 = 0.987 - 0.942$ and the best quality graphene could be obtained at growth temperature of 1050 °C with Ar/H_2/C_2H_2 flow rates of 900/100/0.4 sccm. Until now, graphene has been mostly used as a transparent conductor. Its application as a semiconductor has been limited due to its zero bandgap. Covalent modification was suggested as a possible way to open the bandgap in graphene. Although functionalization of graphene with diazonium salts using surfactant in aqueous solution was trivial for single-layer graphene, degree of functionalization was relatively low in case of multilayer graphene. Herein, substrate and photoinduced effects were investigated. While functionalized graphene on polar SiO_2/Si substrate was contaminated with unreacted diazonium salts, functionalization of graphene on Cu growth substrate afforded the cleaner product. We also found that UVA light from a blacklight lamp or sunlight played important roles in functionalization presumably by generation of hot electrons and phenyl cations from graphene and diazonium salts, respectively. Nonetheless, the reaction could occur slowly without light. In conclusion, diazonium grafting could be enhanced under irradiation of light.

Department:	Chemistry	Student's Signature	....
Field of Study:	Chemistry	Advisor's Signature	....
Academic Year:	2017		

ACKNOWLEDGEMENTS

CONTENTS

LIST OF TABLES

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

LIST OF FIGURES

จุฬาลงกรณ์มหาวิทยาลัย

CHULALONGKORN UNIVERSITY

LIST OF ABBREVIATIONS AND SYMBOLS

%	:	Percentage
2D	:	Two-dimensional
3D	:	Three-dimensional
4NBDT	:	4-Nitrobenzenediazonium tetrafluoroborate
α	:	The probability of type I error
ANOVA	:	Analysis of Variance
APCVD	:	Atmospheric Pressure Chemical Vapor Deposition
Ar	:	Argon
Ar^+	:	Argon ion
Au	:	Gold
β	:	Model coefficient of
BN	:	Hexagonal boron nitride
C_2H_2	:	Acetylene
C_A	:	Raman scattering constant
Ca^{2+}	:	Calcium ion
CCD	:	Central Composite Design
CCF	:	Central Composite Face-centered design
CH_3CN	:	Acetonitrile
CH_4	:	Methane
cm	:	Centimeter
cm^{-1}	:	Reciprocal centimeter
CNT	:	Carbon Nanotube
Co	:	Cobalt
C_S	:	Raman scattering constant
Cu	:	Copper

CuO	:	Copper(II) Oxide
CV	:	Cyclic Voltammetry
CV	:	Cross Validation
CV-RMSE	:	Root-mean-square error of cross validation
DOE	:	Design of Experiment
EDS	:	Energy dispersive spectroscopy
EG	:	Epitaxy growth
E_L	:	Laser energy
Et_2O	:	Diethyl ether
eV	:	Electron volt
Fe	:	Iron
Fe^{3+}	:	Iron(III) ion
$FeCl_3$	:	Ferric chloride
FET	:	Field Effect Transistor
FWHM( )	:	Full width at half maximum
g	:	Gram
GaN	:	Gallium nitride
GC	:	Glassy Carbon
GO	:	Graphene oxide
H_2	:	Hydrogen gas
H_2O	:	Water
H_2SO_4	:	Sulfuric acid
HBF_4	:	Tetrafluoroboric acid
HCl	:	Hydrochloric acid
HfO_2	:	Hafnium(IV) oxide
HNO_3	:	Nitric acid
HOPG	:	Highly Ordered Pristine Graphite

hr	:	Hour
I( )	:	Intensity
iLO	:	In-plane longitudinal optical phonon mode
I_{on}/I_{off}	:	On-to-off current ratio
IR	:	Infrared light
iTO	:	In-plane transverse optical phonon mode
kΩ	:	Kiloohm
L_D	:	Interdefect distance
LOO	:	Leave-one-out
LOOCV	:	Leave-one-out cross validation
LPCVD	:	Low Pressure Chemical Vapor Deposition
LUMO	:	Lowest Unoccupied Molecular Orbital
m	:	Meter
MANOVA	:	Multivariate Analysis of Variance
MeOH	:	Methanol
MFC	:	Mass Flow Controller
min	:	Minute
mK	:	Millikelvin
ml	:	Milliliter
MLR	:	Multiple Linear Regression
mm	:	Millimeter
μm	:	Micrometer
mM	:	Millimolar
mm^2	:	Square micrometer
mS	:	Millisiemen
mW	:	Milliwatt
N$_2$	:	Nitrogen gas

NaNO$_2$	:	Sodium Nitrite
NBu$_4$BF$_4$	:	Tetrabutylammonium tetrafluoroborate
Ni	:	Nickel
nm	:	Nanometer
n	:	Number of experiments
NMR	:	Nuclear Magnetic Resonance
°C	:	Degree Celsius
O$_2$	:	Oxygen gas
Pa	:	Pascal
PMMA	:	Poly(methyl methacrylate)
Pos( )	:	Position
ppm	:	Parts-per-million
PSE	:	Pseudo Standard Error
R^2	:	Coefficient of Determination
r$_A$	:	Radius of circle area that structural is distorted by point defect
RDS	:	Rate Determining Step
rGO	:	Reduced Graphene Oxide
RMSE	:	Root Mean-Square-Error
RO	:	Reversed Osmosis
rpm	:	Round per minute
r$_S$	:	Radius of D-band scattering area
RSM	:	Response Surface Methodology
S	:	Siemens
s	:	Second
sccm	:	Standard Cubic Centimeter per Minute
SD	:	Standard deviation
SDS	:	Sodium dodecylsulfate

SET	:	Single Electron Transfer
Si	:	Silicon
Si_3N_4	:	Silicon nitride
SiC	:	Silicon carbide
SiO_2	:	Silicon dioxide
SiO_2/Si	:	Silicon wafer with silicon dioxide insulating layer
sq	:	Square area
TEM	:	Transmission electron microscope
UV	:	Ultraviolet
UV-A	:	Ultraviolet A
V	:	Volt
W	:	Watt
w/v	:	Weight by volume
$\bar{X}$	:	Mean value
XPS	:	X-ray Photoelectron Spectroscopy
XRD	:	X-ray Difraction
Ω	:	Ohm

CHAPTER I

INTRODUCTION

1.1 Graphene

Graphene is an allotrope of carbon which consists of sp^2-hybridized carbon atoms forming a 2D honeycomb structure [1]. Although it has been investigated since 1940 [2] , this monolayer sheet was predicted to be thermodynamically unstable under ambient condition [3-4]. In 1960, thin lamellae of graphitic carbon, currently named reduced graphene oxide, were prepared by chemical reduction of graphite oxide (GO) [5]. However, such lamellae are not equivalent to graphene. It was not until 2004 that the existence of graphene was proven.
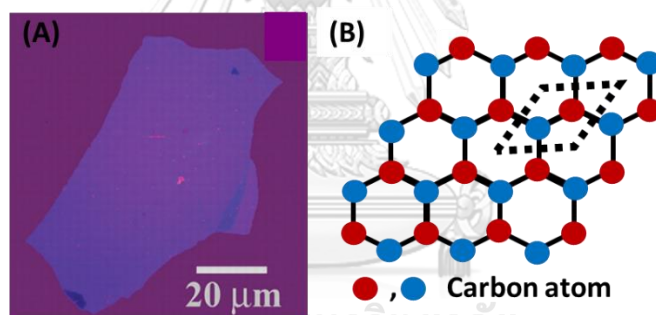
Figure 1.1 (A) Optical micrograph of graphene exfoliated from graphite by Scotch® tape method [6] (Copyright © 2004 American Association for the Advancement of Science) **(B)** Schematic top view of graphene lattice

Graphene was realized by mechanical exfoliation of graphite using Scotch® tape [6] (Figure 1.1(A)). The structure of exfoliated graphene is a single-atom-thick sheet of carbon atoms rearranged into a honeycomb pattern (Figure 1.1(B)). It is considered to be the thinnest material nowadays. Graphene has unique properties which outperform many of the preexisting materials. For example, electrical conductivity of graphene could reach 10^6 S cm^{-1} with extremely high mobility up to 200,000 cm^2 V^{-1} s^{-1} [7]. Meanwhile thermal conductivity of graphene can exceed 3000 W mK^{-1}, which is higher

than that of graphite [8]. Graphene absorbs only 2.3% of visible light making it almost transparent [9]. Due to the nature of strong covalent bonds between carbon atoms in graphene, the material is considered as the strongest one [10]. Nonetheless, it is flexible and has low density. Moreover, it also has large specific surface area of about 2630 $m^2 g^{-1}$ [11]. With some of the aforementioned properties, graphene is capable of being used in electronic and optoelectronic devices.
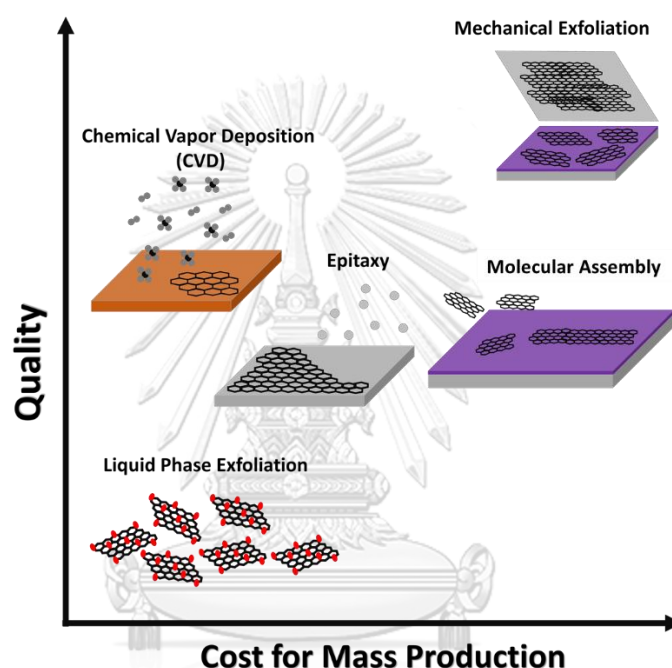
Figure 1.2 Comparison of graphene prepared by different methods in terms of quality and production cost

Despite having excellent properties, graphene sheets prepared by Scotch® tape method can only be produced in tiny amount without any control of film thickness and specific location of deposition. Moreover, the production process is not scalable, so it is not practical for industrial use. Alternative methods have been developed to realize graphene production in industrial scale. Chemical reduction of graphene oxide (GO) is one of the promising approaches because it can be carried out in large scale. According to Hummers' method [12], graphite as a starting material, is first oxidized in strongly acidic and oxidative condition to form graphite oxide which contains many

oxygen-containing functional groups on its basal plane and edge and is later exfoliated by ultrasonication to obtain graphene oxide (GO). GO is then chemically reduced to reduced graphene oxide (rGO) but some functional groups still remain in the structure [13]. Hence, rGO has lower quality, higher defect, smaller grain size and lower electrical conductivity than graphene obtained by Scotch® tape method. rGO can be applied to applications on which large surface area or further functionalization are required.

Due to the fact that graphene synthesized via Hummers' method, which is a top-down approach results in tiny flakes with low conductivity, many researchers has devoted their efforts to synthesize higher quality graphene from smaller precursors. Epitaxial growth (EG) of graphene from silicon carbide (SiC) was proposed as an alternative way [14]. In this method, graphene is formed at high temperature of 1200 – 1500 ℃ [14]. under ultrahigh vacuum when Si atoms evaporate from the surface of SiC and the remaining carbon atoms form a hexagonal graphitic structure. The size of epitaxial graphene is as large as that of the growth substrate. The electron mobility of this graphene could reach 5000 $cm^2 V^{-1} s^{-1}$ which is relatively high compared to rGO but still low compared to exfoliated graphene because of its polycrystallinity [14]. Although, this method affords high quality graphene, SiC is very expensive, and the process needs vacuum pumps.

Chemical vapor deposition (CVD) was then introduced in 2008 to synthesize graphene by feeding small carbonaceous precursor into high temperature tube furnace where metal substrate is heated inside [15]. The gaseous precursor decomposes on metal surface to form carbon film. When the reactor is cooled down, carbon atoms rearrange into a hexagonal pattern to form graphene. The process could be performed under both atmospheric and low pressures. Methane and copper are extensively used as a precursor and a substrate, respectively [16]. The size of CVD graphene is also as large as that of the substrate. The electron mobility ranges from 5000 to 12000 $cm^2 V^{-1} s^{-1}$ [17]. Moreover, this method is industrially scalable. Unfortunately, methane is relatively expensive for using as the precursor. As a result, other precursors were tested, for example, propane [18], acetylene [19] and ethanol [20] .

As illustrated in Figure 1.2, the best quality graphene can be obtained from Scotch® tape method but production cost of this method is relatively high. On the

other hand, liquid phase exfoliation is most economical, but the quality is not good. While CVD and Epitaxy can produce graphene with similar quality. However, CVD has less manufacturing cost than that of epitaxy. Therefore, CVD is most promising approach for industrial production of graphene.

1.2 Chemical Vapor Deposition

Chemical vapor deposition (CVD) is a well-known technique to produce a thin solid film. The film is formed by chemical reaction of vapor-phase precursors on the surface of substrate. The rate of reaction can be accelerated by heat (thermal CVD), UV light (photo-assisted CVD) or plasma (plasma-enhanced CVD). Generally, CVD is used to deposit metal oxide thin film by feeding volatile organometallic compounds on a reactive substrate. The deposited film is widely used in microelectronics and optoelectronics as protective, dielectric or coatings.
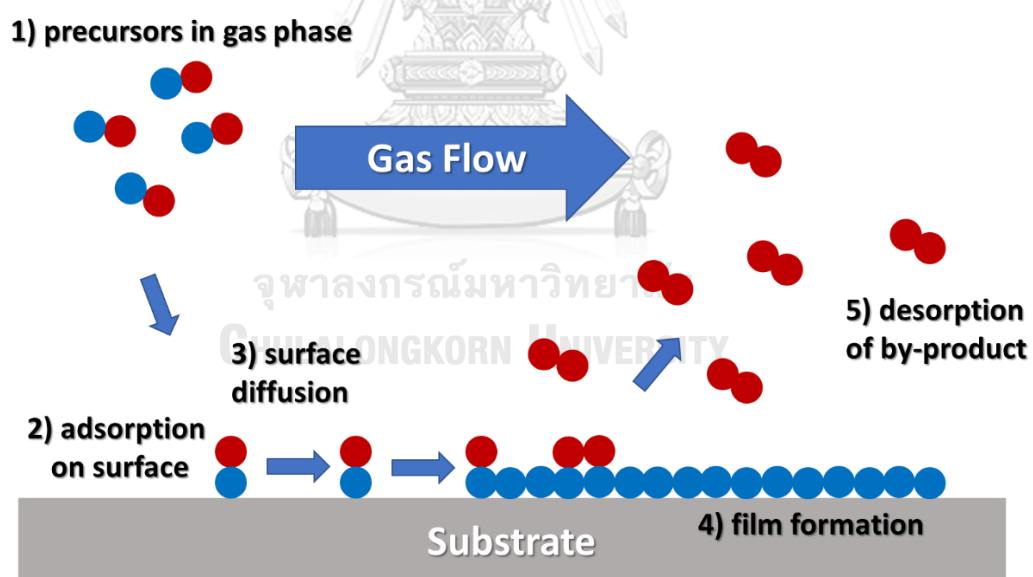
Figure 1.3 General mechanism of CVD processes

In CVD processes, volatile precursors approach and then adsorb onto the surface. The adsorbed precursor could diffuse and transport on surface. At an optimum state, reaction occurs and cleaves some fractions of the precursor molecule which

later desorb from the surface to vapor phase (Figure 1.3). Nucleation and growth of film depend on various factors such as temperature, purity of precursor and pressure.

Graphene can also be synthesized by CVD. This method could synthesize high quality graphene with large area by feeding carbonaceous compounds into the reactor. The precursors are usually hydrocarbon gases such as methane [10], acetylene [19, 21], propylene [22] or even volatile organic compound [23]. Substrate also plays an important role in the reaction. Copper and nickel are widely used because of low cost [24].

First success on CVD-grown graphene production was reported in 2008 [15]. Methane was fed into furnace at 1000 °C into which a nickel foil was placed. After graphene growth, the furnace was quickly cooled with a cooling rate 10 °C s^{-1}. The film was single layer.

Nowadays, methane and copper are the most popular carbon precursor and substrate, respectively, in graphene synthesis. However, methane is relatively expensive, and the gas needs to be imported. On the other hand, acetylene is a lot cheaper than methane. Thus, using acetylene, which is a by-product from petroleum manufacturing, instead of methane would benefit in cost reduction and help promoting graphene production. Therefore, acetylene is recognized as another promising candidate for synthesis of CVD-grown graphene.

1.3 Functionalization of graphene

Generally, defect-free exfoliated graphene has no band gap, thus it cannot be directly utilized as an active material in transistors. Also, even though the structure of CVD-grown graphene is not perfect, the band gap of as-grown graphene is still too small, and so it is commonly used as transparent electrodes

Development of graphene-based electronic devices has been impeded due to the zero band gap nature of the material. To be used especially for electronic and optoelectronic applications, opening band gap is a prerequisite [25].

In order to open the band gap of graphene, breaking symmetry or intervalley mixing [26] are some possibilities. Sublattice symmetry breaking [27], magnetic effect

[28] and quantum confinement effect [29] are considered as causes of band gap opening. Chemical modifications are practical approaches to achieve band gap opening [25, 30]. Two approaches have been proposed including physisorption of organic compounds and chemisorption of reactive species.
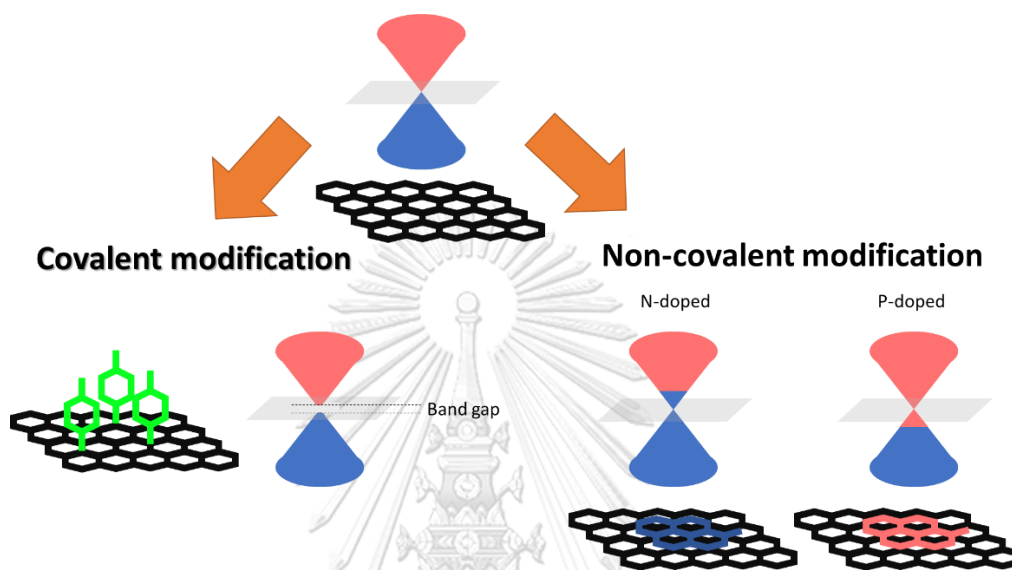
Figure 1.4 Schematic illustration of graphene modification methods: non-covalent modification and covalent modification

The physisorption of organic building blocks on graphene is considered as a non-covalent modification by which π–π interaction between graphene and organic dopants is prominent. Highly π–conjugated compounds such as perylene-3,4,9,10-tetracarboxylic dianhydride [31] and Co-phthalocyanines [32] or long-chained hydrocarbons such as alkylphosphonic acids [33] and pentacosadiynoic acid [34] were applied to graphene surface. According to strong van der Waals interactions, organic building blocks are periodically aligned on the surface. The charge transfer process between graphene and dopants shifting doping level in electronic structure changes charge carrier density resulting in band gap opening. Unfortunately, this method can achieve only a band gap of ~0.1 eV at maximum dopant concentration which is much lower than the desired band gap of 0.4 eV [30].

Another way to modify graphene is covalent modification by which sp^2-hybridized carbon atoms are converted to sp^3–hybridized carbon atoms. Developing methods for graphene functionalization is challenging for chemists, physicists and materials scientists. These methods require reactive species to react with chemically stable graphene. Many dopants which have successfully modified other carbon allotropes such as carbon nanotubes or fullerenes were also tried on graphene. Some of the most successful candidates are diazonium salts. Diazonium salt could also react with reduced graphene oxide (rGO), CVD-grown graphene or even exfoliated graphene [35]. Until now, the band gap of 0.36 eV has been achieved by functionalization of graphene [25]. Unfortunately, only few diazonium salts have been studied.

1.4 Scope of this research

1. Parametric study and optimization of CVD processes for graphene synthesis using acetylene as precursor under atmospheric pressure
2. Chemical functionalization of CVD-grown graphene with reactive diazonium salts

1.5 Objectives of this research

1. To optimize CVD processes for graphene synthesis using acetylene as precursor under atmospheric pressure
2. To develop methods for functionalization of graphene using diazonium salts

CHAPTER II

THEORY AND LITERATURE REVIEW

2.1 Graphene growth using acetylene in Chemical Vapor Deposition

Acetylene (C_2H_2), systemically called ethyne, is colorless, odorless and flammable gas consisting of two carbon atoms bonded together with a triple bond while each atom is terminated with a hydrogen atom (Figure 2.1(A)). It has been produced by cracking of petroleum products as a side product or by hydrolysis of calcium carbide (CaC_2). Generally, it is used as a fuel in a cutting torch because its flame can reach about 3100 °C when mixed with oxygen (O_2). Acetylene can be kept in the form of calcium salt which is easier and safer to handle. This ionic salt could generate acetylene by adding water (H_2O). However, acetylene cannot be kept in copper (Cu) or Cu alloy container because copper (Cu) can catalyze decomposition of acetylene.
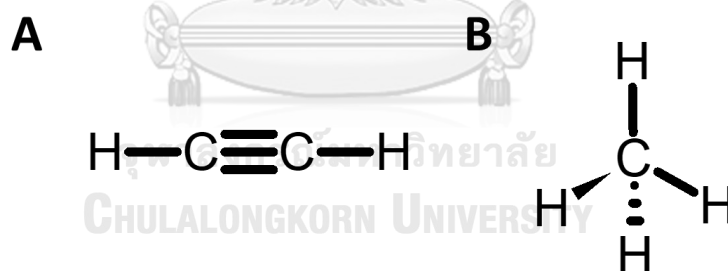
Figure 2.1 Chemical structures of **(A)** acetylene and **(B)** methane

Acetylene is a potential replacement of methane (Figure 2.1(B)) as a carbon precursor for graphene production because acetylene could reduce defect density in as-grown graphene [36]. However, acetylene has higher pyrolysis rate compared to methane. Therefore, graphene produced from acetylene tends to have high defect and small grain size.

Table 2.1 List of publications related to graphene synthesized from acetylene during 2009 – 2017 [19, 37-47]

Year	Authors	Type of CVD	Pressure	Substrate	Growth Temperature (°C)	Quality	No. Layer
2009	Kumar et al.	Thermal	LP	Ni film on SiO_2/Si	750	Defect-free	1
2010	Rümmelli et al.	Thermal	LP	MgO	325	High defect	2-10
2010	Sato et al.	Thermal	LP	Fe film on SiO_2/Si	650	Defect-free	Few
2011	Banerjee et al.	Thermal	LP	Co film on SiO_2/Si	800	Low defect	1
2011	Sun et al.	Thermal	LP	Si_3N_4	1000	High defect	Mono – few
2011	Hofmann et al.	Thermal	LP	Au-Ni film on SiO_2/Si	600	Low defect	1
2013	Qi et al.	Thermal	AP	Cu foil	1000	Low defect	2
2015	Cortés et al.	Thermal	LP	CuO film on SiO_2/Si	1000	High defect	1
2015	Hseih and Chen	Thermal	LP	Ni foil	600	Low defect	Few
2016	Yang et al.	Thermal	LP	CuO on Cu	1035	Low defect	1
2016	Yang et al.	Thermal	LP	CuO on Cu	1035	Low defect	1
2017	Muñoz et al.	Plasma-Enhanced	LP	SiO_2/Si	700	High defect	1
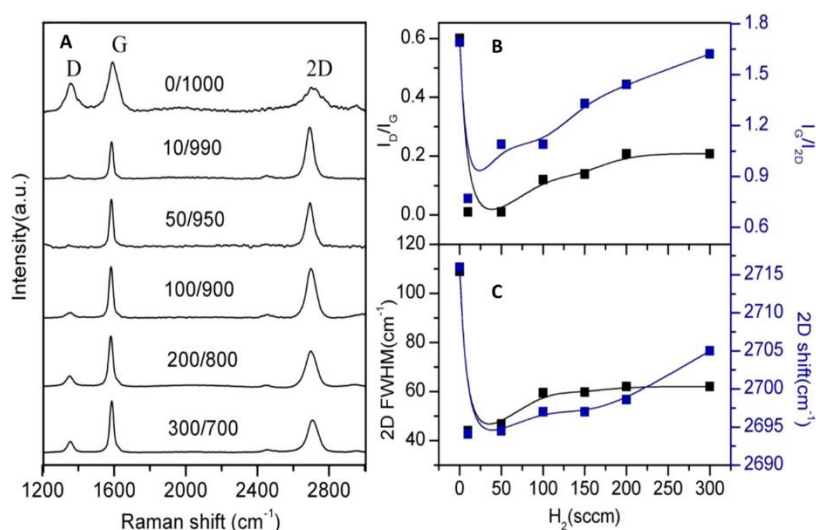
Figure 2.2 (A) Raman spectra, **(B)** I(D)/I(G) and I(G)/I(2D) and **(C)** FWHM(2D) and Pos(2D) of graphene synthesized with different H_2 flow rate [42] (Copyright © 2013, American Chemical Society)

CVD-grown graphene synthesized from this precursor has been developed since 2009 [19]. Thermal CVD under low pressure was the most extensively studied method to produce graphene with good-to-medium quality [19, 37, 39-41]. Metals with high carbon solubility such as Ni and Co were popular choices of substrate [19, 39] because they can retain carbon atoms on their bulk phase [48]. Moreover, growth step can perform at lower temperature (< 1000 °C). Metal alloys were alternative substrates to limit over-growth of graphene by blending the aforementioned metals with other metals with low carbon solubility [41]. Growing graphene on metal oxide, an insulating layer, is ideal for applications in device fabrication [38] because a transfer step is not required. Still, graphene grown on the insulating layer usually has high defect.

Despite the fact that low-pressure CVD (LPCVD) can produce high quality graphene, it takes a long reaction time and requires an expensive high-vacuum pump. Atmospheric pressure CVD (APCVD), on the other hand, is considered as it shortens the reaction time. Moreover, performing reaction under atmospheric pressure does not require any vacuum pump. Nevertheless, APCVD needs high operating temperature in order to synthesize high quality graphene as in LPCVD. APCVD process

for graphene synthesis was first achieved in 2013 on Cu substrate by Mei Qi et al. [42]. In the beginning, they investigated effects of growth temperature (850–1000 °C) and acetylene flow rate (1-12 sccm). They found that high temperature and low acetylene flow rate could reduce defect density. They also observed an effect of H_2/Ar flow rate ratio (Figure 2.2). Bilayer graphene could be synthesized under the flow rate ratio between 10/990 and 100/900. At higher ratios, graphene became multilayer. Consequently, they suggested that H_2 acts as an activator of the surface bound carbon and graphene etchant controlling morphology, nucleation density and nucleation size.

In addition to LPCVD and APCVD, plasma-enhanced CVD (PECVD) is another alternative way to reduce growth temperature. In 2017, Span research teams successfully synthesized graphene on SiO_2/Si through this approach [47]. However, as-grown graphene still had high defect. Furthermore, expensive and complicated instruments were needed.

Most of the CVD processes using acetylene as a precursor were performed at low pressure to reduce its activity and obtain the thinnest graphene. Substrate also played a crucial role on the quality of graphene. Although there was an effort to produce graphene directly on metal oxide, growing graphene on metal usually gave better results. Thermal CVD was more popular than other kind of CVD because of its lower cost. Nonetheless, it required high temperature. The presence of plasma could reduce the operating temperature.

2.2 Covalent functionalization of graphene

Covalently functionalized carbon materials have been tremendously utilized in sensing application [25]. Many methods for modification of carbon nanotubes (CNT), highly oriented pyrolytic graphite (HOPG) and glassy carbon (GC) have been studied. However, unlike other allotropes of carbon, graphene is not quite reactive because of its planarity. Reactive intermediates are required for forming adducts with

graphene due to its low reactivity. Converting sp^2-hybridized carbons to sp^3-hybridized carbons can alter graphene properties.

Free radicals are well known as very reactive intermediates. They can react with a variety of functional groups, for example, alkane, alkene, alkyne and carbonyl. Nevertheless, free-radical reactions cannot be controlled easily. The propagation reaction usually occurs and attributes to formation of oligomers or polymers. Generation of radical is also challenging because of its instability.

One of the most well-known compounds for radical formation is diazonium salt which undergoes electron transfer reaction. Then, the labile radical forms a covalent bond with graphene. Diazonium salt can be prepared from amine. Unfortunately, most of diazonium salts are not stable. It can decompose at room temperature. So, the diazonium salt should be kept under 5 °C or it should be generated in situ and used immediately. However, tetrafluoroborate salts of diazonium compounds are quite stable. They can be isolated in solid form and their solutions can be kept at room temperature.

Another possible method is decomposition of benzoyl peroxide. The grafting is activated by an unstable photogenerated benzoyl radical which then undergoes decarboxylation to form a phenyl radical. The radical can also form adduct with graphene as well as the radical generated from diazonium salt. Nonetheless, benzoyl peroxide is not persistent to light. Therefore, it is difficult to be stored without decomposition.

2.3 Diazonium-mediated functionalization

Surface functionalization with diazonium salts has been extensively studied for many purposes such as protection against corrosion or sensing application. The modified surfaces of carbon, silicon or metal have been achieved. This method could establish a strong C—C bond between surface of substrate and organic layer. Common carbon materials modified by diazonium salt are glassy carbon (GC), highly

Table 2.2 List of publications related to diazonium functionalization of graphene during 2009 – 2014 [49-61]

Year	Authors	Type of Graphene	Substrate	Diazonium salt	Solvent	Additive	Technique	Atmosphere	Temperature	Time
2009	Haddon et al.	EG	Si	4NBDT	CH_3CN	0.1 M NBu_4PF_6	Solution Chemistry	Ar	rt	20 hr
2010	Strano et al.	ME	SiO_2/Si	4NBDT	H_2O	1 % SDS	Solution Chemistry	Air	35 °C	7 hr
2010	Tour et al.	GNR	SiO_2/Si	4NBDT	CH_3CN	N/A	Solution Chemistry	N_2	rt	min to hr
2010	Choi et al.	ME	SiO_2/Si	4BrBDT	50 % v/v CH_3OH/H_2O	N/A	Solution Chemistry	N/A	rt	30 min
2011	Strano et al.	CVD	SiO_2/Si	4PPOBDT	H_2O	0.5 % SDS	Solution Chemistry	N/A	45 °C	8 hr
2011	Nouchi et al.	ME	HDMS-coated SiO_2/Si	4NBDT	H_2O	1 % SDS	Solution Chemistry	N/A	36 - 40 °C	7 hr
2011	Gao et al.	ME	SiO_2/Si	4NBDT	CH_3CN	NBu_4PF_6	Solution Chemistry	Ar	rt	10 hr
2012	Guo et al.	CVD	SiO_2/Si	4NBDT	H_2O	0.1 M H_2SO_4	Electrochemistry	Ar	rt	5 s
2012	Strano et al.	CVD	BN, Al_2O_3, OTS, SiO_2	4NBDT	H_2O	0.5 % SDS	Solution Chemistry	N/A	35 °C	16.5 hr
2013	Tour et al.	CVD	SiO_2/Si	4NBDT, 4DMABDT, 4MCBDT	CH_3CN	N/A	Solution Chemistry	N/A	N/A	5 – 60 min
2013	Ago et al.	CVD	PDMS	4NBDT	H_2O	N/A	Mechanics	N/A	N/A	10 s – 180 min
2013	Ruoff et al.	CVD	Si	4NBDT	CH_3CN	N/A	Solution Chemistry	Ar	N/A	180 min
2014	Daasbjerg et al.	CVD	Ni and Cu	AQDT	CH_3CN	0.1 M NBu_4BF_4 0.01 M NBu_4I	Electrochemistry	N/A	N/A	N/A

Table 2.3 List of publications related to diazonium functionalization of graphene during 2015 – 2017 [62-67]

Year	Authors	Type of Graphene	Substrate	Diazonium salt	Solvent	Additive	Technique	Atmosphere	Temperature	Time
2015	Mooste et al.	CVD	Ni	4NBDT, 4BrBDT, ABD, Fast Garnet GBC, Fast Black K	CH_3CN	0.1 M NBu_4BF_4	Electrochemistry	N/A	23 ± 1 °C	N/A
2015	Siaj et al.	CVD	SiO_2	4CBDCl	H_2O	0.5 M HCl	Electrochemistry	N/A	N/A	N/A
2016	Assaud et al.	CVD	Pt film on SiO_2/Si	4TPBDCl	H_2O	0.1 M HCl	Electrochemistry	N/A	N/A	N/A
2016	Palermo et al.	CVD	SiO_2/Si, PET and Quartz	4DCOBDT	H_2O	0.1 M H_2SO_4	Patterning and Electrochemistry	N/A	rt	N/A
2016	Siaj et al.	CVD	Cu	4NBDT	CH_3CN	N/A	Electrochemistry	N/A	N/A	N/A
2017	Kalbáč et al.	CVD	Cu	4NBDCl, 4CBDCl 4ECBDCl	30 % v/v H_2O/CH_3CN	N/A	Solution Chemistry	N/A	N/A	2 hr

oriented pyrolytic graphite (HOPG), carbon fiber, carbon blacks, carbon nanotubes and diamond. [68]

The grafting mechanism was studied by observing electrochemical behavior of diazonium salts. Most of grafting experiments were carried out in 0.1 M NBu_4BF_4 in CH_3CN or 0.1 M H_2SO_4 in H_2O. Diazonium salts have particularly low cathodic reduction potential. The aryl radical was considered as a reactive intermediate produced directly via concerted electron transfer from the electrode with dinitrogen cleavage. Then, the electrode surface was electro-grafted by the aryl radical to obtain C-surface bond. However, for most efficient grafting, surface must be clean and free from oxide. Deoxygenated solutions must be used in all processes to prevent oxide formation.

