การประยุกต์เทคนิคการวิเคราะห์สัญญาณ เพื่อตรวจสอบการน็อคของเครื่องยนต์สันดาปภายใน รุ่น 1200 Coupe จากสัญญาณการสั่นสะเทือน

Application of Signal Analysis for Checking Knock Combustion of 1200 Coupe Engine by Vibration Signal

ไพจิตร กชกรจารุพงศ์

Paijit Kochakornjarupong

วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

มหาวิทยาลัยสงขลานครินทร์

Master of Engineering Thesis in Electrical Engineering

Prince of Songkla University

2541

ชื่อวิทยานิพนธ์	การประยุกต์เทคนิคการวิเคราะห์สัญญาณ เพื่อตรวจสอบการน็อคของเครื่องยนต์สันดาปภายใน รุ่น 1200 Coupe จากสัญญาณการสั่นสะเทือน
ผู้เขียน	นายไพจิตร กชกรจารุพงศ์
สาขาวิชา	วิศวกรรมไฟฟ้า

คณะกรรมการที่ปรึกษา

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.กิตติพัฒน์ ตันตระรุ่งโรจน์)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ชัชวาลย์ ยนต์หงส์)

.....กรรมการ
(อาจารย์ปัญญรักษ์ งามศรีตระกูล)

คณะกรรมการสอบ

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.กิตติพัฒน์ ตันตระรุ่งโรจน์)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ชัชวาลย์ ยนต์หงส์)

.....กรรมการ
(อาจารย์ปัญญรักษ์ งามศรีตระกูล)

.....กรรมการ
(อาจารย์วีระพันธุ์ มุสิกสาร)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์แสวง กะระณา)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์อนุมัติให้นับวิทยานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

.....
(รองศาสตราจารย์ ดร.ก้าน จันทร์พรหมมา)
คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์	การประยุกต์เทคนิคการวิเคราะห์สัญญาณ เพื่อตรวจสอบการน็อคของเครื่องยนต์สันดาปภายใน รุ่น 1200 Coupe จากสัญญาณการสั่นสะเทือน
ผู้เขียน	นายไพจิตร กชกรจารุพงศ์
สาขาวิชา	วิศวกรรมไฟฟ้า
ปีการศึกษา	2541

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้กล่าวถึง การประยุกต์ใช้เทคนิคการวิเคราะห์สัญญาณในทางเวลาเต็มหน่วยเพื่อตรวจสอบการน็อคของเครื่องยนต์ TOYOTA Corolla 1200 โดยบันทึกสัญญาณการสั่นสะเทือนจากตัวตรวจรู้ที่วัดการสั่นสะเทือนของเครื่องยนต์ด้วยความเร็วรอบ 1800, 2000, 2200, 2400 และ 2600 รอบ/นาที ในกรณีเครื่องยนต์ปกติจะตั้งจังหวะการจุดระเบิดของเครื่องยนต์ไว้ก่อนหน้า TDC 20 องศา และก่อนจะวัดสัญญาณเครื่องยนต์ที่เกิดการน็อคจะตั้งจังหวะการจุดระเบิดไว้ก่อนหน้า TDC 30 องศา จากนั้นนำข้อมูลไปแปลงจากโดเมนเวลาเป็นโดเมนความถี่ด้วยตัวแปลงฟูริเยอร์แบบเร็ว และตัวแปลงเวฟเลตเต็มหน่วย โดยใช้โปรแกรมคอมพิวเตอร์ที่เขียนขึ้น ผลที่ได้จากการแปลงมีช่วงความถี่ที่มีค่าแอมพลิจูดที่แตกต่างกันมากของสัญญาณปกติ และสัญญาณที่มีการน็อค ที่ความถี่ 4.3 กิโลเฮิรตซ์ ถึง 6.3 กิโลเฮิรตซ์ เมื่อนำค่าแอมพลิจูดที่ได้จากตัวแปลงเวฟเลตเต็มหน่วยในช่วงความถี่ที่พบว่าต่างกันมาหาค่าเฉลี่ยเพื่อวิเคราะห์สัญญาณการน็อคของเครื่องยนต์ พบว่าค่าเฉลี่ยที่ได้จากสัญญาณการสั่นสะเทือนของเครื่องยนต์ปกติ และค่าเฉลี่ยจากสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อคมีช่วงของค่าแตกต่างกัน ซึ่งสามารถแยกสัญญาณการสั่นสะเทือนของเครื่องยนต์ปกติ และสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อคได้

Thesis Tittle	Application of Signal Analysis for Checking Knock Combustion of 1200 Coupe Engine by Vibration Signal
Author	Mr. Paijit Kochakornjarupong
Major Program	Electrical Engineering
Academic Year	1998

Abstract

This thesis describes an application of signal analysis in discrete time system for checking combustion knock of a TOYOTA Corolla 1200 engine by recording vibration signals of the engine from a sensor at an engine speeds 1800, 2000, 2200, 2400 and 2600 cycles/minute. In a normal combustion case, an engine ignition timing is adjusted to before TDC 20 degrees. For a combustion knock case, an engine ignition timing is adjusted to before TDC 30 degrees. The vibration signals are transformed to a frequency domain with the Fast Fourier Transform and the Discrete Wavelet Transform by a computer program. It was found that at a range of frequencies between 4.3 kHz and 6.3 kHz, there were distinctive differences between amplitudes of a normal signal and a knock signal. The amplitudes of the Discrete Wavelet Transform are then averaged and have a range of values that can separate a normal vibration signal and a knock vibration signal.

กิตติกรรมประกาศ

ขอขอบพระคุณ อาจารย์ผศ.ดร.กิตติพัฒน์ ตันตระรุ่งโรจน์ อาจารย์ที่ปรึกษา อาจารย์ปัญญรักษ์ งามศรีตระกูล อาจารย์ผศ.ชัชวาลย์ ยนต์หงส์ อาจารย์ที่ปรึกษาร่วม อาจารย์ดร. นิตยา นินทรกิจ อาจารย์วีระพันธุ์ มุสิกสาร ที่กรุณาให้คำปรึกษาชี้แนะ และช่วยแก้ไขวิทยานิพนธ์นี้จนสำเร็จตามวัตถุประสงค์ ขอขอบพระคุณอาจารย์ผศ.แสวง กะระณา คณะกรรมการสอบที่กรุณาให้คำแนะนำและตรวจแก้ไขวิทยานิพนธ์นี้จนสำเร็จตามวัตถุประสงค์ ขอขอบพระคุณ บิดา มารดา พี่ ที่ให้กำลังใจ สนับสนุนและช่วยเหลือ ขอขอบคุณ ครูและบุคลากรในภาควิชาวิศวกรรมไฟฟ้า วิศวกรรมคอมพิวเตอร์ และ วิศวกรรมเครื่องกลที่ช่วยเหลือและอำนวยความสะดวกในการทำวิทยานิพนธ์ ขอขอบคุณ เพื่อนๆ และ น้องๆ ที่ช่วยเหลือด้านต่างๆ

ไพจิตร กชกรจารุพงศ์

สารบาญ

สารบาญ

รายการตาราง

รายการภาพประกอบ

หน้า

หน้า

หน้า

ความหมายของศัพท์เฉพาะในวิทยานิพนธ์นี้

หมวดไทย-อังกฤษ

กราฟการไหลของสัญญาณ	signal flow graph
กาบอร์ฟังก์ชัน	Gabor function
การกลับบิต	bit reversed
การชักตัวอย่าง	sampling
การน็อคเนื่องจากการจุดตัวเอง	Detonation Knock
การน็อคเนื่องจากการชิงจุดระเบิด	Pre-ignition Knock
ข้อมูลเฉพาะที่	localization
ควอตซ์	Quartz
คอนจูเกตเชิงซ้อน	complex conjugate
ค่าคงที่ของโหมดของการสั่นสะเทือน	vibration mode constant
ค่ารากของกำลังสองเฉลี่ย	Root Mean Square
ค่าสูงสุดไปยังค่าต่ำสุด	Peak to Peak
ค่าเวลาการแปลงผัน	conversion time
จังหวะคาย	Exhaust stroke
จังหวะดูด	Suction stroke
จังหวะระเบิด	Power stroke
จังหวะอัด	Compression stroke
โดเมนความถี่	Frequency domain
ตัวตรวจรู้	Sensor
ตัวแปลงฟูริเยอร์แบบเร็ว	FFT (Fast Fourier Transform)
ตัวแปลงฟูริเยอร์เต็มหน่วย	DFT (Discrete Fourier Transform)
ตัวแปลงเวฟเลต	WT (Wavelet Transform)
ตัวแปลงเวฟเลตเต็มหน่วย	DWT (Discrete Wavelet Transform)
แผงวงจรแปลงผันแอนะล็อกเป็นดิจิตอล	Analog to Digital converter card
พารามิเตอร์การขยาย	dilation parameter

ความหมายของศัพท์เฉพาะในวิทยานิพนธ์นี้

หมวดไทย-อังกฤษ (ต่อ)

พารามิเตอร์การเลื่อน	translation parameter
โพลาไรเซชัน	Polarization Direction
ฟิโซอิเลกตริก	Piezoelectric
ฟูริเยอร์	Fourier
ฟังก์ชันอิมพัลส์	impulse function
ฟังก์ชันเวฟเลตพื้นฐาน	basic wavelet function
เฟอร์โรอิเลกตริกเซรามิกส์	Ferroelectric Ceramics
ไฟตั้งองศาจุดระเบิด	Timing light
ระยะการเคลื่อนที่	Displacement
ระดับค่าเฉลี่ย	Average Level
ระดับยอดสูงสุด	Peak Level
ระดับยอดสูงสุดถึงยอดสูงสุดอีกยอดหนึ่ง	Peak to Peak Level
ระดับสีเทา	gray level
ระบบเวลาต่อเนื่อง	continuous time system
ระบบเวลาเต็มหน่วย	discrete time system
เลขฐานสอง	binary number
ศูนย์ตายบน	TDC (Top Dead Center)
ศูนย์ตายล่าง	BDC (Bottom Dead Center)
สมการของเดรปเปอร์	Draper's equation
สัญญาณการสั่นสะเทือน	Vibration Signal
สัญญาณไซน์	sine signal
อัตราการชักตัวอย่าง	sampling rate
อัตราไนควีซต์	Nyquist rate
แอมพลิจูด	amplitude
เฮิรตซ์	Hz (Hertz)

ความหมายของศัพท์เฉพาะในวิทยานิพนธ์นี้

หมวด อังกฤษ-ไทย

Amplitude	แอมพลิจูด
Analog to Digital converter card	แผงวงจรแปลงผันแอนะล็อกเป็นดิจิตอล
Average Level	ระดับค่าเฉลี่ย
basic wavelet function	ฟังก์ชันเวฟเลตพื้นฐาน
BDC (Bottom Dead Center)	ศูนย์ตายล่าง
binary number	เลขฐานสอง
bit reversed	การกลับบิต
complex conjugate	คอนจูเกตเชิงซ้อน
Compression stroke	จังหวะอัด
continuous time system	ระบบเวลาต่อเนื่อง
conversion time	ค่าเวลาการแปลงผัน
Detonation Knock	การน็อคเนื่องจากการจุดตัวเอง
DFT (Discrete Fourier Transform)	ตัวแปลงฟูริเยอร์เต็มหน่วย
dilation parameter	พารามิเตอร์การขยาย
discrete time system	ระบบเวลาเต็มหน่วย
Displacement	ระยะการเคลื่อนที่
Draper's equation	สมการของเดรปเปอร์
DWT (Discrete Wavelet Transform)	ตัวแปลงเวฟเลตเต็มหน่วย
Exhaust stroke	จังหวะคาย
Ferroelectric Ceramics	เฟอร์โรอิเลกตริกเซรามิกส์
FFT (Fast Fourier Transform)	ตัวแปลงฟูริเยอร์แบบเร็ว
Fourier	ฟูริเยอร์
Frequency domain	โดเมนความถี่
Gabor function	กาบอร์ฟังก์ชัน
gray level	ระดับสีเทา

ความหมายของศัพท์เฉพาะในวิทยานิพนธ์นี้

หมวด อังกฤษ-ไทย (ต่อ)

Hz (Hertz)	เฮิรตซ์
impulse function	ฟังก์ชันอิมพัลส์
Localization	ข้อมูลเฉพาะที่
Nyquist rate	อัตราไนควีซต์
Peak Level	ระดับยอดสูงสุด
Peak to Peak	ค่าสูงสุดไปยังค่าต่ำสุด
Peak to Peak Level	ระดับยอดสูงสุดถึงยอดสูงสุดอีกยอดหนึ่ง
Piezoelectric	ฟีโซอิเลกตริก
Polarization Direction	โพลาไรเซชัน
Power stroke	จังหวะระเบิด
Pre-ignition Knock	การน็อคเนื่องจากการชิงจุดระเบิด
Quartz	ควอตซ์
Root Mean Square	ค่ารากของกำลังสองเฉลี่ย
sampling	การชักตัวอย่าง
sampling rate	อัตราการชักตัวอย่าง
Sensor	ตัวตรวจรู้
signal flow graph	กราฟการไหลของสัญญาณ
sine signal	สัญญาณไซน์
Suction stroke	จังหวะดูด
TDC (Top Dead Center)	ศูนย์ตายบน
Timing light	ไฟตั้งองศาจุดระเบิด
translation parameter	พารามิเตอร์การเลื่อน
vibration mode constant	ค่าคงที่ของโหมดของการสั่นสะเทือน
Vibration Signal	สัญญาณการสั่นสะเทือน
WT (Wavelet Transform)	ตัวแปลงเวฟเลต

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของหัวข้อวิจัย

เครื่องยนต์สันดาปภายใน เป็นเครื่องกลสำหรับเปลี่ยนพลังงานความร้อนให้เป็นพลังงานกลที่จะนำไปใช้งานต่างๆได้ จากระบบการทำงานที่ต้องใช้งานอุปกรณ์ต่างๆอย่างต่อเนื่อง เครื่องยนต์ที่เกิดการน็อคขึ้นถ้าปล่อยไว้ไม่แก้ไขจะทำให้เครื่องยนต์นั้นเกิดการเสียหายได้ ซึ่งทำให้เกิดความสูญเสียในการทำงานและเกิดการเสียหายกับเครื่องยนต์ที่ต้องการใช้งานอย่างต่อเนื่องตลอดเวลา เช่น อุปกรณ์ต่างๆที่ใช้ในโรงงานที่มีเครื่องยนต์เป็นส่วนประกอบเพื่อใช้ในการผลิตสืบเนื่องกัน ดังนั้นการตรวจสอบการน็อคของเครื่องยนต์ จึงเป็นส่วนสำคัญที่จะป้องกันการเสียหายที่อาจจะเกิดขึ้น

เครื่องยนต์ที่ทำงานอยู่ในสภาพปกติ จะมีลักษณะของการสั่นสะเทือนที่คงที่ แต่ถึงแม้เครื่องยนต์มีการน็อคเกิดขึ้น ถ้าไม่มีการสังเกต หรือไม่มีเครื่องมือวัดที่ถูกต้อง ก็จะไม่สามารถทราบได้ จึงจำเป็นต้องมีเครื่องมือที่สามารถรับค่าความเปลี่ยนแปลงได้อย่างต่อเนื่องและรวดเร็ว เนื่องจากอุปกรณ์อิเล็กทรอนิกส์สามารถใช้อ่านค่าการเปลี่ยนทางฟิสิกส์ได้ถูกต้องและรวดเร็ว เมื่อนำอุปกรณ์อิเล็กทรอนิกส์ต่อกับคอมพิวเตอร์จะสามารถนำข้อมูลที่อ่านค่าได้ มาประมวลผลโดยการใช้ซอฟต์แวร์ ทำให้สามารถวิเคราะห์ข้อมูลสภาพของเครื่องยนต์ซึ่งจะช่วยให้การตรวจสอบทำได้อย่างสะดวกและรวดเร็ว และจะเป็นประโยชน์ในการป้องกันความเสียหายอันอาจจะเกิดจากการทำงานของเครื่องยนต์ที่เกิดการน็อค

1.2 วัตถุประสงค์

1.2.1 เพื่อทำการศึกษาลักษณะของข้อมูลด้วยโดเมนความถี่(frequency domain) ของสัญญาณที่ได้จากการสั่นสะเทือนของเครื่องยนต์ที่ทำการทดลอง สำหรับวิเคราะห์สัญญาณ เพื่อเป็นแบบอย่าง ในการตรวจสอบการน็อคของเครื่องยนต์

1.2.2 เพื่อศึกษาและออกแบบ วิธีการวิเคราะห์สัญญาณที่ทำการทดลอง ในการตรวจสอบเครื่องยนต์ ว่ามีการน็อคของเครื่องยนต์หรือไม่

1.3 ขอบเขตการวิจัย

1.3.1 เก็บข้อมูลสัญญาณที่ได้จากการสั่นสะเทือนของเครื่องยนต์ TOYOTA Corolla 1200 ในกรณีปกติ และในกรณีที่เกิดการน็อค

1.3.2 ออกแบบวิธีการวิเคราะห์สัญญาณเพื่อตรวจสอบสัญญาณการน็อคของเครื่องยนต์ TOYOTA Corolla 1200

1.4 ขั้นตอนและวิธีดำเนินการวิจัย

1.4.1 ศึกษาทฤษฎีและหลักการการน็อคของเครื่องยนต์

1.4.2 บันทึกสัญญาณการสั่นสะเทือนของเครื่องยนต์จากกรณีปกติ และกรณีที่มีการน็อค

1.4.3 ศึกษาลักษณะสัญญาณที่ได้จากการทดลองโดยการวิเคราะห์ในโดเมนความถี่

1.4.4 ศึกษาวิธีการวิเคราะห์สัญญาณ เพื่อตัดสินว่าเครื่องยนต์เกิดการน็อคหรือไม่

1.4.5 ออกแบบวิธีการวิเคราะห์สัญญาณ

1.4.6 ทดสอบ แก้ไข และปรับปรุงการทำงานของการวิเคราะห์สัญญาณ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้ทราบถึงลักษณะสัญญาณการน็อคของเครื่องยนต์ และวิธีการวิเคราะห์สัญญาณในโดเมนความถี่เพื่อตรวจสอบการน็อคของเครื่องยนต์

1.5.2 ได้ข้อมูลพื้นฐานสำหรับพัฒนาการวิเคราะห์สำหรับตรวจสอบการน็อคของเครื่องยนต์ หรือ การวิจัยในลักษณะใกล้เคียงกันต่อไป

บทที่ 2

การน็อคของเครื่องยนต์และสัญญาณการสั่นสะเทือนของเครื่องยนต์

2.1 คำนำ

เครื่องยนต์เป็นเครื่องมืออย่างหนึ่งที่ใช้เปลี่ยนพลังงานความร้อนจากเชื้อเพลิงให้เป็นพลังงานกล ถ้าจะกล่าวอย่างกว้าง ๆ จะสามารถแบ่งเป็น 2 ชนิดคือ เครื่องยนต์สันดาปภายใน และ เครื่องยนต์สันดาปภายนอก ถ้าเครื่องยนต์ใช้กระบวนการเปลี่ยนพลังงานความร้อนเป็นพลังงานกลโดยการใส่เชื้อเพลิงเข้าไปในเครื่องยนต์ และทำให้เกิดการสันดาปภายในเครื่องยนต์ เรียกว่า เป็นเครื่องยนต์สันดาปภายใน เช่น เครื่องยนต์แก๊สโซลีน เครื่องยนต์ดีเซล ในทางกลับกันถ้าการสันดาปของเครื่องยนต์เกิดขึ้นนอกเครื่องยนต์ โดยอาศัยตัวกลางเป็นตัวนำความร้อน และเปลี่ยนไปเป็นพลังงานในเครื่องยนต์ เรียกเครื่องยนต์นั้นว่า เครื่องยนต์สันดาปภายนอก เช่น เครื่องจักรไอน้ำ ที่อาศัยไอน้ำเป็นสารทำงาน นำความร้อนจากหม้อไอน้ำผ่านให้เครื่องจักรไอน้ำเพื่อผลิตพลังงานกลออกมา

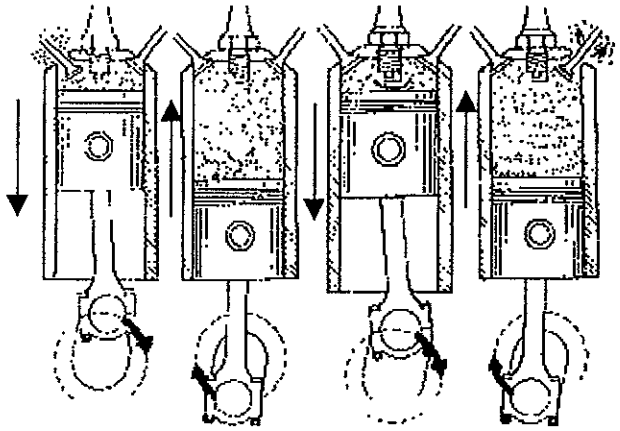
การเคลื่อนที่ของส่วนประกอบของเครื่องยนต์ที่รับพลังงานความร้อนมาเปลี่ยนเป็นพลังงานกล แยกเป็น 2 ลักษณะ คือ การเคลื่อนที่เชิงเส้นตรง กลับไป-กลับมา และการเคลื่อนที่เชิงการหมุน เครื่องยนต์ที่เป็นลักษณะแรกเรียกว่าเครื่องยนต์ลูกสูบ เช่น เครื่องยนต์จุดระเบิดด้วยหัวเทียน เครื่องยนต์จุดระเบิดด้วยแรงอัด ส่วนเครื่องยนต์ในลักษณะหลัง เช่น เครื่องกังหันไอน้ำ กังหันแก๊ส เครื่องยนต์ไอพ่น เป็นต้น

สำหรับเครื่องยนต์ที่จะนำมาวิเคราะห์หาการน็อคในการวิจัยนี้ คือ เครื่องยนต์ TOYOTA Corolla 1200 นี้เป็นเครื่องยนต์แก๊สโซลีนประเภทสันดาปภายในแบบ 4 จังหวะ ที่ใช้การเคลื่อนที่เชิงเส้นตรง และจุดระเบิดด้วยหัวเทียน

2.2 จังหวะการทำงานของเครื่องยนต์

เครื่องยนต์แก๊สโซลีนแบบ 4 จังหวะ เป็นเครื่องยนต์แก๊สโซลีนที่มีจังหวะในการทำงานคือ จังหวะดูด(Suction stroke) จังหวะอัด(Compression stroke) จังหวะระเบิด(Power

stroke) และจังหวะคาย(Exhaust stroke) รวมกันเป็นการทำงานครบ 1 วัฏจักรการทำงาน มีขั้นตอนการทำงานดังแสดงในรูป 2-1



ก ข ค ง

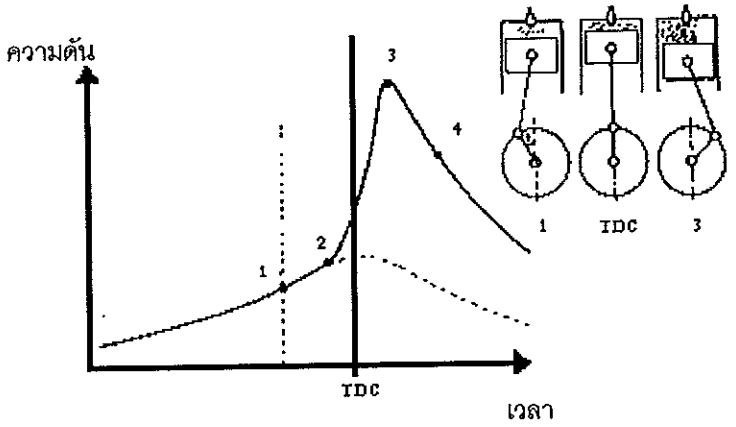
จังหวะดูด จังหวะอัด จังหวะระเบิด จังหวะคาย

ภาพประกอบ 2-1 แสดงส่วนประกอบและการทำงานของเครื่องยนต์ก๊าซโซลีน 4 จังหวะ

ในภาพประกอบ 2-1 ก เป็นจังหวะดูด ลูกสูบเริ่มเลื่อนจากตำแหน่งศูนย์ตายบน (Top Dead Center, TDC)ลงสู่ตำแหน่งศูนย์ตายล่าง(Bottom Dead Center, BDC) เป็นจังหวะที่ลิ้นไอดีเปิด ลูกสูบจะทำหน้าที่ดูดส่วนผสมระหว่างเชื้อเพลิงกับอากาศหรือไอดี ผ่านลิ้นไอดีเข้าในกระบอกสูบ จนสุดระยะเลื่อนลงของลูกสูบ ภาพประกอบ 2-1 ข เป็นจังหวะอัด เมื่อลูกสูบเลื่อนผ่านตำแหน่งศูนย์ตายล่าง และเริ่มเลื่อนขึ้นและลิ้นไอดีปิด เป็นการอัดส่วนผสมให้มีความดันและความร้อนสูงขึ้น ภาพประกอบ 2-1 ค เป็นจังหวะระเบิด เมื่อลูกสูบเคลื่อนที่เกือบถึงตำแหน่งศูนย์ตายบน ในตำแหน่งที่ตั้งจังหวะการจุดระเบิดไว้ชุดอุปกรณ์จุดระเบิดจะจุดประกายไฟด้วยหัวเทียน ทำให้เกิดการสันดาปของส่วนผสมเชื้อเพลิงกับอากาศ ความดันสูงนั้นจะดันลูกสูบให้เคลื่อนที่ลงเป็นกำลังหมุนเพลาข้อเหวี่ยง ภาพประกอบ 2-1 ง เป็นจังหวะคาย เมื่อลูกสูบเลื่อนลงจงถึงตำแหน่งที่กำหนดก่อนถึงศูนย์ตายล่าง ลิ้นไอเสียจะเปิด ให้ก๊าซที่เกิดจากการสันดาปหรือไอเสีย ออกนอกกระบอกสูบ จากนั้นลูกสูบจะเลื่อนขึ้นเพื่อคายไอเสียจนลูกสูบเลื่อนขึ้นเกือบถึงตำแหน่งศูนย์ตายบนที่กำหนด ลิ้นไอดีจะเริ่มเปิดก่อน และลิ้นไอเสียจะปิดเมื่อลูกสูบเลื่อนเลยตำแหน่งศูนย์ตายบนไปเล็กน้อยเป็นการเริ่มต้นวัฏจักรการทำงานใหม่ต่อไป