To characterize a modified surface, many methods have been used to proof the existence of organic layer. Fourier transform infrared spectroscopy (FTIR) is very useful in observing vibrational modes of grafted organic layers for example two strong bands of nitro group at ~1530 and 1350 cm^{-1}. This technique can also detect the adsorbed diazonium salts which N≡N bond stretching locates between 2300 and 2130 cm^{-1}. X-ray photoelectron spectroscopy (XPS) is one of the most powerful techniques in surface characterization. The information obtained from XPS spectra shows atomic composition at the surface and its oxidation state. After modification, surface morphology is certainly changed. Atomic force microscopy can reveal the formation of organic layer with variable height and roughness. Furthermore, energy dispersive spectroscopy (EDS) could show the dispersion and density of atom on the surface.

In the past decade, methods for diazonium functionalization on graphene were developed. Various types pf graphene such as mechanical exfoliated graphene [51, 53-54], epitaxial graphene [49], graphene nanoribbons [52] and CVD graphene [55-67], were functionalized. Since 2009, high quality graphenes such as mechanical exfoliated graphene or epitaxial graphene were employed. After 2011, CVD graphene was extensively applied because it holds considerable promise for the next generation of electronics. Various diazonium salts have also been functionalized. However, the structure of diazonium salts has been limited to only benzene core structure with different substituents.
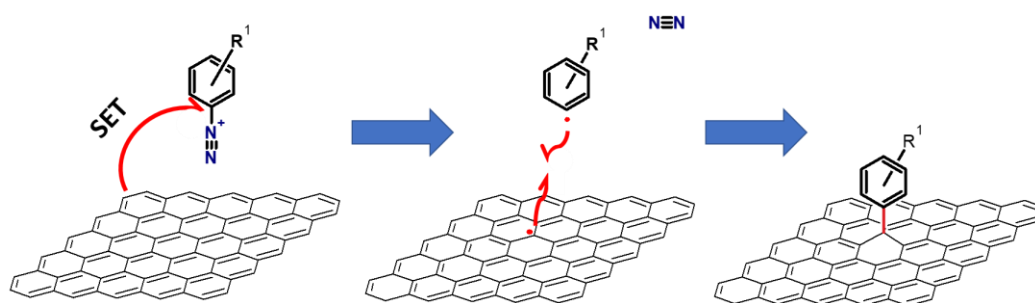
Figure 2.3 Schematic mechanism of diazonium grafting on graphene

The proposed reaction mechanism of diazonium grafting on graphene consists of two steps as shown in Figure 2.3. First, graphene transfers an electron to diazonium salt resulting in radical formation and nitrogen gas cleavage. This step was considered as rate determining step (RDS). Thereafter, it immediately forms a covalent bond with graphene radical.

Many approaches have been developed to enhance reaction rate of the first step including solution chemistry [49-55, 57, 59-60, 67], electrochemistry [56, 61-66] and mechanochemistry [58]. Solution chemistry is simple but it usually takes a long time. For electrochemical approach, it is the quickest way to perform functionalization within minute. However, this approach required expensive instruments with expertise in the field. In case of mechanochemical approach only one work was published, and this method was not favorable. Basically, the reactivity of graphene was enhanced by stretching graphene on flexible substrate.

The model chemical for developing methods for graphene functionalization method is 4-nitrobenzenediazonium salt. Its tetrafluoroborate salt was frequently used due to its high stability.

Reactivity studies are usually carried out by solution chemistry approach because there is no enhancement factor from external sources. The reactivity of functionalization solely depends on graphene and diazonium salt themselves. The single-layer graphene is more reactive than multilayer graphene. As the number of

graphene layer increases, the reactivity turns low [51]. Moreover, diazonium salt prefers reacting at the edge to basal plane [51]. In addition to the thickness of graphene, the substrate also plays an important role in the functionalization. Polar substrates such as SiO_2/Si and sapphire were found to enhance reactivity. On the other hand, functionalization on non-polar substrates, for instance, boron nitride, were nearly impossible [57]. Interestingly, the electrical conductivity of graphene deposited on non-polar substrates could be enhanced better than that deposited on polar substrates [69].

Raman parameters including I(D)/I(G), I(2D)/I(G), Pos(G), FWHM(G), Pos(2D) and FWHM(2D) frequently change after functionalization. I(D)/I(G) increases and I(2D)/I(G) decreases as shown in Figure 2,4(A) and 2.4(E), respectively, while Pos(G) and Pos(2D) shift to higher wavenumbers (Figure 2.4(C)). Moreover, G and 2D peaks usually broaden (Figure 2.4(B) and 2.4(D)).
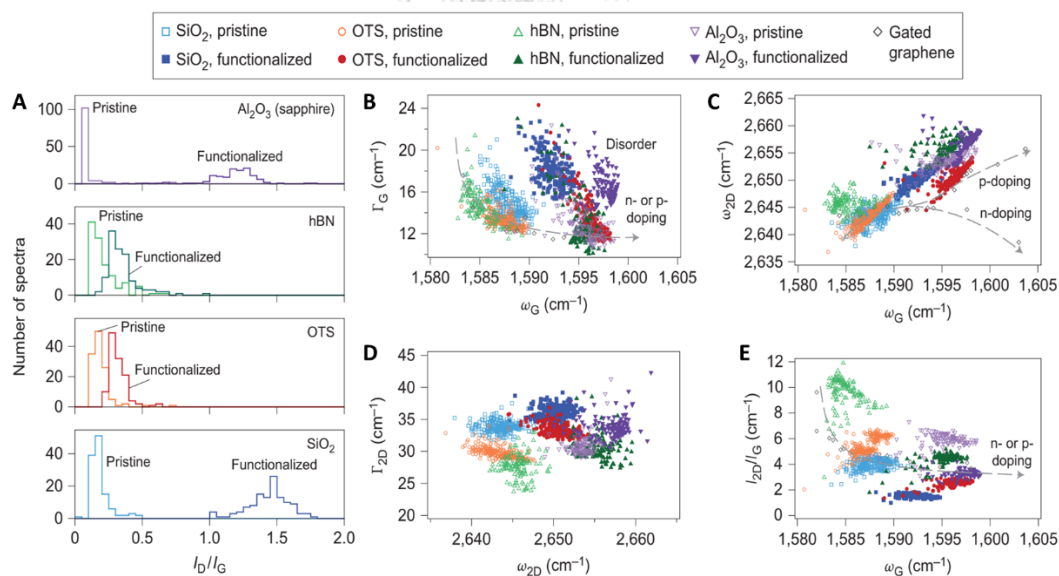
Figure 2.4 (A) Histograms of I(D)/I(G) ratios graphene before and after functionalization on different substrates **(B-E)** scatter plots of Raman peak parameters of graphene before and after functionalization on different substrates [57] (Copyright © 2012, Springer Nature)

Although solution chemistry approach exhibits low reactivity, it is simple and does not require any complicated instrument. According to the studies by Haddon [49] and Gao [54], O_2-free CH_3CN and NBu_4PF_6 were used as solvent and organic electrolyte, respectively. In addition, the reaction had to be performed in O_2-free atmosphere to prevent O_2 adsorption on graphene surface [49, 54]. Later in 2010, Strano and co-workers [51] reported that the reaction could perform in H_2O in the presence of sodium dodecyl sulfate (SDS) [51]. The surfactant molecules could bring diazonium species closer to graphene such that electron transfers could occur more easily. The use of mixed solvents was also interesting, however, it has not been well studied.

Electrochemical approach is very effective in functionalizing graphene. Basically, graphene is functionalized by cyclic voltammetry (CV) where graphene is utilized as the working electrode in a three-electrode system. The functionalization could be carried out in aqueous solution [56, 62, 64]. Due to its fast process, it has expanded the horizon of diazonium salts used in graphene functionalization from small benzene ring derivatives to large polyaromatic derivatives. The diazonium salt could be readily prepared in situ or used in the form of stable tetrafluoroborate salt. In addition to H_2O-based electrochemistry, electrochemical process in organic solvent was also possible [61, 63, 65].

The last approach is mechanochemical approach first introduced in 2013 [58]. This approach is the least studied one in spite of its simplicity. This requires a stretching tool and graphene needs to be transferred to a flexible substrate such as a PDMS stamp. So far, the only diazonium salt tested with this approach was 4-nitrobenzenediazonium tetrafluoroborate (4NBDT).

2.3 Raman spectroscopy for graphene characterization

Raman spectroscopy is one of the most powerful tools for characterization of graphene. The characterization is both fast and non-destructive. Moreover, Raman spectrum provides structural information as well as electronic information. However,

to obtain accurate information, spectral interpretation needs to be carefully done. Raman spectroscopy is well-known method for distinguishing carbon allotropes. While Raman spectrum of graphite was first measured over 40 years ago, it was not until 2006 for the first Raman spectrum of graphene to be recorded.
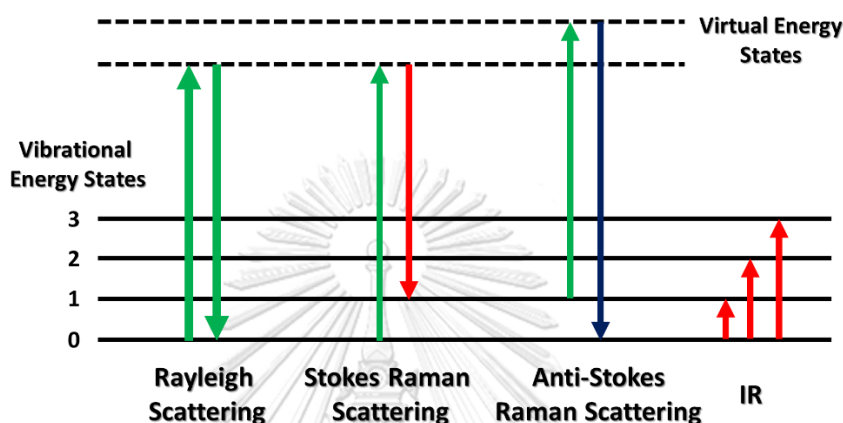
Figure 2.5 Schematic mechanism of Rayleigh scattering, Raman scattering and IR

In principle, Raman scattering occurs after an incident photon is absorbed by a Raman active material. The incident photon excited an electron to a virtual excited state and creates an electron-hole pair. The electron is scattered by phonon and loses or gain some energy before electron-hole recombination. The recombination generates a photon with lower or higher energy than the incident photon as depicted in Figure 2.5. Nonetheless, the scattering process rarely takes place. Therefore, the high intensity laser is required for Raman measurement. If the emitted photon has higher energy than the incident photon, the process is called Stoke scattering. On the other hand, if the emitted photon has lower energy than the incident photon, the process is called anti-Stoke scattering. However, most incident photons undergo elastic scattering process called Rayleigh process resulting in no energy shift. For general measurement, spectrum is recorded in Stoke region.

Two characteristic bands found in graphene include G and 2D (G') peaks as shown in Figure 2.6(B). G peak (graphitic peak), located approximately at 1600 cm^{-1}, is

generally observed in carbon materials consisting of sp^2-hybridized carbon such graphite, graphene, fullerene and carbon nanotube (Figure2.6(A)), while 2D peak appears around 2700 cm^{-1}. The intensity ratio between 2D peak and G peak (I(2D)/I(G)) reflects the number of graphene layers.
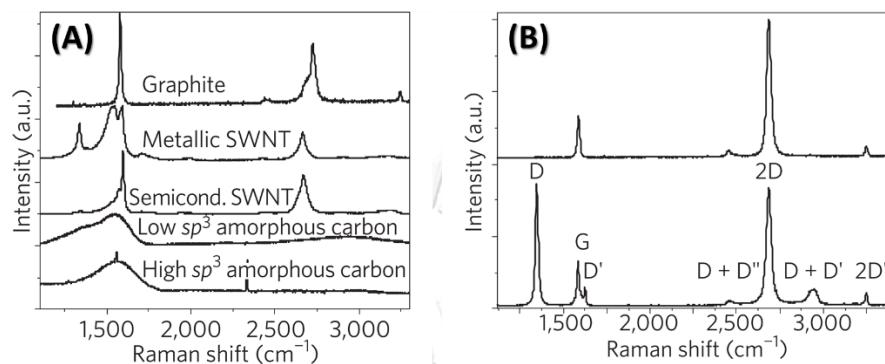
Figure 2.6 Raman spectra of **(A)** various carbon allotropes, **(B)** pristine graphene (top) and defective graphene (bottom) [70] (Copyright © 2013, Springer Nature)

G band is attributed to first order Raman scattering involving in-plane transverse optical phonon mode (iTO) or in-plane longitudinal optical phonon mode (iLO) at Γ point (Figured 2.7). G band is sensitive to doping and the number of layers. When graphene gets thicker, the position of G band shifts to lower Raman shift. It is more complicated to interpret the spectrum when doping is involved since the position shift is also dependent to the number of graphene layers. Nonetheless, doping level could be possibly indicated by observing FWHM(G).
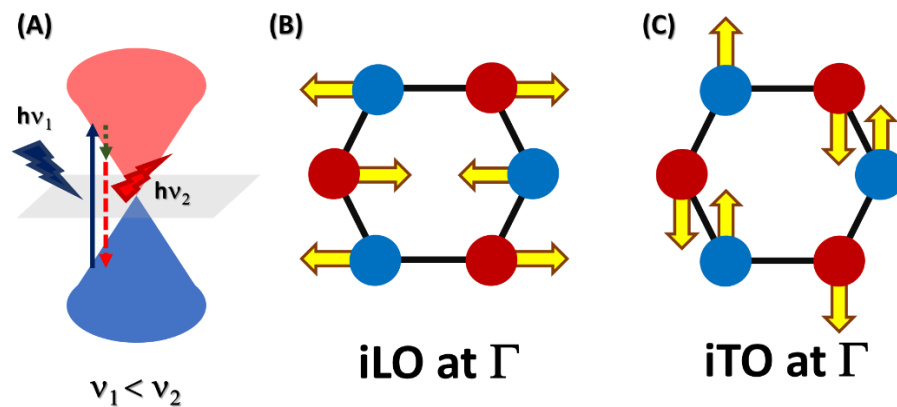
Figure 2.7 (A) Raman process of G band, sketch of phonon vibration of **(B)** inverse longitudinal optical mode at Γ and **(C)** inverse transverse optical mode at Γ

2D band is associated with double resonance process. It is a two-phonon scattering between K and K′ points (Figure 2.8). After forming electron-hole pair, electron is inelastically scattered by in-plane transverse optical phonon mode (iTO) at K point and then inelastically scattered again by in-plane transverse optical phonon mode (iTO) before recombination.
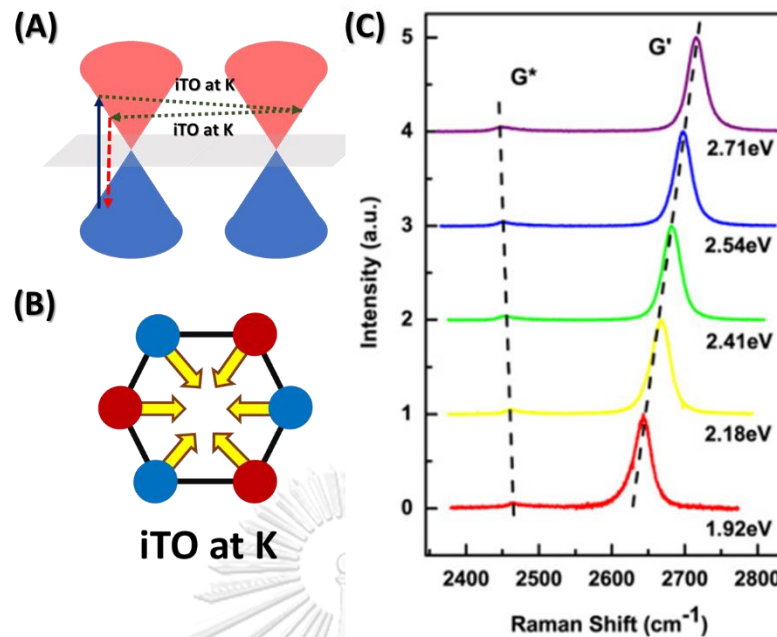
Figure 2.8 (A) Raman process of 2D peak, **(B)** sketch of phonon vibration of inverse transverse optical mode at K point and **(C)** Raman spectra showing shifts of 2D band according to different laser wavelengths [71] (Copyright © 2009 Elsevier B.V. All rights reserved.)

2D band is an important feature to estimate the number of graphene layers. When I(2D)/I(G) ratio is greater than 1, it suggests that graphene is single-layer. If I(2D)/I(G) ratio is approximately 1, it is bilayer. In case that I(2D)/I(G) is much lower than 1, the film is very thick such that its properties become similar to those of graphite. However, I(2D)/I(G) ratio can be easily disturbed by other factors such as doping and strain. The position and FWHM of 2D band (Pos(2D) and FWHM(2D)) are alternative choices for determining the number of graphene layers. However, the position is less sensitive to the number of layers than I(2D)/I(G). Therefore FWHM(2D) is a good choice because it is not only sensitive to the number of graphene layers, but it is also rarely affected by other factors. In addition to parametric variables such as I(2D)/I(G), FWHM(2D) and Pos(2D), shape of 2D band is a useful information to determine the number of graphene layers. By deconvoluting 2D band with Lorentzian

peaks, the number of peaks needed for fitting 2D band could support the estimation of number of graphene layers from FWHM(2D).

In case of pristine graphene obtained from mechanical exfoliation, only G and 2D bands are present in its Raman spectrum. However, graphene synthesized by other methods usually yields defective graphene. Raman spectrum of graphene with defects shows an additional peak around 1300 cm⁻¹ called D band.
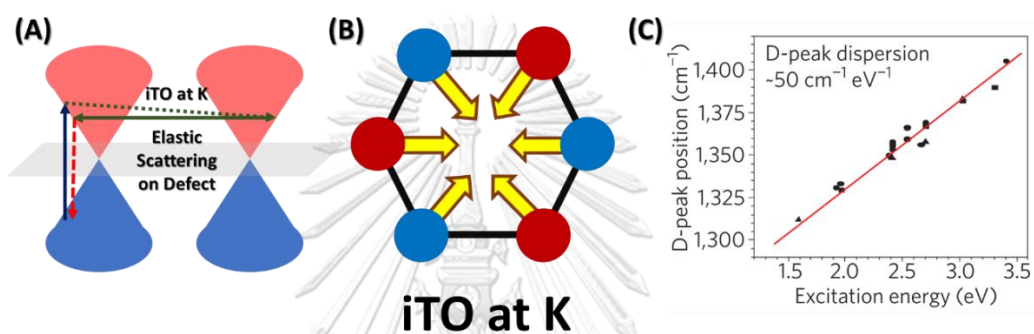
Figure 2.9 (A) Raman process of D band, **(B)** sketch of phonon vibration of inverse transverse optical mode at K point and **(C)** D peak dispersion with excitation energy [70] (Copyright © 2013, Springer Nature)

D band only appears when defects are present in graphene. The Raman process behind D band is like that behind 2D band. The electron scattered by iTO phonon in the first step as in 2D band is elastically scattered by defect before recombination (Figure 2.9). The energy loss is half of that in case of 2D band. That is the reason why G' band is alternatively called 2D band. D band is very dispersive to excitation energy as well as 2D band. The intensity ratio of D band to G band (I(D)/I(G)) is extensively used as the indicator to determine the amount of defects in graphene. In 2010, Lucchese and co-workers [72] successfully expressed the relationship between (I(D)/I(G)) and interdefect distance (L$_D$) as shown in Equation 2.1

$$\frac{I_D}{I_G} = C_A \frac{(r_A^2 - r_s^2)}{(r_A^2 - 2r_s^2)} \left[e^{-\pi r_s^2/L_D^2} - e^{-\pi(r_s^2 - r_s^2)/L_D^2} \right] + C_s \left[e^{-\pi r_s^2/L_D^2} \right] \qquad \text{... (Equation 2.1)}$$

where C_A = 4.2, C_S = 0.87, r_A = 3 nm and r_S = 1 nm. r_S is the radius of circle area that graphene structure is distorted by point defects while r_A is the radius of D-band scattering area. The other 2 parameters were fitted from experimental data (Figure 2.10(A)).
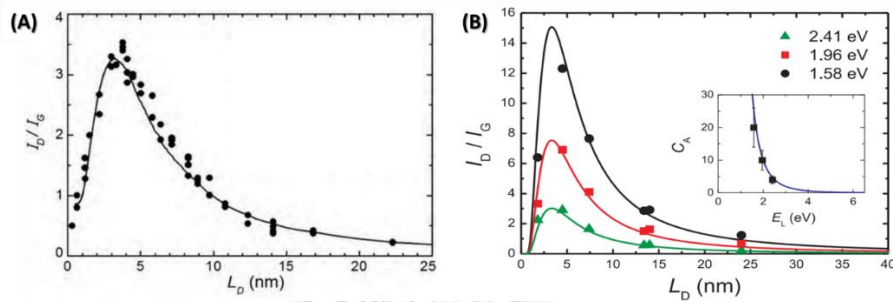
Figure 2.10 (A) The plot of I(D)/I(G) ratio against interdefect distance (L_D) for samples exposed to distinct Ar$^+$ doses [72] (Copyright © 2010 Elsevier Ltd. All rights reserved.) and **(B)** I(D)/I(G) ratio for monolayer graphene exposed to distinct Ar$^+$ doses obtained using different excitation lasers [73] (Copyright © 2011, American Chemical Society)

According to the dispersive property of D band, I(D)/I(G) ratios measured from different laser wavelengths are not comparable. As the laser energy increases, C_A, obtained by fitting, decreases. The relations between C_A and laser energy (E_L) was derived from experimental data (Figure 2.10(B)) as shown in equation (2.2)

$$C_A = \frac{160}{E_L^4} \quad\quad \text{... (Equation 2.2)}$$

By considering L_D > 10 nm and $L_D \gg r_A$ and r_S, equation (2.1) can be simplified as

$$\frac{I_D}{I_G} \approx C_A \frac{\pi(r_A^2 - r_S^2)}{L_D^2} \quad\quad \text{... (Equation 2.3)}$$

By substituting r_A = 3.1 nm, r_S = 1 nm and equation (2.2) into equation (2.3), the equation (2.3) can be written as

$$L_D^2(nm^2) = \frac{4.3 \times 10^3}{E_L^4}\left(\frac{I_D}{I_G}\right)^{-1} \quad\quad \text{... (Equation 2.4)}$$

By converting laser energy to laser wavelength, we have

$$L_D^2 (nm^2) = \frac{1.8 \times 10^{-9} \lambda_L^4}{E_L^4} \left(\frac{I_D}{I_G}\right)^{-1} \qquad \text{... (Equation 2.5)}$$

Those three major bands are commonly observed in graphene. However, there are some other bands. For example, in pristine graphene, 2D′ band could be observed around 3250 cm^{-1}. This band shows a weak signal compared to three major bands. Moreover, some phonon modes are activated by defects introduced on graphene including D band, D′ band, D + D′ band and D + D″ band. Their phonon modes related to inelastic or elastic scattering of electrons on defects. Upon introduction of high defects, these peaks could appear apparently.

2.4 Experimental design for process optimization

Generally, "One-factor-at-a-time" method is carried out to observe effect of the factor and optimize condition. The experiments are performed by varying single factor while other factors are fixed. The optimum condition can be achieved in case of no interaction between factors. The method is favored by scientists and engineers. However, in some systems for example a CVD system, interactions is present and affects responses.

As mentioned above, there are interactions between factors in CVD systems. One-factor-at-a-time method might not give the best condition. Experimental design and response surface methodology should be used instead.

Experimental design, also called design of experiments (DOE) is an experimental plan carefully designed in advance. The aim of experimental design is to explain or describe variation of data under certain conditions. Many designs have been developed, for example, Box-Behnken design, central composite design (CCD) and central composite face-centered design (CCF). Each design has its own advantages and disadvantages. Hence, one design should be carefully selected for each study.

Response surface methodology (RSM) is utilized to explore the relationships between factors. Moreover, maximum or minimum of responses under investigating

space could be acquired by plotting response surface: 3D-contour of response of one factor against another factor.

At the beginning, mathematical model is constructed from data obtained from experimental design by a technique called multiple linear regression (MLR). The appropriate mathematical model could predict response well and suggest precise effect of factors to response.

In brief, a function of response depending to factors is guessed. It is usually composed of linear terms, quadratic terms and interaction terms as shown in Equation (2.6).

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \beta_{ii} x_i^2 + \sum_{i=1}^{n} \sum_{i<j} \beta_{ij} x_i x_j \qquad \text{... (Equation 2.6)}$$

where β_0, β_i, β_{ii}, β_{ij} are coefficients, x_i and x_j are factors, and y is response. After obtaining a model equation, coefficient of determination (R^2) can be calculated. The coefficient indicates the proportion of variance in the response that could be predicted by factors. Thereby, good model must have coefficient of determination (R^2) close to 1.

However, the number of terms in function is limited by the number of experiments in design. Moreover, each term has different impacts on response. Significance of factors could be evaluated by analysis of variance (ANOVA) or student's t-test.

Some insignificant terms could be eliminated to improve predictive ability. Root mean-square-error (RMSE) is a tool to grade predictive ability. It can be calculated as shown in Equation (2.7), where y_i is actual response, $\hat{y}_i$ is predicted response and n is total number of experiments

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \qquad \text{... (Equation 2.7)}$$

However, good prediction in the model does not mean that it could accurately predict other experiments. Generally, further experiments are conducted, and actual response and predicted response are compared. If the mathematical model precisely predicts a response but poorly predicts responses from other

experiments, the model is overfitted. The overfit usually occurs when too many terms are added into the model.

There is another method for observing overfit, called cross validation. The data are divided into test set and training set. A model is developed from training set and RMSE are calculated from test set. The RMSE is specifically called root mean square error of cross-validation (CV-RMSE). K-fold cross validation (k-fold CV) is a very popular method. It divides data into k sets. One set is selected to be the test set. However, it requires large number of experiments. Leave-one-out cross validation (LOOCV) is an interesting method when the number of experiments is small. The test set consists of only one experiment. The disadvantage of this CV is excessive computation.

2.4.1 Plackett-Burman design

This design was introduced in 1946 by 2 British statisticians. The design is popular and economical for ruggedness testing. The main feature is that $4n-1$ factors can be included in the study with only $4n$ experiments, where n is a positive integer. For example, 12 experiments are needed for screening the effect of 11 factors. The advantage of this design is that the number of experiments is less than other designs with equal factors. In case that the number of factors does not match to $4n-1$, dummy factor can be integrated in the experiments. More dummy factors could improve estimation of error. Furthermore, each factor must be divided into two levels including high level and low level denoted as "+" and "−", respectively.

Table 2.4 The first set of factors for Plackett-Burman design

Number of factors	Set of factors
3	− + +
7	− − + + + + −
11	− − − − + + + − + + +
15	− − − + + + + + + + + − − − −
19	− − − + − + − + + + + − − + + − + + −

The set of factors for each experiment combines 2n−1 factors at low level with 2n factors at high level. The first sets of factors are available in the literature as shown in Table 2.4. Other experiments are assigned in a cyclical manner from the first experiment. For last experiment, all factors are set to low level. In case of 7 factors, set of factors for 8 experiments are listed in Table 2.5.

After all, there are 4 experiments with high level and 4 experiments with low level for each factor. The effect of each parameter was calculated as shown in Equation (2.8)

$$\text{Effect} = \frac{2[\sum y_{(+)} - \sum y_{(-)}]}{N}$$ ... (Equation 2.8)

where $y_{(+)}$ is a response when a given factor is at high level, $y_{(-)}$ is a response when a given factor is at low level and N is total number of experiments.

Table 2.5 Examples of the experimental design using Box-Behnken design with 3 factors

Run	X_1	X_2	X_3	X_4	X_5	X_6	X_7
1	−	−	+	+	+	+	−
2	−	−	−	+	+	+	+
3	+	−	−	−	+	+	+
4	+	+	−	−	−	+	+
5	+	+	+	−	−	−	+
6	+	+	+	+	−	−	−
7	−	+	+	+	+	−	−
8	−	−	−	−	−	−	−

There are many ways to decide which factor has a significant impact on response. The most effective method is F–test. However, this method requires the presence of dummy factor in the design. When no dummy factor is included, Lenth's pseudo standard error (PSE) is an alternative method. However, it assumes that the factor with least effect has no impact on the experiment. The variance of factor with least effect is presumably pure error variance.

2.4.2 Box-Behnken design

Box-Behnken Design introduced in 1960 by George E. P. Box and Donald Behnken is a three-level fractional factorial design. The advantage of this design is the limit of sample size as the number of factors grows. The sample size is sufficient to make regression including linear terms, interaction terms and quadratic terms. Three levels are denoted as "−1", "0", "+1" for low level, medium level and high level, respectively. This design is nearly rotatable because distances from the center point to other points are equal.
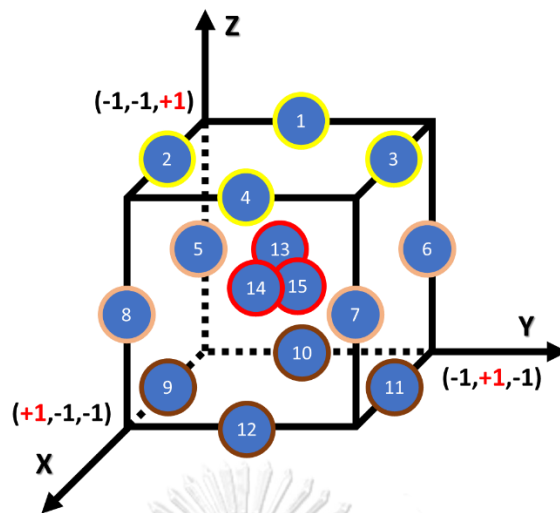
Figure 2.11 Graphical representation of Box-Behnken design

In case of 3-factored Box-Behnken model, 15 experiments are needed to carry out. As graphically illustrated in Figure 2.11, one point represents one experiment and three coordinates of each point indicate the value of three factors. 12 points are at the edge of the box while 3 points are at the center of the box. All 12 points have the same distance from the center while points at the corner are relatively far comparing to those 12 points. This might cause inaccuracy in prediction around that area. This problem could be solved by central composite design (CCD). However, due to the requirement of 5 levels per factor, CCD might not be suitable for our work.