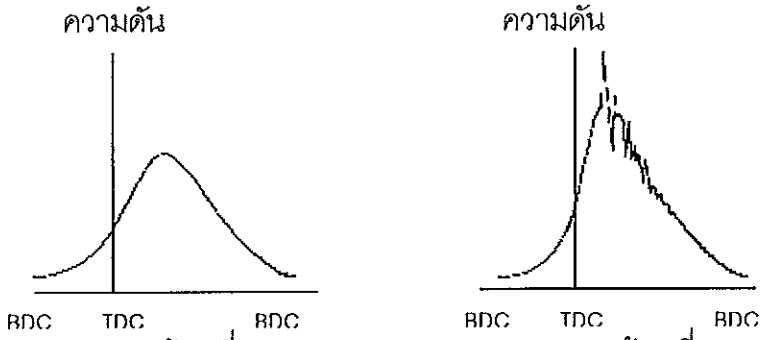
2.3 การน็อคของเครื่องยนต์

เครื่องยนต์จะสามารถให้พลังงานได้เมื่อเกิดการสันดาปภายในขึ้น ซึ่งเครื่องยนต์สันดาปภายในจะมีขั้นตอนการสันดาปโดยเริ่มจากการที่หัวเทียนจุดประกายไฟที่ตำแหน่งที่กำหนด เพื่อเริ่มเผาไหม้ให้แก่เชื้อเพลิง โดยตำแหน่งที่กำหนดนี้ต้องเผื่อเวลาให้กับการลุกลามของเปลวไฟหรือการเผาไหม้ด้วย ซึ่งเรียกว่า จังหวะการจุดระเบิดล่วงหน้า จากนั้นเชื้อเพลิงจะใช้เวลาช่วงหนึ่งเพื่อขยายตัว และขยายตัวสูงสุดเพื่อให้กำลังงานมากที่สุด ดังจะเห็นได้จากภาพประกอบ 2-2 จุดที่ 1 คือจุดที่หัวเทียนจุดประกายไฟให้แก่เชื้อเพลิง จุดที่ 2 คือจุดที่การเผาไหม้เริ่มขยายตัวซึ่งจะเกิดเมื่อลูกสูบเคลื่อนที่เกือบจะถึงตำแหน่งศูนย์ตายบน จุดที่ 3 คือจุดที่การเผาไหม้ขยายตัวได้สูงสุด และ จุดที่ 4 คือจุดสิ้นสุดการสันดาป



ภาพประกอบ 2-2 แสดงขั้นตอนการเผาไหม้ของเชื้อเพลิงกับความดันในกระบอกสูบ

โดยทั่วไป การน็อคในเครื่องยนต์จะมี 2 ชนิด คือ การน็อคเนื่องจากการชิงจุดระเบิด (Pre-ignition Knock)อันเป็นการน็อคเนื่องจากการชิงจุดระเบิดของเชื้อเพลิงก่อนที่หัวเทียนจะจุดประกายไฟ และการน็อคเนื่องจากการจุดตัวเอง(Detonation Knock)ซึ่งเป็นการน็อคที่เกิดภายหลังที่หัวเทียนจุดประกายไฟแล้ว เป็นอาการการเกิดชิงจุดหลายตำแหน่งในห้องเผาไหม้ อาการน็อคดังกล่าวจะใช้ในความหมายที่แสดงถึงเสียงที่ผิดปกติ ที่เกิดขึ้นในกระบวนการสันดาป ซึ่งสามารถแสดงด้วยกราฟของความดันในกระบอกสูบดังนี้



ภาพประกอบ 2-3 แสดงกราฟความดันในกระบอกสูบระหว่างการสันดาป
เปรียบเทียบกรณีปกติและเมื่อเกิดการน็อคเนื่องจากการจุดตัวเอง

จากภาพประกอบ 2-3 ก แสดงให้เห็นกราฟการสันดาปปกติที่เรียบ และ ภาพประกอบ 2-3 ข แสดงกราฟความดันในกระบอกสูบที่เกิดการน็อคที่มีความดันขึ้นสูงในเวลาอันรวดเร็ว ซึ่งคลื่นความดันนี้จะไปกระแทกผนังห้องเผาไหม้และหัวลูกสูบ ทำให้เกิดการสั่นสะเทือนและทำให้เกิดเสียงดังขึ้น

สาเหตุการน็อคของเครื่องยนต์สามารถเกิดจากการตั้งเวลาการจุดระเบิดของหัวเทียนล่วงหน้าไว้มากเกินไป และคุณสมบัติของน้ำมันที่มีจุดติดไฟได้เองต่ำกว่าปกติ การเพิ่มองศาการจุดระเบิดของหัวเทียนจะเพิ่มความแรงของการน็อคเนื่องจากการจุดตัวเอง และการลดองศาของการจุดระเบิดจะลดความแรงของการน็อคลง การตั้งเวลาจุดระเบิดของเครื่องยนต์ล่วงหน้าไว้มาก ๆ มีแนวโน้มทำให้การเกิดการน็อคเนื่องจากการจุดตัวเองมากขึ้น ดังนั้น เพื่อลดปัญหาดังกล่าวจึงจำเป็นต้องลดเวลาการจุดระเบิดล่วงหน้าลง ให้อยู่ในระดับที่พอเหมาะ

การน็อคทำให้เกิดข้อเสียดังนี้

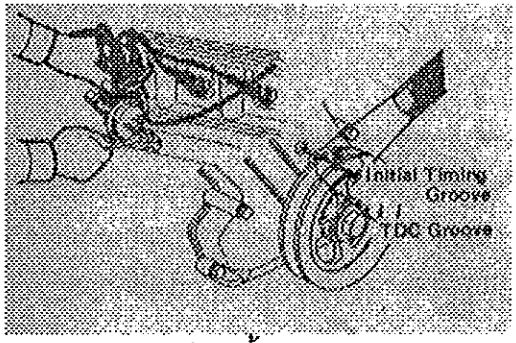
1. ทำให้เกิดแรงกระแทกบนชิ้นส่วน โดยเฉพาะบนหัวลูกสูบ ซึ่งแรงกระแทกดังกล่าวจะถูกส่งต่อไปยังก้านสูบและเพลาข้อเหวี่ยง ทำให้แบริ่งสึกหรอ ก้านสูบคดงอหากรุนแรงมาก ๆ อาจทำให้ลูกสูบแตกหรือทะลุได้

2. ทำให้เกิดเสียงดัง ซึ่งเป็นลักษณะเสียงเคาะหรือเสียงน็อค ส่วนมากมาจากชิ้นส่วนต่างๆ ของเครื่องยนต์ที่อยู่ภายใต้คลื่นความดันอย่างทันทีทันใด

3. ความแตกต่างความดันมีผลทำให้เกิดคลื่นความดัน แก๊สจะขยายตัวและหดตัวไล่ฟิล์มน้ำมันที่ติดอยู่กับผนังกระบอกสูบ ทำให้สูญเสียความร้อนไปกับการระบายความร้อนมาก

4. ทำให้กำลังของเครื่องยนต์ลดลง การน็อคถ้าเกิดก่อนการจุดระเบิดของหัวเทียน จะทำให้เกิดการต้านทานการหมุนของเครื่องยนต์ แต่ถ้าเกิดคล้อยหลังมาก ๆ ก็จะทำให้สูญเสียเชื้อเพลิงมาก โดยไม่เกิดประโยชน์แต่อย่างใด

การตั้งเวลาการจุดระเบิดมีผลต่อการน็อคของเครื่องยนต์ตามที่กล่าวมาแล้ว วิธีการตั้งเวลาการจุดระเบิดทำได้โดยการตั้งไฟและตรวจสอบโดยใช้เครื่องมือที่เรียกว่าไฟตั้งองศาจุดระเบิด (Timing light) ทั้งนี้เมื่อตั้งไฟเครื่องยนต์โดยใช้ไฟตั้งองศาจุดระเบิดดูองศาการจุดระเบิดที่แท้จริง ถ้าองศาที่ปรากฏไม่ตรงกับที่ต้องการ ให้ขยับหมุนเรือนจานจ่ายจนกระทั่งได้ตำแหน่งตามต้องการ จากนั้นจึงล็อคน็อตยึดเรือนจานจ่ายให้แน่นดังภาพประกอบ 2-4



ภาพประกอบ.2-4 แสดงอุปกรณ์และวิธีการตั้งเวลาการจุดระเบิด
(สุจิตต์ สนองคุณ, 2531)

2.4 สัญญาณการสั่นสะเทือน (Vibration Signal)

เนื่องจากการน็อคสามารถสังเกตุได้จากการฟังเสียงผิดปกติที่เกิดขึ้นในกระบวนการสันดาป แต่มีข้อเสียในกรณีที่มีเสียงรบกวนที่มีระดับสูงกว่าเสียงน็อคจะแยกแยะได้ยาก สำหรับการแก้ไขสามารถทำได้โดยการใช้การวิเคราะห์จากสัญญาณการสั่นสะเทือน เนื่องจากการน็อคจะทำให้เกิดความดันในกระบอกสูบไปกระแทกกับโครงสร้างของเครื่องยนต์ ถ้าเกิดการน็อคมากจะมีการสั่นสะเทือนรุนแรงขึ้นทำให้สามารถสังเกตุสัญญาณการน็อคได้ สัญญาณการสั่นสะเทือนที่วัดได้จากเครื่องยนต์ เพื่อนำมาช่วยในการวิเคราะห์มีพารามิเตอร์ที่ใช้ในการวัดประกอบด้วย ความถี่ และ ขนาดของการสั่นสะเทือน

1. ความถี่ของการสั่นสะเทือน หากพิจารณาสัญญาณการสั่นสะเทือนบนโดเมนเวลาก็จะหมายถึง จำนวนครั้งของการสั่นสะเทือนต่อหน่วยเวลา ซึ่งหน่วยที่ใช้จะเป็น วินาที $^{-1}$ หรือ เฮิรตซ์

(Hertz, Hz) พารามิเตอร์ความถี่นี้จะเป็นส่วนสำคัญที่จะนำไปใช้กับการวิเคราะห์สัญญาณการสั่นสะเทือนบนโดเมนความถี่

2. ขนาดของการสั่นสะเทือน เป็นพารามิเตอร์ที่ใช้บอกถึงสภาพการสั่นสะเทือนของเครื่องยนต์ โดยการใช้ระยะการเคลื่อนที่ (Displacement) หรือระยะขจัดจากจุดสมดุล ในกรณีที่วัดค่าจากค่าสูงสุดไปยังค่าต่ำสุด (Peak to Peak) จะเป็นค่าระยะทางทั้งหมดที่ตัวมวลเคลื่อนที่จากจุดสูงสุดไปสู่จุดต่ำสุดในแต่ละครั้ง ซึ่งจะมีทั้งค่าบวกและลบและมีค่าแปรเปลี่ยนตามเวลา การบอกขนาดของสัญญาณการสั่นสะเทือนในลักษณะการบอกขนาดโดยรวมของการสั่นสะเทือนที่นิยมใช้กันมีดังต่อไปนี้คือ

ก. ระดับยอดสูงสุด (Peak Level) เป็นการบอกค่าระดับสูงสุดของสัญญาณที่เบี่ยงเบนไปจากระดับศูนย์ในทางบวก โดยไม่ใช้ค่าทางลบ ค่านี้มักนิยมใช้วัดการสั่นสะเทือนที่เกิดจากการกระแทกในช่วงเวลาสั้น ๆ

ข. ระดับยอดสูงสุดถึงยอดสูงสุดอีกยอดหนึ่ง (Peak to Peak Level) ซึ่งก็คือการบอกค่าขนาดของสัญญาณที่วัดจากจุดสูงสุดทางบวกกับจุดต่ำสุดทางลบ

ค. ระดับค่าเฉลี่ย (Average Level) เป็นค่าเฉลี่ยของสัญญาณที่เกิดขึ้นในช่วงเวลาหนึ่ง ๆ ในกรณีทั่ว ๆ ไปที่สัญญาณการสั่นสะเทือนมีค่าเป็นทั้งบวกและลบเมื่อเทียบกับตำแหน่งสมดุล ซึ่งจะทำให้ค่าเฉลี่ยเป็นศูนย์ ค่าเฉลี่ยในกรณีนี้จะคิดโดยทำการเปลี่ยนเครื่องหมายของขนาดสัญญาณที่ลบให้เป็นบวก แล้วจึงคิดค่าเฉลี่ยออกมา