CHAPTER III

EXPERIMENTAL

3.1 CVD instrumentation
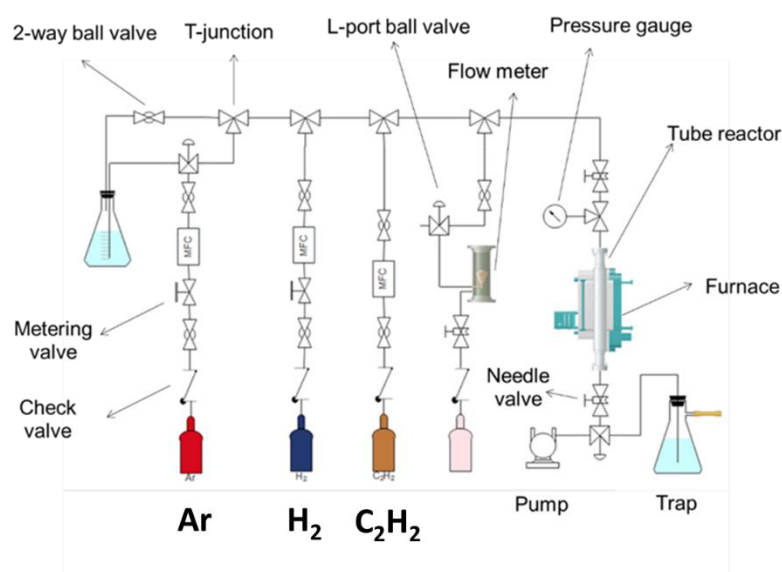
Figure 3.1 Schematic design of our CVD reactor

Our CVD reactor was built by assembling two components including a gas control panel and a furnace reactor (Figure 3.1). Three gases including argon (Ar), hydrogen gas (H_2) and acetylene (C_2H_2) were used for graphene synthesis. Acetylene serves as carbon precursor while H_2 and Ar act as catalyst and diluting gas, respectively. In addition to a gas regulator attached to each gas cylinder, check valves were installed to prevent backflow and contamination. 2-way ball valves were used to allow and stop gas flows. Gas flow rates were regulated by Thermal Mass Flow Controllers (MFCs). Metering valves between 2-way ball valves and MFCs have no function in the process. Before gas mixing at T-junction, other 2-way ball valves were installed to avoid backflow of the mixed gases. A photograph of our gas control panel is shown in Figure 3.2. Then, the mixed gases flowed through a needle valve, a pressure gauge and the tube reactor, respectively. At the exit end of the tube reactor,

there were another needle valve and L-port ball valve to selectively direct leftover gases to either a water trap or a vacuum pump.
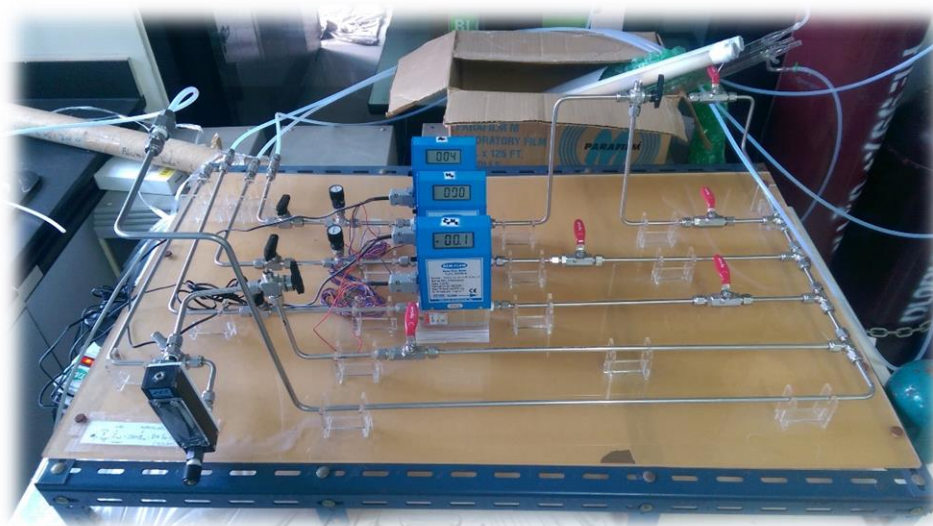
Figure 3.2 Our gas control panel

All gases were purchased from Praxair. Argon and hydrogen gases were obtained as ultra-high-purity (99.999 %), while, acetylene gas was of 99.6 % purity. The mixed gases were fed into a mullite tube which was placed in a commercial horizontal tube furnace (Lindberg/Blue M^{TM}) as shown in Figure 3.3. Leftover gases after the process were bubbled into water prior to be released into the atmosphere.

Figure 3.3 A CVD reactor

3.2 Optimization of CVD conditions

The process to synthesize graphene via CVD consists of 4 main steps including heating, annealing, growth and cooling. The temperature profile of each step is displayed in Figure 3.4. We note that the last 3 steps are very crucial to the quality of graphene. During annealing step, substrate is cleaned by hydrogen and its surface is reconstructed. Clean specific facets are important to obtaining high quality graphene. During growth and cooling steps, carbon atoms are deposited and rearranged on the substrate, respectively. Here, controlling nucleation rate and growth rates is the key to success.

In our work, a copper foil (purchased from Alfa Aesar as 99.8 % purity with a thickness of 25 μm) was cut into 1.5 x 1.5 cm^2 pieces and used as our graphene growth substrate. The substrate was extensively pre-cleaned before performing CVD process by sonicated in isopropanol for 5 min and acetone for 5 min, respectively. The substrate was then blow-dried by an Ar gun. The copper substrate was inserted into a mullite tube and positioned at center of the tube. After connecting the reactor to the gas lines, Ar was flushed into the tube at the flow rate of 1000 sccm for 5 min to purge air from the tube. Finally, composition of the gas mixture was adjusted before starting a CVD process.
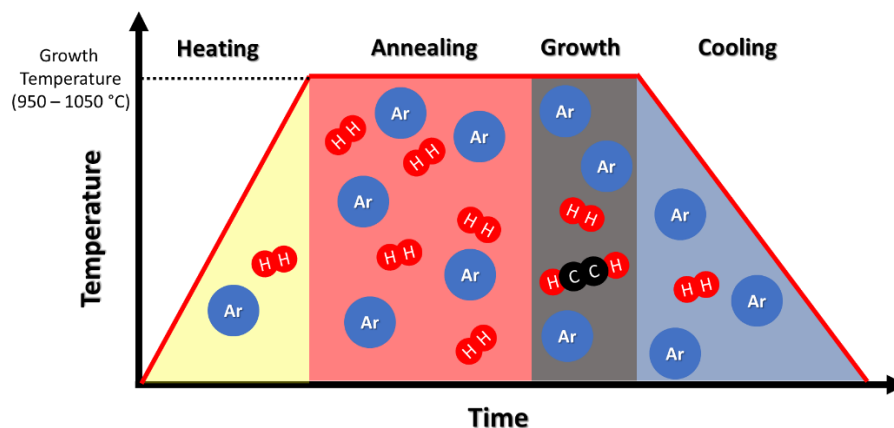
Figure 3.4 The temperature profile of a CVD process for graphene synthesis

Quality of graphene was estimated from Raman spectra. Every Raman spectrum was processed by Python script (appendix) to obtain 6 Raman parameters of graphene sample. They included intensity ratio of D peak to G peak (I(D)/I(G)), intensity ratio of 2D peak to G peak (I(2D)/I(G)), position of G peak (Pos(G)), full width at half maximum of G peak (FWHM(G)), position of 2D peak (Pos(2D)) and full width at half maximum of 2D peak (FWHM(2D)). In optimization process, all statistical calculations in Plackett-Burman screening experiment and response surface modeling were performed in Microsoft Excel and SPSS licensed by Chulalongkorn University.

3.2.1 Plackett-Burman screening experiments

Many parameters (or factors) in CVD process could affect the quality of as-grown graphene. Screening experiment was conducted to eliminate insignificant factors and focus on important factors.

Plackett-Burman design was used in this screening process due to its high performance with fewer experiments. 7 factors as listed in Table 3.1 were selected. The high and low levels were determined by our preliminary study and previous literatures. The coded factors were summarized in Table 3.2.

Table 3.1 CVD Factors for screening experiments

Label	Factors	Low Level (−)	High Level (+)	Unit
X_1	pre-growth composition (Ar/H$_2$)	150/150	300/300	sccm/sccm
X_2	annealing time	20	60	min
X_3	growth temperature	950	1050	°C
X_4	post-annealing composition (Ar/H$_2$)	450/50	900/100	sccm/sccm
X_5	acetylene flow rate	0.4	1.0	sccm
X_6	growth time	10	20	min
X_7	cooling time	5	10	°C/min

For each experiment, heating rate was set to 25 °C/min, while flow rate ratios of Ar to H$_2$ in pre-growth stage (heating and annealing steps) and post-annealing stage (growth and cooling steps) were set to 1:1 and 9:1, respectively. Once the temperature was below 500 °C, it was dropped uncontrollably under Ar atmosphere with a flow rate of 150 sccm. Each experiment was repeated thrice.

Table 3.2 Coded factors for Plackett-Burman screening experiments

Run	X_1	X_2	X_3	X_4	X_5	X_6	X_7
1	−	−	−	−	−	−	−
2	−	−	+	+	+	+	−
3	−	−	−	+	+	+	+
4	+	−	−	−	+	+	+
5	+	+	−	−	−	+	+
6	+	+	+	−	−	−	+
7	+	+	+	+	−	−	−
8	−	+	+	+	+	−	−

3.2.2 Response surface methodology

After screening experiments, 3 significant factors were selected for studying their effects and interaction effects on dependent variables. Box-Behnken design was used in this section because the design required less number of experiments (15 runs) and instrument limitation. Each factor was divided into 3 levels (−1, 0 and +1). −1 and +1 represented low and high level, respectively, from the screening experiments whereas 0 represented middle point between low and high level. The coded parameters in each experiment were listed in Table 3.3. The other factors in the previous experiments were determined by considering their effects on the 6 dependent variables obtained from Raman spectra.

3.3 Transfer method of graphene

For Raman measurement, graphene on Cu growth substrate had to be transferred to a SiO_2/Si (285 or 300 nm SiO_2, B-doped Si) substrate by PMMA-mediated transfer method (Figure 3.5). First, graphene on Cu was attached to a glass slide by Scotch® tape in order to prevent PMMA coating on back side of copper. As-grown or modified graphene on copper was coated with 4 % w/w PMMA (purchased from Aldrich with average molecular weight ~996000 Da) in toluene by spin-coating at 500 rpm for 5 s followed by 3000 rpm for 30 s. The PMMA-coated graphene on Cu was baked in the oven at 100 °C for at least 30 min to remove remaining solvent. The coated copper was detached from glass slide and 4 edges of the coated substrate were then trimmed. In addition, graphene on back side of substrate was removed by diluted HNO_3 and Cu substrate was etched by floating on 1 M $FeCl_3$. The floating PMMA/graphene hybrid film was subsequently transferred to beakers filled with RO water, 0.25 % w/w HCl, and milli-Q water, respectively, in order to remove contaminated Fe^{3+} ions. The film was finally taken up on SiO_2/Si substrate. The water between PMMA/graphene film and SiO_2/Si substrate was removed by heating in an oven at 80 °C for at least 1 hr. PMMA was removed with acetone 5 times and isopropanol once. The wet graphene film on the new substrate was blow-dried by an Ar gun.

Table 3.3 Coded factors for 15 experiments of 3-level Box-Behnken design

Run	X	Y	Z
1	−1	0	+1
2	0	−1	+1
3	0	+1	+1
4	+1	0	+1
5	−1	−1	0
6	−1	+1	0
7	+1	+1	0
8	+1	−1	0
9	0	−1	−1
10	−1	0	−1
11	0	+1	−1
12	+1	0	−1
13	0	0	0
14	0	0	0
15	0	0	0
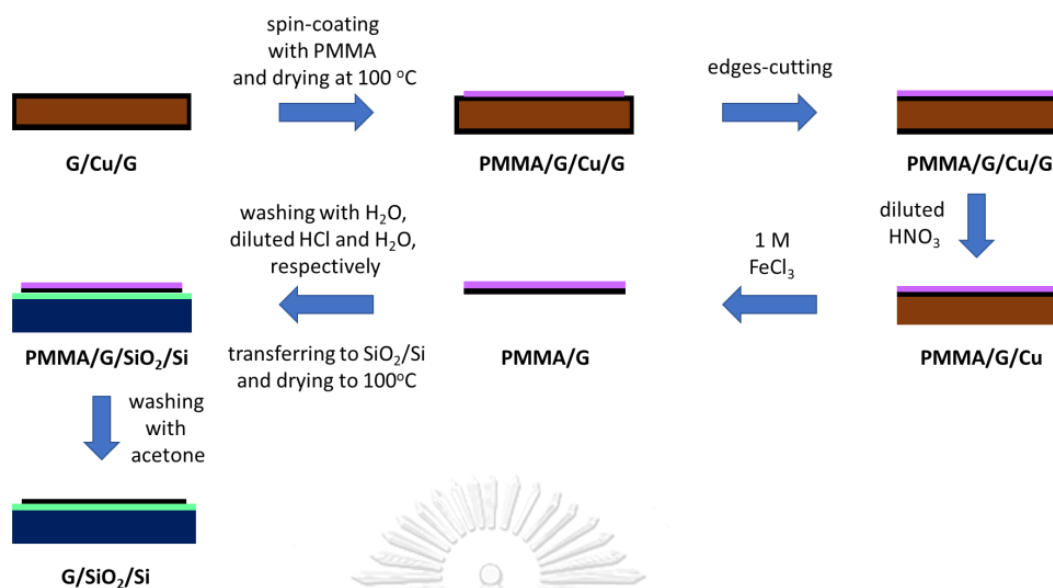
Figure 3.5 Schematic process of PMMA-mediated transfer method

3.4 Raman measurement

Raman spectra were recorded using DXR Raman microscope (Thermo scientific) equipped with a 532-nm excitation laser. All samples were analyzed under 100X-objective lens with laser spot size of 1 μm. The typical laser power was 10 mW with a pinhole aperture of 50 μm. The exposure time was set to 2 s with 2 accumulations. 250 spectra were measured on each sample.

3.5 Synthesis of 4-nitrobenzenediazonium tetrafluoroborate

4-Nitrobenzenediazonium tetrafluoroborate (4NBDT) was prepared from 4-nitroaniline (Figure 3.6). First, 4-nitroaniline (1.38 g, 10 mmol) was added into 50 % w/w HBF_4 solution (3.5 ml) and then, RO water (3.5 ml) was added into the mixture. Once the mixture became homogeneous, the solution was cooled down to 0 °C. The solution of $NaNO_2$ (0.695 g, 10.1 mmol) in water (1.5 ml) was added dropwise.
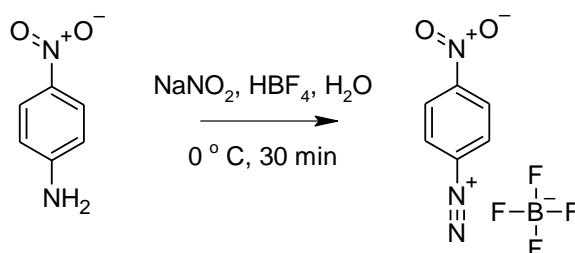
Figure 3.6 Synthesis of 4-nitrobenzenediazonium tetrafluoroborate (4NBDT) from 4-nitroaniline

The mixture was stirred for 30 min. The precipitate was filtrated through filter paper. The precipitate was washed with small amounts of cold methanol (MeOH) and copious amounts of diethylether (Et$_2$O). The product was recrystallized in acetone/Et$_2$O and then dried in vacuum. The diazonium salt was kept in refrigerator to prevent decomposition.

3.6 Functionalization of graphene

3.6.1 Conventional method

4-Nitrobenzenediazonium tetrafluoroborate (31 mg, 0.125 mmol) was dissolved in 5 ml of 1 % w/v SDS solution. The mixture was sonicated for 5 min. The diazonium solution was dropped into the 100-ml beaker which a magnetic bar and graphene on substrate (SiO$_2$/Si or Cu) were placed on the opposite sides. The solution was gently stirred at room temperature. After 7 hr of continuous stir, the solution was carefully removed by dropper. The functionalized graphene was washed thrice with Milli-Q water. Then, the film was soaked in Milli-Q water overnight in order to completely remove surfactant molecules from the surface. Finally, the film was dried by an Ar gun.

3.6.2 Photoinduced method

Figure 3.7 A fluorescence box installed with 2 blacklights on the top of the box

The same diazonium solution as in 3.6.1 was dropped into beaker in which graphene film was placed. Then the solution was irradiated inside a fluorescence box under blacklight (364 nm) for 30 min. The solution was poured out and the film was washed thrice with Milli-Q water and then soaked in Milli-Q water overnight. Finally, the film was dried by an Ar gun.

CHAPTER IV

RESULTS AND DISCUSSION

As aforementioned in chapter II, copper (Cu), nickel (Ni) and cobalt (Co) are promising substrates for graphene synthesis. At high temperature, carbon atoms can dissolve in these metals. Because Cu has lowest carbon solubility as stated in Table 4.1, self-limiting growth can be achieved easier than other metal. Hence, Cu is widely used as a substrate for graphene synthesis.

Table 4.1 Carbon solubility in nickel (Ni), cobalt (Co) and copper (Cu) [48]

Metal Substrate	% C atom	Temperature (°C)
Ni	0.19	600
Co	0.13	700
Cu	0.0007–0.0280	1000
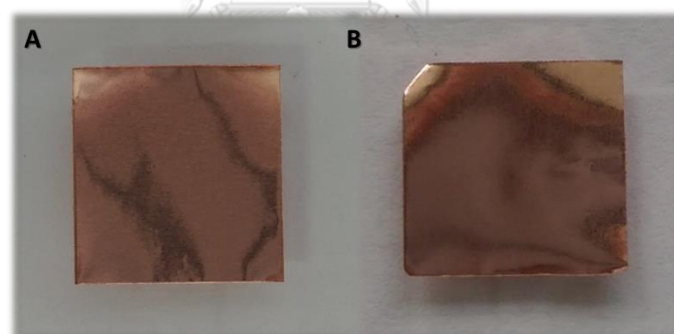
Figure 4.1 Photographs of 25-μm-thick Cu foils (A) before and (B) after graphene
growth by CVD

Cu from Alfa Aesar is bright reddish brown as shown in Figure 4.1(A). After graphene growth there was no noticeable change when observing by naked eyes (Figure 4.1(B)). Optical micrographs of Cu surface revealed that, Cu surface became smoother and formed microscopic grains after CVD process (Figure 4.2). However, the quality of graphene formed could not be verified by those micrographs.
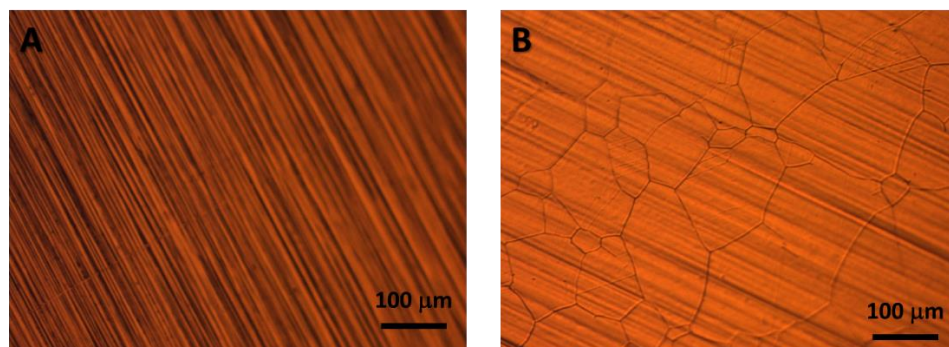
Figure 4.2 Optical micrographs of Cu foils (A) before and (B) after graphene growth by CVD

Due to the fact that Raman measurement of graphene on Cu gives low signal with strong fluorescent background, graphene needs to be transfer to SiO_2/Si substrate prior to the Raman measurement.
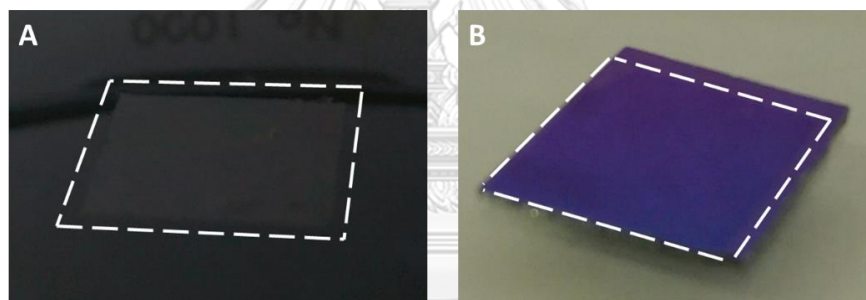
Figure 4.3 (A) PMMA/graphene hybrid film floating on 1 M $FeCl_3$ **solution (B)** graphene film transferred on SiO_2/Si substrate (285 nm SiO_2)

As described in chapter III, graphene was transferred by PMMA-mediated method. PMMA acts as a support for graphene after Cu-etching (Figure 4.3(A)). Finally, PMMA was removed by acetone after being taken up onto SiO_2/Si substrate. As illustrated in Figure 4.3(B), graphene can be clearly discriminated from SiO_2/Si substrate. On 285 nm SiO_2 wafer, graphene can be seen more easily than that on 300 nm SiO_2 wafer by naked eyes. However, graphene can be clearly observed on both wafers under microscope (Figure 4.4)
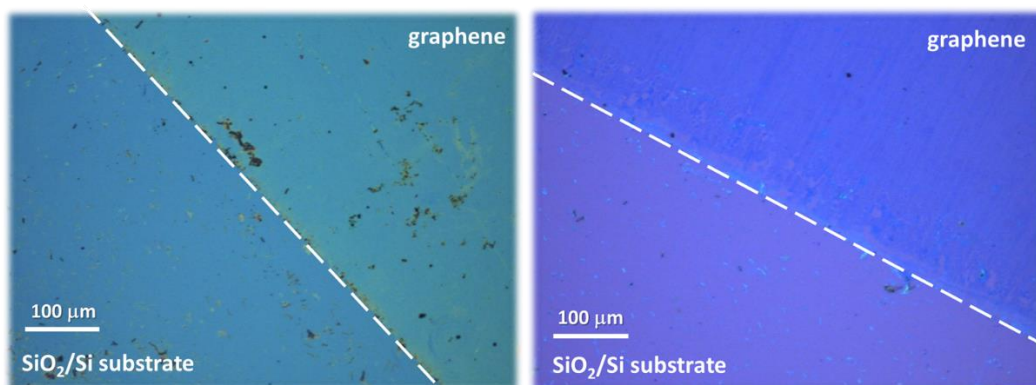
Figure 4.4 Graphene on **(A)** 300 nm and **(B)** 285 nm SiO$_2$ wafer

4.1 Plackett-Burman screening experiment

First, optimizing direction of 6 dependent variables, whether a higher or lower value is preferable, needed to be decided. I(D)/I(G) ratio indicating the amount of defect on graphene should be minimized since the quality of graphene strongly depends on this variable. On the contrary, I(2D)/I(G) ratio should be maximized because graphene with fewer layers is more desirable. Positions of G and 2D peaks are shifted by 2 reasons including increase in the number of layer and doping. As the number of layer increases, the red shift of G peak and the blue shift of 2D peak are observed. For doping, the position of G always goes red shift while the position of 2D can possibly go red or blue shift depending on the type of doping. Despite the fact that Ar, H$_2$ and C$_2$H$_2$ or combinations of them were continuously fed throughout the process, trace amount of air could possibly remain in the reactor. Therefore, as-grown graphene could probably be doped by N$_2$ or O$_2$. As a result, high quality pristine graphene usually shows G and 2D peak at higher and lower wavenumber, respectively, compared to our graphene. Moreover, 2D peak broadens as the number of layer increases while G peak broadens with increasing level of doping. Thus, the full width at half maximum of G and 2D peaks should be lessened. Favorable trends for each dependent variable are summarized in Table 4.2.

Table 4.2 Dependent variables and their optimizing directions

Dependent Variable	Optimizing Direction
I(D)/I(G)	Minimize
I(2D)/I(G)	Maximize
Pos(G)	Maximize
FWHM(G)	Minimize
Pos(2D)	Minimize
FWHM(2D)	Minimize

The experiments were conducted according to parameters in previous chapter. Each run was repeated thrice. Each sample was characterized by a confocal Raman spectrometer and 250 spectra were collected. The representative value of each parameter as shown in Table 4.3 was averaged from 750 spectra.

To decide which factors significantly influenced dependent variable, Lenth's Pseudo Standard Error (Lenth's PSE) was chosen because it could justify our decision without the need of a dummy factor (a factor without any effect on dependent variable). This method assumes that a factor with the least effect is the dummy.

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

Table 4.3 Summary of parameters from Raman spectra for Plackett-Burman experiments

Run	I(D)/I(G)			I(2D)/I(G)			Pos(G)			FWHM(G)			Pos(2D)			FWHM(2D)		
	$\bar{x}$	SD	n	$\bar{x}$	SD	n	$\bar{x}$	SD	n	$\bar{x}$	SD	n	$\bar{x}$	SD	n	$\bar{x}$	SD	n
1	0.94	0.17	3	0.41	0.05	3	1586.3	1.0	3	50.31	10.18	3	2696.9	0.3	3	86.22	4.39	3
2	0.27	0.05	3	1.05	0.10	3	1587.3	1.8	3	24.90	0.74	3	2691.7	1.5	3	56.88	1.02	3
3	1.05	0.04	3	1.05	0.04	3	1587.3	2.5	3	24.90	0.78	3	2691.7	2.0	3	56.88	0.13	3
4	1.10	0.07	3	0.30	0.07	3	1586.6	3.3	3	69.70	7.31	3	2694.2	5.2	3	98.67	4.55	3
5	1.03	0.05	3	0.44	0.05	3	1587.7	1.6	3	57.40	4.08	3	2694.7	1.8	3	88.28	2.28	3
6	0.19	0.04	3	0.99	0.28	3	1585.0	5.3	3	23.78	2.18	3	2691.0	6.3	3	56.79	3.79	3
7	0.21	0.07	3	1.21	0.05	3	1586.1	2.4	3	22.91	1.73	3	2689.3	2.6	3	52.90	3.96	3
8	0.28	0.05	3	0.94	0.15	3	1588.2	3.4	3	26.99	2.02	3	2690.5	2.4	3	59.15	6.52	3

As stated in Table 4.3, I(D)/I(G) ratios range from 0.19 to 1.10. This indicates that inter-defect distances vary from 11.5 to 27.6 nm. While I(2D)/I(G) ratios, interpreting the number of graphene layer, imply that only bilayer graphene up to multilayer graphene were produced under various growth conditions. The interpretation according to I(2D)/I(G) ratios was also consistent with that according to FWHM(2D) values, which can also be used to determine the thickness of graphene. When considering Pos(G) values, all of samples synthesized under these conditions were not doped by trace amount of nitrogen remained in the reactor. This is in consistent with a previous report [74] in that Pos(G) of N-doped graphene is lower than 1583 cm^{-1}. Interestingly, there is a remarkable relationship between I(D)/I(G) ratios and FWHM(G) values. In high defect samples, their G peaks are usually broadened. This suggests that as defect density increases, graphene becomes closer to amorphous carbon.
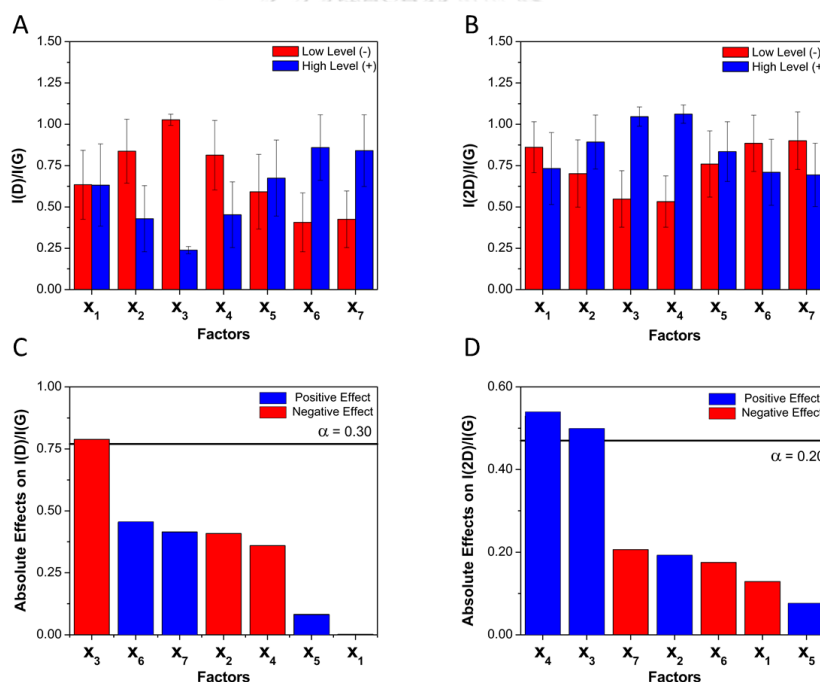
Figure 4.5 Bar charts showing average **(A)** I(D)/I(G) and **(B)** I(2D)/I(G) ratios at high and low levels of independent variables and Pareto charts of absolute effect of independent variables on **(C)** I(D)/I(G) and **(D)** I(2D)/I(G) values

After acquiring data from Raman spectra, they were processed as described in the literature [75]. As shown in Figure 4.5(A), annealing time (X_2), growth temperature (X_3) and Ar-H_2 composition during post-annealing steps (X_4) showed negative effect on I(D)/I(G). On the other hand, growth time (X_5) and cooling rate (X_6) positively affected I(D)/I(G). While Ar-H_2 composition before growth (X_1) and acetylene flow rate (X_5) had no much effect on I(D)/I(G). The Pareto chart in Figure 4.5(C) showed sorted absolute effect on I(D)/I(G). The most effective factor was growth temperature (X_3) followed by growth time (X_6), cooling rate (X_7), annealing time (X_2), Ar-H_2 composition during post-annealing steps (X_4), acetylene flow rate (X_5) and Ar-H_2 composition before growth (X_1), respectively. At 70% level of confidence, growth temperature (X_3) was the only significant factor affecting I(D)/I(G).

As seen in Figure 4.5(B), I(2D)/I(G) could be enhanced by increasing annealing time (X_2), growth temperature (X_3) and Ar-H_2 composition during post-annealing steps (X_4). On the contrary, increasing cooling rate (X_7), growth time (X_6) and Ar-H_2 composition before growth (X_1) resulted in decrease of I(2D)/I(G), while acetylene flow rate (X_5) did not make any noticeable change in I(2D)/I(G). We also found that growth temperature (X_3) and Ar-H_2 composition during post-annealing steps (X_4) had much higher impact on I(2D)/I(G) than the other affecting factors had at 80% level of confidence (Figure 4.5(D)).
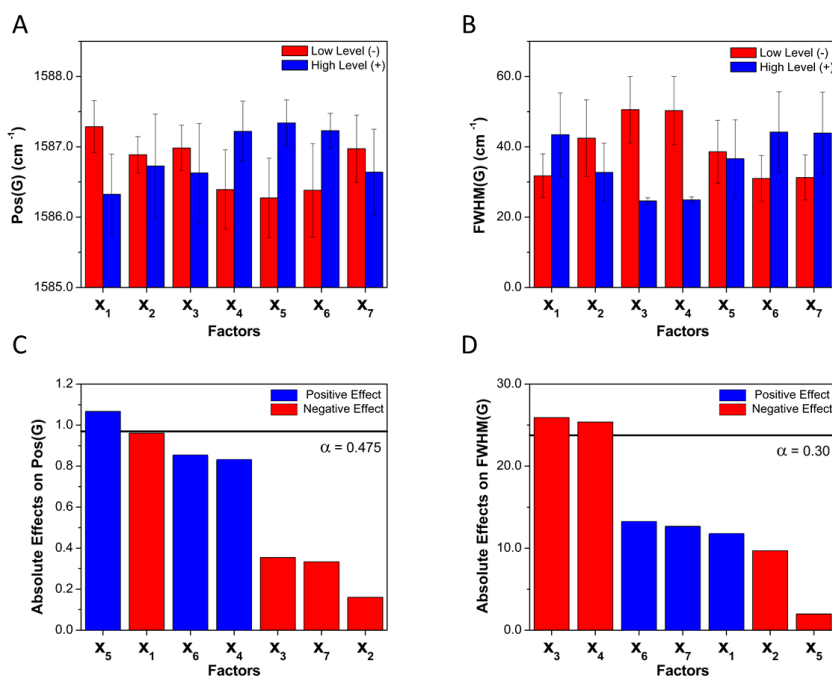
Figure 4.6 Bar charts showing average **(A)** Pos(G) and **(B)** FWHM(G) values at high and
low levels of independent variables and Pareto charts of absolute effect of
independent variables on **(C)** Pos(G) and **(D)** FWHM(G) values

Figure 4.6(A) shows that the growth temperature (X_3), cooling rate (X_7) and
annealing time (X_2) were clearly insignificant to Pos(G). While acetylene flow rate (X_5),
Ar-H_2 composition before growth (X_1), growth time (X_6) and Ar-H_2 composition during
post-annealing step (X_4) showed small effects on Pos(G). Nonetheless, only acetylene
flow rate (X_5) was predicted to be significant at 52.5% level of confidence (Figure
4.6(C)). However, effects of all factors on Pos(G) were less than the resolution of
Raman spectrometer (7 cm^{-1}). Therefore, the results were not valid.

FWHM(G), suggesting doping level on graphene, was negatively affected by
annealing time (X_2), growth temperature (X_3) and Ar-H_2 composition during post-
annealing steps (X_4) as shown in Figure 4.6(B). On the contrary, Ar-H_2 composition
before growth (X_1), growth time (X_6) and cooling rate (X_7) had positive influences on
FWHM(G), whereas acetylene flow rate (X_5) did not show any significant effect on
FWHM(G). Comparing absolute effects of independent variables, growth temperature

(X_3) and Ar-H_2 composition during post-annealing steps (X_4) were anticipated as significant factors at 70% level of confidence as shown in Figure 4.6(D). Growth time (X_6), cooling rate, (G) Ar-H_2 composition before growth (X_1) and annealing time (X_2) had similar absolute effects.

Information on peak 2D is very useful for indicating the thickness of graphene. I(2D)/I(G) ratio is usually considered first because it is the factor that can be easily observed. Still, I(2D)/I(G) may be remarkably disturbed by charge particles adsorbed on graphene. Therefore, many researchers commonly scrutinize Pos(2D) as well as FWHM(2D) due to their less charge effects.
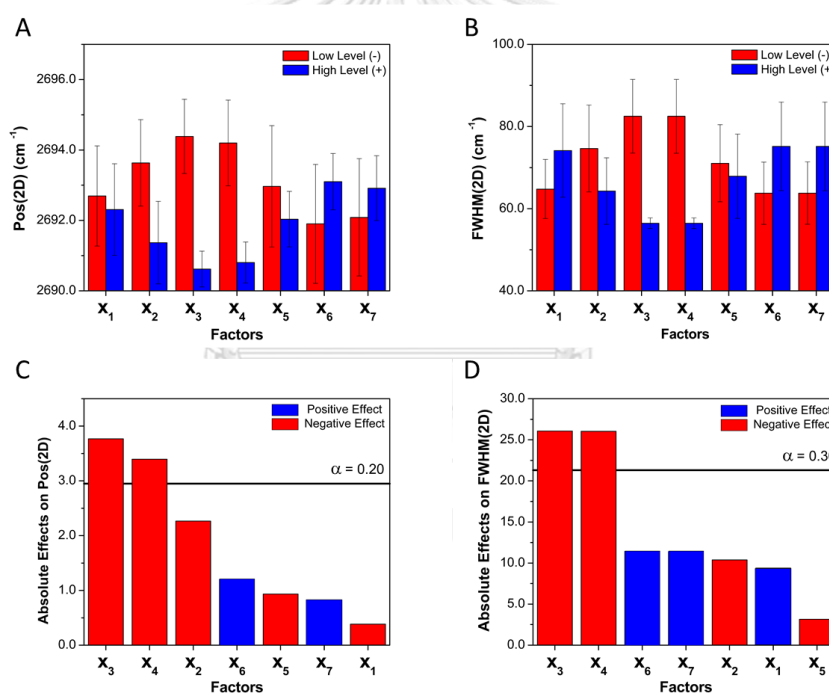
Figure 4.7 Bar charts showing average **(A)** Pos(2D) and **(B)** FWHM(2D) values at high and low levels of independent variables and Pareto charts of absolute effect of independent variables on **(C)** Pos(2D) and **(D)** FWHM(2D) values

As demonstrated in Figure 4.7, only growth time (X_6) and cooling rate (X_7) had positive effect on Pos(2D). Meanwhile, the other factors negatively affected Pos(2D). However, the effective factors for tuning Pos(2D) were growth temperature (X_3) and

Ar-H$_2$ composition during post-annealing steps (X$_4$) at 80% level of confidence. As well as Pos(2D), growth temperature (X$_3$) and Ar-H$_2$ composition during post-annealing steps (X$_4$) were considered as significant factors for reducing FWHM(2D). In addition to Pos(G), Pos(2D) showed variations which were less than the resolution of Raman spectrometer. Hence, this information was also not reliable to describe the variation of Pos(2D).

According to the above studies as summarized in Table 4.4, the factors having significant on various dependent variables were growth temperature (X$_3$) and Ar-H$_2$ composition during post-annealing steps (X$_4$). Both of them played important roles in tuning I(2D)/I(G), FWHM(G), Pos(2D) and FWHM(2D). In addition, growth temperature (X$_3$) took an important part in decreasing defect density estimated by I(D)/I(G). Therefore, growth temperature (X$_3$) and Ar-H$_2$ composition during post-annealing steps (X$_4$) were selected for further study based on response surface methodology.

Nevertheless, optimization of Pos(G) was still unclear. The most effective parameter was acetylene flow rate (X$_5$), which was significant at 52.5% level of confidence. This low level of confidence meant it was possible that none of the factors studied here was a significant factor affecting Pos(G). Still, acetylene flow rate is one of the crucial factors in graphene synthesis, because it controls nucleation rate and growth rate [47]. In addition, there was a report [42] that studied the effect of acetylene flow rate on quality of as-grown graphene. Thus, acetylene flow rate (X$_5$) was also selected for further study using response surface methodology.

Table 4.4 Summary of optimum direction of independent factors to achieve optimizing direction of dependent variables (P = Positive, N = Negative)

Dependent variable	Optimizing direction	Optimum direction						
		X_1	X_2	X_3	X_4	X_5	X_6	X_7
I(D)/I(G)	N	P	P	P	P	N	N	N
I(2D)/I(G)	P	N	P	P	P	P	N	N
FWHM(G)	N	N	P	P	P	P	N	N
FWHM(2D)	N	N	P	P	P	P	N	N

Apart from selecting three factors for constructing response surfaces, the remaining factors that would not be included in the study needed to be fixed by considering optimum direction, as in Table 4.4. Firstly, Ar-H_2 composition before growth (X_1) was fixed to 150/150 sccm/sccm because setting the factor at low level satisfied most of dependent variables except I(D)/I(G). Furthermore, using such flow rates could reduce gas cost. In addition, low total flow rates could reduce retention time which makes cleaning process more efficient. Secondly, annealing time (X_2) was set at 60 min. The results were in consistent with previous reports in that extension of annealing time could improve the quality of graphene [76-78]. Prolonging annealing time could afford cleaner reconstructed surface. Thirdly, growth time (X_6) was specified to 10 min as reported by Chinese research team [42]. Normally, CVD process using acetylene is not self-limited, due to high pyrolysis rate of acetylene, so long growth time could directly affect graphene thickness. Finally, cooling rate was set at 5 °C, to satisfy all dependent variables except Pos(2D). This results is in consistent with previous report [15] that low cooling rate could reduce both defect density and thickness of graphene. As temperature drops, carbon atoms could probably form 5-, 6-, 7- or 8-membered rings. However, only 6-membered ring is thermodynamically favored. Other ring types could form but they are not stable. Slow cooling provides more time for rearrangement and more energy to overcome energy barrier to form stable graphene.

Noticeably, for these experiments, high value of α was used to determine whether or not each factor was significant. Generally, α was set at 0.05 to 0.10, which corresponded to 90 to 95% level of confidence. This also meant that there was high possibility that type I error could occur. This might be because the dependent variables were quite sensitive to those independent factors and dummy factors were not added in the experiments. To guarantee correctness of collected data, a great number of Raman spectra had to be recorded and interpreted.

4.2 Response surface methodology

The results from screening experiments suggested that growth temperature (X_3) and Ar-H_2 composition during post-annealing steps (X_4) be used for response surface modeling.

According to previous report [42], growth temperature (X_3) was already known as the main factor controlling the quality of graphene. For Ar-H_2 flow rates (X_4), they did not only dilute acetylene gas but also controlled residence time of acetylene gas. Last but not least, acetylene flow rate (X_5) also directly affected nucleation and growth rates. Thus, acetylene flow rate (X_5) was selected as another factor for response surface modeling.

Normally, highest growth temperature of graphene synthesized via CVD process from methane is 1050 °C. However, the other processes operate below 1000 °C. Also, some researchers anticipated that CVD graphene synthesized from acetylene could perform at lower temperature. Thus, the effect of growth temperature (X_3) in our experiments was observed between 950 to 1050 °C

Furthermore, the total flow rates of input gases in previous reports [10, 19, 42] were very different depending on operating pressure. At atmospheric pressure, Ar-H_2 total flow rates of 1000 sccm were confirmed to produce low defect bilayer graphene. Nonetheless, H_2/Ar flow rate ratio must be 0.010-0.111. Hence, the flow rate ratio of H_2/Ar in experiments was kept at 0.111 and total flow rates were investigated.

As stated earlier that acetylene, as carbon precursor, has high pyrolysis rate, so small amounts of acetylene were needed in the process. The lowest flow rate of acetylene fed into the reactor was 0.24 sccm under low pressure. While at atmospheric pressure, it was a little bit higher at 1.0 sccm. Therefore, in these experiments, acetylene flow rate (X_5) was varied lower than previous reports [42, 47].

Table 4.5 Three factors for response surface modelling (RSM)

Factor	Unit	Level (−1)	Level (0)	Level (+1)
C	°C	950	1000	1050
D	sccm/sccm	450/50	675/75	900/100
E	sccm	0.4	0.7	1.0

Box-Behnken design was employed in this study. Although this model does not cover all over spaces, the required number of experiments for this design is less than that of other designs such as central composite design (CCD) or full factorial design. Also, high level of temperature expected to be a main factor for controlling the quality of graphene was limited due to furnace specification.

According to Box-Behnken design, three levels of each factor had to be specified. Thus, middle point between high and low level in Plackett-Burman design was set as Level (0) as shown in Table 4.5.

15 experiments were conducted with variable parameters as specified in Table 4.6 and fixed parameters as discussed in previous section. The as-grown graphene on Cu was transferred to SiO_2/Si before Raman measurement. 250 Raman spectra were collected for each run and 6 Raman parameters were extracted for modeling as have been done in the screening experiments.

Table 4.6 Coded and actual parameters in Box-Behnken Design

Run	Coded Value			Actual Value		
	C	D	E	C (°C)	D (sccm/sccm)	E (sccm)
1	−1	0	+1	950	675/75	1.0
2	0	−1	+1	1000	450/50	1.0
3	0	+1	+1	1000	900/100	1.0
4	+1	0	+1	1050	675/75	1.0
5	−1	−1	0	950	450/50	0.7
6	−1	+1	0	950	900/100	0.7
7	+1	+1	0	1050	900/100	0.7
8	+1	−1	0	1050	450/50	0.7
9	0	−1	−1	1000	450/50	0.4
10	−1	0	−1	950	675/75	0.4
11	0	+1	−1	1000	900/100	0.4
12	+1	0	−1	1050	675/75	0.4
13	0	0	0	1000	675/75	0.7
14	0	0	0	1000	675/75	0.7
15	0	0	0	1000	675/75	0.7

As shown in Table 4.7, the quality of graphene varied from medium to low. The best sample was from Run 12 with I(D)/I(G) of 0.28, (I2D)/I(G) of 1.45, Pos(G) of 1587.42 cm^{-1}, FWHM(G) of 23.21 cm^{-1}, Pos(2D) of 2687.85 cm^{-1} and FWHM(2D) of 47.40 cm^{-1}. Even though, defect density of the best sample in these experiments was greater than that of the best sample from Plackett-Burman screening experiments, the thickness of graphene from Run 12 as indicated by I(2D)/I(G) and FWHM(2D) were much better than that from the screening experiments. Unfortunately, ranges of Pos(G) and Pos(2D) were less than the resolution of Raman spectrometer. Consequently, models developed from these data for predicting Pos(G) and Pos(2D) were not reliable.

Table 4.7 Summary of Raman parameters from experiments of Box-Behnken Design

Run	I(D)/I(G)		I(2D)/I(G)		Pos(G)		FWHM(G)		Pos(2D)		FWHM(2D)	
	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD	$\bar{X}$	SD
1	1.05	0.13	0.35	0.07	1586.45	2.41	60.93	14.35	2696.33	4.20	89.39	7.58
2	0.94	0.33	0.46	0.12	1587.05	2.25	47.56	15.36	2690.67	3.50	74.08	9.77
3	0.72	0.28	0.70	0.15	1586.66	2.38	39.29	13.42	2693.71	3.65	71.00	7.77
4	0.41	0.33	0.83	0.27	1586.44	2.34	32.83	9.36	2693.47	5.19	66.50	6.56
5	1.06	0.14	0.37	0.07	1582.38	2.37	62.07	13.70	2691.41	3.75	91.60	7.14
6	1.09	0.16	0.41	0.07	1582.90	2.07	56.94	13.95	2691.05	4.25	87.24	6.17
7	0.34	0.27	1.19	0.36	1587.34	2.73	26.07	4.27	2688.60	4.07	50.68	5.13
8	0.74	0.49	1.25	0.45	1584.57	1.98	34.57	12.31	2683.88	5.37	57.25	7.92
9	0.82	0.34	0.74	0.20	1585.67	2.57	41.44	15.74	2689.05	3.93	69.80	9.42
10	1.04	0.17	0.42	0.09	1584.97	2.44	55.00	14.57	2693.53	4.43	85.72	7.41
11	0.64	0.31	0.91	0.25	1585.62	2.11	36.49	11.86	2686.54	4.34	64.67	8.14
12	0.28	0.29	1.45	0.57	1587.42	2.13	23.21	3.77	2687.85	3.94	47.40	5.28
13	0.86	0.32	0.76	0.20	1585.73	2.60	44.45	14.78	2687.33	4.58	72.06	9.08
14	0.91	0.30	0.71	0.18	1584.86	2.26	44.98	14.34	2687.19	4.60	72.05	7.44
15	0.94	0.38	0.66	0.19	1584.86	2.78	47.03	17.19	2688.72	4.52	74.26	10.01

The average parameters including I(D)/I(G), (I2D)/I(G), FWHM(G) and FWHM(2D) were then analyzed by multiple linear regression (MLR) using SPSS software provided by Chulalongkorn University. The full quadratic model was selected for every dependent variable because linear terms, interaction terms and quadratic terms were included. The calculated regression coefficients were summarized in Table 4.8.

Coefficient of determination was calculated for investigating fitness of model. All models had high coefficient of determination up to 0.97. This implied that the mathematical model could predict dependent variables well.

In addition to coefficient of determination, analysis of variance (ANOVA) and lack-of-fit test were performed to observe degree of fit. ANOVA results were shown in Table 4.9. P-values of the corrected models of I(D)/I(G), I(2D)/I(G), FWHM(G) and FWHM(2D) were less than 0.05 indicating that those models were significant and well-described by the factors.

Table 4.8 Summary of coefficients calculated from multiple linear regression (MLR)

Independent variable	Coefficients			
	I(D)/I(G)	I(2D)/I(G)	FWHM(G)	FWHM(2D)
Intercept	0.903	0.708	45.485	72.792
X_3	−0.31	0.395	−14.782	−16.515
X_4	−0.095	0.049	−3.356	−2.391
X_5	0.044	−0.146	3.059	4.172
X_3*X_4	−0.107	−0.025	−0.842	−0.551
X_3*X_5	0.03	−0.135	0.92	3.86
X_4*X_5	−0.01	0.019	−0.831	0.511
X_3^2	−0.09	0.08	0.613	0.634
X_4^2	−0.006	0.016	−1.183	−1.733
X_5^2	−0.117	−0.024	−3.104	−1.174

Then, the effect of each term was considered. Growth temperature (X_3) apparently affected all Raman parameters as p-value was less than 0.05. While Ar-H_2 flow rates (X_4) had influences only on I(D)/I(G), FWHM(G) and FWHM(2D). Interestingly, acetylene flow rate (X_5), which seemed to have no effect on any dependent variable according to the screening experiments, showed effects on I(2D)/I(G) and Pos(2D).

Three factors (X_3, X_4 and X_5) have three interaction terms including X_3X_4, X_3X_5 and X_4X_5. The interaction between growth temperature (X_3) and Ar-H_2 flow rates (X_4) played important roles in I(D)/I(G) and Pos(2D) whereas the interaction between growth temperature (X_3) and acetylene flow rate (X_5) affected I(2D)/I(G) and FWHM(2D).

For quadratic terms (X_3^2, X_4^2 and X_5^2), they can suggest non-linear relationships. In models of I(2D)/I(G) and FWHM(2D), the quadratic terms were not required for good

fit. On the contrary, the quadratic terms of growth temperature (X_3^2) and acetylene flow rate (X_5^2) showed significant roles on I(D)/I(G) models. Meanwhile, only quadratic term of acetylene flow rate (X_5^2) significantly described changes in FWHM(G).

The results from ANOVA, as stated in Table 4.9, suggested that each model needed different sets of terms to describe the dependent variable, even though coefficient of determination were close to 1 and there were no lack-of-fit in every model, as shown in Table 4.10. This might indicate that our current models were overfitted. So, leave-one-out cross validation (LOOCV) was carried out to observe the overfit. As illustrated in Figure 4.8, there were variations in every model.

Due to unnecessary terms in the full quadratic models, new models were developed by removing terms with p-value higher than 0.05 according to ANOVA results.
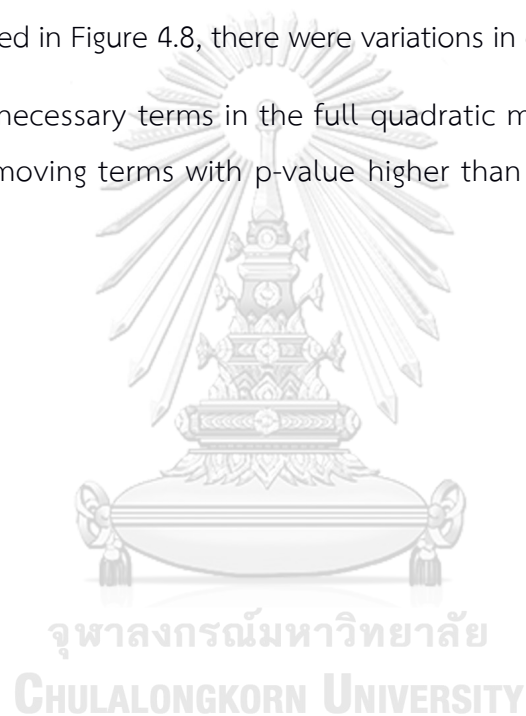
Table 4.9 ANOVA of full quadratic models for I(D)/I(G), I(2D)/I(G), FWHM(G), and FWHM(2D)

Source		Sum of Squares	df	Mean Square	F	p-value
Corrected Model	I(D)/I(G)	.981[a]	9	0.109	29.861	**0.001**
	I(2D)/I(G)	1.543[b]	9	0.171	24.636	**0.001**
	FWHM(G)	1964.070[d]	9	218.23	114.668	**0**
	FWHM(2D)	2446.634[f]	9	271.848	39.505	**0**
Intercept	I(D)/I(G)	2.445	1	2.445	669.523	**0**
	I(2D)/I(G)	1.505	1	1.505	216.217	**0**
	FWHM(G)	6206.602	1	6206.602	3261.219	**0**
	FWHM(2D)	15896.02	1	15896.02	2309.986	**0**
X_1	I(D)/I(G)	0.769	1	0.769	210.661	**0**
	I(2D)/I(G)	1.248	1	1.248	179.299	**0**
	FWHM(G)	1748.104	1	1748.104	918.53	**0**
	FWHM(2D)	2182.075	1	2182.075	317.096	**0**
X_2	I(D)/I(G)	0.072	1	0.072	19.784	**0.007**
	I(2D)/I(G)	0.019	1	0.019	2.792	0.156
	FWHM(G)	90.102	1	90.102	47.343	**0.001**
	FWHM(2D)	45.729	1	45.729	6.645	**0.05**

Table 4.9 ANOVA of full quadratic models for I(D)/I(G), I(2D)/I(G), FWHM(G), and FWHM(2D) (continued)

Source		Sum of Squares	df	Mean Square	F	p-value
X_5	I(D)/I(G)	0.015	1	0.015	4.187	0.096
	I(2D)/I(G)	0.171	1	0.171	24.601	**0.004**
	FWHM(G)	74.836	1	74.836	39.322	**0.002**
	FWHM(2D)	139.228	1	139.228	20.232	**0.006**
$X_3{*}X_4$	I(D)/I(G)	0.045	1	0.045	12.454	**0.017**
	I(2D)/I(G)	0.002	1	0.002	0.349	0.58
	FWHM(G)	2.834	1	2.834	1.489	0.277
	FWHM(2D)	1.213	1	1.213	0.176	0.692
$X_3{*}X_5$	I(D)/I(G)	0.004	1	0.004	0.993	0.365
	I(2D)/I(G)	0.073	1	0.073	10.528	**0.023**
	FWHM(G)	3.384	1	3.384	1.778	0.24
	FWHM(2D)	59.593	1	59.593	8.66	**0.032**
$X_4{*}X_5$	I(D)/I(G)	0	1	0	0.118	0.746
	I(2D)/I(G)	0.001	1	0.001	0.198	0.675
	FWHM(G)	2.763	1	2.763	1.452	0.282
	FWHM(2D)	1.045	1	1.045	0.152	0.713

Table 4.9 ANOVA of full quadratic models for I(D)/I(G), I(2D)/I(G), FWHM(G), and FWHM(2D) (continued)

Source		Sum of Squares	df	Mean Square	F	p-value
X_3^2	I(D)/I(G)	0.03	1	0.03	8.16	**0.036**
	I(2D)/I(G)	0.024	1	0.024	3.401	0.124
	FWHM(G)	1.387	1	1.387	0.729	0.432
	FWHM(2D)	1.486	1	1.486	0.216	0.662
X_4^2	I(D)/I(G)	0	1	0	0.037	0.856
	I(2D)/I(G)	0.001	1	0.001	0.131	0.732
	FWHM(G)	5.164	1	5.164	2.714	0.16
	FWHM(2D)	11.086	1	11.086	1.611	0.26
X_5^2	I(D)/I(G)	0.05	1	0.05	13.829	**0.014**
	I(2D)/I(G)	0.002	1	0.002	0.303	0.606
	FWHM(G)	35.584	1	35.584	18.697	**0.008**
	FWHM(2D)	5.085	1	5.085	0.739	0.429
Error	I(D)/I(G)	0.018	5	0.004		
	I(2D)/I(G)	0.035	5	0.007		
	FWHM(G)	9.516	5	1.903		
	FWHM(2D)	34.407	5	6.881		

Table 4.10 Lack-of-fit analysis of I(D)/I(G), I(2D)/I(G), FWHM(G), and FWHM(2D) quadratic
models

Dependent Variable		Sum of Squares	df	Mean Square	F	p-value
I(D)/I(G)	Lack of Fit	.015	3	.005	3.566	**.227**
	Pure Error	.003	2	.001		
I(2D)/I(G)	Lack of Fit	.030	3	.010	4.056	**.204**
	Pure Error	.005	2	.002		
FWHM(G)	Lack of Fit	5.788	3	1.929	1.035	**.526**
	Pure Error	3.728	2	1.864		
FWHM(2D)	Lack of Fit	31.168	3	10.389	6.415	**.138**
	Pure Error	3.239	2	1.620		

The coefficients of reduced models were shown in Table 4.11. The coefficients
had slight variations from the full model, but the sign of each term remained the
same. Because Box-Behnken design is near rotatable. Although terms were added or
removed, the coefficients would not be significantly varied.

ANOVA was performed to observe the significances of models and each term
(Table 4.12 – 4.15). After removing inactive terms, all models still remained significant.
Coefficient of determination of every reduced model was lower than those of full
models. Nevertheless, all coefficients of determination were greater than 0.90.
Moreover, no lack-of-fit was found for every model.
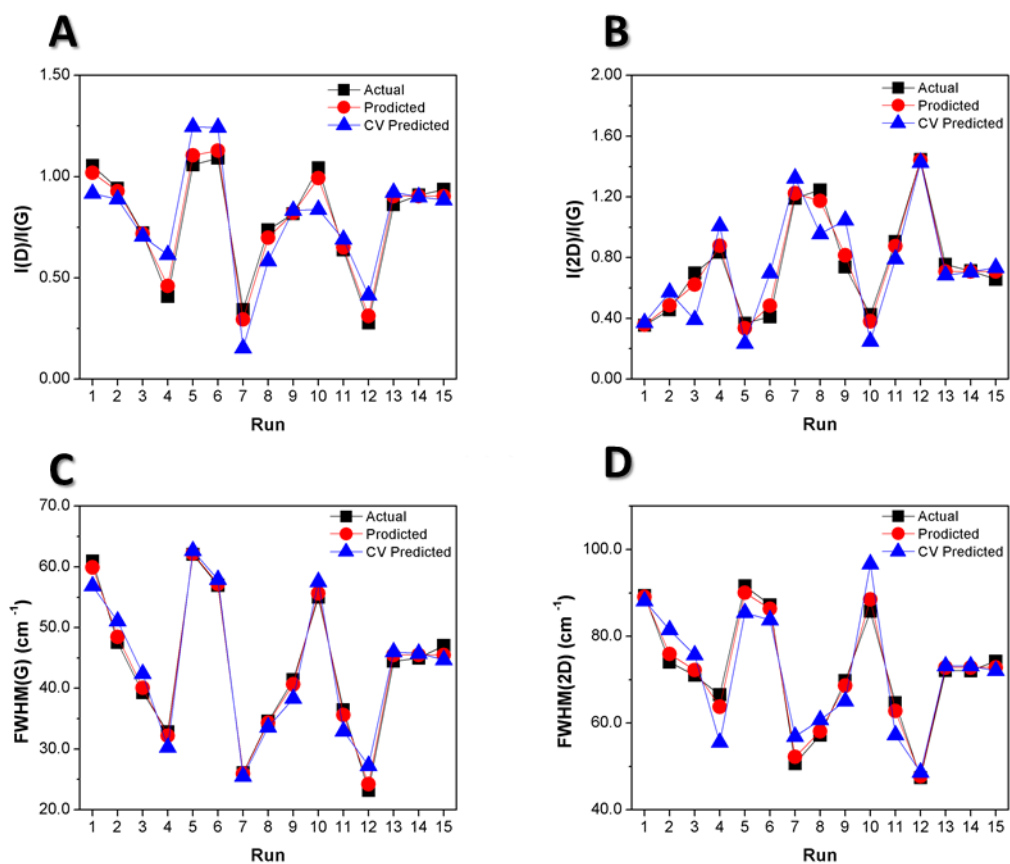
Figure 4.8 Leave-one-out cross validation of **(A)** I(D)/I(G), **(B)** I(2D)/I(G), **(C)** FWHM(G) and **(D)** FWHM(2D) models

Table 4.11 Summary of coefficients calculated from multiple linear regression (MLR) after removing unnecessary terms

Variables	Coefficients			
	I(D)/I(G)	I(2D)/I(G)	FWHM(G)	FWHM(2D)
Constant	0.899	0.747	45.159	71.58
X_3	−0.31	0.395	−14.782	−16.515
X_4	−0.095	−	−3.356	−2.391
X_5	−	−0.146	3.059	4.172
X_3*X_4	−0.107	−	−	−
X_3*X_5	−	−0.135	−	3.86
X_4*X_5	−	−	−	−
X_3^2	−0.089	−	−	−
X_4^2	−	−	−	−
X_5^2	−0.116	−	−3.064	−

The remaining terms in most models significantly influenced the dependent variables. The cross validations of reduced models are shown in Figure 4.9. The predicted values became closer to the actual values. LOOCV-RMSE would be utilized to observe predictive ability of the models.

By comparing LOOCV-RMSE of full quadratic models to reduced models, the RMSE became smaller. This indicated that the predictive ability increased after removing unnecessary terms.

Table 4.12 ANOVA and Lack-of-fit analysis for reduced model of I(D)/I(G)

Source	Type III Sum of Squares	df	Mean Square	F	p-value
Corrected Model	.962[a]	5	.192	45.881	**.000**
Intercept	3.502	1	3.502	835.318	**.000**
X_3	.769	1	.769	183.462	**.000**
X_4	.072	1	.072	17.230	**.002**
X_3*X_4	.045	1	.045	10.846	**.009**
X_3^2	.030	1	.030	7.076	**.026**
X_5^2	.050	1	.050	12.019	**.007**
Error	.038	9	.004		
Total	10.343	15			
Corrected Total	1.000	14			
Lack of Fit	.015	3	.005	1.307	**.356**
Pure Error	.023	6	.004		

a. R Squared = .962

Table 4.13 ANOVA and Lack-of-fit analysis for reduced model of I(2D)/I(G)

Source	Type III Sum of Squares	df	Mean Square	F	p-value
Corrected Model	1.492[a]	3	.497	63.939	**.000**
Intercept	8.360	1	8.360	1074.643	**.000**
X_3	1.248	1	1.248	160.393	**.000**
X_5	.171	1	.171	22.007	**.001**
X_3*X_5	.073	1	.073	9.418	**.011**
Error	.086	11	.008		
Total	9.938	15			
Corrected Total	1.578	14			
Lack of Fit	.035	5	.007	.841	**.566**
Pure Error	.050	6	.008		

a. R Squared = .946

Table 4.14 ANOVA and Lack-of-fit analysis for reduced model of FWHM(G)

Source	Type III Sum of Squares	df	Mean Square	F	p-value
Corrected Model	1948.083[a]	4	487.021	190.974	.000
Intercept	14275.467	1	14275.467	5597.790	.000
X_3	1748.104	1	1748.104	685.478	.000
X_4	90.102	1	90.102	35.331	.000
X_5	74.836	1	74.836	29.345	.000
X_5^2	35.042	1	35.042	13.741	.004
Error	25.502	10	2.550		
Total	30390.261	15			
Corrected Total	1973.585	14			
Lack of Fit	21.774	8	2.722	1.460	.469
Pure Error	3.728	2	1.864		

a. R Squared = .987

Table 4.15 ANOVA and Lack-of-fit analysis for reduced model of FWHM(2D)

Source	Type III Sum of Squares	df	Mean Square	F	p-value
Corrected Model	2426.625[a]	4	606.656	111.485	.000
Intercept	76856.147	1	76856.147	14123.813	.000
X_3	2182.075	1	2182.075	400.999	.000
X_4	45.729	1	45.729	8.404	.016
X_5	139.228	1	139.228	25.586	.000
X_3*X_5	59.593	1	59.593	10.951	.008
Error	54.416	10	5.442		
Total	79337.189	15			
Corrected Total	2481.041	14			
Lack of Fit	51.177	8	6.397	3.950	.218
Pure Error	3.239	2	1.620		

a. R Squared = .978
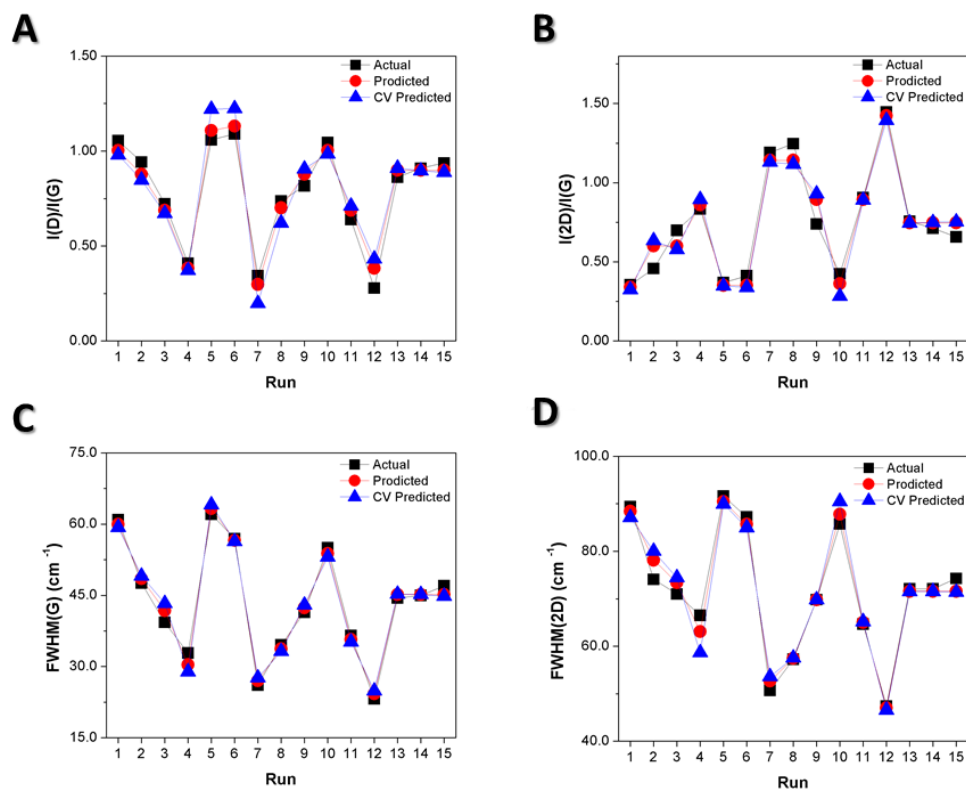
Figure 4.9 Leave-one-out cross validation of **(A)** I(D)/I(G), **(B)** I(2D)/I(G), **(C)** FWHM(G) and **(D)** FWHM(2D) reduced models

Table 4.16 Comparison of root-mean-square error (RMSE) of leave-one-out cross validation (LOOCV), coefficient of determination

Model	LOOCV-RMSE		Coefficient of determination	
	Full	Reduced	Full	Reduced
I(D)/I(G)	0.130	0.097	0.982	0.962
I(2D)/I(G)	0.181	0.099	0.978	0.946
FWHM(G)	2.595	2.015	0.995	0.987
FWHM(2D)	5.808	3.295	0.986	0.978

In summary, the reduced models were good tools for approximating I(D)/I(G), I(2D)/I(G), FWHM(G) and FWHM(2D).