ง. ค่ารากของกำลังสองเฉลี่ย (Root Mean Square) เป็นค่าที่ได้จากการนำสัญญาณที่วัดได้ในโดเมนเวลาที่เกิดขึ้นในช่วงหนึ่ง ๆ มายกกำลังสองแล้วทำการเฉลี่ยตลอดคาบและถอดรากที่สอง

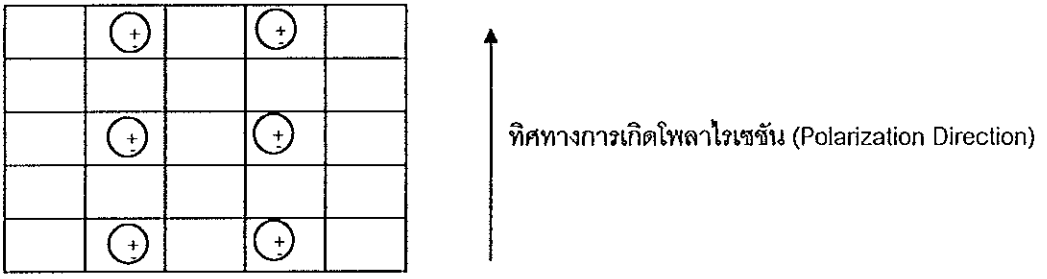
$$x_{rms} = \sqrt{\frac{1}{\tau}\int_0^{\tau} x^2(t)dt}$$

เมื่อ x(t) เป็นค่าสัญญาณ ณ เวลา t และ τ เป็นช่วงเวลาที่ทำการวัดสัญญาณหรือคาบของสัญญาณ

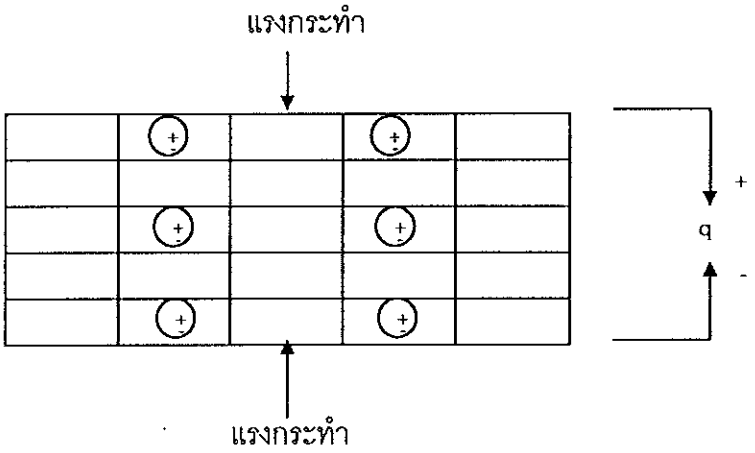
โดยทั่วไปการสั่นสะเทือนไม่ได้เป็นลักษณะของคลื่นรูปไซน์ แต่อย่างไรก็ดีสัญญาณเหล่านี้ก็จะสามารถแยกออกเป็นผลรวมของสัญญาณรูปไซน์หลายๆ ความถี่ได้ตามทฤษฎีของฟูริเยอร์ (Fourier)

2.5 การวัดการสั่นสะเทือนโดยใช้ตัวตรวจรู้ (Sensor)

เครื่องยนต์เมื่อเกิดการน็อคจะมีการสั่นสะเทือนที่ต่างไปจากเครื่องยนต์ปกติ [Bahman 1996] ดังนั้นจึงได้ทำการจับสัญญาณการสั่นสะเทือนของเครื่องยนต์มาทำการวิเคราะห์เพื่อตรวจหาการน็อคได้ด้วยตัวตรวจรู้ ตัวตรวจรู้จะทำหน้าที่แปลงการสั่นสะเทือนที่ได้มีการจับสัญญาณเป็นสัญญาณทางไฟฟ้า โดยการวัดขนาดของการสั่นสะเทือนโดยตรง ในที่นี้เลือกใช้ตัวตรวจรู้ที่ใช้หลักการของพิโซอิเลกตริก(Piezoelectric)ซึ่งทำด้วยวัสดุเฟอร์โรอิเลกตริกเซรามิกส์(Ferroelectric Ceramics) ที่มีคุณสมบัติให้ประจุไฟฟ้าเมื่อถูกแรงกด ดังภาพประกอบ 2-5



ก.



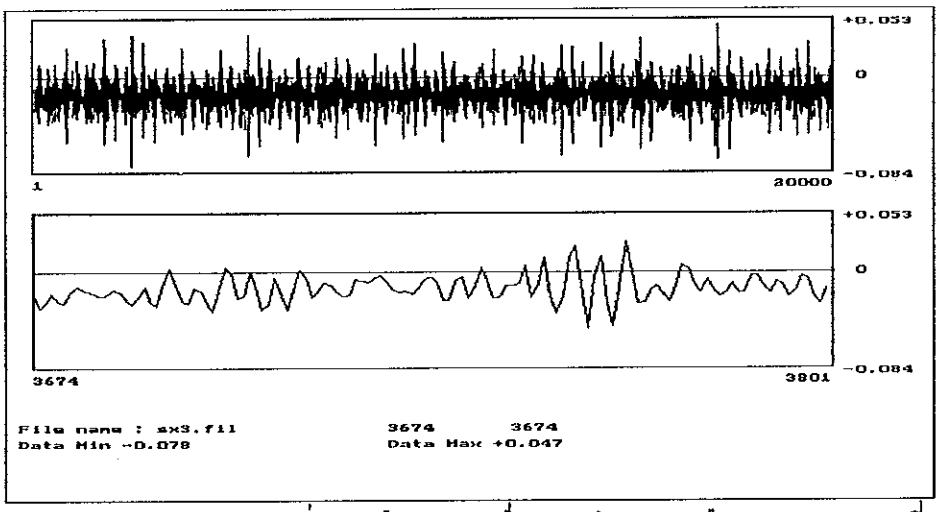
ข.

ภาพประกอบ 2-5 แสดงลักษณะของแรงกระทำต่อผลึกพิโซอิเล็กตริก ก. สภาพที่ยังไม่เปลี่ยนรูป และ ข. สภาพการเปลี่ยนรูปจากแรงกระทำ

จากคุณสมบัติของตัวตรวจรู้ที่ใช้หลักการพิโซอิเล็กตริกโดยการใช้วัสดุบางประเภท เช่น ควอตซ์(Quartz) และ เฟอร์โรอิเล็กตริกเซรามิกส์(Ferroelectric Ceramics)บางชนิดที่จะเกิดประจุไฟฟ้าเมื่อถูกแรงกด วัสดุพิโซอิเล็กตริกทำหน้าที่เหมือนสปริงรับแรงกดจากมวลซึ่งสั่นทำให้

เกิดปริมาณประจุไฟฟ้าตามแรงกดจากการสั่นสะเทือนจากแรงกดดังแสดงในรูป 2-5 ซึ่งสามารถตอบสนองต่อการสั่นสะเทือนที่เกิดขึ้นได้ จึงสามารถนำไปวัดการสั่นสะเทือนของเครื่องยนต์ สัญญาณที่ได้นี้สามารถนำไปวิเคราะห์เพื่อตรวจสอบว่าเกิดการน็อคของเครื่องยนต์หรือไม่

ในการวิเคราะห์สัญญาณจะนำสัญญาณการสั่นสะเทือนที่วัดได้จากเครื่องยนต์ มาประมวลโดยโปรแกรมที่เขียนขึ้นโดยใช้ ตัวแปลภาษาเทอร์โบซี รุ่น 2.0 (Turbo C version 2.0) โปรแกรมจะทำการอ่านแฟ้มข้อมูลที่ต้องการแล้วจึงทำการเลือกช่วงของข้อมูลที่ต้องการจากการดูสัญญาณที่บอกการจุดระเบิดจากหัวเทียนดังภาพประกอบ 2-6



ภาพประกอบ 2-6 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์และการเลือกช่วงสัญญาณที่ต้องการจากโปรแกรมคอมพิวเตอร์

บทที่ 3

การวิเคราะห์สัญญาณในโดเมนความถี่

จากการศึกษาจากเอกสารเกี่ยวกับการวิเคราะห์สัญญาณเครื่องยนต์พบว่าการใช้ตัวตรวจรู้รับสัญญาณการสั่นสะเทือนจากเครื่องยนต์เป็นสัญญาณในทางโดเมนของเวลา และมีข้อมูลโดเมนของความถี่ด้วย ทั้งนี้จำเป็นต้องใช้สมการทางคณิตศาสตร์เข้ามาแปลงข้อมูลจากโดเมนเวลาให้เป็นโดเมนความถี่เพื่อทำการวิเคราะห์สัญญาณในโดเมนความถี่ ซึ่งจะทำการศึกษาวิธีของตัวแปลงฟูริเยอร์แบบเร็ว (Fast Fourier Transform, FFT) และ ตัวแปลงเวฟเลต (Wavelet Transform, WT) เพื่อนำมาวิเคราะห์ข้อมูลในโดเมนความถี่ต่อไป

3.1 การวิเคราะห์โดยใช้ตัวแปลงฟูริเยอร์แบบเร็ว (Fast Fourier Transform, FFT)

ตัวแปลงฟูริเยอร์เป็นวิธีการวิเคราะห์สัญญาณที่สามารถแสดงข้อมูลของสัญญาณในแกนของความถี่ ในระบบเวลาต่อเนื่อง(continuous time system) ตัวแปลงฟูริเยอร์จะหาได้ดังสมการที่ 3.1

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-j2\pi\, ft} dt \qquad \ldots (3.1)$$

เมื่อ x(t)เป็นฟังก์ชันในแกนเวลา และ X(f)เป็นฟังก์ชันในแกนความถี่ ซึ่งเป็นค่าตัวแปลงฟูริเยอร์ ของx(t) สามารถเขียนตัวแปลงฟูริเยอร์ ในรูปของจำนวนเชิงซ้อนได้เป็น

$$X(f) = R(f) + jI(f) = |X(f)|\, e^{j\,\theta(f)} \qquad \cdots (3.2)$$

เมื่อ R(f) คือ ส่วนจำนวนจริงของตัวแปลงฟูริเยอร์

I(f) คือ ส่วนจินตภาพของตัวแปลงฟูริเยอร์

|X(f)| คือ ค่าแอมพลิจูด(amplitude)ของตัวแปลงฟูริเยอร์มีค่าเท่ากับ $\sqrt{R^2(f) + I^2(f)}$

θ(f) คือ ค่ามุมของตัวแปลงฟูริเยอร์ มีค่าเท่ากับ $\tan^{-1}[I(f)/R(f)]$

เพื่อให้สามารถนำตัวแปลงฟูริเยอร์ มาใช้กับระบบเวลาเต็มหน่วย(discrete time system) จึงใช้วิธีการชักตัวอย่าง(sampling)สัญญาณ เพื่อหาตัวแปลงฟูริเยอร์เต็มหน่วย (Discrete Fourier Transform, DFT)

โดยการใช้ฟังก์ชันอิมพัลส์(impulse function)ที่มีข้อกำหนดว่า

$$\delta(t-\gamma)=0 \quad ;t\neq\gamma$$

ซึ่งมีคุณสมบัติ 1. $\int_{-\infty} f(t)\ \delta\ (t-\gamma)dt = f(\gamma)$; γ เป็น จำนวนจริง

2. $\int_{-\infty} \delta\ (t-\gamma)dt = 1$; γ เป็น จำนวนจริง

จะได้ สัญญาณจากการการชักตัวอย่าง คือ

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(t)\ \delta(t-nt) = \sum_{n=\infty}^{\infty} x(nT)\ \delta(t-nt)$$; T เป็นคาบของการชักตัวอย่าง

จากตัวแปลงฟูริเยอร์

$$X(f) = \int_{-\infty}^{\infty} x(t)\ e^{-j2\pi\ ft} dt$$

เมื่อให้ x(t)เป็นสัญญาณที่สร้างจากสัญญาณสุ่ม แล้วทำการชักตัวอย่างจะได้ว่า

$$X_s(f) = \int_{-\infty}^{\infty}\left[\sum_{-\infty}^{\infty} x(nT)\delta(t-nT)\right] e^{-j2\pi\ ft} dt$$

เมื่อจากคุณสมบัติของฟังก์ชันอิมพัลส์ $\delta(t)$ จะได้

$$X_s(f) = \sum_{n=-\infty}^{\infty} x(nT)\ e^{-j2\pi\ ft}$$

นั่นคือจะได้ตัวแปลงฟูริเยอร์เต็มหน่วย(DFT)คือสมการที่ 3.3

$$X(k) = \sum_{n=0}^{N-1} x(n)\ e^{-j2\pi\ nk/N} \qquad \ldots (3.3)$$

เมื่อให้ $W^k = e^{-j2\pi\ k/N}$ และ $W^k = -W^{k+N/2}$ จากสมการที่ 3.3 จะได้

$$X(k) = \sum_{n=0}^{N-1} x(n)\ W^{nk} \qquad \ldots (3.4)$$

ลักษณะที่สำคัญของวิธี FFT นี้คือการเป็นคาบของค่า W การคำนวณจะใช้การพิจารณาถึงค่าที่ซ้ำกันและค่าที่ยกกำลังของ W ดังเช่นให้ $N = 4 = 2^2$ นำมาพิจารณาเป็นเมตริกซ์ของสมการ3.4 จะได้ผลคือ

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad \ldots (3.5)$$

เมื่อแทนค่า $W^0=1$ ในสมการ 3.5 จะได้

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^4 & W^6 \\ 1 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3 \end{bmatrix} \quad \text{... (3.6)}$$

จากคุณสมบัติของ $W^{nk} = W^{nk \bmod(N)}$ เมื่อ nk mod(N) คือการหาค่าเศษจากการหาร nk ด้วย N แทนค่าให้สมการ 3.6 จะได้

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^0 & W^2 \\ 1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3 \end{bmatrix} \quad \text{... (3.7)}$$

นำสมการ 3.7 มาแยกตัวประกอบการคูณของเมตริกซ์ เพื่อหาการคูณและการบวกที่ซ้ำซ้อนโดยการสลับตำแหน่งดังนี้

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \quad \text{... (3.8)}$$

พิจารณาการแยกเมตริกซ์ของสมการ 3.8 กำหนดให้

$$\begin{bmatrix} x_1(0) \\ x_1(2) \\ x_1(1) \\ x_1(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \quad \text{... (3.9)}$$

นั่นคือการคำนวณหา $x_1(0) = x(0) + W^0 x(2)$ มีการคูณกัน 1 ครั้ง และบวก 1 ครั้ง เมื่อต้องการหา $x_1(2)$ ใช้การบวกครั้งเดียวเพราะ $W^0 = -W^2$ ทำให้ $x_1(2) = x(0) + W^2 x(2) = x(0) - W^0 x(2)$ จะเห็นว่าไม่ต้องคูณ $W^0 x(2)$ อีกเพราะสามารถนำค่าที่ได้มาจากการคำนวณของ $x_1(0)$ นั่นคือการหาค่า $x_1(k)$ มีการคูณ 2 ครั้ง และ บวก 4ครั้ง ต่อไปคำนวณการคูณของส่วนที่เหลือ

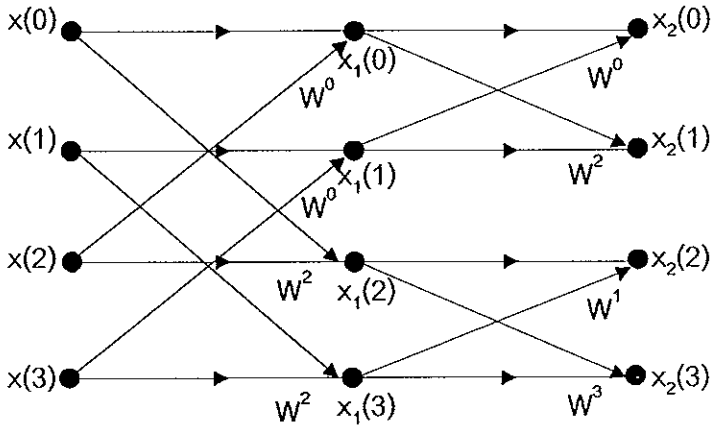
$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} x_2(0) \\ x_2(2) \\ x_2(1) \\ x_2(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 0 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} \quad \text{... (3.10)}$$

พิจารณาการคำนวณ $x_2(0) = x_1(0) + W^0 x_1(1)$ มีการคูณกัน 1 ครั้ง และบวก 1 ครั้ง เมื่อต้องการหา $x_2(1)$ ใช้การบวกครั้งเดียวเพราะ $W^0 = -W^2$ ได้จากคำนวณของ $x_2(0)$ สำหรับ $x_2(2)$จะมีการคูณและบวกอย่างละครั้ง และ $x_2(3)$จะมีการบวกครั้งเดียว

นั่นคือถ้าใช้วิธีการนี้จะใช้การคูณ 4 ครั้ง และ บวก 8 ครั้ง เมื่อเทียบกับการทำ DFTแล้วจะต้องคูณ 16 ครั้ง และ บวก 12 ครั้ง จากนั้นต้องมีการนำผลลัพธ์มาสลับตำแหน่งให้เหมือนเดิมโดยใช้วิธีการกลับบิต(bit reversed) ของตำแหน่งข้อมูลที่แปลงให้เป็นเลขฐานสองแล้วเรียงลำดับบิตใหม่จากหลังไปหน้าดังนี้

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} \xrightarrow{\text{แปลงตำแหน่งเป็นเลขฐานสอง}} \begin{bmatrix} X(00) \\ X(10) \\ X(01) \\ X(11) \end{bmatrix} \xrightarrow{\text{กลับบิต}} \begin{bmatrix} X(00) \\ X(01) \\ X(10) \\ X(11) \end{bmatrix}$$

สามารถนำขั้นตอนการทำงานของ FFT เขียนกราฟการไหลของสัญญาณ(signal flow graph)ได้ดังต่อไปนี้



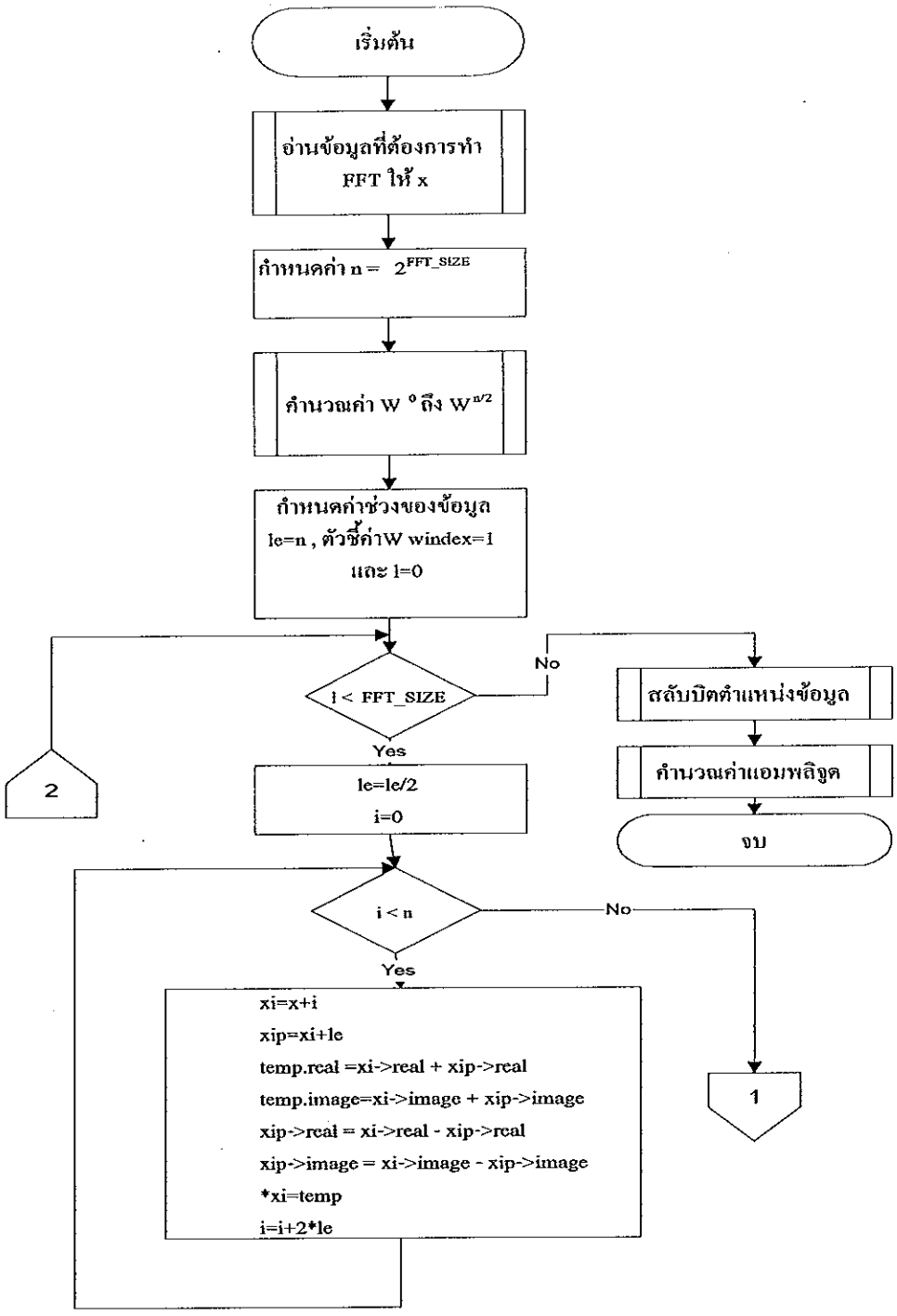
ภาพประกอบ3-1 แสดงกราฟการไหลของสัญญาณ ของ การทำ FFT ที่มีค่า N=4

จากกราฟการไหลของสัญญาณที่เป็นการแสดงการคูณกันและบวกกันตามตำแหน่งลูกศร เช่น $x_1(1) = x(0) + W^0 x(2)$ และนำมาหาการข้ามการคูณของค่าที่มีการคำนวณไว้แล้วได้ เช่น ค่า $W^0 x(2)$ ได้จากการคำนวณของ $x_1(0) = x(0) + W^0 x(2)$ สามารถข้ามได้เมื่อไปหาค่า $x_1(2) = x(0) + W^2 x(2) = x(0) - W^0 x(2)$ แล้วนำผลลัพธ์มากลับบิตก็จะได้ค่า FFT

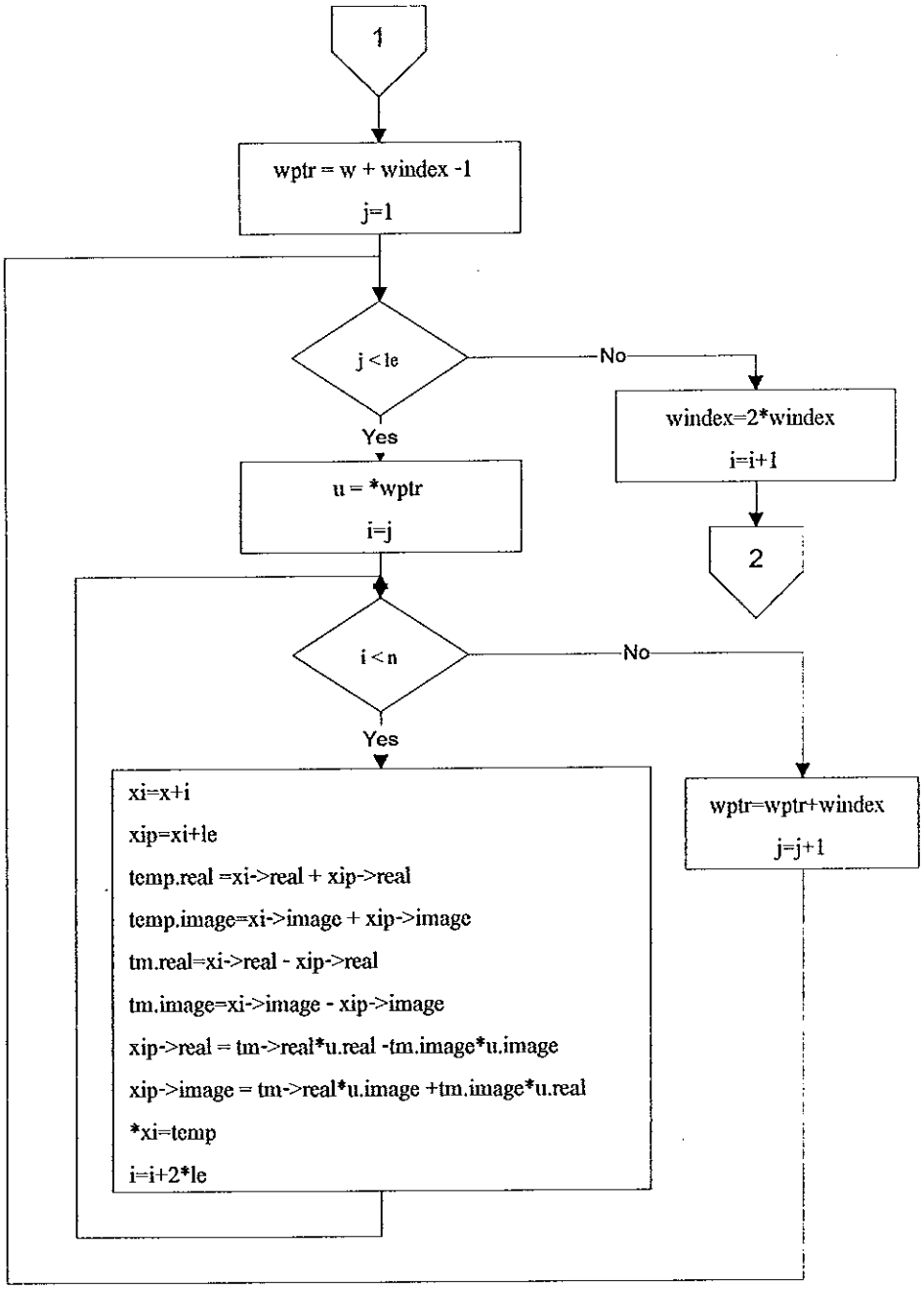
เมื่อต้องการหาค่า FFT ของข้อมูล N จำนวน สามารถทำได้โดยการจับคู่กันของค่า W ที่เหมือนกัน โดยจะจับคู่กันในการคำนวณระดับแรกด้วยระยะห่างของข้อมูล N/2 จำนวน เช่น ในภาพประกอบ 3-1 ค่า N=4 ดังนั้น $x_1(0)$ จับจับคู่กับ $x_1(2)$ และ $x_1(1)$ จะจับคู่กับ $x_1(3)$ เมื่อคำนวณเสร็จแล้ว ในการจับคู่ในการคำนวณระดับต่อไปให้ลดระยะห่างอีกครึ่งหนึ่งเป็น N/4 เช่นในภาพประกอบ 3-1 $x_2(0)$ จับจับคู่กับ $x_2(1)$ และ $x_2(2)$ จะจับคู่กับ $x_2(3)$ แล้วทำขั้นตอนการคำนวณโดยการจับคู่ต่อไปจนกระทั่งเหลือระยะห่างเป็น 1 จะได้ผลลัพธ์ ที่เมื่อนำไปกลับบิตของตำแหน่งเป็นค่า FFT