After obtaining the mathematical models, contour plots (Figure 4.10 – 4.14) were constructed to observe effect of each parameter. In models of I(2D)/I(G), only growth temperature (X_3) and acetylene flow rate (X_5) were considered while all three parameters were considered in the other models. Furthermore, the parameters would be optimized to achieve highest quality graphene under investigating space.

Obviously, growth temperature (X_3) was the most effective factor to control I(D)/I(G) as seen in Figure 4.10(A) and 4.10(B). This is in consistent with previous report [42]. I(D)/I(G) gradually increased as temperature dropped. While Ar-H_2 flow rates (X_4) had small negative effect on I(D)/I(G), however, the interaction between growth temperature (X_3) and Ar-H_2 flow (D) had more effect than only Ar-H_2 flow rates (X_4). At high Ar-H_2 flow rate (X_4) and high temperature (X_3), I(D)/I(G) reached minimum within investigating space.
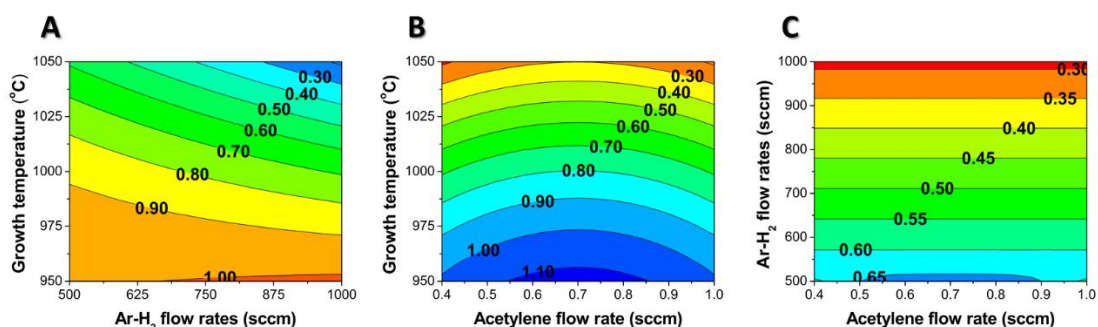
**Figure 4.10** Response surface plots of I(D)/I(G) for **(A)** growth temperature and Ar-H$_2$ flow rates, **(B)** growth temperature and acetylene flow rate and **(C)** Ar-H$_2$ flow rates and acetylene flow rate

Although the linear term of acetylene flow rate (X_5) was not considered as active component in the model, the quadratic term of acetylene flow rate (X_5^2) was an important element for better approximation. Interestingly, both high and low acetylene flow rates could afford graphene with low defect. On the other hand, medium acetylene flow rate (X_5) could increase I(D)/I(G). The interaction between acetylene flow rate (X_5) and Ar-H$_2$ flow rates (X_4) was suppressed at high Ar-H$_2$ flow rates (X_4) and parabolic change was observed at very low Ar-H$_2$ flow rates (X_4) (Figure 4.10(C)).
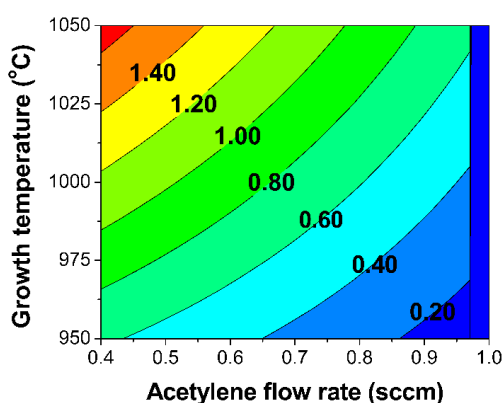
Figure 4.11 Response surface plots of I(2D)/I(G) for growth temperature and acetylene flow rate

Model of I(2D)/I(G) concerned only to growth temperature (X_3) and acetylene flow rate (X_5). Figure 4.11(A) shows that growth temperature (X_3) was still a dominant factor. Interestingly, I(2D)/I(G) rapidly changed with acetylene flow rate (X_5) when it was greater than 0.95 sccm but the change became gradual when acetylene flow rate (X_5) was lower than 0.975 sccm. Such observation was more obvious at higher growth temperature (X_3). The maximum I(2D)/I(G) was observed at lowest acetylene flow rate (X_5) and highest growth temperature within investigating space.
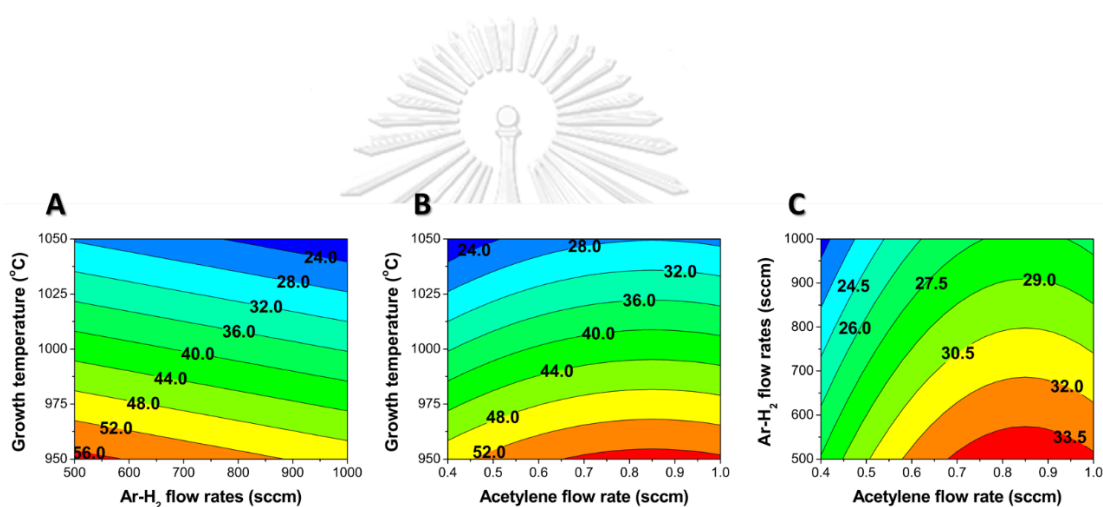
Figure 4.12 Response surface plots of FWHM(G) for **(A)** growth temperature and Ar-H$_2$ flow rates, **(B)** growth temperature and acetylene flow rate and **(C)** Ar-H$_2$ flow rates and acetylene flow rate

For FWHM(G), no interaction term was present in the model. There were only three linear terms (X_3, X_4 and X_5) and a quadratic term of acetylene flow rate (X_5^2). Growth temperature was once again a key factor to tune FWHM(G). The lowest FWHM(G) was obtained at high Ar-H$_2$ flow rates (X_4) and growth temperature (X_3) (Figure 4.12(A)). Decreasing acetylene flow rate (X_5) can also decrease FWHM(G) (Figure 4.12(B)). This observation was more obvious when acetylene flow rate (X_5) was lower than 0.6 sccm.
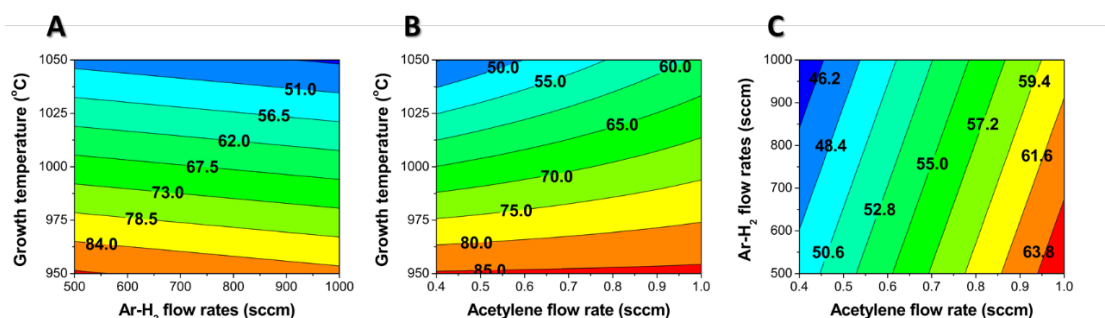
Figure 4.13 Response surface plots of FWHM(2D) for **(A)** growth temperature and Ar-H_2 flow rates, **(B)** growth temperature and acetylene flow rate and **(C)** Ar-H_2 flow rates and acetylene flow rate

FWHM(2D) is the most precise indicator for approximating the number of graphene layer because it is not disturbed efficiently by charge impurity. Figure 4.13(B) showed that the best quality graphene was produced at high temperature (X_3) with low acetylene flow rate (X_5). While high flow rates of Ar and H_2 (X_4) showed small negative effect on FWHM(2D) (Figure 4.13(A) and 4.13(B)). The thinnest graphene synthesized under investigating space was bilayer with FWHM(2D) of 40-50 cm^{-1}.

In summary, growth temperature (X_3), acetylene flow rate (X_5) and Ar-H_2 flow rates (X_4) are important factors to achieve the thinnest graphene with lowest defect. The response surface models suggested that the best quality graphene could be produced at 1050 °C with Ar-H_2-acetylene flow rate ratio of 900/100/0.4.

Nucleation and growth rates are kinetic parameters controlling growth of graphene. Low nucleation and high growth rates are preferred because they could lead to large graphene grain with low defect. Defects on graphene usually refer to graphene edges or non-6-membered ring structures. Defect density can be estimated by I(D)/I(G) ratio from Raman spectra. Due to high pyrolysis rate of acetylene, nucleation rate of the process is often high resulting in high edge defects. Furthermore, high nucleation rate could create additional graphene layer resulting in multilayer structure. The effects of CVD parameters must be investigated in order to reduce nucleation rate

According to this study, growth temperature (X_3), Ar-H_2 flow rates (X_4) and acetylene flow rate (X_5) were involved in the models. The most convenient way to

reduce nucleation rate is to dilute acetylene gas by increasing Ar-H_2 flow rates (X_4) and decreasing acetylene flow rate (X_5). In addition, increasing Ar-H_2 flow rates (X_4) could also reduce resident time of gases. Consequently, acetylene has a time limit to adsorb on Cu surface. However, those two factors (X_4 and X_5) had less effects than growth temperature (X_3). Based on Bertran's report [79], nucleation rate drops as temperature increases. Therefore, synthesizing CVD process at high temperature could afford graphene with low defect.

4.3 Functionalization of CVD-grown graphene
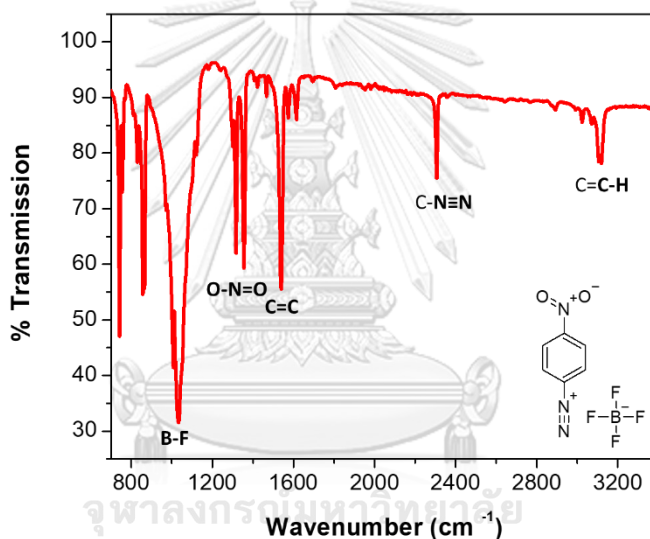
Figure 4.14 IR spectrum of 4-nitrobenzenediazonium tetrafluoroborate (4NBDT)

As grown graphene was functionalized with 4-nitrobenzenediazonium tetrafluoroborate (4NBDT). Firstly, 4NBDT was synthesized from 4-nitroaniline and the product was then characterized by IR spectroscopy and NMR spectroscopy. IR spectra provide information about the functional groups of substance while NMR spectra confirm the structure of organic compound.

The IR spectrum of 4NDBT as shown in Figure 4.14 exhibited 5 characteristic peaks. The diazonium vibration (N≡N) was clearly observed at 2307 cm^{-1} while no

amine peak was observed around 3600 cm^{-1}. This indicated that the amine reactant was successfully converted to diazonium salt. Moreover, high intense peak of B—F vibration appeared at 1033 cm^{-1} suggesting that tetrafluoroborate ion was the counterion of diazonium salt. Meanwhile, the vibrational peak of O—N=O group was also observed at 1356 and 1317 cm^{-1} and the C=C and C—H vibrations of phenyl core structure also appeared at 1540 and 3118 cm^{-1}.
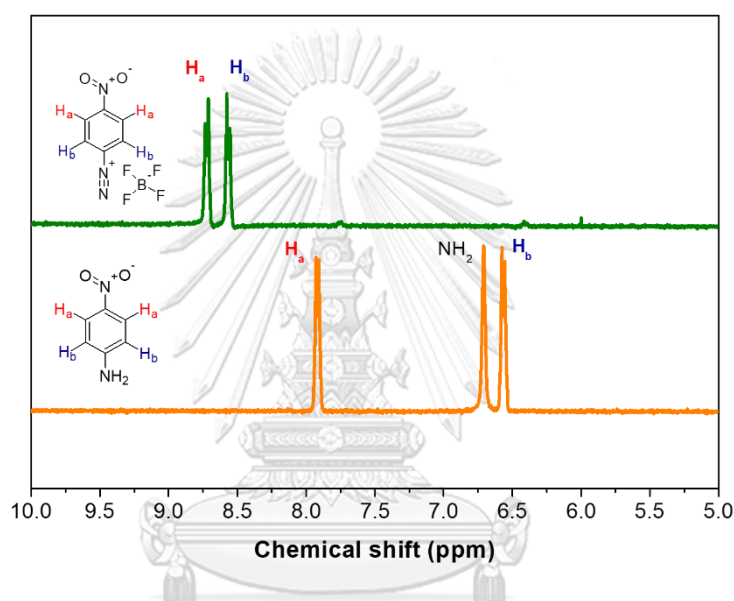

Figure 4.15 NMR spectra of 4-nitrobenzenediazonium tetrafluoroborate (4NBDT) (top) and 4-nitroaniline (bottom)

NMR spectra of 4NBDT and 4-nitroaniline were compared in Figure 4.15. After diazotization, the doublet amino protons around 8 ppm disappeared and phenyl protons were shifted to lower field for approximately 2 ppm. This suggested that after amine was converted to diazonium ion, protons on phenyl ring became more electron-deficient. These NMR results confirmed that 4NBDT was successfully synthesized.

4.3.1 Effect of substrate on functionalization
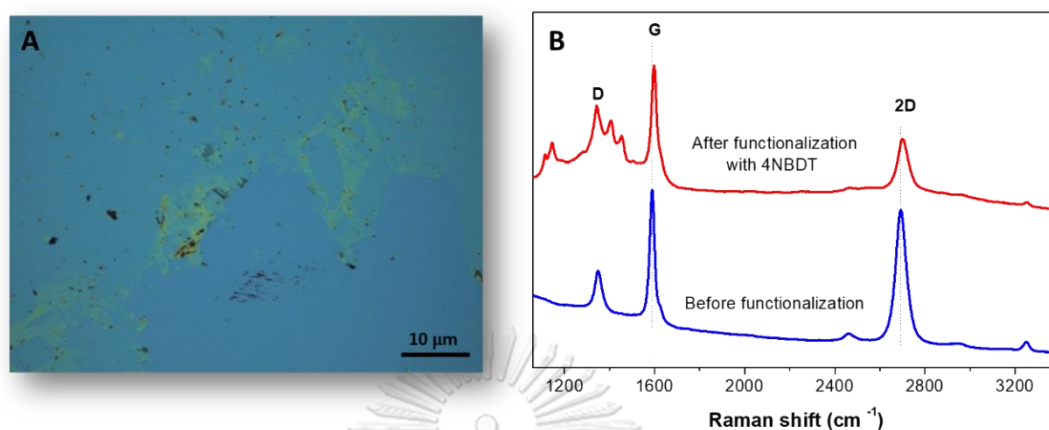
Figure 4.16 (A) Optical micrograph of modified graphene and **(B)** Raman spectra of graphene before functionalization (top) and after functionalization on SiO_2/Si (bottom)

The procedure reported by Strano and co-worker [51] was tested with graphene synthesized from acetylene. Graphene on SiO_2/Si was incubated in stirred 25 mM 4NBDT in 1 % w/v SDS/H_2O for 7 hr. The reaction was carried out in a dark fume hood. After the reaction was completed, the sample was washed with copious amounts of water before incubating in water overnight to make sure that all unreacted reactants were removed. However, in their publication, they used mechanical exfoliated single layer graphene while the graphene used in our experiment was CVD graphene. Although our graphene film was not entirely uniform, its average thickness was bilayer.

Interestingly, after modification, some parts of graphene film were lost as illustrated in Figure 4.16(A). This might be due to weak interactions between substrate and graphene.

Modified graphene remained on the substrate was subjected to Raman measurement. Figure 4.16(B) showed Raman spectra of graphene before and after functionalization where three characteristic peaks were found. Clearly, their positions

and intensity changed significantly. The intensity of D peak apparently increased indicating that some sp^2-hybridized carbons were converted to sp^3-hybridized carbons. This implied that functionalization was successful. However, there were messy peaks found between D and G peak and around 1200 cm^{-1}. These peaks were attributed to π-π interaction between unreacted diazonium salt and graphene. Moreover, the intensity of 2D peak significantly dropped due to doping. As well as Pos(G) and Pos(2D), they were also shifted whereas; FWHM(G) and FWHM(2D) had small increase as shown in Table 4.17.

Table 4.17 Average Raman parameters of graphene before functionalization and after functionalization with 4NBDT on SiO$_2$/Si

Raman parameter	Before functionalization	After functionalization
I(D)/I(G)	0.31 ± 0.01	0.44 ± 0.01
I(2D)/I(G)	1.03 ± 0.02	0.55 ± 0.01
Pos(G)	1589.5 ± 0.2	1596.9 ± 0.1
FWHM(G)	24.2 ± 0.3	25.1 ± 2.7
Pos(2D)	2693.4 ± 0.3	2701.7 ± 0.3
FWHM(2D)	56.5 ± 0.5	64.9 ± 0.6

The XPS spectra (Figure 4.17) also confirmed that the functionalization was successful as the N1s peak appeared at 401 and 407 eV where two types of nitrogen had similar amounts. The peak at higher binding energy was assigned to nitro group while the peak at lower binding energy was attributed to adsorbed nitrogen gas or diazene. Even though XPS measurement was performed at ultrahigh vacuum, some nitrogen or oxygen molecules can still adsorb and their peaks can be present in the spectra. The diazene might stem from diazo coupling resulting in multilayer functionalization. According to XPS spectra, a nitro group could be found each every 21 carbon atoms of graphene.

The existence of unreacted diazonium salt might affect conductivity according to doping. Strong interaction between diazonium salt and graphene and low reactivity of graphene were considered as cause of this problem. However, we could not produce single layer graphene from acetylene. To solve the problem, the strategy was changed from graphene to substrate.
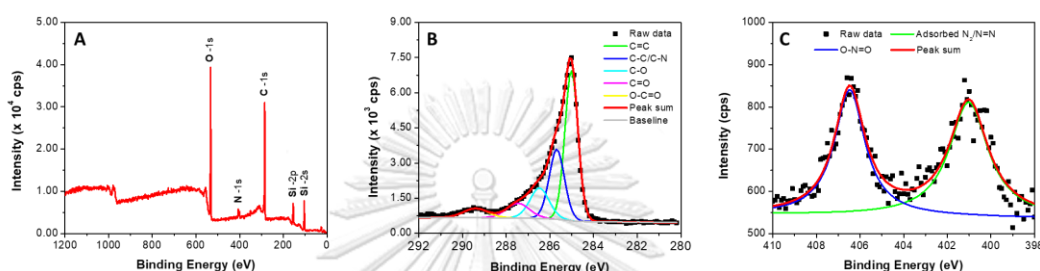

Figure 4.17 (A) Survey XPS spectrum **(B)** C1s XPS spectrum and **(C)** N1s XPS spectrum of functionalized graphene on SiO_2/Si

SiO_2/Si is a promising substrate for functionalization of graphene thank to its polarity. Since graphene is an extremely thin material, the substrate could possibly retain 4NBDT on the surface of graphene. Changing substrate to less polar was an alternative way.

Cu is another interesting candidate for graphene functionalization since no transfer step is required prior to functionalization. Moreover, graphene and Cu had lattice-matching because graphene was CVD grown directly on Cu. This could prevent loss of graphene during functionalization. More importantly, Cu is inert to diazonium functionalization since diazonium cannot react with Cu.

Graphene on Cu was functionalized under same condition as graphene on SiO_2/Si. After overnight of incubation in water, graphene was transferred to SiO_2/Si for Raman measurement. The image taken from an optical microscope as in Figure 4.18(A) revealed smooth and continuous surface of functionalized graphene.
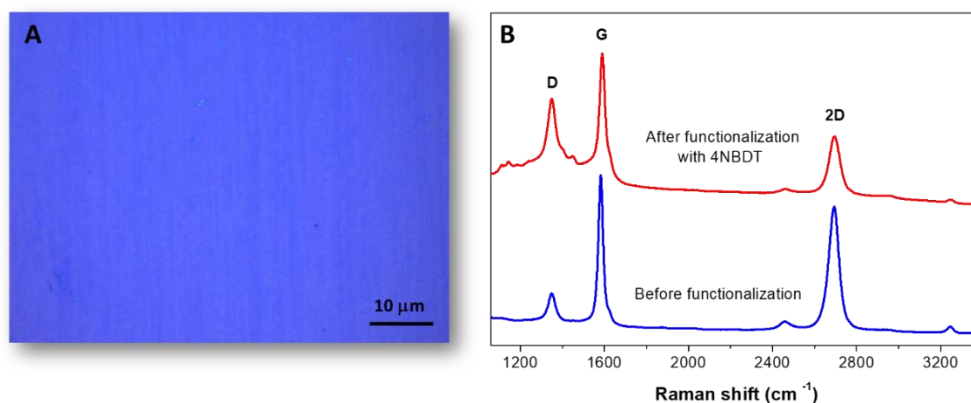
Figure 4.18 (A) Optical micrograph of modified graphene and **(B)** Raman spectra of graphene before functionalization (top) and after functionalization on Cu (bottom). All samples were transferred to SiO_2/Si substrate prior to taking an optical micrograph and measuring Raman spectra

After functionalization, the intensity of D peak distinctly increased while the intensity of 2D peak significantly declined as shown in Figure 4.18(B). In addition, both G and 2D peaks were red-shifted. Intriguingly, messy peaks found in graphene modified on SiO_2/Si were not observed on graphene modified on Cu. However, degree of functionalization estimated by increase of $I(D)/I(G)$ in case of functionalization on SiO_2/Si and Cu were not comparable because of different quality of starting graphene. Whilst the change of other Raman parameters (Table 4.18) corresponded to the modification on SiO_2/Si.

Table 4.18 Average Raman parameters of graphene before functionalization and after functionalization with 4NBDT on Cu

Raman parameter	Before functionalization	After functionalization
I(D)/I(G)	0.19 ± 0.01	0.57 ± 0.02
I(2D)/I(G)	0.99 ± 0.02	0.48 ± 0.01
Pos(G)	1585.0 ± 0.2	1590.4 ± 0.2
FWHM(G)	23.8 ± 0.2	31.4 ± 0.3
Pos(2D)	2691.0 ± 0.3	2695.2 ± 0.3
FWHM(2D)	56.8 ± 0.4	63.5 ± 0.5

XPS spectra of graphene functionalized on Cu were similar to that of graphene functionalized on SiO_2/Si as seen in Figure 4.19. However, most of nitrogen signal came from nitro group rather than adsorbed nitrogen gas with diazene. The calculated ratio of nitro group to carbon atoms of graphene was 1 to 27.
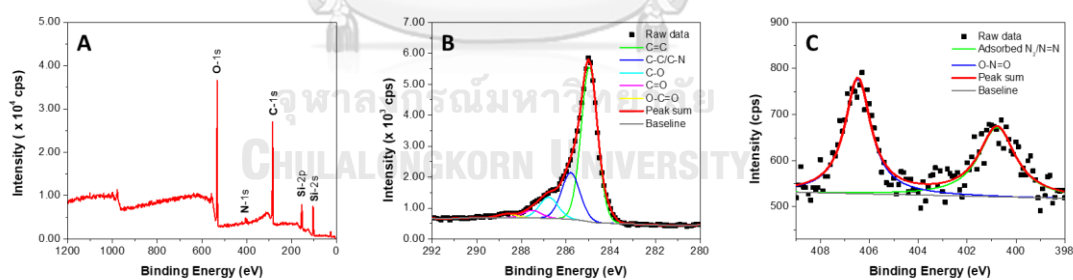
Figure 4.19 (A) Survey XPS spectrum **(B)** C1s XPS spectrum and **(C)** N1s XPS spectrum of graphene functionalized on Cu

The XPS results indicated that functionalization on SiO_2/Si could give higher degree of functionalization than on Cu. This observation is in consistent with those in other reports [57]. However, functionalization on SiO_2/Si also had drawbacks. First, it was difficult to remove unreacted diazonium salt from graphene surface by simple

washing. Second, some parts of graphene were torn off due to lattice-mismatching and weak interactions between graphene and substrate. Therefore, functionalization on Cu was a better choice.

4.3.2 Effect of light on functionalization

Diazonium salts are utilized in organic synthesis of aromatic compounds as versatile intermediates. Diazonium group can be substituted by hydrogen, halide, cyanide and hydroxyl groups or coupled with electron-rich aromatic ring. It has been taught in organic classes for a long time, however, a property of diazonium salt is usually neglected. That is sensitivity of diazonium salt to light [80]. Under irradiation of suitable wavelength (350-450 nm), nitrogen could cleave, and aryl cation are formed. The aryl cation can readily react with nucleophiles.

However, grafting mechanism of a diazonium salt to graphene is quite different. First, an electron from graphene transfers to a diazonium salt. Nitrogen gas is released, and aryl radical is formed. Second, the active radical reacts with graphene by forming a covalent bond. The first step is considered as the rate determining step and many approaches were made to push diazonium salt close to graphene surface. Another approach is to promote electron transfer from graphene to diazonium salt.

The conventional method uses SDS as surfactant to bring diazonium salt close to graphene surface so electron transfer takes place easily. However, the reaction takes 7 hr to complete with stirring. Additionally, there is risk that magnetic bar would damage graphene surface.

As mentioned earlier, diazonium salt could decompose under light irradiation. Nonetheless, there is no any clear report about the role of light in diazonium functionalization on graphene. Therefore, light would be employed in the following experiments in order to observe the effect of light on graphene functionalization.
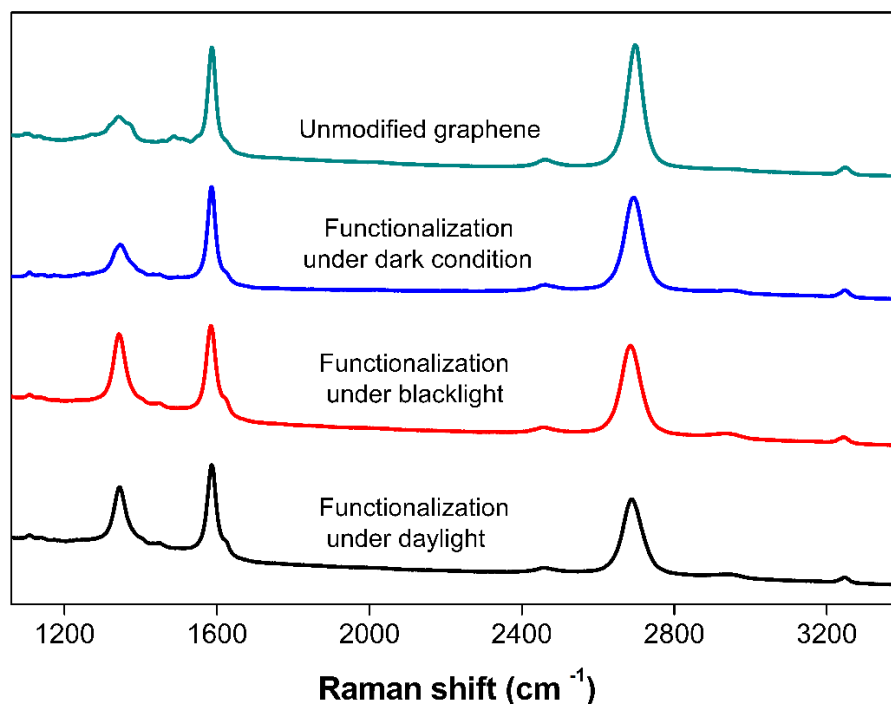
Figure 4.20 Raman spectra of unmodified graphene (cyan) and modified graphene under dark condition (blue), blacklight (red) and daylight (black)

Three experiments were conducted in order to compare modification methods under dark, daylight and blacklight conditions. As-grown graphene on Cu was incubated in 25 mM of 4NBDT under different light condition for 30 min. Then, all modified graphene was washed with copious amounts of water before incubating in water overnight. Afterwards, graphene was transferred to SiO_2/Si substrate for Raman measurement.

As illustrated in Figure 4.20, under dark condition, the intensity of D peak slightly increase, whereas the intensity of 2D peak marginally lessened. This implied that the grafting reaction could slowly take place without light.

As expected, under irradiation of light, D peak tremendously increased. However, unlike conventional method, 2D peak did not immensely decrease. This suggested that only covalent modification was dominant while diazonium salt rarely remained on graphene. Although the rise of D peak in case of irradiation under

blacklight was similar to that under daylight, I(D)/I(G) of graphene modified under blacklight was slightly higher. This indicated that light took an important role in enhancement of diazonium functionalization.
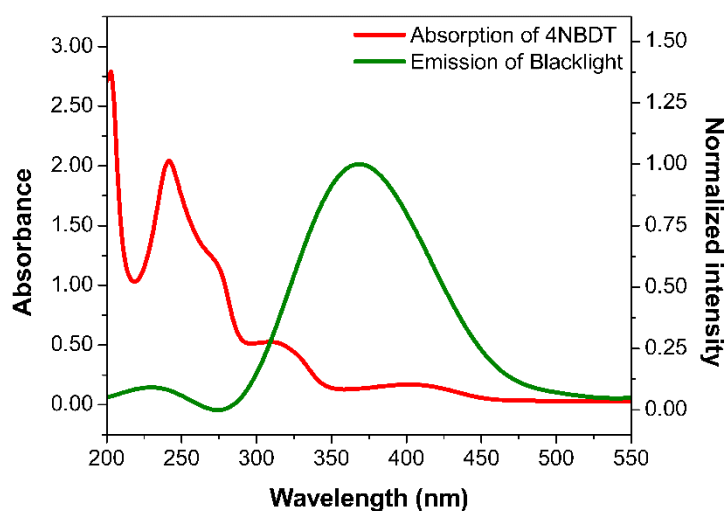
Figure 4.21 UV spectrum of 4-nitrobenzenediazonium tetrafluoroborate (4NBDT) and fluorescent spectrum of blacklight used in these experiments

For detailed discussion, the emission spectra of blacklight was recorded using a UV-Vis spectrophotometer. The blacklight emitted a broad spectrum from 275 to 525 nm as shown in Figure 4.21. The maximum emission wavelength of 368 nm with full width at half maximum of 101 nm made the light falling into a UV-A region. Since the UV spectrum of 4NBDT revealed an apparent absorption peak at 241 nm and shoulder peaks at 276 and 308 nm, the absorption spectrum of 4NBDT and the emission spectrum of blacklight were overlapping from 300 to 350 nm. Therefore, it was possible that 4NBDT could absorb photon energy from blacklight to release nitrogen gas and form 4-nitrophenyl cation. Then, the cation would accept an electron from graphene via single-electron transfer (SET) and become 4-nitrophenyl radical as displayed in Figure 4.22.

Additionally, it was also possible that blacklight could simultaneously interact with graphene. For example, photoinduced functionalization of graphene with

benzoyl peroxide was reported [81]. In that work, hot electrons generated from a 514.5 nm Raman laser were considered as activators for functionalization. Similarly, hot electrons could be generated from graphene by blacklight in our case. Therefore, SET from graphene to 4-nitrophenyl cation could be accelerated. Finally, a covalent bond between 4-nitrophenyl radical and graphene radical was formed.
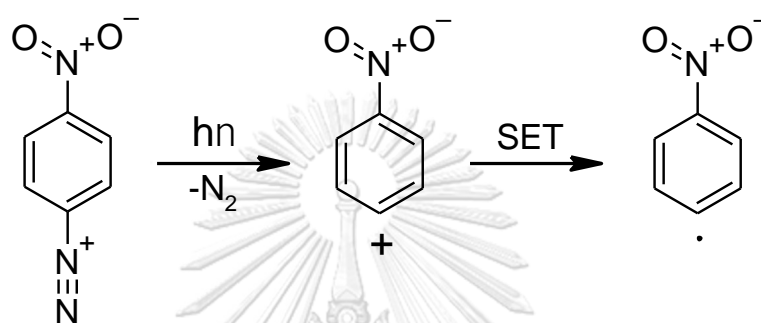

Figure 4.22 Schematic steps explaining the generation of 4-nitrophenyl radical under radiation of light

CHAPTER V

CONCLUSION

To optimize an atmospheric CVD process for graphene synthesis on copper substrate from acetylene precursor, six Raman parameters of graphene including $I(D)/I(G)$, $I(2D)/I(G)$, $Pos(G)$, $FWHM(G)$, $Pos(2D)$ and $FWHM(2D)$ were selected as dependent variables. According to Plackett-Burman screening experiments, growth temperature (X_3) and Ar-H_2 flow rates (X_4) were expected to be significant at moderate level of confidence (70 – 80%). Together with acetylene flow rate (X_5), three parameters of CVD process for graphene synthesis were subject to response surface modeling (RSM). Only four approximation functions were obtained by performing multiple linear regression (MLR) of data according to Box-Behnken design;

$$I(D)/I(G) = 0.899 - 0.310C - 0.095D - 0.107C*D - 0.089C^2 - 0.116E^2 \quad : R^2 = 0.962$$

$$I(2D)/I(G) = 0.747 + 0.395C - 0.146E - 0.135C*E \quad : R^2 = 0.946$$

$$FWHM(G)/cm^{-1} = 45.159 - 14.782C - 3.356D + 3.059E - 3.064E^2 \quad : R^2 = 0.987$$

$$FWHM(2D)/cm^{-1} = 71.580 - 16.515C - 2.391D + 4.172E + 3.860C*E \quad : R^2 = 0.978$$

where $C = 0.02*((growth\ temperature)/°C) - 20$, $D = 0.004*((Ar\ flow\ rate + H_2\ flow\ rate)/sccm) - 3$ and $E = 3.333*((acetylene\ flow\ rate)/sccm) - 2.333$. Growth temperature is a key factor to control both defect amount and number of graphene layers by adjusting nucleation rate. The best condition for synthesizing graphene under our investigating space was growth temperature of 1050 °C and Ar, H_2, C_2H_2 flow rates of 900, 100, 0.4 sccm, respectively.

Graphene samples prepared by above condition on different substrates including SiO_2/Si and Cu were then functionalized by 4-nitrobenzenediazonium tetrafluoroborate (4NBDT) and the functionalized graphene samples were characterized by Raman spectroscopy and X-ray photoelectron spectroscopy (XPS). Unreacted diazonium salts were remained on functionalized graphene samples on SiO_2/Si. On the other hand, functionalized graphene samples on Cu afforded cleaner surfaces. Moreover, in the presence of light (UV-A region), degree of functionalization was enhanced whereas the grafting reaction slowly occurred in the dark. Light in such

region was presumably an electron-transfer activator because it could generate hot electrons from graphene and they could be transferred to 4-nitrophenyl cations generated simultaneously from diazonium salt also accelerated by blacklight.

REFERENCES

1. Geim, A. K. and Novoselov, K. S., The Rise of Graphene. *Nature Materials* **2007,** *6*, 183.

2. Wallace, P. R., The Band Theory of Graphite. *Physical Review* **1947,** *71* (9), 622-634.

3. Peierls, R., Quelques Propriétés Typiques des Corps Solides. **1935**.

4. Landau, L. D., Zur Theorie der Phasenumwandlungen II. *Phys. Z. Sowjetunion* **1937,** *11*, 26-35.

5. Boehm, H. P., Clauss, A., Fischer, G. O. and Hofmann, U., Das Adsorptionsverhalten sehr dünner Kohlenstoff-Folien. *Zeitschrift für anorganische und allgemeine Chemie* **1962,** *316* (3-4), 119-127.

6. Novoselov, K. S., Geim, A. K., Morozov, S. V., Jiang, D., Zhang, Y., Dubonos, S. V., Grigorieva, I. V. and Firsov, A. A., Electric Field Effect in Atomically Thin Carbon Films. *Science* **2004,** *306* (5696), 666.

7. Novoselov, K. S., Jiang, D., Schedin, F., Booth, T. J., Khotkevich, V. V., Morozov, S. V. and Geim, A. K., Two-dimensional Atomic Crystals. *Proceedings of the National Academy of Sciences of the United States of America* **2005,** *102* (30), 10451.

8. Shahil, K. M. F. and Balandin, A. A., Thermal Properties of Graphene and Multilayer Graphene: Applications in Thermal Interface Materials. *Solid State Communications* **2012,** *152* (15), 1331-1340.

9. Kim, K. S., Zhao, Y., Jang, H., Lee, S. Y., Kim, J. M., Kim, K. S., Ahn, J.-H., Kim, P., Choi, J.-Y. and Hong, B. H., Large-scale Pattern Growth of Graphene Films for Stretchable Transparent Electrodes. *Nature* **2009,** *457*, 706.

10. Li, X., Cai, W., An, J., Kim, S., Nah, J., Yang, D., Piner, R., Velamakanni, A., Jung, I., Tutuc, E., Banerjee, S. K., Colombo, L. and Ruoff, R. S., Large-Area Synthesis of High-Quality and Uniform Graphene Films on Copper Foils. *Science* **2009,** *324* (5932), 1312.

11. Rao, C. N. R., Sood, A. K., Subrahmanyam, K. S. and Govindaraj, A., Graphene: The New Two‐Dimensional Nanomaterial. *Angewandte Chemie International Edition* **2009,** *48* (42), 7752-7777.

12. Hummers, W. S. and Offeman, R. E., Preparation of Graphitic Oxide. *Journal of the American Chemical Society* **1958,** *80* (6), 1339-1339.

13. Pei, S. and Cheng, H.-M., The Reduction of Graphene Oxide. *Carbon* **2012,** *50* (9), 3210-3228.

14. Mishra, N., Boeckl, J., Motta, N. and Iacopi, F., Graphene Growth on Silicon Carbide: A Review *Physica Status Solidi A* **2016,** *213* (9), 2269-2269.

15. Yu, Q., Lian, J., Siriponglert, S., Li, H., Chen, Y. P. and Pei, S.-S., Graphene Segregated on Ni Surfaces and Transferred to Insulators. *Applied Physics Letters* **2008,** *93* (11), 113103.

16. Lee, H. C., Liu, W.-W., Chai, S.-P., Mohamed, A. R., Lai, C. W., Khe, C.-S., Voon, C. H., Hashim, U. and Hidayah, N. M. S., Synthesis of Single-layer Graphene: A Review of Recent Development. *Procedia Chemistry* **2016,** *19*, 916-921.

17. Chen, B., Huang, H., Ma, X., Huang, L., Zhang, Z. and Peng, L.-M., How Good Can CVD-grown Monolayer Braphene Be? *Nanoscale* **2014,** *6* (24), 15255-15261.

18. Wassei Jonathan, K., Mecklenburg, M., Torres Jaime, A., Fowler Jesse, D., Regan, B. C., Kaner Richard, B. and Weiller Bruce, H., Chemical Vapor Deposition of Graphene on Copper from Methane, Ethane and Propane: Evidence for Bilayer Selectivity. *Small* **2012,** *8* (9), 1415-1422.

19. Kumar, S., McEvoy, N., Lutz, T., Keeley, G., Whiteside, N., Blau, W. and Duesberg, G. S., Low Temperature Graphene Growth. *ECS Transactions* **2009,** *19* (5), 175-181.

20. Zhao, P., Kumamoto, A., Kim, S., Chen, X., Hou, B., Chiashi, S., Einarsson, E., Ikuhara, Y. and Maruyama, S., Self-Limiting Chemical Vapor Deposition Growth of Monolayer Graphene from Ethanol. *The Journal of Physical Chemistry C* **2013,** *117* (20), 10755-10763.

21. Osikoya, A. O., Parlak, O., Murugan, N. A., Dikio, E. D., Moloto, H., Uzun, L., Turner, A. P. F. and Tiwari, A., Acetylene-sourced CVD-synthesised Catalytically Active

Graphene for Electrochemical Biosensing. *Biosensors and Bioelectronics* **2017,** *89*, 496-504.

22.   Alexander, G., Kurt, K. and Denis, V. V., Dynamics of Graphene Growth on a Metal Surface: a Time-dependent Photoemission Study. *New Journal of Physics* **2009,** *11* (7), 073050.

23.   Seo, D. H., Pineda, S., Fang, J., Gozukara, Y., Yick, S., Bendavid, A., Lam, S. K. H., Murdock, A. T., Murphy, A. B., Han, Z. J. and Ostrikov, K., Single-step Ambient-air Synthesis of Graphene from Renewable Precursors as Electrochemical Genosensor. *Nature Communications* **2017,** *8*, 14217.

24.   Zhang, Y., Zhang, L. and Zhou, C., Review of Chemical Vapor Deposition of Graphene and Related Applications. *Accounts of Chemical Research* **2013,** *46* (10), 2329-2339.

25.   Park, J. and Yan, M., Covalent Functionalization of Graphene with Reactive Intermediates. *Accounts of Chemical Research* **2013,** *46* (1), 181-189.

26.   Lee, S.-H., Chung, H.-J., Heo, J., Yang, H., Shin, J., Chung, U. I. and Seo, S., Band Gap Opening by Two-Dimensional Manifestation of Peierls Instability in Graphene. *ACS Nano* **2011,** *5* (4), 2964-2969.

27.   Skomski, R., Dowben, P. A., Sky Driver, M. and Kelber, J. A., Sublattice-induced Symmetry Breaking and Band-gap Formation in Graphene. *Materials Horizons* **2014,** *1* (6), 563-571.

28.   Kandemir, B. S. and Mogulkoc, A., Chiral Symmetry Breaking by a Magnetic Field in Graphene. *Physics Letters A* **2015,** *379* (36), 2120-2124.

29.   Villamagua, L., Carini, M., Stashans, A. and Vacacela Gomez, C., Band Gap Engineering of Graphene through Quantum Confinement and Edge Distortions. *Ricerche di Matematica* **2016,** *65* (2), 579-584.

30.   Mali, K. S., Greenwood, J., Adisoejoso, J., Phillipson, R. and De Feyter, S., Nanostructuring Graphene for Controlled and Reproducible Functionalization. *Nanoscale* **2015,** *7* (5), 1566-1585.

31.   Wang, Q. H. and Hersam, M. C., Room-temperature Molecular-resolution Characterization of Self-assembled Organic Monolayers on Epitaxial Graphene. *Nature Chemistry* **2009,** *1*, 206.

32.   Järvinen, P., Hämäläinen, S. K., Banerjee, K., Häkkinen, P., Ijäs, M., Harju, A. and Liljeroth, P., Molecular Self-Assembly on Graphene on SiO_2 and h-BN Substrates. *Nano Letters* **2013,** *13* (7), 3199-3204.

33.   Prado, M. C., Nascimento, R., Moura, L. G., Matos, M. J. S., Mazzoni, M. S. C., Cancado, L. G., Chacham, H. and Neves, B. R. A., Two-Dimensional Molecular Crystals of Phosphonic Acids on Graphene. *ACS Nano* **2011,** *5* (1), 394-398.

34.   Alaboson, J. M. P., Sham, C.-H., Kewalramani, S., Emery, J. D., Johns, J. E., Deshpande, A., Chien, T., Bedzyk, M. J., Elam, J. W., Pellin, M. J. and Hersam, M. C., Templating Sub-10 nm Atomic Layer Deposited Oxide Nanostructures on Graphene via One-Dimensional Organic Self-Assembled Monolayers. *Nano Letters* **2013,** *13* (12), 5763-5770.

35.   Georgakilas, V., Otyepka, M., Bourlinos, A. B., Chandra, V., Kim, N., Kemp, K. C., Hobza, P., Zboril, R. and Kim, K. S., Functionalization of Graphene: Covalent and Non-Covalent Approaches, Derivatives and Applications. *Chemical Reviews* **2012,** *112* (11), 6156-6214.

36.   Wang, C., Xiao, B. and Ding, Y.-h., Theoretical Investigation on the Healing Mechanism of Divacancy Defect in Graphene Growth by Reaction with Ethylene and Acetylene. *New Journal of Chemistry* **2013,** *37* (3), 640-645.

37.   Daiyu, K., Shintaro, S., Katsunori, Y., Naoki, H., Motonobu, S., Mizuhisa, N. and Naoki, Y., Low-Temperature Synthesis of Graphene and Fabrication of Top-Gated Field Effect Transistors without Using Transfer Processes. *Applied Physics Express* **2010,** *3* (2), 025102.

38.   Rümmeli, M. H., Bachmatiuk, A., Scott, A., Börrnert, F., Warner, J. H., Hoffman, V., Lin, J.-H., Cuniberti, G. and Büchner, B., Direct Low-Temperature Nanographene CVD Synthesis over a Dielectric Insulator. *ACS Nano* **2010,** *4* (7), 4206-4210.

39.   Ramón, M. E., Gupta, A., Corbet, C., Ferrer, D. A., Movva, H. C. P., Carpenter, G., Colombo, L., Bourianoff, G., Doczy, M., Akinwande, D., Tutuc, E. and Banerjee, S. K., CMOS-Compatible Synthesis of Large-Area, High-Mobility Graphene by Chemical Vapor Deposition of Acetylene on Cobalt Thin Films. *ACS Nano* **2011,** *5* (9), 7198-7204.

40. Sun, J., Lindvall, N., Cole, M. T., Teo, K. B. K. and Yurgens, A., Large-area Uniform Graphene-like Thin Films Grown by Chemical Vapor Deposition Directly on Silicon Nitride. *Applied Physics Letters* **2011,** *98* (25), 252107.

41. Weatherup, R. S., Bayer, B. C., Blume, R., Ducati, C., Baehtz, C., Schlögl, R. and Hofmann, S., In Situ Characterization of Alloy Catalysts for Low-Temperature Graphene Growth. *Nano Letters* **2011,** *11* (10), 4154-4160.

42. Qi, M., Ren, Z., Jiao, Y., Zhou, Y., Xu, X., Li, W., Li, J., Zheng, X. and Bai, J., Hydrogen Kinetics on Scalable Graphene Growth by Atmospheric Pressure Chemical Vapor Deposition with Acetylene. *The Journal of Physical Chemistry C* **2013,** *117* (27), 14348-14353.

43. Chen, C.-S. and Hsieh, C.-K., Effects of Acetylene Flow Rate and Processing Temperature on Graphene Films Grown by Thermal Chemical Vapor Deposition. *Thin Solid Films* **2015,** *584*, 265-269.

44. Cortés, A., Celedón, C. and Zarate, R., CVD Synthesis of Graphene from Acetylene Catalyzed by a Reduced CuO Thin Film Deposited on SiO_2 Substrate. *Journal of the Chilean Chemical Society* **2015,** *60*, 2911-2913.

45. Meng, Y., Shinichirou, S., Masato, O., Ken, S. and Hideo, M., Electronic Properties and Strain Sensitivity of CVD-grown Graphene with Acetylene. *Japanese Journal of Applied Physics* **2016,** *55* (4S), 04EP05.

46. Yang, M., Sasaki, S., Suzuki, K. and Miura, H., Control of the Nucleation and Quality of Graphene Grown by Low-pressure Chemical Vapor Deposition with Acetylene. *Applied Surface Science* **2016,** *366*, 219-226.

47. Muñoz, R., Munuera, C., Martínez, J. I., Azpeitia, J., Gómez-Aleixandre, C. and García-Hernández, M., Low Temperature Metal Free Growth of Graphene on Insulating Substrates by Plasma Assisted Chemical Vapor Deposition. *2D Materials* **2017,** *4* (1), 015009.

48. Cabrero-Vilatela, A., Weatherup, R. S., Braeuninger-Weimer, P., Caneva, S. and Hofmann, S., Towards a General Growth Model for Graphene CVD on Transition Metal Catalysts. *Nanoscale* **2016,** *8* (4), 2149-2158.

49. Bekyarova, E., Itkis, M. E., Ramesh, P., Berger, C., Sprinkle, M., de Heer, W. A. and Haddon, R. C., Chemical Modification of Epitaxial Graphene: Spontaneous

Grafting of Aryl Groups. *Journal of the American Chemical Society* **2009,** *131* (4), 1336-1337.

50.   Lim, H., Lee, J. S., Shin, H.-J., Shin, H. S. and Choi, H. C., Spatially Resolved Spontaneous Reactivity of Diazonium Salt on Edge and Basal Plane of Graphene without Surfactant and Its Doping Effect. *Langmuir* **2010,** *26* (14), 12278-12284.

51.   Sharma, R., Baik, J. H., Perera, C. J. and Strano, M. S., Anomalously Large Reactivity of Single Graphene Layers and Edges toward Electron Transfer Chemistries. *Nano Letters* **2010,** *10* (2), 398-405.

52.   Sinitskii, A., Dimiev, A., Corley, D. A., Fursina, A. A., Kosynkin, D. V. and Tour, J. M., Kinetics of Diazonium Functionalization of Chemically Converted Graphene Nanoribbons. *ACS Nano* **2010,** *4* (4), 1949-1954.

53.   Fan, X., Nouchi, R. and Tanigaki, K., Effect of Charge Puddles and Ripples on the Chemical Reactivity of Single Layer Graphene Supported by SiO2/Si Substrate. *The Journal of Physical Chemistry C* **2011,** *115* (26), 12960-12964.

54.   Huang, P., Zhu, H., Jing, L., Zhao, Y. and Gao, X., Graphene Covalently Binding Aryl Groups: Conductivity Increases Rather than Decreases. *ACS Nano* **2011,** *5* (10), 7945-7949.

55.   Jin, Z., McNicholas, T. P., Shih, C.-J., Wang, Q. H., Paulus, G. L. C., Hilmer, A. J., Shimizu, S. and Strano, M. S., Click Chemistry on Solution-Dispersed Graphene and Monolayer CVD Graphene. *Chemistry of Materials* **2011,** *23* (14), 3362-3370.

56.   Gan, L., Zhang, D. and Guo, X., Electrochemistry: An Efficient Way to Chemically Modify Individual Monolayers of Graphene. *Small* **2012,** *8* (9), 1326-1330.

57.   Wang, Q. H., Jin, Z., Kim, K. K., Hilmer, A. J., Paulus, G. L. C., Shih, C.-J., Ham, M.-H., Sanchez-Yamagishi, J. D., Watanabe, K., Taniguchi, T., Kong, J., Jarillo-Herrero, P. and Strano, M. S., Understanding and Controlling the Substrate Effect on Graphene Electron-transfer Chemistry via Reactivity Imprint Lithography. *Nature Chemistry* **2012,** *4*, 724.

58.   Bissett, M. A., Konabe, S., Okada, S., Tsuji, M. and Ago, H., Enhanced Chemical Reactivity of Graphene Induced by Mechanical Strain. *ACS Nano* **2013,** *7* (11), 10335-10343.

59.   Pembroke, E., Ruan, G., Sinitskii, A., Corley, D. A., Yan, Z., Sun, Z. and Tour, J. M., Effect of Anchor and Functional Groups in Functionalized Graphene Devices. *Nano Research* **2013,** *6* (2), 138-148.

60.   Wu, Q., Wu, Y., Hao, Y., Geng, J., Charlton, M., Chen, S., Ren, Y., Ji, H., Li, H., Boukhvalov, D. W., Piner, R. D., Bielawski, C. W. and Ruoff, R. S., Selective Surface Functionalization at Regions of High Local Curvature in Graphene. *Chemical Communications* **2013,** *49* (7), 677-679.

61.   Kibena, E., Marandi, M., Sammelselg, V., Tammeveski, K., Jensen Bjarke, B. E., Mortensen Anders, B., Lillethorup, M., Kongsfelt, M., Pedersen Steen, U. and Daasbjerg, K., Electrochemical Behaviour of HOPG and CVD-Grown Graphene Electrodes Modified with Thick Anthraquinone Films by Diazonium Reduction. *Electroanalysis* **2014,** *26* (12), 2619-2630.

62.   Eissa, S., Jimenez, G. C., Mahvash, F., Guermoune, A., Tlili, C., Szkopek, T., Zourob, M. and Siaj, M., Functionalized CVD Monolayer Graphene for Label-free Impedimetric Biosensing. *Nano Research* **2015,** *8* (5), 1698-1709.

63.   Mooste, M., Kibena, E., Kozlova, J., Marandi, M., Matisen, L., Niilisk, A., Sammelselg, V. and Tammeveski, K., Electrografting and Morphological Studies of Chemical Vapour Deposition Grown Graphene Sheets Modified by Electroreduction of Aryldiazonium Salts. *Electrochimica Acta* **2015,** *161*, 195-204.

64.   Assaud, L., Massonnet, N., Evrard, D., Vergnes, H., Salvagnac, L., Conédéra, V., Noé, L., Monthioux, M., Gros, P., Temple-Boyer, P. and Caussat, B., A New Route for the Integration of a Graphene/Diazonium/PEDOT Electrode towards Antioxidant Biomarker Detection. *Journal of Electroanalytical Chemistry* **2016,** *771*, 73-79.

65.   N'Diaye, J., Hmam, O., Zidi, M., Tavares Ana, C., Izquierdo, R., Szkopek, T. and Siaj, M., One-Step In-Situ Growth of Core–Shell SiC@Graphene Nanoparticles/Graphene Hybrids by Chemical Vapor Deposition. *Advanced Materials Interfaces* **2016,** *3* (8), 1500806.

66. Xia, Z., Leonardi, F., Gobbi, M., Liu, Y., Bellani, V., Liscio, A., Kovtun, A., Li, R., Feng, X., Orgiu, E., Samorì, P., Treossi, E. and Palermo, V., Electrochemical Functionalization of Graphene at the Nanoscale with Self-Assembling Diazonium Salts. *ACS Nano* **2016**, *10* (7), 7125-7134.

67. Matulková, I., Kovaříček, P., Šlouf, M., Němec, I. and Kalbáč, M., Surface Enhanced Infrared Absorption Spectroscopy for Graphene Functionalization on Copper. *Carbon* **2017**, *124*, 250-255.

68. Pinson, J. and Podvorica, F., Attachment of Organic Layers to Conductive or Semiconductive Surfaces by Reduction of Diazonium Salts. *Chemical Society Reviews* **2005**, *34* (5), 429-439.

69. Jain, N., Bansal, T., Durcan, C. and Yu, B., Graphene-Based Interconnects on Hexagonal Boron Nitride Substrate. *IEEE Electron Device Letters* **2012**, *33* (7), 925-927.

70. Ferrari, A. C. and Basko, D. M., Raman Spectroscopy as a Versatile Tool for Studying the Properties of Graphene. *Nature Nanotechnology* **2013**, *8*, 235.

71. Malard, L. M., Pimenta, M. A., Dresselhaus, G. and Dresselhaus, M. S., Raman Spectroscopy in Graphene. *Physics Reports* **2009**, *473* (5), 51-87.

72. Lucchese, M. M., Stavale, F., Ferreira, E. H. M., Vilani, C., Moutinho, M. V. O., Capaz, R. B., Achete, C. A. and Jorio, A., Quantifying Ion-induced Defects and Raman Relaxation Length in Graphene. *Carbon* **2010**, *48* (5), 1592-1597.

73. Cançado, L. G., Jorio, A., Ferreira, E. H. M., Stavale, F., Achete, C. A., Capaz, R. B., Moutinho, M. V. O., Lombardo, A., Kulmala, T. S. and Ferrari, A. C., Quantifying Defects in Graphene via Raman Spectroscopy at Different Excitation Energies. *Nano Letters* **2011**, *11* (8), 3190-3196.

74. Wei, D., Liu, Y., Wang, Y., Zhang, H., Huang, L. and Yu, G., Synthesis of N-Doped Graphene by Chemical Vapor Deposition and Its Electrical Properties. *Nano Letters* **2009**, *9* (5), 1752-1758.

75. Analytical Methods Committee, A. N., Experimental Design and Optimisation (4): Plackett-Burman Designs. *Analytical Methods* **2013**, *5* (8), 1901-1903.

76. Prekodravac, J., Marković, Z., Jovanović, S., Budimir, M., Peruško, D., Holclajtner-Antunović, I., Pavlović, V., Syrgiannis, Z., Bonasera, A. and Todorović-Marković,

B., The Effect of Annealing Temperature and Time on Synthesis of Graphene Thin Films by Rapid Thermal Annealing. *Synthetic Metals* **2015,** *209*, 461-467.

77. Jin, Y., Hu, B., Wei, Z., Luo, Z., Wei, D., Xi, Y., Zhang, Y. and Liu, Y., Roles of H_2 in Annealing and Growth Times of Graphene CVD Synthesis over Copper Foil. *Journal of Materials Chemistry A* **2014,** *2* (38), 16208-16216.

78. Papon, R., Pierlot, C., Sharma, S., Shinde Sachin, M., Kalita, G. and Tanemura, M., Optimization of CVD Parameters for Graphene Synthesis through Design of Experiments. *Physica Status Solidi B* **2016,** *254* (5), 1600629.

79. Chaitoglou, S. and Bertran, E., Effect of Temperature on Graphene Grown by Chemical Vapor Deposition. *Journal of Materials Science* **2017,** *52* (13), 8348-8356.

80. Parmeggiani, F., Fear of the Dark: Diazo Printing by Photochemical Decomposition of Aryldiazonium Tetrafluoroborates. *Journal of Chemical Education* **2014,** *91* (5), 692-695.

81. Liu, H., Ryu, S., Chen, Z., Steigerwald, M. L., Nuckolls, C. and Brus, L. E., Photochemical Reactivity of Graphene. *Journal of the American Chemical Society* **2009,** *131* (47), 17099-17101.