การหาค่าของ DFT โดยการหาค่าตัวประกอบของ W เมื่อพิจารณาการทำงานของDFT ถ้าต้องการหาค่าผลลัพธ์ต้องทำการคูณจำนวน N^2 ครั้ง และต้องทำการบวกกัน N(N-1) ครั้ง เวลาที่ต้องใช้ในการคำนวณจะเป็น N^2 เทียบกับการทำ FFT แล้วจะต้องคูณจำนวน $(N/2)\log_2 N$ ครั้ง และต้องทำการบวกกัน $N \log_2 N$ ครั้ง เวลาที่ต้องใช้ในการคำนวณจะเป็น $N \log_2 N$ เมื่อข้อมูลมีขนาดเพิ่มขึ้นการใช้ FFT จะช่วยลดเวลาในการคำนวณลงได้มาก

การวิเคราะห์โดยวิธี FFT จะทำโดยใช้โปรแกรมที่เขียนขึ้นตามผังงานในภาพประกอบ3-2

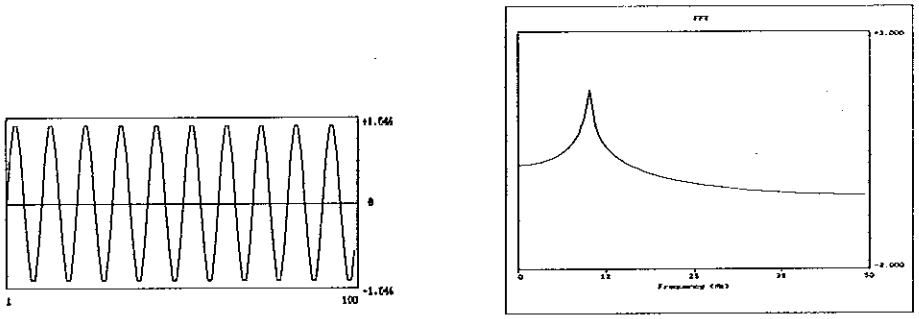


ภาพประกอบ 3-2 แสดงผังงานโปรแกรมทำ FFT



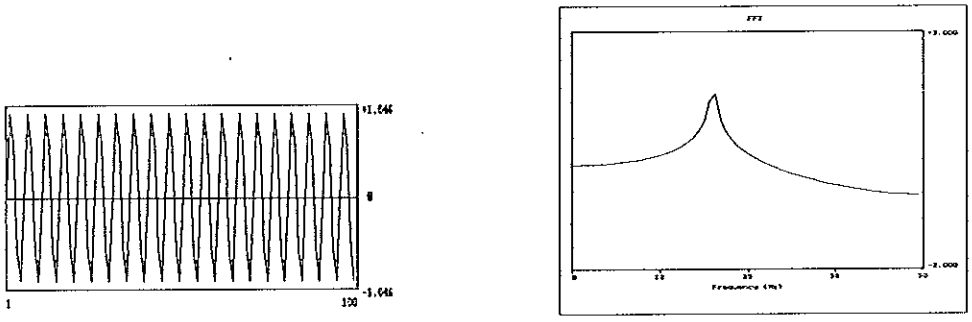
ภาพประกอบ 3-2 แสดงผังงานโปรแกรมทำ FFT (ต่อ)

เมื่อทำการวิเคราะห์ด้วยโปรแกรม FFT จะได้ผลลัพธ์เป็นข้อมูลในแกนเวลา และ แกนความถี่ในลักษณะดังนี้



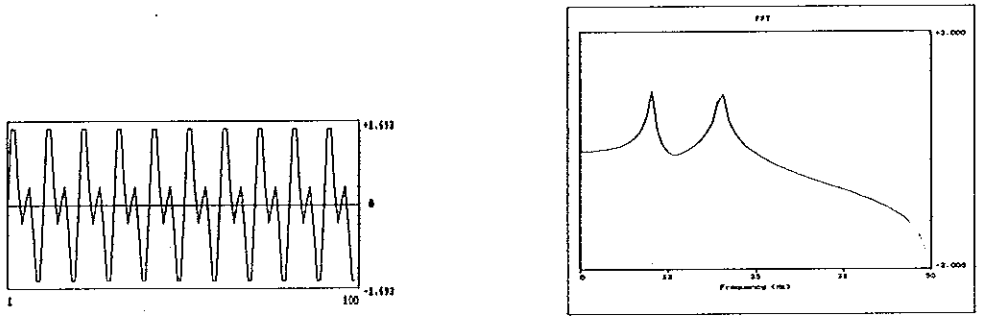
ก ข

ภาพประกอบ 3-3 แสดงสัญญาณไซน์ 10 เฮิรตซ์และแอมพลิจูดที่ได้จากการทำ FFT



ก ข

ภาพประกอบ 3-4 แสดงสัญญาณไซน์ 20 เฮิรตซ์และแอมพลิจูดที่ได้จากการทำ FFT



ก ข

ภาพประกอบ 3-5 แสดงสัญญาณรวม 2 สัญญาณ และแอมพลิจูดที่ได้จากการทำ FFT

จากภาพประกอบ 3-3 ก เป็นการแสดงภาพสัญญาณไซน์(sine signal) ความถี่ 10 เฮิรตซ์ ด้วยความเร็วการสุ่มสัญญาณ 100 ครั้งต่อวินาที จำนวนข้อมูล 128 จุด เพื่อหาค่า FFT ในภาพประกอบ 3-3 ข ได้ผลลัพธ์เป็นค่าแอมพลิจูดสูงสุดที่ความถี่ของข้อมูลที่บริเวณ 10 เฮิรตซ์ ภาพประกอบ 3-4 ก แสดงภาพสัญญาณไซน์ ความถี่ 20 เฮิรตซ์ ด้วยความเร็วการสุ่มสัญญาณ 100 ครั้งต่อวินาที จำนวนข้อมูล 128 จุด เพื่อหาค่า FFT ในภาพประกอบ 3-4 ข ได้ผลลัพธ์เป็นค่าแอมพลิจูดที่มีค่าสูงสุดที่ความถี่ของข้อมูลที่บริเวณ 20 เฮิรตซ์ ภาพประกอบ 3-5 ก แสดงสัญญาณ

ไซน์ ความถี่ 10 เฮิรตซ์ และ 20 เฮิรตซ์ รวมกันแล้วสุ่มสัญญาณ 100 ครั้งต่อวินาที จำนวนข้อมูล 128 จุด มาหาค่า FFT ในภาพประกอบ 3-5 ข ได้ผลลัพธ์เป็นค่าแอมพลิจูดสูงที่บริเวณ 10 เฮิรตซ์ และ 20 เฮิรตซ์

3.2 การวิเคราะห์โดยใช้ตัวแปลงเวฟเลต (Wavelet Transform, WT)

ตัวแปลงเวฟเลต เป็นวิธีการวิเคราะห์สัญญาณที่สามารถวิเคราะห์ข้อมูลโดยการแบ่งข้อมูลเป็นส่วนประกอบของความถี่ที่ต่างกัน และสามารถศึกษาแต่ละส่วนของข้อมูล ซึ่งเป็นเครื่องมือที่ใช้สำหรับศึกษาข้อมูลเฉพาะที่(localization)ทั้งในแกนเวลาและแกนความถี่

ตัวแปลงเวฟเลตในระบบเวลาต่อเนื่อง มีสมการดังนี้

$$S(a,b)=\frac{1}{\sqrt{a}}\int \bar{g}\left(\frac{t-b}{a}\right)s(t)dt \qquad \ldots (3.11)$$

เมื่อ g(t) คือ ฟังก์ชันพื้นฐานที่ใช้ในการแปลงเวฟเลต

$\bar{g}(t)$ คือ คอนจูเกตเชิงซ้อน(complex conjugate) ของ g(t)

s(t) คือ ข้อมูลในระบบเวลาต่อเนื่องที่ต้องการแปลง

a คือ พารามิเตอร์การขยาย(dilation parameter)

b คือ พารามิเตอร์การเลื่อน(translation parameter)

และเรียก $g_{a,b}(t)=\frac{1}{\sqrt{a}}g\left(\frac{t-b}{a}\right)$ ว่า ฟังก์ชันเวฟเลตพื้นฐาน(basic wavelet function)

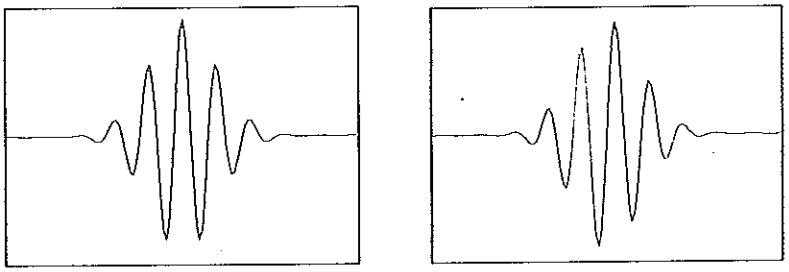
พิจารณาการทำตัวแปลงเวฟเลตในระบบเวลาเต็มหน่วย(discrete time system) โดยการสุ่มตัวอย่างของสมการ 3.11 จะได้ตัวแปลงเวฟเลตเต็มหน่วยคือ

$$S(a,b)=\sum_{n=-\infty}^{\infty}\bar{g}(nT)s(nT) \qquad \text{;T เป็นคาบของการชักสัญญาณ} \qquad \ldots (3.12)$$

ในการแปลงเวฟเลตเต็มหน่วย (Discrete Wavelet Transform, DWT) จะเลือกใช้กาบอร์ฟังก์ชัน (Gabor function) เป็นฟังก์ชันพื้นฐานของเวฟเลต เพื่อประโยชน์ในการได้ข้อมูลเฉพาะที่ทั้งในโดเมนเวลาและโดเมนความถี่ของสัญญาณ ซึ่ง กาบอร์ฟังก์ชันมีสมการดังนี้คือ

$$g(t)=e^{j2\pi t}e^{-t^2/2} \qquad \ldots (3.13)$$

จากสมการ 3.13 สามารถแสดงลักษณะของฟังก์ชันนี้ได้ดังภาพประกอบที่ 3-6



ก ข

ภาพประกอบ 3-6 แสดงฟังก์ชันพื้นฐานของเวฟเลตที่ใช้ในการวิเคราะห์

จากภาพประกอบ 3-6 แสดงให้เห็นถึงลักษณะของกาบอร์ฟังก์ชัน ที่ใช้เป็นฟังก์ชันพื้นฐานของเวฟเลต โดย ภาพประกอบ 3-6 ก เป็นส่วนของจำนวนจริง และ ภาพประกอบ 3-6 ข เป็นส่วนของจำนวนจินตภาพ เมื่อแทนค่าของ g(t) ลงในสมการ 3.12 จะได้

$$S(a,b) = \sum_{n=0}^{N-1} \bar{g}_{a,b}[n]s[n]$$

$$= \sum_{n=0}^{N-1} \left(\frac{1}{\sqrt{a}}\cos w_0\left(\frac{n-b}{a}\right)e^{-\left(\frac{n-b}{a}\right)^2/2} + i\frac{1}{\sqrt{a}}\sin w_0\left(\frac{n-b}{a}\right)e^{-\left(\frac{n-b}{a}\right)^2/2}\right) s[n]$$

$$= \frac{1}{\sqrt{a}}\sum_{n=0}^{N-1}\cos w_0\left(\frac{n-b}{a}\right)e^{-\left(\frac{n-b}{a}\right)^2/2}s[n] + i\frac{1}{\sqrt{a}}\sum_{n=0}^{N-1}\sin w_0\left(\frac{n-b}{a}\right)e^{-\left(\frac{n-b}{a}\right)^2/2}s[n]$$