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

APPENDIX

Python script for analysis of Raman spectra

```python
1   import os
2   import subprocess
3   import RI_DG_2D_version_1_22
4   import AG_ver_1_02
5   import AF_1_02
6   import SA_1_01
7   import shutil
8   import Histogram_version_1_02
9
10  #RI_DG_2D_version_1_22.interpretator('Prefix file', First number, Last number, 'output
    filename')
11  RI_DG_2D_version_1_22.interpretator('data1_', 1, 100, '00001')
12  RI_DG_2D_version_1_22.interpretator('data2_', 1, 100, '00002')
13  RI_DG_2D_version_1_22.interpretator('data3_', 1, 100, '00003')
14  #AG_ver_1_02.AverageGraph('Prefix file', First number, Last number)
15  AG_ver_1_02.AverageGraph('data1_', 1, 100)
16  AG_ver_1_02.AverageGraph('data2_', 1, 100)
17  AG_ver_1_02.AverageGraph('data3_', 1, 100)
18  #os.rename('Original name', 'new name')
19  shutil.move('Average_data1_from1to100.csv', 'Average_data_0001.csv')
20  shutil.move('Average_data2_from1to100.csv', 'Average_data_0002.csv')
21  shutil.move('Average_data3_from1to100.csv', 'Average_data_0003.csv')
22  AG_ver_1_02.AverageGraph('Average_data_', 1, 3)
23  shutil.move('Average_Average_data_from1to3.csv', 'Average_graph.csv')
24  #AF_1_02.AssemblyFile(No file, No spectra collected , 'Filename1', 'Filename2',
    'Filename3', 'output filename')
25  AF_1_02.AssemblyFile(3, 250, '00001', '00002', '00003', 'StatRawdata')
26  Histogram_version_1_02.HistogramMaker('StatRawdata')
27  SA_1_01.StatAnalysis('StatRawdata')
```

Python function for calculating Raman parameters of graphene

```python
def interpretator(Prefix, first_file_number, Last_file_number, Net_file_name):
    import pandas as pd
    import numpy as np
    import csv
    from math import factorial

    print("Graphene's Raman Spectra Interpretator")
    print("by Bhobnibhit C updated Feb 13th 2018 version 1.21")
    print("\nThis program will calculate I(D)/I(G), I(2D)/I(G), Pos(G), FWHM(G),
    Pos(2D) and FWHM(2D) from Raman spectra\n")
    print("\nThis version, automatically save all files\n")

    #create array for keeping all data
    Net = []
    for i in range(1000):
        Net.append([])
        for j in range(7):
            Net[i].append(0)
    h = 0


    #Prefix = input('Enter input file prefix: ')
    #first_file_number = Current_file_number = int(input('First file number: '))
    #Last_file_number = int(input('Last file number: '))
    Current_file_number =  first_file_number
    filename1 = 'love'

    print("\n#########################################################################
    ###########################\n")
    print("****Analysis in Progress****")
    #print("\n")
    #print(Prefix)
    #print(first_file_number)
    #print(Last_file_number)
    #print(filename1)

    #making file name
    while first_file_number-1 < Current_file_number < Last_file_number +1:
        if 0 < Current_file_number < 10:
            filename1 = Prefix + "000" + str(Current_file_number) + ".csv"
            filename_input = Prefix + "000" + str(Current_file_number)
            #print(filename1)
            Current_file_number = Current_file_number + 1
        elif 9 < Current_file_number < 100:
            filename1 = Prefix + "00" + str(Current_file_number) + ".csv"
            filename_input = Prefix + "000" + str(Current_file_number)
            #print(filename1)
            Current_file_number = Current_file_number + 1
        elif 99 < Current_file_number < 1000:
            filename1 = Prefix + "0" + str(Current_file_number) + ".csv"
            filename_input = Prefix + "000" + str(Current_file_number)
            #print(filename1)
            Current_file_number = Current_file_number + 1
        c_reader = csv.reader(open(filename1, 'r'), delimiter=',')
        RamanShift_Full_list = list(zip(*c_reader))[0]
        c_reader = csv.reader(open(filename1, 'r'), delimiter=',')
        Intensity_Full_list = list(zip(*c_reader))[1]


        #print(RamanShift_Full)
        #print("\n\n\n\n")
        #print(Intensity_Full)

        #df = pd.read_csv("point2_0001.csv")
        #df = pd.read_csv(filename)
        #RamanShift_Full = df.RamanShift #you can also use df['column_name']
        #Intensity_Full = df.Intensity #you can also use df['column_name']
```

```
65
66          #print(RamanShift_Full)
67          RamanShift_real = [] #declare array for redording raman shift
68          Intensity_real = [] #declare array for redording intensity
69          Intensity_Smooth = []
70          baseline_2D = []
71          RamanShift_baseline_2D = []
72          Intensity_baseline_2D = []
73          baseline_2D = []
74          baseline_2D_corrected = []
75          RamanShift_Full = []
76          Intensity_Full = []
77
78          #making zero array
79          for k in range(0,100000):
80              RamanShift_Full.append(0)
81          for k in range(0,100000):
82              Intensity_Full.append(0)
83          for k in range(0,100000):
84              RamanShift_real.append(0)
85          for k in range(0,100000):
86              Intensity_real.append(0)
87          for k in range(0,3400):
88              Intensity_Smooth.append(0)
89          for k in range(0,1300):
90              RamanShift_baseline_2D.append(0)
91          for k in range(0,1300):
92              Intensity_baseline_2D.append(0)
93          for k in range(0,1300):
94              baseline_2D.append(0)
95          for k in range(0,1300):
96              baseline_2D_corrected.append(0)
97
98          for i in range (0, len(RamanShift_Full_list)):
99              RamanShift_Full[i] = float(RamanShift_Full_list[i])
100             Intensity_Full[i] = float(Intensity_Full_list[i])
101         #print(RamanShift_Full)
102         #print("\n\n\n\n\n\n\n\n\n")
103         #print(Intensity_Full)
104         #print("Input file: ", filename1)
105         #print("\n")
106         #print("****Analysis in Progress****")
107         #print("\n")
108         #extracting only 1000-3400
109         #print(len(RamanShift_Full))
110         filename_reserved =  "spectra1000to3400"+ filename_input +".csv"
111         with open(filename_reserved, "w") as sp:
112             for i in range (0, len(RamanShift_Full)):
113                 while 1000 < RamanShift_Full[i] < 3400 :
114                     print(RamanShift_Full[i],",",Intensity_Full[i], file=sp)
115                     RamanShift_real[int(RamanShift_Full[i])] = float(RamanShift_Full[i])
116                     Intensity_real[int(Intensity_Full[i])] = float(Intensity_Full[i])
117                     break
118         sp.close()
119         RamanShift_real = np.array(RamanShift_real)
120         Intensity_real = np.array(Intensity_real)
121         q = 0
122
123         #marking 2 points
124         for i in range (0, len(RamanShift_Full)):
125             while 2199 < RamanShift_Full[i] < 2200 :
126                 #print(RamanShift_Full[i],",",Intensity_Full[i])
127                 x1 = RamanShift_Full[i]
128                 y1 = Intensity_Full[i]
129                 break
130         for i in range (0, len(RamanShift_Full)):
131             while 3099 < RamanShift_Full[i] < 3100 :
```

```python
132                      #print(RamanShift_Full[i],",",Intensity_Full[i])
133                      x2 = RamanShift_Full[i]
134                      y2 = Intensity_Full[i]
135                      break
136
137          #print(x1,y1,x2,y2)
138          #calculating slope and intercept
139          slope_2D = (y2-y1)/(x2-x1)
140          intercept_2D = ((y2 - slope_2D*x2) + (y1 - slope_2D*x1))/2
141          #print(slope_2D,intercept_2D)
142          #create baseline array
143          i = 0
144          j = 0
145          k = 0
146          filename_reserved =  "peak_2D"+ filename_input +".csv"
147          with open(filename_reserved, "w") as sp:
148              for i in range (0, len(RamanShift_Full)):
149                  while 2000 < RamanShift_Full[i] < 3200 :
150                      j = j +1
151                      print(RamanShift_Full[i],",",Intensity_Full[i], file=sp)
152                      RamanShift_baseline_2D[k] = RamanShift_Full[i]
153                      Intensity_baseline_2D[k] = Intensity_Full[i]
154                      k = k + 1
155                      break
156          sp.close()
157          #print(RamanShift_baseline_2D)
158          #print(len(RamanShift_baseline_2D))
159          #creating fitting array
160          filename_reserved =  "2DbaselineFit"+ filename_input +".csv"
161          with open(filename_reserved, "w") as sp:
162              for i in range (0, len(RamanShift_baseline_2D)):
163                  #print(baseline_2D[i])
164                  baseline_2D[i] = float((slope_2D)*(RamanShift_baseline_2D[i]) +
                     (intercept_2D))
165                  print(RamanShift_baseline_2D[i],",",baseline_2D[i], file=sp)
166                  #print(RamanShift_baseline_2D[i],",",baseline_2D[i])
167          sp.close()
168          #print(baseline_2D)
169          #print("\n\n\n\n")
170          #print(Intensity_baseline_2D)
171
172          #Substract baseline
173          filename_reserved =  "2D_baseline_corrected"+ filename_input +".csv"
174          with open(filename_reserved, "w") as sp:
175              for i in range (0, len(RamanShift_baseline_2D)):
176                  baseline_2D_corrected[i] = Intensity_baseline_2D[i] - baseline_2D[i]
177                  print(RamanShift_baseline_2D[i],",",baseline_2D_corrected[i], file=sp)
178          sp.close()
179
180          #finding peak_2D
181          max_2D = []
182          for k in range(0,20):
183              max_2D.append(0)
184          k = 0
185          #finding maximum of 2D
186          for i in range (1, len(RamanShift_baseline_2D)-1):
187              if baseline_2D_corrected[i-1] < baseline_2D_corrected[i] and
                 baseline_2D_corrected[i] > baseline_2D_corrected[i+1] and 2650 <
                 RamanShift_baseline_2D[i] < 2750:
188                  #print(RamanShift_baseline_2D[i],",",baseline_2D_corrected[i])
189                  max_2D[k] = baseline_2D_corrected[i]
190                  k = k +1
191          #print(max_2D)
192          int_2D = np.amax(max_2D)
193
194          for i in range (0, len(RamanShift_baseline_2D)):
195              if baseline_2D_corrected[i] == int_2D:
```

```
196                        pos_2D = RamanShift_baseline_2D[i]
197                #print(pos_2D,",",int_2D)
198                #print("Position of 2D peak is at", pos_2D,"cm-1")
199                #print("Intensity of 2D peak is", int_2D)
200                if int_2D > 0.015:
201                    #finding FWHM_2D
202                    half_height_2D = int_2D/2.0
203                    Pos_at_half_height_2D = []
204                    for j in range(0,20):
205                        Pos_at_half_height_2D.append(0)
206                    k = 0
207                    #print(half_height_2D)
208                    EF = 0.0001
209                    while Pos_at_half_height_2D[1] == 0:
210                        ErrorFWHM = float(EF)/100
211                        k = 0
212                        for j in range(0,20):
213                            Pos_at_half_height_2D.append(0)
214                        for i in range (0, len(RamanShift_baseline_2D)):
215                            if half_height_2D - half_height_2D*ErrorFWHM <
                               baseline_2D_corrected[i] < half_height_2D + half_height_2D*ErrorFWHM:
216                                #print(RamanShift_baseline_2D[i],",",baseline_2D_corrected[i])
217                                Pos_at_half_height_2D[k] = RamanShift_baseline_2D[i]
218                                k = k + 1
219                        EF = EF + 0.0001
220                    #print(Pos_at_half_height_2D)
221                    #recheck that FWHM is correct
222                    EF = 0.0001
223                    if Pos_at_half_height_2D[0] < pos_2D and Pos_at_half_height_2D[1] < pos_2D:
224                        Pos_at_half_height_2D[1] = 0
225                        #create UL and LL
226                        for i in range (0, len(RamanShift_baseline_2D)):
227                            if RamanShift_baseline_2D[i] == pos_2D:
228                                LL = i
229                            elif 2800 < RamanShift_baseline_2D[i] < 2801:
230                                UL = i
231                        while Pos_at_half_height_2D[1] == 0:
232                            ErrorFWHM = float(EF)/100
233                            for i in range (LL, UL):
234                                #print("both positions are less than pos_2D")
235                                if RamanShift_baseline_2D[i] > pos_2D:
236                                    if half_height_2D - half_height_2D*ErrorFWHM <
                                       baseline_2D_corrected[i] < half_height_2D +
                                       half_height_2D*ErrorFWHM:
237
                                        #print(RamanShift_baseline_2D[i],",",baseline_2D_correcte
                                        d[i])
238                                        Pos_at_half_height_2D[1] = RamanShift_baseline_2D[i]
239                            EF = EF + 0.0001
240                    elif Pos_at_half_height_2D[1] > pos_2D and Pos_at_half_height_2D[0] > pos_2D:
241                        Pos_at_half_height_2D[1] = 0
242                        #create UL and LL
243                        for i in range (0, len(RamanShift_baseline_2D)):
244                            if RamanShift_baseline_2D[i] == pos_2D:
245                                UL = i
246                            elif 2500 < RamanShift_baseline_2D[i] < 2501:
247                                LL = i
248                        while Pos_at_half_height_2D[1] == 0:
249                            ErrorFWHM = float(EF)/100
250                            for i in range (LL, UL):
251                                #print("both positions are greater than pos_2D")
252                                if RamanShift_baseline_2D[i] < pos_2D:
253                                    #print(RamanShift_baseline_2D[i],baseline_2D_corrected[i])
254                                    if half_height_2D - half_height_2D*ErrorFWHM <
                                       baseline_2D_corrected[i] < half_height_2D +
                                       half_height_2D*ErrorFWHM:
255
```

```
                                        #print(RamanShift_baseline_2D[i],",",baseline_2D_correcte
                                        d[i])
256                                     Pos_at_half_height_2D[1] = RamanShift_baseline_2D[i]
257                         EF = EF + 0.0001
258                 #print(Pos_at_half_height_2D[0],Pos_at_half_height_2D[1])
259                 #print("\n", Pos_at_half_height_2D)
260                 FWHM_2D = abs(float(Pos_at_half_height_2D[1]) -
                    float(Pos_at_half_height_2D[0]))
261                 #print(FWHM_2D)
262                 #print("Full width at half maximum of 2D peak is", FWHM_2D,"cm-1")
263                 #print("\n\n\n\n\n\n\n\n\n\n\n")
264
265                 baseline_DandG = []
266                 RamanShift_baseline_DandG = []
267                 Intensity_baseline_DandG = []
268                 baseline_DandG_corrected = []
269
270                 for k in range(0,726):
271                     RamanShift_baseline_DandG.append(0)
272                 for k in range(0,726):
273                     Intensity_baseline_DandG.append(0)
274                 for k in range(0,726):
275                     baseline_DandG.append(0)
276                 for k in range(0,726):
277                     baseline_DandG_corrected.append(0)
278
279                 #create baseline array for D ang G peak
280                 i = 0
281                 j = 0
282                 k = 0
283                 filename_reserved =  "peak_D_and_G"+ filename_input +".csv"
284                 with open(filename_reserved, "w") as sp:
285                     for i in range (0, len(RamanShift_Full)):
286                         while 1100 < RamanShift_Full[i] < 1800 :
287                             j = j +1
288                             print(RamanShift_Full[i],",",Intensity_Full[i], file=sp)
289                             RamanShift_baseline_DandG[k] = RamanShift_Full[i]
290                             Intensity_baseline_DandG[k] = Intensity_Full[i]
291                             k = k + 1
292                             break
293                 sp.close()
294                 #print(RamanShift_baseline_DandG)
295                 #print(j)
296                 #print(len(RamanShift_baseline_DandG))
297                 #print(Intensity_baseline_DandG)
298                 #print(len(Intensity_baseline_DandG))
299
300                 #marking 2 points for D and G
301                 for i in range (0, len(RamanShift_Full)):
302                         while 1100 < RamanShift_Full[i] < 1102 :
303                             #print(RamanShift_Full[i],",",Intensity_Full[i])
304                             x1 = RamanShift_Full[i]
305                             y1 = Intensity_Full[i]
306                             break
307                 for i in range (0, len(RamanShift_Full)):
308                         while 1798 < RamanShift_Full[i] < 1800 :
309                             #print(RamanShift_Full[i],",",Intensity_Full[i])
310                             x2 = RamanShift_Full[i]
311                             y2 = Intensity_Full[i]
312                             break
313
314                 #print(x1,y1,x2,y2)
315                 #calculating slope and intercept
316                 slope_DandG = (y2-y1)/(x2-x1)
317                 intercept_DandG = ((y2 - slope_DandG*x2) + (y1 - slope_DandG*x1))/2
318                 #print(slope_DandG,intercept_DandG)
319
```

```
320                        #creating fitting array
321                        filename_reserved =  "DandG_baselineFit"+ filename_input +".csv"
322                        with open(filename_reserved, "w") as sp:
323                            for i in range (0, len(RamanShift_baseline_DandG)):
324                                #print(baseline_DandG[i])
325                                baseline_DandG[i] =
                                   float((slope_DandG)*(RamanShift_baseline_DandG[i]) +
                                   (intercept_DandG))
326                                print(RamanShift_baseline_DandG[i],",",baseline_DandG[i], file=sp)
327                                #print(RamanShift_baseline_DandG[i],",",baseline_DandG[i])
328                        sp.close()
329                        #print(baseline_DandG)
330                        #print("\n\n\n\n")
331                        #print(Intensity_baseline_DandG)
332                        #Substract baseline
333                        filename_reserved =  "DandG_baseline_corrected"+ filename_input +".csv"
334                        with open(filename_reserved, "w") as sp:
335                            for i in range (0, len(RamanShift_baseline_DandG)):
336                                baseline_DandG_corrected[i] = Intensity_baseline_DandG[i] -
                                   baseline_DandG[i]
337                                print(RamanShift_baseline_DandG[i],",",baseline_DandG_corrected[i],
                                   file=sp)
338                        sp.close()
339
340
341
342
343                        #finding peak_D
344                        max_D = []
345                        for k in range(0,20):
346                            max_D.append(0)
347                        k = 0
348                        #finding maximum of D
349
350                        for i in range (1, len(baseline_DandG_corrected)-1):
351                            if baseline_DandG_corrected[i-1] < baseline_DandG_corrected[i] and
                               baseline_DandG_corrected[i] > baseline_DandG_corrected[i+1] and 1300 <
                               RamanShift_baseline_DandG[i] < 1400:
352                                #print(RamanShift_baseline_D[i],",",baseline_D_corrected[i])
353                                max_D[k] = baseline_DandG_corrected[i]
354                                k = k +1
355                        #print(max_D)
356                        int_D = np.amax(max_D)
357
358                        for i in range (0, len(RamanShift_baseline_DandG)):
359                            if baseline_DandG_corrected[i] == int_D:
360                                pos_D = RamanShift_baseline_DandG[i]
361                            #elif int_D == 0:
362                                #pos_D = 0
363                        #print(pos_D,",",int_D)
364                        #print("Position of D peak is at", pos_D,"cm-1")
365                        #print("Intensity of D peak is", int_D)
366
367                        #finding peak_G
368                        max_G = []
369                        for k in range(0,20):
370                            max_G.append(0)
371                        k = 0
372                        #finding maximum of G
373                        int_G = np.amax(max_G)
374                        for i in range (1, len(RamanShift_baseline_DandG)-1):
375                            if baseline_DandG_corrected[i-1] < baseline_DandG_corrected[i] and
                               baseline_DandG_corrected[i] > baseline_DandG_corrected[i+1] and 1550 <
                               RamanShift_baseline_DandG[i] < 1600:
376                                #print(RamanShift_baseline_DandG[i],",",baseline_DandG_corrected[i])
377                                max_G[k] = baseline_DandG_corrected[i]
378                                k = k +1
```

```python
379                    #print(max_G)
380                    int_G = np.amax(max_G)
381
382                    for i in range (0, len(RamanShift_baseline_DandG)):
383                        if baseline_DandG_corrected[i] == int_G:
384                            pos_G = RamanShift_baseline_DandG[i]
385                        #elif int_G == 0:
386                            #pos_G = 0
387                    #print(pos_G,",",int_G)
388                    #print("Position of G peak is at", pos_G,"cm-1")
389                    #print("Intensity of G peak is", int_G)
390
391                    #finding FWHM_G
392                    half_height_G = int_G/2
393                    Pos_at_half_height_G = []
394                    for k in range(0,20):
395                        Pos_at_half_height_G.append(0)
396                    k = 0
397                    #print(half_height_G)
398                    EF = 0.001
399                    #define range of G searching
400                    for i in range (0, len(RamanShift_baseline_DandG)):
401                        if 1450 < RamanShift_baseline_DandG[i] < 1451:
402                            LL = i
403                        elif 1669 < RamanShift_baseline_DandG[i] < 1700:
404                            UL = i
405                    #print(LL,UL)
406                    if int_G != 0:
407                        while Pos_at_half_height_G[1] == 0:
408                            ErrorFWHM = float(EF)/100
409                            #print("FWHM_G 1st loop")
410                            #print(ErrorFWHM)
411                            k = 0
412                            for j in range(0,20):
413                                Pos_at_half_height_G.append(0)
414                            for i in range (LL, UL):
415                                #print(RamanShift_baseline_DandG[i])
416                                if half_height_G - half_height_G*ErrorFWHM <
                                   baseline_DandG_corrected[i] < half_height_G +
                                   half_height_G*ErrorFWHM:
417
                                    #print(RamanShift_baseline_DandG[i],",",baseline_DandG_correc
                                    ted[i])
418                                    Pos_at_half_height_G[k] = RamanShift_baseline_DandG[i]
419                                    k = k + 1
420                            EF = EF + 0.0001
421                        #recheck that FWHM is correct
422                        EF = 0.0001
423                        if Pos_at_half_height_G[0] < pos_G and Pos_at_half_height_G[1] < pos_G:
424                            Pos_at_half_height_G[1] = 0
425                            #print("FWHM_G 2nd loop")
426                            while Pos_at_half_height_G[1] == 0:
427                                ErrorFWHM = float(EF)/100
428                                for i in range (LL, UL):
429                                    if RamanShift_baseline_DandG[i] > pos_G:
430                                        if half_height_G - half_height_G*ErrorFWHM <
                                           baseline_DandG_corrected[i] < half_height_G +
                                           half_height_G*ErrorFWHM:
431
                                            #print(RamanShift_baseline_2D[i],",",baseline_2D_corr
                                            ected[i])
432                                            Pos_at_half_height_G[1] =
                                            RamanShift_baseline_DandG[i]
433                                EF = EF + 0.0001
434                        elif Pos_at_half_height_G[1] > pos_G and Pos_at_half_height_G[0] > pos_G:
435                            Pos_at_half_height_G[1] = 0
436                            #print("FWHM_G 3rd loop")
```

```
437                         while Pos_at_half_height_G[1] == 0:
438                             ErrorFWHM = float(EF)/100
439                             #print(ErrorFWHM)
440                             for i in range (LL, UL):
441                                 if RamanShift_baseline_DandG[i] < pos_G:
442
                                        #print(RamanShift_baseline_DandG[i],baseline_DandG_correc
                                        ted[i])
443                                     if half_height_G - half_height_G*ErrorFWHM <
                                        baseline_DandG_corrected[i] < half_height_G +
                                        half_height_G*ErrorFWHM:
444                                         #print("3rd record loop")
445
                                            #print(RamanShift_baseline_DandG[i],",",baseline_Dand
                                            G_corrected[i])
446                                         Pos_at_half_height_G[1] =
                                            RamanShift_baseline_DandG[i]
447                                 EF = EF + 0.1
448                     FWHM_G = abs(float(Pos_at_half_height_G[1]) -
                        float(Pos_at_half_height_G[0]))
449                 #print(FWHM_G)
450                 if int_G != 0:
451                     ID_IG = int_D/int_G
452                     I2D_IG = int_2D/int_G
453                 elif int_G == 0:
454                     ID_IG = "N/A"
455                     I2D_IG = "N/A"
456                     FWHM_G = 0
457                 #print("Full width at half maximum of G peak is", FWHM_G,"cm-1")
458                 #print("\n")
459                 #print("****The end of Analysis****")
460                 #print("\n")
461                 #print("****Conclusions****")
462                 #print("\n")
463                 #print("I(D)/I(G) = ",ID_IG)
464                 #print("I(2D)/I(G) = ",I2D_IG)
465                 #print("Pos(G) = ",pos_G,"       cm-1")
466                 #print("FWHM(G) = ",FWHM_G,"         cm-1")
467                 #print("Pos(2D) = ",pos_2D,"     cm-1")
468                 #print("FWHM(2D) = ",FWHM_2D,"  cm-1")
469                 #print("\n")
470                 Response = "y"
471                 #Conclusions
472                 if 5 < FWHM_G < 100 and 10 < FWHM_2D < 120 and 1570 < pos_G < 1600 and 2650
                    < pos_2D < 2725:
473                     #Response = input('Do you want to save the results? ("n" not to save)')
474                     filename_saved = "R" + filename_input + ".csv"
475                     if Response == "n":
476                         print("The result wasn't be saved")
477                     else:
478                         #filename = input('Enter filename of the results: ')
479                         with open(filename_saved, "w") as sp:
480                             print("I(D)/I(G),I(2D)/I(G),Pos(G),FWHM(G),Pos(2D),FWHM(2D)",
                                file=sp)
481
                                print(ID_IG,",",I2D_IG,",",pos_G,",",FWHM_G,",",pos_2D,",",FWHM_2
                                D, file=sp)
482                         Net[h][0] = Current_file_number - 1
483                         Net[h][1] = ID_IG
484                         Net[h][2] = I2D_IG
485                         Net[h][3] = pos_G
486                         Net[h][4] = FWHM_G
487                         Net[h][5] = pos_2D
488                         Net[h][6] = FWHM_2D
489                         h = h +1
490                     #print("already saved output as", filename_saved)
491             #else:
```

```
492                    #print("####bad input data####\n####It won't be saved####")
493                sp.close()
494                print("@",Current_file_number-1)
495                #print("\n")
496
                   #print("###################################################################
                   #################################")
497                #print("\n")
498
499        #printing all data in file
500        print("****The End of Analysis****")
501
           print("\n###################################################################
           ##########################\n")
502        #print("\n")
503        #Net_file_name = input('We would save all data in one file\n\nEnter output file
           name: ')
504        Net_file_name_with_csv = Net_file_name + ".csv"
505        with open(Net_file_name_with_csv, "w") as sp:
506            print("No,I(D)/I(G),I(2D)/I(G),Pos(G),FWHM(G),Pos(2D),FWHM(2D)", file=sp)
507            for i in range (0, len(Net)):
508                if Net[i][1] != 0 or Net[i][2] != 0 or Net[i][3] != 0 or Net[i][4] != 0 or
                   Net[i][5] != 0 or Net[i][6] != 0:
509
                       print(Net[i][0],",",Net[i][1],",",Net[i][2],",",Net[i][3],",",Net[i][4],"
                       ,",Net[i][5],",",Net[i][6], file=sp)
510                else:
511                    break
512        sp.close()
513        print("\nThe",Net_file_name_with_csv,"has been saved.\n")
514
           print("###################################################################
           ########################")
515        print("\nThank you, See you next time\n")
516
517
```

Python function for collecting data from files

```python
1   def AssemblyFile(NoFile, NoSpectra, File1, File2, File3, Net_file_name):
2       import pandas as pd
3       import numpy as np
4       import csv
5       from math import factorial
6       import random
7
8       print("\nAutomatic Assembling File ")
9       print("by Bhobnibhit C updated Mar 6nd 2018 version 1.02\n")
10
11      print("This program will assembly files (up to 10 files and 250 spectra per file)
        into single file.\n")
12      print("Instructions\n****Input files must be .csv with commas as dividers****\n ")
13
14      #Getting file name
15      #NoFile = int(input("How many files do you want to assembly?\n=====>"))
16      #NoSpectra = int(input("How many spectra do you want to record per file?\n=====>"))
17      FileName = []
18      FileNamewocsv = []
19      Net = []
20      for i in range(1000):
21          Net.append([])
22          for j in range(8):
23              Net[i].append(0)
24      for i in range(0,10):
25          FileName.append(0)
26          FileNamewocsv.append(0)
27      FileNamewocsv[0] = File1
28      FileNamewocsv[1] = File2
29      FileNamewocsv[2] = File3
30      for i in range(0, NoFile):
31          print("Please enter file name No.",str(i + 1),"without .csv")
32          #FileNamewocsv[i] = input("----->")
33          FileName[i] = str(FileNamewocsv[i]) + ".csv"
34          #Deleting First line of file
35          with open(FileName[i], 'r') as fin:
36              data = fin.read().splitlines(True)
37          FileName[i]
38          with open(FileNamewocsv[i] + "_DFL.csv", 'w') as fout:
39              fout.writelines(data[1:])
40      #print(FileName)
41
42
        print("\n###############################################################################
        ###########################\n")
43      print("*****We are transfering your data*****\n")
44      CountFile = 0
45      CountNoSpectra = 0
46      CountID_IG = 0
47      CountI2D_IG = 0
48      CountPosG = 0
49      CountFWHMG = 0
50      CountPos2D = 0
51      CountFWHM2D = 0
52
53      for i in range(0, NoFile):
54          #print(NoFile)
55          c_reader = csv.reader(open(FileNamewocsv[i] + "_DFL.csv", 'r'), delimiter=',')
56          Filename = np.asarray(list(zip(*c_reader))[0])
57          for k in range(0, len(Filename)):
58              Net[CountNoSpectra][1] =  Filename[k]
59              CountNoSpectra = CountNoSpectra + 1
60          for k in range(0, len(Filename)):
61              Net[CountFile][0] =  i + 1
62              CountFile = CountFile + 1
63          c_reader = csv.reader(open(FileNamewocsv[i] + "_DFL.csv", 'r'), delimiter=',')
64          Filename = np.asarray(list(zip(*c_reader))[1])
```

```
65              Filename = [float(i) for i in Filename]
66              for k in range(0, len(Filename)):
67                  Net[CountID_IG][2] =  Filename[k]
68                  CountID_IG = CountID_IG + 1
69              c_reader = csv.reader(open(FileNamewocsv[i] + "_DFL.csv", 'r'), delimiter=',')
70              Filename = np.asarray(list(zip(*c_reader))[2])
71              Filename = [float(i) for i in Filename]
72              for k in range(0, len(Filename)):
73                  Net[CountI2D_IG][3] =  Filename[k]
74                  CountI2D_IG = CountI2D_IG + 1
75              c_reader = csv.reader(open(FileNamewocsv[i] + "_DFL.csv", 'r'), delimiter=',')
76              Filename = np.asarray(list(zip(*c_reader))[3])
77              Filename = [float(i) for i in Filename]
78              for k in range(0, len(Filename)):
79                  Net[CountPosG][4] =  Filename[k]
80                  CountPosG = CountPosG + 1
81              c_reader = csv.reader(open(FileNamewocsv[i] + "_DFL.csv", 'r'), delimiter=',')
82              Filename = np.asarray(list(zip(*c_reader))[4])
83              Filename = [float(i) for i in Filename]
84              for k in range(0, len(Filename)):
85                  Net[CountFWHMG][5] =  Filename[k]
86                  CountFWHMG = CountFWHMG + 1
87              c_reader = csv.reader(open(FileNamewocsv[i] + "_DFL.csv", 'r'), delimiter=',')
88              Filename = np.asarray(list(zip(*c_reader))[5])
89              Filename = [float(i) for i in Filename]
90              for k in range(0, len(Filename)):
91                  Net[CountPos2D][6] =  Filename[k]
92                  CountPos2D = CountPos2D + 1
93              c_reader = csv.reader(open(FileNamewocsv[i] + "_DFL.csv", 'r'), delimiter=',')
94              Filename = np.asarray(list(zip(*c_reader))[6])
95              Filename = [float(i) for i in Filename]
96              for k in range(0, len(Filename)):
97                  Net[CountFWHM2D][7] =  Filename[k]
98                  CountFWHM2D = CountFWHM2D + 1
99
100         #random number
101         randoms = random.sample(range(CountNoSpectra), k = NoSpectra)
102         sorted_randoms = sorted(randoms)
103         print(sorted_randoms)
104
105         #print data on screen
106         for k in range (0, len(sorted_randoms)):
107             if Net[sorted_randoms[k]][1] != 0 or Net[sorted_randoms[k]][2] != 0 or
                Net[sorted_randoms[k]][3] != 0 or Net[sorted_randoms[k]][4] != 0 or
                Net[sorted_randoms[k]][5] != 0 or Net[sorted_randoms[k]][6] != 0 or
                Net[sorted_randoms[k]][7] != 0:
108
                    print(Net[sorted_randoms[k]][0],",",Net[sorted_randoms[k]][1],",",Net[sorted_
                    randoms[k]][2],",",Net[sorted_randoms[k]][3],",",Net[sorted_randoms[k]][4],",
                    ",Net[sorted_randoms[k]][5],",",Net[sorted_randoms[k]][6],",",Net[sorted_rand
                    oms[k]][7])
109             else:
110                 break
111     print("\n****The End of Transfer****")
112
        print("\n#####################################################################################
        ###########################\n")
113     #print("\n")
114     #print data as file
115     #Net_file_name = input('We would save all data in one file\n\nEnter output file
        name: ')
116     Net_file_name_with_csv = Net_file_name + ".csv"
117     with open(Net_file_name_with_csv, "w") as sp:
118         for k in range (0, len(sorted_randoms)):
119             if Net[sorted_randoms[k]][1] != 0 or Net[sorted_randoms[k]][2] != 0 or
                Net[sorted_randoms[k]][3] != 0 or Net[sorted_randoms[k]][4] != 0 or
                Net[sorted_randoms[k]][5] != 0 or Net[sorted_randoms[k]][6] != 0 or
```

```
                    Net[sorted_randoms[k]][7] != 0:
120
                        print(Net[sorted_randoms[k]][0],",",Net[sorted_randoms[k]][1],",",Net[sor
                        ted_randoms[k]][2],",",Net[sorted_randoms[k]][3],",",Net[sorted_randoms[k
                        ]][4],",",Net[sorted_randoms[k]][5],",",Net[sorted_randoms[k]][6],",",Net
                        [sorted_randoms[k]][7], file=sp)
121                 else:
122                     break
123         sp.close()
124
125
        print("\n##################################################################
        #########################\n")
126     print("The",Net_file_name_with_csv,"has been saved.\n")
127
        print("##################################################################
        #######################\n")
128     print("C'est fini. Merci,au revoir")
129     #print(Net)
130
131
```