เมื่อ $a = \frac{f_s}{\omega}$ และ $a = 2^l$; f_s คือ อัตราการชักตัวอย่าง(sampling rate)

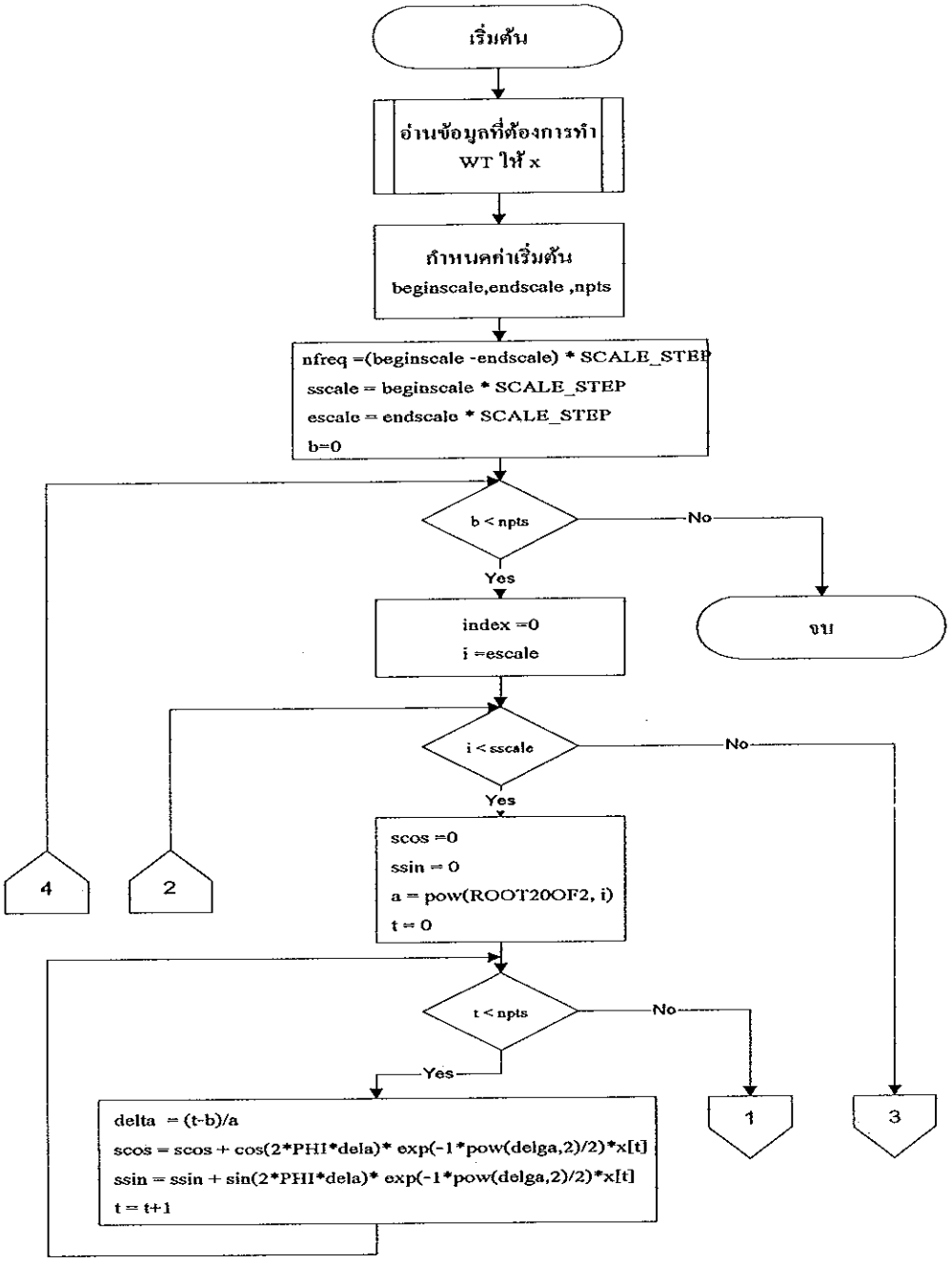
$\omega = \frac{f_s}{a}$; ω คือ ความถี่ที่ต้องการ

$\omega_0 = 2\pi$; a คือ พารามิเตอร์การขยาย(dilation parameter)

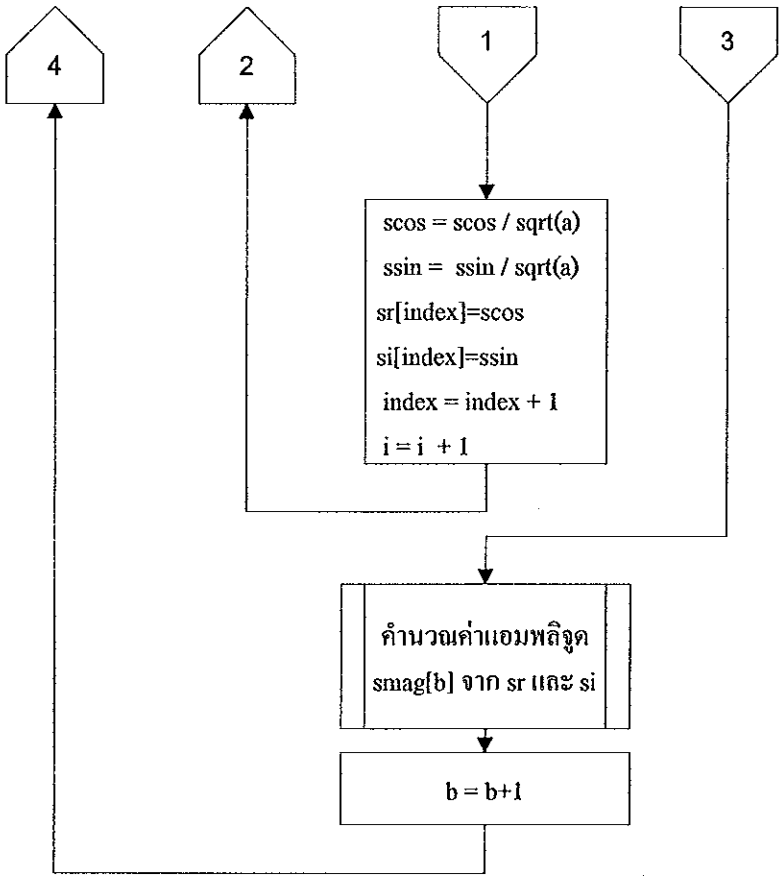
s [n] คือ ข้อมูลที่ต้องการทำ DWT

S(a,b) คือ ผลการทำ DWT

จากการวิเคราะห์โดยวิธีDWT โดยใช้โปรแกรมที่เขียนขึ้นตามผังงานในภาพประกอบ 3-7 ดังนี้

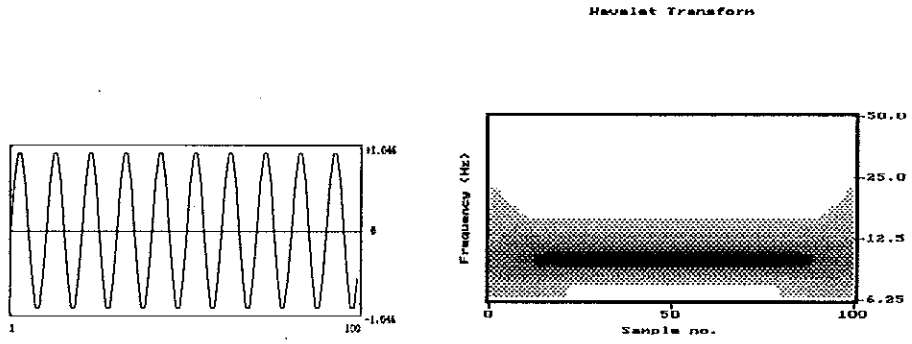


ภาพประกอบ 3-7 แสดงผังงานการทำ DWT



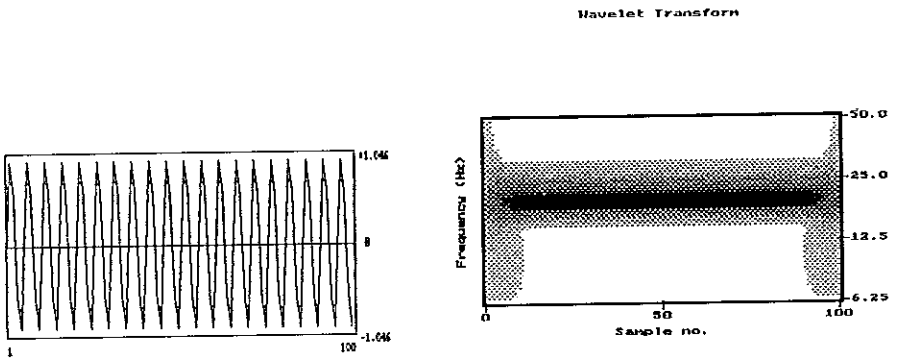
ภาพประกอบ 3-7 แสดงผังงานการทำ DWT (ต่อ)

เมื่อประมวลผลโปรแกรม DWT จะได้ผลลัพธ์เป็นข้อมูลในแกนตั้งเป็นค่าความถี่ สำหรับแกนนอนจะเป็นค่าตำแหน่งของข้อมูลที่นำมาแปลง และข้อมูลของแอมพลิจูดจะแสดงด้วยค่าระดับสีเทา(gray level) โดยแอมพลิจูดที่มีค่ามากจะมีค่าสีเข้มกว่า ดังสามารถแสดงในภาพประกอบ 3.8 ถึง 3.10



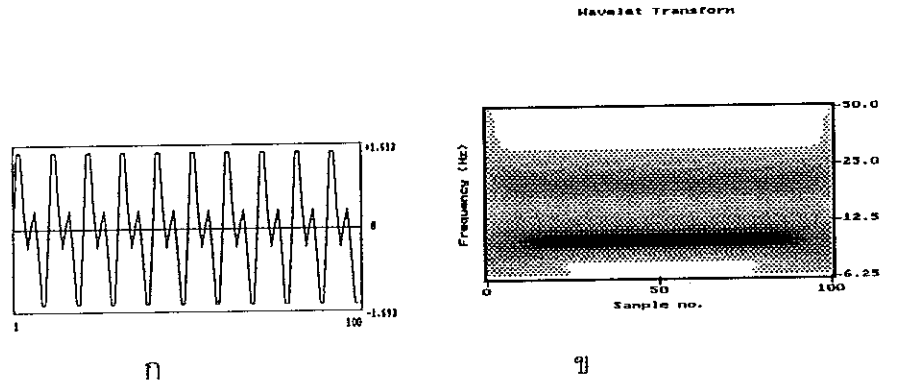
ก ข

ภาพประกอบ 3-11 แสดงสัญญาณไซน์ 10 เฮิรตซ์และแอมพลิจูดที่ได้จากการทำ DWT



ก ข

ภาพประกอบ 3-9 แสดงสัญญาณไซน์ 20 เฮิรตซ์และแอมพลิจูดที่ได้จากการทำ DWT



ก ข

ภาพประกอบ 3-10 แสดงสัญญาณรวม 2 สัญญาณและแอมพลิจูดที่ได้จากการทำ DWT

ภาพประกอบ 3-8 ก แสดงภาพการนำสัญญาณไซน์ ความถี่ 10 เฮิรตซ์ ด้วยความเร็วการสุ่มสัญญาณ 100 ครั้งต่อวินาที จำนวนข้อมูล 100 จุด มาทำ DWT ในภาพประกอบ 3-8 ข เป็นผลลัพธ์ของค่าแอมพลิจูดที่แสดงด้วยค่าระดับความเข้มของสี มีค่าของแอมพลิจูดสูงสุดที่ความถี่ของข้อมูลที่บริเวณ 10 เฮิรตซ์ ภาพประกอบ 3-9 ก แสดงภาพการนำสัญญาณไซน์ ความถี่ 20 เฮิรตซ์ ด้วยความเร็วการสุ่มสัญญาณ 100 ครั้งต่อวินาที จำนวนข้อมูล 100 จุด มาทำ DWT ในภาพประกอบ 3-9 ข เป็นผลลัพธ์ของค่าแอมพลิจูดที่แสดงด้วยค่าระดับความเข้มของสี มีค่าของแอมพลิจูดสูงสุดที่ความถี่ของข้อมูลที่บริเวณ 10 เฮิรตซ์ ภาพประกอบ 3-10 ก แสดงการนำสัญญาณไซน์ ความถี่ 10 เฮิรตซ์ และ 20 เฮิรตซ์ รวมกันแล้วสุ่มสัญญาณ 100 ครั้งต่อวินาที จำนวนข้อมูล 100 จุด มาทำ DWT ได้ผลลัพธ์ดังภาพประกอบ 3-10 ข โดยได้เป็นค่าแอมพลิจูดที่แสดงด้วยค่าระดับความเข้มของสี มีค่าของแอมพลิจูดสูงที่บริเวณ 10 เฮิรตซ์ และ 20 เฮิรตซ์