Python function for creating histograms

```python
1    def HistogramMaker(Filename1pre):
2        import pandas as pd
3        import numpy as np
4        import csv
5        import scipy.stats
6        import statistics
7        from math import factorial
8        from time import gmtime, strftime
9        from datetime import datetime
10
11       print("\nHistogram maker")
12
         print("\n#################################################################
         ############################\n")
13       print("by Bhobnibhit C updated Mar 28rd 2018 version 1.01\n\n")
14
15       #Input Data
16       #Filename1pre = input("Please enter first filename: ")
17       Filename1 = Filename1pre + ".csv"
18
19       #Collect data file 1
20       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
21       NoSpectra_1 = np.asarray(list(zip(*c_reader))[0])
22       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
23       ID_IG_1 = np.asarray(list(zip(*c_reader))[2])
24       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
25       I2D_IG_1 = np.asarray(list(zip(*c_reader))[3])
26       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
27       PosG_1 = np.asarray(list(zip(*c_reader))[4])
28       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
29       FWHMG_1 = np.asarray(list(zip(*c_reader))[5])
30       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
31       Pos2D_1 = np.asarray(list(zip(*c_reader))[6])
32       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
33       FWHM2D_1 = np.asarray(list(zip(*c_reader))[7])
34       ID_IG_1 = [float(i) for i in ID_IG_1]
35       I2D_IG_1 = [float(i) for i in I2D_IG_1]
36       PosG_1 = [float(i) for i in PosG_1]
37       FWHMG_1 = [float(i) for i in FWHMG_1]
38       Pos2D_1 = [float(i) for i in Pos2D_1]
39       FWHM2D_1 = [float(i) for i in FWHM2D_1]
40
41
         print("\n#################################################################
         ############################\n")
42       print("\nDescriptives\n\nVariables Max Min Average")
43       print("ID/IG", np.amax(ID_IG_1), np.amin(ID_IG_1), np.mean(ID_IG_1))
44       print("I2D/IG", np.amax(I2D_IG_1), np.amin(I2D_IG_1), np.mean(I2D_IG_1))
45       print("PosG", np.amax(PosG_1), np.amin(PosG_1), np.mean(PosG_1))
46       print("FWHMG", np.amax(FWHMG_1), np.amin(FWHMG_1), np.mean(FWHMG_1))
47       print("Pos2D", np.amax(Pos2D_1), np.amin(Pos2D_1), np.mean(Pos2D_1))
48       print("FWHM2D", np.amax(FWHM2D_1), np.amin(FWHM2D_1), np.mean(FWHM2D_1))
49
50
         print("\n#################################################################
         ############################\n")
51       print("Please enter these informations\n")
52       print("ID/IG")
53       First_ID_IG = 0 #float(input("Start Bin: "))
54       Last_ID_IG = 2.4 #float(input("End Bin: "))
55       Bin_ID_IG = 0.15 #float(input("Bin Width: "))
56       Level_ID_IG = int((Last_ID_IG - First_ID_IG)/Bin_ID_IG)
57       print("I2D/IG")
58       First_I2D_IG = 0 #float(input("Start Bin: "))
59       Last_I2D_IG = 4 #float(input("End Bin: "))
60       Bin_I2D_IG = 0.2 #float(input("Bin Width: "))
61       Level_I2D_IG = int((Last_I2D_IG - First_I2D_IG)/Bin_I2D_IG)
```

```python
62          print("PosG")
63          First_PosG = 1570 #float(input("Start Bin: "))
64          Last_PosG = 1600 #float(input("End Bin: "))
65          Bin_PosG = 5 #float(input("Bin Width: "))
66          Level_PosG = int((Last_PosG - First_PosG)/Bin_PosG)
67          print("FWHMG")
68          First_FWHMG = 0 #float(input("Start Bin: "))
69          Last_FWHMG = 100 #float(input("End Bin: "))
70          Bin_FWHMG = 5 #float(input("Bin Width: "))
71          Level_FWHMG = int((Last_FWHMG - First_FWHMG)/Bin_FWHMG)
72          print("Pos2D")
73          First_Pos2D = 2650 #float(input("Start Bin: "))
74          Last_Pos2D = 2720 #float(input("End Bin: "))
75          Bin_Pos2D= 10 #float(input("Bin Width: "))
76          Level_Pos2D = int((Last_Pos2D - First_Pos2D)/Bin_Pos2D)
77          print("FWHM2D")
78          First_FWHM2D = 20 #float(input("Start Bin: "))
79          Last_FWHM2D = 120 #float(input("End Bin: "))
80          Bin_FWHM2D = 10 #float(input("Bin Width: "))
81          Level_FWHM2D = int((Last_FWHM2D - First_FWHM2D)/Bin_FWHM2D)
82
83          row = [Level_ID_IG, Level_I2D_IG, Level_PosG, Level_FWHMG, Level_Pos2D, Level_FWHM2D]
84          row_sum = int(np.amax(row))
85
86          sum = []
87          for i in range(row_sum):
88              sum.append([])
89              for j in range(12):
90                  sum[i].append(0)
91          #print(sum)
92
93          #ID_IG
94          Edge_ID_IG = []
95          for i in range(0,Level_ID_IG +1):
96              Edge_ID_IG.append(0)
97          for i in range(0, len(Edge_ID_IG)):
98              if i == 0:
99                  Edge_ID_IG[i] = First_ID_IG
100             elif i == len(Edge_ID_IG):
101                 Edge_ID_IG[i] = Last_ID_IG
102             else:
103                 Edge_ID_IG[i] = Edge_ID_IG[i-1] + Bin_ID_IG
104         j = 0
105         for i in range(0, len(Edge_ID_IG)-1):
106             sum[j][0] = (Edge_ID_IG[i] + Edge_ID_IG[i+1])/2
107             j = j +1
108         for i in range(0, len(ID_IG_1)):
109             for j in range(1, len(Edge_ID_IG)):
110                 if Edge_ID_IG[j-1] < ID_IG_1[i] < Edge_ID_IG[j]:
111                     sum[j-1][1] = sum[j-1][1] +1
112         #max_ID_IG = []
113         #for i in range(0, row_sum):
114             #max_ID_IG.append(0)
115         #for i in range(0, row_sum):
116             #max_ID_IG[i] = sum[i][1]
117         #Vmax_ID_IG = np.amax(max_ID_IG)
118         #for i in range(0, row_sum):
119             #sum[i][1] = (sum[i][1])*100/Vmax_ID_IG
120
121         #for j in range(1, row_sum):
122             #print(sum[j-1][0],sum[j-1][1])
123
124         #I2D_IG
125         Edge_I2D_IG = []
126         for i in range(0,Level_I2D_IG +1):
127             Edge_I2D_IG.append(0)
128         for i in range(0, len(Edge_I2D_IG)):
```

```
129            if i == 0:
130                Edge_I2D_IG[i] = First_I2D_IG
131            elif i == len(Edge_I2D_IG):
132                Edge_I2D_IG[i] = Last_I2D_IG
133            else:
134                Edge_I2D_IG[i] = Edge_I2D_IG[i-1] + Bin_I2D_IG
135        j = 0
136        for i in range(0, len(Edge_I2D_IG)-1):
137            sum[j][2] = (Edge_I2D_IG[i] + Edge_I2D_IG[i+1])/2
138            j = j +1
139        for i in range(0, len(I2D_IG_1)):
140            for j in range(1, len(Edge_I2D_IG)):
141                if Edge_I2D_IG[j-1] < I2D_IG_1[i] < Edge_I2D_IG[j]:
142                    sum[j-1][3] = sum[j-1][3] +1
143
144        #for j in range(1, row_sum):
145            #print(sum[j-1][2],sum[j-1][3])
146
147        #PosG
148        Edge_PosG = []
149        for i in range(0,Level_PosG +1):
150            Edge_PosG.append(0)
151        for i in range(0, len(Edge_PosG)):
152            if i == 0:
153                Edge_PosG[i] = First_PosG
154            elif i == len(Edge_PosG):
155                Edge_PosG[i] = Last_PosG
156            else:
157                Edge_PosG[i] = Edge_PosG[i-1] + Bin_PosG
158        j = 0
159        for i in range(0, len(Edge_PosG)-1):
160            sum[j][4] = (Edge_PosG[i] + Edge_PosG[i+1])/2
161            j = j +1
162        for i in range(0, len(PosG_1)):
163            for j in range(1, len(Edge_PosG)):
164                if Edge_PosG[j-1] < PosG_1[i] < Edge_PosG[j]:
165                    sum[j-1][5] = sum[j-1][5] +1
166
167        #for j in range(1, row_sum):
168            #print(sum[j-1][4],sum[j-1][5])
169
170        #FWHMG
171        Edge_FWHMG = []
172        for i in range(0,Level_FWHMG +1):
173            Edge_FWHMG.append(0)
174        for i in range(0, len(Edge_FWHMG)):
175            if i == 0:
176                Edge_FWHMG[i] = First_FWHMG
177            elif i == len(Edge_FWHMG):
178                Edge_FWHMG[i] = Last_FWHMG
179            else:
180                Edge_FWHMG[i] = Edge_FWHMG[i-1] + Bin_FWHMG
181        j = 0
182        for i in range(0, len(Edge_FWHMG)-1):
183            sum[j][6] = (Edge_FWHMG[i] + Edge_FWHMG[i+1])/2
184            j = j +1
185        for i in range(0, len(FWHMG_1)):
186            for j in range(1, len(Edge_FWHMG)):
187                if Edge_FWHMG[j-1] < FWHMG_1[i] < Edge_FWHMG[j]:
188                    sum[j-1][7] = sum[j-1][7] +1
189
190        #for j in range(1, row_sum):
191            #print(sum[j-1][6],sum[j-1][7])
192
193        #Pos2D
194        Edge_Pos2D = []
195        for i in range(0,Level_Pos2D +1):
```

```
196              Edge_Pos2D.append(0)
197         for i in range(0, len(Edge_Pos2D)):
198             if i == 0:
199                 Edge_Pos2D[i] = First_Pos2D
200             elif i == len(Edge_Pos2D):
201                 Edge_Pos2D[i] = Last_Pos2D
202             else:
203                 Edge_Pos2D[i] = Edge_Pos2D[i-1] + Bin_Pos2D
204         j = 0
205         for i in range(0, len(Edge_Pos2D)-1):
206             sum[j][8] = (Edge_Pos2D[i] + Edge_Pos2D[i+1])/2
207             j = j +1
208         for i in range(0, len(Pos2D_1)):
209             for j in range(1, len(Edge_Pos2D)):
210                 if Edge_Pos2D[j-1] < Pos2D_1[i] < Edge_Pos2D[j]:
211                     sum[j-1][9] = sum[j-1][9] +1
212
213         #for j in range(1, row_sum):
214             #print(sum[j-1][8],sum[j-1][9])
215
216         #FWHM2D
217         Edge_FWHM2D = []
218         for i in range(0,Level_FWHM2D +1):
219             Edge_FWHM2D.append(0)
220         for i in range(0, len(Edge_FWHM2D)):
221             if i == 0:
222                 Edge_FWHM2D[i] = First_FWHM2D
223             elif i == len(Edge_FWHM2D):
224                 Edge_FWHM2D[i] = Last_FWHM2D
225             else:
226                 Edge_FWHM2D[i] = Edge_FWHM2D[i-1] + Bin_FWHM2D
227         j = 0
228         for i in range(0, len(Edge_FWHM2D)-1):
229             sum[j][10] = (Edge_FWHM2D[i] + Edge_FWHM2D[i+1])/2
230             j = j +1
231         for i in range(0, len(FWHM2D_1)):
232             for j in range(1, len(Edge_FWHM2D)):
233                 if Edge_FWHM2D[j-1] < FWHM2D_1[i] < Edge_FWHM2D[j]:
234                     sum[j-1][11] = sum[j-1][11] +1
235
236         #for j in range(1, row_sum):
237             #print(sum[j-1][10],sum[j-1][11])
238
239         #print(sum)
240
         print("\n##################################################################
         ###########################\n")
241         for i in range (0, len(sum)):
242
             print(sum[i][0],",",sum[i][1],",",sum[i][2],",",sum[i][3],",",sum[i][4],",",sum[i
             ][5],",",sum[i][6],",",sum[i][7],",",sum[i][8],",",sum[i][9],",",sum[i][10],",",s
             um[i][11])
243
         print("\n##################################################################
         ###########################\n")
244     Net_file_name_with_csv = "Histograms_" + Filename1pre + ".csv"
245     with open(Net_file_name_with_csv, "w") as sp:
246         #print("I(D)/I(G),I(2D)/I(G),Pos(G),FWHM(G),Pos(2D),FWHM(2D)", file=sp)
247         for i in range (0, len(sum)):
248
             print(sum[i][0],",",sum[i][1],",",sum[i][2],",",sum[i][3],",",sum[i][4],",",s
             um[i][5],",",sum[i][6],",",sum[i][7],",",sum[i][8],",",sum[i][9],",",sum[i][1
             0],",",sum[i][11], file=sp)
249     sp.close()
250     print(Net_file_name_with_csv," has been saved. I hope you enjoy this world.")
251     #K = input()
252     #if K != "bye":
```

```
253        print("Au revoir")
```

Python function for statistical analysis of data

```python
1    def StatAnalysis(Filename1pre):
2        import pandas as pd
3        import numpy as np
4        import csv
5        import scipy.stats
6        import statistics
7        from math import factorial
8        from time import gmtime, strftime
9        from datetime import datetime
10
11       print("\nSTATISTICAL ANALYSIS")
12
         print("\n####################################################################
         ###########################\n")
13       print("by Bhobnibhit C updated Mar 28rd 2018 version 1.01\n")
14
15       print("This program will calculate statistical parameters for you.\n")
16       print("Instructions\n****Input files must be .csv with commas as dividers****")
17
         #print("\n###################################################################
         ##########################\n")
18       #print("List of Analysis Method\n1) Descriptives including means, median, standard
         deviation, skewness, sample size and standard error of mean\n2) Levene's test of
         variance equality\n3) One-way ANOVA\n4) t-test (both equal and unequal variance)\n")
19
         print("###################################################################
         ##########################\n")
20       #Input Data
21       #Filename1pre = input("Please enter first filename: ")
22       #Filename2pre = input("Please enter second filename: ")
23       Filename1 = Filename1pre + ".csv"
24       #Filename2 = Filename2pre + ".csv"
25
26       #Collect data file 1
27       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
28       NoSpectra_1 = np.asarray(list(zip(*c_reader))[0])
29       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
30       ID_IG_1 = np.asarray(list(zip(*c_reader))[2])
31       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
32       I2D_IG_1 = np.asarray(list(zip(*c_reader))[3])
33       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
34       PosG_1 = np.asarray(list(zip(*c_reader))[4])
35       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
36       FWHMG_1 = np.asarray(list(zip(*c_reader))[5])
37       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
38       Pos2D_1 = np.asarray(list(zip(*c_reader))[6])
39       c_reader = csv.reader(open(Filename1, 'r'), delimiter=',')
40       FWHM2D_1 = np.asarray(list(zip(*c_reader))[7])
41       ID_IG_1 = [float(i) for i in ID_IG_1]
42       I2D_IG_1 = [float(i) for i in I2D_IG_1]
43       PosG_1 = [float(i) for i in PosG_1]
44       FWHMG_1 = [float(i) for i in FWHMG_1]
45       Pos2D_1 = [float(i) for i in Pos2D_1]
46       FWHM2D_1 = [float(i) for i in FWHM2D_1]
47
48       #print(ID_IG_1)
49       #print(I2D_IG_1)
50       #print(PosG_1)
51       #print(FWHMG_1)
52       #print(Pos2D_1)
53       #print(FWHM2D_1)
54
55       #Collect data file 2
56       #c_reader = csv.reader(open(Filename2, 'r'), delimiter=',')
57       #NoSpectra_2 = np.asarray(list(zip(*c_reader))[0])
58       #c_reader = csv.reader(open(Filename2, 'r'), delimiter=',')
59       #ID_IG_2 = np.asarray(list(zip(*c_reader))[2])
```

```python
60          #c_reader = csv.reader(open(Filename2, 'r'), delimiter=',')
61          #I2D_IG_2 = np.asarray(list(zip(*c_reader))[3])
62          #c_reader = csv.reader(open(Filename2, 'r'), delimiter=',')
63          #PosG_2 = np.asarray(list(zip(*c_reader))[4])
64          #c_reader = csv.reader(open(Filename2, 'r'), delimiter=',')
65          #FWHMG_2 = np.asarray(list(zip(*c_reader))[5])
66          #c_reader = csv.reader(open(Filename2, 'r'), delimiter=',')
67          #Pos2D_2 = np.asarray(list(zip(*c_reader))[6])
68          #c_reader = csv.reader(open(Filename2, 'r'), delimiter=',')
69          #FWHM2D_2 = np.asarray(list(zip(*c_reader))[7])
70          #ID_IG_2 = [float(i) for i in ID_IG_2]
71          #I2D_IG_2 = [float(i) for i in I2D_IG_2]
72          #PosG_2 = [float(i) for i in PosG_2]
73          #FWHMG_2 = [float(i) for i in FWHMG_2]
74          #Pos2D_2 = [float(i) for i in Pos2D_2]
75          #FWHM2D_2 = [float(i) for i in FWHM2D_2]
76
77          #print(ID_IG_2)
78          #print(I2D_IG_2)
79          #print(PosG_2)
80          #print(FWHMG_2)
81          #print(Pos2D_2)
82          #print(FWHM2D_2)
83
84          #Descriptives
85          print("\nStatiscal Analysis\n")
86          print("File 1:", Filename1)
87
            print("\n#######################################################################
            ###########################")
88          print("\nDescrptives\n")
89          print("Mean")
90          print("Group
            1,",np.mean(ID_IG_1),np.mean(I2D_IG_1),np.mean(PosG_1),np.mean(FWHMG_1),np.mean(Pos2D
            _1),np.mean(FWHM2D_1))
91          #print("Group
            2,",np.mean(ID_IG_2),np.mean(I2D_IG_2),np.mean(PosG_2),np.mean(FWHMG_2),np.mean(Pos2D
            _2),np.mean(FWHM2D_2))
92          print("Median")
93          print("Group
            1,",np.median(ID_IG_1),np.median(I2D_IG_1),np.median(PosG_1),np.median(FWHMG_1),np.me
            dian(Pos2D_1),np.median(FWHM2D_1))
94          #print("Group
            2,",np.median(ID_IG_2),np.median(I2D_IG_2),np.median(PosG_2),np.median(FWHMG_2),np.me
            dian(Pos2D_2),np.median(FWHM2D_2))
95          print("Mode")
96          print("Group
            1,","\n",scipy.stats.mode(ID_IG_1),"\n",scipy.stats.mode(I2D_IG_1),"\n",scipy.stats.m
            ode(PosG_1),"\n",scipy.stats.mode(FWHMG_1),"\n",scipy.stats.mode(Pos2D_1),"\n",scipy.
            stats.mode(FWHM2D_1))
97          #print("Group
            2,","\n",scipy.stats.mode(ID_IG_2),"\n",scipy.stats.mode(I2D_IG_2),"\n",scipy.stats.m
            ode(PosG_2),"\n",scipy.stats.mode(FWHMG_2),"\n",scipy.stats.mode(Pos2D_2),"\n",scipy.
            stats.mode(FWHM2D_2))
98          print("Standard Deviation")
99          print("Group 1,",np.std(ID_IG_1,ddof = 1),np.std(I2D_IG_1,ddof =
            1),np.std(PosG_1,ddof = 1),np.std(FWHMG_1,ddof = 1),np.std(Pos2D_1,ddof =
            1),np.std(FWHM2D_1,ddof = 1))
100         #print("Group 2,",np.std(ID_IG_2,ddof = 1),np.std(I2D_IG_2,ddof =
            1),np.std(PosG_2,ddof = 1),np.std(FWHMG_2,ddof = 1),np.std(Pos2D_2,ddof =
            1),np.std(FWHM2D_2,ddof = 1))
101         print("Sample Size")
102         print("Group
            1,",len(ID_IG_1),len(I2D_IG_1),len(PosG_1),len(FWHMG_1),len(Pos2D_1),len(FWHM2D_1))
103         #print("Group
            2,",len(ID_IG_2),len(I2D_IG_2),len(PosG_2),len(FWHMG_2),len(Pos2D_2),len(FWHM2D_2))
104         print("Standard Error")
```

```python
105     print("Group 1,",scipy.stats.sem(ID_IG_1,ddof = 1),scipy.stats.sem(I2D_IG_1,ddof =
        1),scipy.stats.sem(PosG_1,ddof = 1),scipy.stats.sem(FWHMG_1,ddof =
        1),scipy.stats.sem(Pos2D_1,ddof = 1),scipy.stats.sem(FWHM2D_1,ddof = 1))
106     #print("Group 2,",scipy.stats.sem(ID_IG_2,ddof = 1),scipy.stats.sem(I2D_IG_2,ddof =
        1),scipy.stats.sem(PosG_2,ddof = 1),scipy.stats.sem(FWHMG_2,ddof =
        1),scipy.stats.sem(Pos2D_2,ddof = 1),scipy.stats.sem(FWHM2D_2,ddof = 1))
107     print("Skewness")
108     print("Group
        1,",scipy.stats.skew(ID_IG_1),scipy.stats.skew(I2D_IG_1),scipy.stats.skew(PosG_1),sci
        py.stats.skew(FWHMG_1),scipy.stats.skew(Pos2D_1),scipy.stats.skew(FWHM2D_1))
109     #print("Group
        2,",scipy.stats.skew(ID_IG_2),scipy.stats.skew(I2D_IG_2),scipy.stats.skew(PosG_2),sci
        py.stats.skew(FWHMG_2),scipy.stats.skew(Pos2D_2),scipy.stats.skew(FWHM2D_2))
110
111     Net_file_name = "StatisticalReport_ " + Filename1pre + ".txt"
112     with open(Net_file_name, "w") as sp:
113         print("Statiscal Analysis Report\n", file = sp)
114         print("File 1:", Filename1, file = sp)
115
            #print("\n############################################################################
            ##############################", file = sp)
116
            print("\n#############################################################################
            ##############################", file = sp)
117         print("\nDescrptives\n", file = sp)
118         print("Mean", file = sp)
119         print("Group
            1,",np.mean(ID_IG_1),np.mean(I2D_IG_1),np.mean(PosG_1),np.mean(FWHMG_1),np.mean(P
            os2D_1),np.mean(FWHM2D_1), file = sp)
120         #print("Group
            2,",np.mean(ID_IG_2),np.mean(I2D_IG_2),np.mean(PosG_2),np.mean(FWHMG_2),np.mean(P
            os2D_2),np.mean(FWHM2D_2), file = sp)
121         print("Median", file = sp)
122         print("Group
            1,",np.median(ID_IG_1),np.median(I2D_IG_1),np.median(PosG_1),np.median(FWHMG_1),n
            p.median(Pos2D_1),np.median(FWHM2D_1), file = sp)
123         #print("Group
            2,",np.median(ID_IG_2),np.median(I2D_IG_2),np.median(PosG_2),np.median(FWHMG_2),n
            p.median(Pos2D_2),np.median(FWHM2D_2), file = sp)
124         print("Mode", file = sp)
125         print("Group
            1,","\n",scipy.stats.mode(ID_IG_1),"\n",scipy.stats.mode(I2D_IG_1),"\n",scipy.sta
            ts.mode(PosG_1),"\n",scipy.stats.mode(FWHMG_1),"\n",scipy.stats.mode(Pos2D_1),"\n
            ",scipy.stats.mode(FWHM2D_1), file = sp)
126         #print("Group
            2,","\n",scipy.stats.mode(ID_IG_2),"\n",scipy.stats.mode(I2D_IG_2),"\n",scipy.sta
            ts.mode(PosG_2),"\n",scipy.stats.mode(FWHMG_2),"\n",scipy.stats.mode(Pos2D_2),"\n
            ",scipy.stats.mode(FWHM2D_2), file = sp)
127         print("Standard Deviation", file = sp)
128         print("Group 1,",np.std(ID_IG_1,ddof = 1),np.std(I2D_IG_1,ddof =
            1),np.std(PosG_1,ddof = 1),np.std(FWHMG_1,ddof = 1),np.std(Pos2D_1,ddof =
            1),np.std(FWHM2D_1,ddof = 1), file = sp)
129         #print("Group 2,",np.std(ID_IG_2,ddof = 1),np.std(I2D_IG_2,ddof =
            1),np.std(PosG_2,ddof = 1),np.std(FWHMG_2,ddof = 1),np.std(Pos2D_2,ddof =
            1),np.std(FWHM2D_2,ddof = 1), file = sp)
130         print("Sample Size", file = sp)
131         print("Group
            1,",len(ID_IG_1),len(I2D_IG_1),len(PosG_1),len(FWHMG_1),len(Pos2D_1),len(FWHM2D_1
            ), file = sp)
132         #print("Group
            2,",len(ID_IG_2),len(I2D_IG_2),len(PosG_2),len(FWHMG_2),len(Pos2D_2),len(FWHM2D_2
            ), file = sp)
133         print("Standard Error", file = sp)
134         print("Group 1,",scipy.stats.sem(ID_IG_1,ddof =
            1),scipy.stats.sem(I2D_IG_1,ddof = 1),scipy.stats.sem(PosG_1,ddof =
            1),scipy.stats.sem(FWHMG_1,ddof = 1),scipy.stats.sem(Pos2D_1,ddof =
            1),scipy.stats.sem(FWHM2D_1,ddof = 1), file = sp)
```

```python
135         #print("Group 2,",scipy.stats.sem(ID_IG_2,ddof =
            1),scipy.stats.sem(I2D_IG_2,ddof = 1),scipy.stats.sem(PosG_2,ddof =
            1),scipy.stats.sem(FWHMG_2,ddof = 1),scipy.stats.sem(Pos2D_2,ddof =
            1),scipy.stats.sem(FWHM2D_2,ddof = 1), file = sp)
136         print("Skewness", file = sp)
137         print("Group
            1,",scipy.stats.skew(ID_IG_1),scipy.stats.skew(I2D_IG_1),scipy.stats.skew(PosG_1)
            ,scipy.stats.skew(FWHMG_1),scipy.stats.skew(Pos2D_1),scipy.stats.skew(FWHM2D_1),
            file = sp)
138         #print("Group
            2,",scipy.stats.skew(ID_IG_2),scipy.stats.skew(I2D_IG_2),scipy.stats.skew(PosG_2)
            ,scipy.stats.skew(FWHMG_2),scipy.stats.skew(Pos2D_2),scipy.stats.skew(FWHM2D_2),
            file = sp)
139         print("\n#################################################For
            Excel#####################################################\n", file = sp)
140         print(np.mean(ID_IG_1),",",np.std(ID_IG_1,ddof =
            1),",",len(ID_IG_1),",",np.mean(I2D_IG_1),",",np.std(I2D_IG_1,ddof =
            1),",",len(I2D_IG_1),",",np.mean(PosG_1),",",np.std(PosG_1,ddof =
            1),",",len(PosG_1),",",np.mean(FWHMG_1),",",np.std(FWHMG_1,ddof =
            1),",",len(FWHMG_1),",",np.mean(Pos2D_1),",",np.std(Pos2D_1,ddof =
            1),",",len(Pos2D_1),",",np.mean(FWHM2D_1),",",np.std(FWHM2D_1,ddof =
            1),",",len(FWHM2D_1), file = sp)
141
            print("\n##############################################################
            ##############################", file = sp)
142         print("\nRaw data\n",file=sp)
143         print("File,I(D)/I(G),I(2D)/I(G),Pos(G),FWHM(G),Pos(2D),FWHM(2D)",file=sp)
144         for i in range(0, len(NoSpectra_1)):
145
                print("1,",ID_IG_1[i],",",I2D_IG_1[i],",",PosG_1[i],",",FWHMG_1[i],",",Pos2D_
                1[i],",",FWHM2D_1[i],file = sp)
146         #for i in range(0, len(NoSpectra_2)):
147
                #print("2,",ID_IG_2[i],",",I2D_IG_2[i],",",PosG_2[i],",",FWHMG_2[i],",",Pos2D
                _2[i],",",FWHM2D_2[i],file = sp)
148
            print("\n\n##############################################################
            ################################\n\nThis file was recorded
            on",str(datetime.now()), file = sp)
149
            print("\n##############################################################
            ###############################\n\nEnd of File", file=sp)
150     sp.close()
151
        print("\n##############################################################
        ############################\n\nEnd of Analysis\nYour results has been saved as",
        Net_file_name)
152
        print("\n##############################################################
        ###########################\n\nC'est fini. Merci, au revoir")
```

Python function for averaging Raman spectra

```python
def AverageGraph(Prefix, first_file_number, Last_file_number):
    import pandas as pd
    import numpy as np
    import csv
    from math import factorial

    print("\nAutomatic Graph Averager")
    print("by Bhobnibhit C updated Mar 2nd 2018 version 1.02\n")

    print("This program will average your graphs for representing a set of data.\n")
    print("Instructions\n****Input files must be .csv with commas as dividers****\n ")
    print("****Normalization will be done before average****\n")

    #create array for keeping all data
    R_RamanShift = []
    for i in range(10000):
        R_RamanShift.append(0)
    R_Intensity = []
    Sum_Intensity = []
    for i in range(10000):
        R_Intensity.append(0)
        Sum_Intensity.append(0)
    NoData = 0


    #Prefix = input('Enter input file prefix: ')
    #first_file_number = Current_file_number = int(input('First file number: '))
    #Last_file_number = int(input('Last file number: '))
    Current_file_number = first_file_number
    filename1 = 'love'

    print("\n##############################################################################
###########################\n")
    print("****Analysis in Progress****\n")
    #print("\n")
    #print(Prefix)
    #print(first_file_number)
    #print(Last_file_number)
    #print(filename1)

    filename1 = Prefix + "000" + str(Current_file_number) + ".csv"
    c_reader = csv.reader(open(filename1, 'r'), delimiter=',')
    R_RamanShiftpre = list(zip(*c_reader))[0]
    for i in range(0, len(R_RamanShiftpre)):
        R_RamanShift[i] = float(R_RamanShiftpre[i])

    #print(R_RamanShift)
    RamanShift_Full = []
    Intensity_Full = []
    for i in range(0,100000):
        RamanShift_Full.append(0)
        Intensity_Full.append(0)

    #making file name
    while first_file_number-1 < Current_file_number < Last_file_number +1:
        if 0 < Current_file_number < 10:
            filename1 = Prefix + "000" + str(Current_file_number) + ".csv"
            filename_input = Prefix + "000" + str(Current_file_number)
            #print(filename1)
            Current_file_number = Current_file_number + 1
        elif 9 < Current_file_number < 100:
            filename1 = Prefix + "00" + str(Current_file_number) + ".csv"
            filename_input = Prefix + "000" + str(Current_file_number)
            #print(filename1)
            Current_file_number = Current_file_number + 1
        elif 99 < Current_file_number < 1000:
            filename1 = Prefix + "0" + str(Current_file_number) + ".csv"
```

```
66                    filename_input = Prefix + "000" + str(Current_file_number)
67                    #print(filename1)
68                    Current_file_number = Current_file_number + 1
69            c_reader = csv.reader(open(filename1, 'r'), delimiter=',')
70            RamanShift_Fullpre = list(zip(*c_reader))[0]
71            c_reader = csv.reader(open(filename1, 'r'), delimiter=',')
72            Intensity_Fullpre = list(zip(*c_reader))[1]
73
74            #cut range into 1000 to 3400 and finding minimum
75            for i in range (0, len(RamanShift_Fullpre)):
76                RamanShift_Full[i] = float(RamanShift_Fullpre[i])
77                Intensity_Full[i] = float(Intensity_Fullpre[i])
78            #print(RamanShift_Full)
79            #print(Intensity_Full)
80            RamanShift_1000to3400 = []
81            Intensity_1000to3400 = []
82            for k in range(0,100000):
83                RamanShift_1000to3400.append(0)
84                Intensity_1000to3400.append(0)
85            k = 0
86            for i in range (0, len(RamanShift_Full)):
87                while 1000 < RamanShift_Full[i] < 3400 :
88                    #print(RamanShift_Full[i],",",Intensity_Full[i])
89                    RamanShift_1000to3400[k] = float(RamanShift_Full[i])
90                    Intensity_1000to3400[k] = float(Intensity_Full[i])
91                    k = k + 1
92                    break
93            BaselineCorrector = np.amin(Intensity_1000to3400)
94
95            #finding multiplier
96            RamanShift_1450to1700 = []
97            Intensity_1450to1700 = []
98            for k in range(0,100000):
99                RamanShift_1450to1700.append(0)
100               Intensity_1450to1700.append(0)
101           k = 0
102           for i in range (0, len(RamanShift_Full)):
103               while 1450 < RamanShift_Full[i] < 1700 :
104                   RamanShift_1450to1700[k] = float(RamanShift_Full[i])
105                   Intensity_1450to1700[k] = float(Intensity_Full[i])
106                   k = k + 1
107                   break
108           SignalModulator = np.amax(Intensity_1450to1700)
109           #print(R_RamanShift)
110           #print(Intensity_Full)
111           #print(RamanShift_Full)
112           for i in range(0, len(R_RamanShift)):
113               #print("Check")
114               #print(RamanShift_Full[i],R_RamanShift[i])
115               if RamanShift_Full[i] == R_RamanShift[i]:
116                   #print("Correct")
117                   Sum_Intensity[i] =  float(Sum_Intensity[i]) + (Intensity_Full[i] -
                      BaselineCorrector)/SignalModulator
118           print("@",str(Current_file_number-1))
119           NoData = NoData + 1
120       #print(NoData)
121       #print(Sum_Intensity)
122       for i in range(0, len(Sum_Intensity)):
123           R_Intensity[i] = Sum_Intensity[i]/NoData
124
125       filename_reserved =   "Average_"+ Prefix + "from" + str(first_file_number) + "to" +
          str(Current_file_number - 1) +".csv"
126       with open(filename_reserved, "w") as sp:
127           for i in range (0, len(R_RamanShift)):
128               if R_RamanShift != 0 and R_Intensity[i] != 0:
129                   print(R_RamanShift[i],",",R_Intensity[i], file=sp)
130                   #print(R_RamanShift[i],",",R_Intensity[i])
```

```
131        sp.close()
132
133
           print("\n#################################################################
           #########################\n")
134        print("Your result has been recorded as ",filename_reserved)
135        print("****C'est fini!!! Merci, Au revoir****")
136
```

VITA

Name: Mr. Bhobnibhit Chatmaneerungcharoen

Date of Birth: January 28, 1992   Birth place: Bangkok, Thailand

Contact Address: 50/949 Moo 6 Baan Mai, Parkkred, Nonthaburi 11120,

THAILAND

Education:

2014 B.S. (Chemistry) with First class honor, Kasetsart University

2018 M.Sc. (Chemistry), Chulalongkorn University

Conference:

February 2-3, 2017   Pure and Applied Chemistry International

Conference 2017

Proceedings:

Chatmaneerungcharoen, B.; Unarunotai, S., Effect of cooling rate on quality

of graphene films grown by atmospheric pressure chemical vapor

deposition with acetylene on copper foils, Proceedings of Pure and Applied

Chemistry International Conference 2017 (PACCON 2017), Bangkok,

Thailand.