จากวิธีการวิเคราะห์ด้วย FFT และ DWT จะนำไปวิเคราะห์สัญญาณการสั่นสะเทือนของเครื่องยนต์ปกติ และ เครื่องยนต์ที่เกิดการน็อค ซึ่งผลการใช้วิธีการนี้จะนำเสนอในบทต่อไป

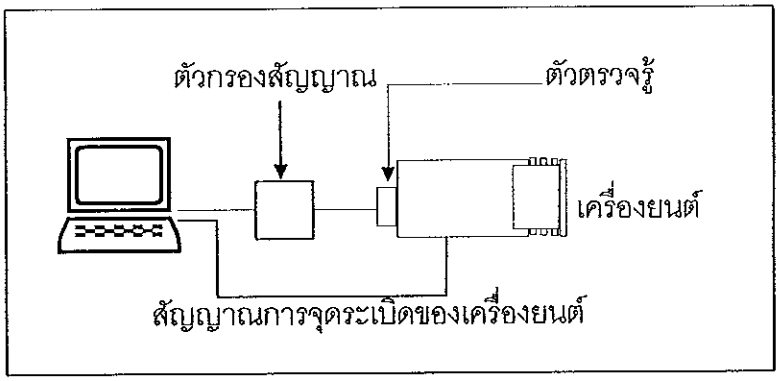
บทที่ 4

ผลการวัดสัญญาณ การวิเคราะห์ในโดเมนความถี่และสรุป

การวิเคราะห์เครื่องยนต์ในปัจจุบันมีการนำคอมพิวเตอร์มาประยุกต์ใช้งานมากขึ้นเพราะราคาที่ถูก ในเรื่องเครื่องยนต์การซ่อมบำรุงเป็นเรื่องหลักที่จะต้องดำเนินการตลอดเพื่อลดค่าใช้จ่ายในการซ่อมบำรุงและยืดอายุเครื่องยนต์ การตรวจสอบที่เป็นที่นิยมมักจะวิเคราะห์จากความสัญญาณการสั่นสะเทือนหรือสัญญาณเสียง โดยการใช้ตัวตรวจรู้ในการวัดสัญญาณ การวิเคราะห์สัญญาณในแกนความถี่เป็นการหาสเปรคตรัมที่เป็นคาบของการเคลื่อนไหวเรื่อยๆ การน็อคเป็นปัญหาหนึ่งซึ่งพบได้มากในการใช้งานเครื่องยนต์จริง ๆ และเป็นปัญหาหลักที่เป็นข้อจำกัดในการใช้เชื้อเพลิงให้มีประสิทธิภาพสูงสุดในเครื่องยนต์ที่ใช้การจุดระเบิด ซึ่งการที่เครื่องยนต์น็อคเกิดจากการระเบิดตัวเองของแก๊สในระหว่างการสันดาปภายใน การปล่อยพลังงานออกมาทันทีทันใดของการจุดระเบิดตัวเองนั้นทำให้เกิดความดันสูง และเกิดคลื่นช็อค ซึ่งจะทำให้เกิดเสียงความถี่ธรรมชาติของกระบอกสูบเป็นเสียงของการน็อค ซึ่งจะเกิดหลังจากตำแหน่งจุดศูนย์ตายบนของลูกสูบในจังหวะอัด ดังนั้นในการทดลองนี้จะทำการวัดสัญญาณของเครื่องยนต์โดยใช้การวัดการสั่นสะเทือนของเครื่องยนต์แล้วประมวลผลสัญญาณโดยโปรแกรมคอมพิวเตอร์ต่อไป

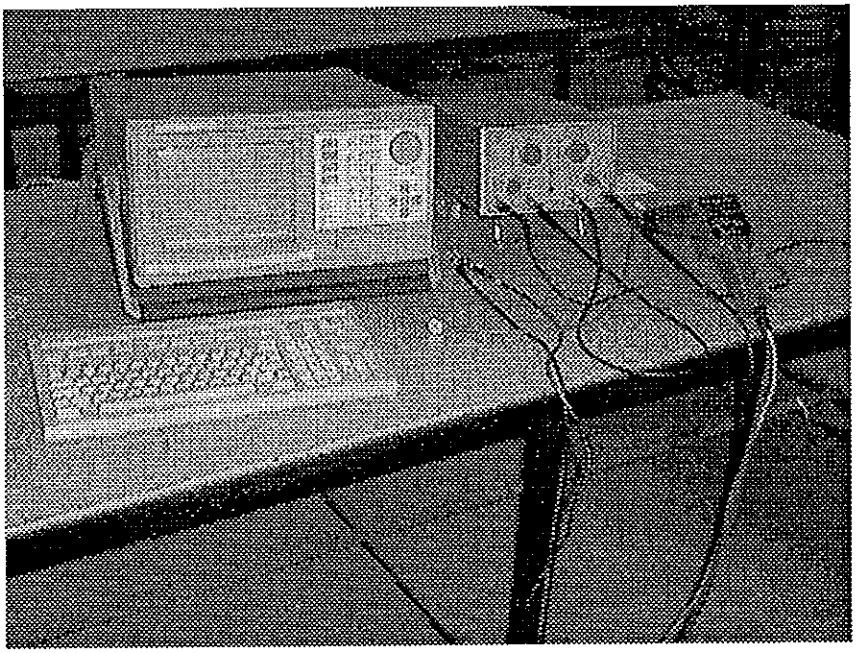
1. วิธีการวัดสัญญาณและอุปกรณ์ที่ใช้

ในการวัดสัญญาณเพื่อตรวจสอบการน็อคของเครื่องยนต์เพื่อให้สามารถวัดความเปลี่ยนแปลงทางกลของเครื่องยนต์ได้สะดวก ถูกต้องรวดเร็วจึงออกแบบให้ใช้ตัวตรวจรู้จับสัญญาณการสั่นสะเทือนจากเครื่องยนต์มาให้เครื่องบันทึกสัญญาณเพื่อเก็บข้อมูลไว้ ลักษณะของการทดลองได้เป็นดังภาพประกอบ 4-1



ภาพประกอบ 4-1 แสดงการวัดสัญญาณจากเครื่องยนต์โดยใช้ตัวตรวจรู้ส่งข้อมูลให้เครื่องเก็บสัญญาณ

จากภาพประกอบ 4-1 แสดงอุปกรณ์และวิธีการวัดสัญญาณ โดยใช้อุปกรณ์ประกอบด้วยตัวตรวจรู้เพื่อรับสัญญาณการสั่นสะเทือนของเครื่องยนต์ เครื่องกรองความถี่เพื่อกรองสัญญาณที่มีความถี่ในช่วงที่ต้องการเป็นเครื่องฟิลเตอร์ Krohn-Hite 3202 และเครื่องบันทึกสัญญาณ Tektronix 2050 ดิจิตอล ออสซิลโลสโคป และ ไฟตั้งองศาจุดระเบิดของเครื่องยนต์ ดังภาพประกอบที่4-2



ภาพประกอบ 4-2 แสดงอุปกรณ์ที่ใช้ในการวัดสัญญาณ

2. การบันทึกสัญญาณการสั่นสะเทือนจากเครื่องยนต์

ความถี่ของสัญญาณการน็อคของเครื่องยนต์สามารถประมาณได้อย่างหยาบ ๆ โดยใช้สมการของเดรปเปอร์ (Draper's equation) ดังนี้

$$f_r=(P_{mn}* C)/(\pi* B) \quad ...(4.1)$$

เมื่อ f_r คือ ความถี่การน็อค

P_{mn} คือ ค่าคงที่ของโหมดของการสั่นสะเทือน(vibration mode constant) มีค่า 2.007

C คือ ความเร็วของเสียงในแก๊สในกระบอกสูบ มีค่า 900 เมตร/วินาที

B คือ รัศมีของกระบอกสูบ

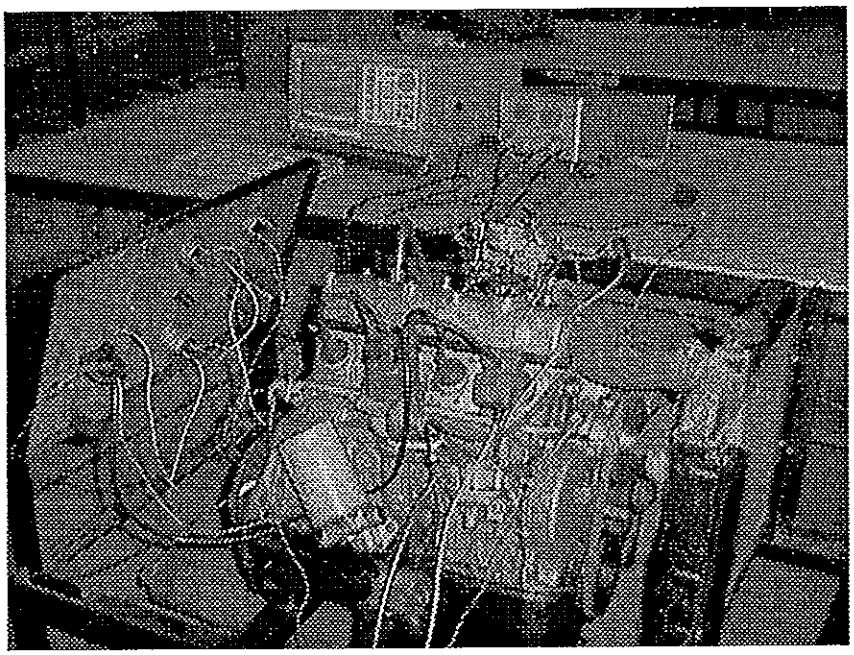
เมื่อค่ารัศมีของกระบอกสูบเครื่องยนต์ที่จับสัญญาณเท่ากับ 7.5 เซนติเมตร แทนค่าในสมการ 4.1จะได้ $f_r = (2.007 * 900) / (3.1416 * .075)$

$= 7,666$ Hz

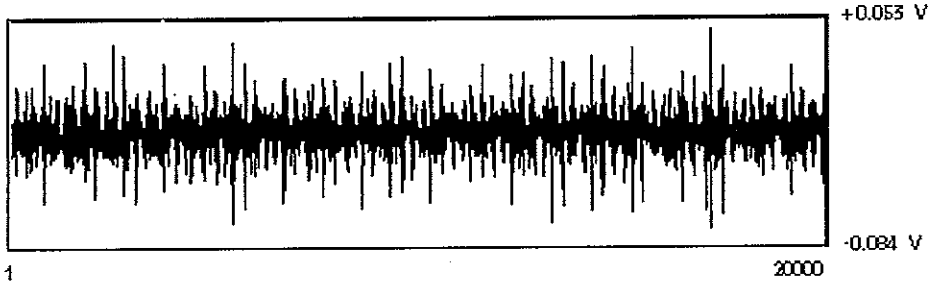
ในการบันทึกสัญญาณการสั่นสะเทือนของเครื่องยนต์เป็นการเก็บข้อมูลในรูปแบบของเวลาเต็มหน่วย เพื่อให้ข้อมูลที่เก็บได้มีข้อมูลความถี่ของสัญญาณเวลาต่อเนื่องที่ไปชักตัวอย่างครบ จากค่าความถี่ของสัญญาณการน็อคของเครื่องยนต์ที่ได้จากสมการ 4.1 ประมาณได้ 7.67 กิโลเฮิรตซ์ เมื่อใช้อัตราการชักตัวอย่างด้วยอัตราไนควีซต์ (Nyquist rate) จะต้องใช้ความถี่การชักตัวอย่างอย่างน้อย 2 เท่าของ f_r ดังนั้นจึงใช้การชักตัวอย่างด้วยอัตรา 20000 ตัวอย่างต่อวินาที

การวัดสัญญาณการสั่นสะเทือนจากตัวตรวจรู้ค่าที่ได้เป็นค่าที่สัมพันธ์โดยตรงกับความเร่งของการสั่นสะเทือนหรือแรงที่ทำกับตัวตรวจรู้ (Daniel J. Inman, 1996) ได้ค่าเป็นแรงดันไฟฟ้า ซึ่งสามารถแสดงถึงการสั่นสะเทือนของเครื่องยนต์ได้

สัญญาณที่บันทึกประกอบด้วย สัญญาณบอกการจุดระเบิดจากระบบจุดระเบิดของเครื่องยนต์จากคอยล์จุดระเบิด และสัญญาณการสั่นสะเทือนของเครื่องยนต์โดยติดตั้งตัวตรวจรู้ไว้ที่เครื่องยนต์ที่ผนังของเสื้อสูบ แล้วจึงต่อสายสัญญาณไปยังเครื่องฟิลเตอร์กรองสัญญาณที่ความถี่ 1000 เฮิรตซ์ ถึง 10,000 เฮิรตซ์ แล้วต่อสายสัญญาณไปยังเครื่องบันทึกสัญญาณ Tektronix 2050 ดิจิตอล ออสซิลโลสโคป โดยใช้อัตราการชักตัวอย่าง 20000 ตัวอย่างครั้งต่อวินาที ดังภาพประกอบที่ 4-3 และบันทึกข้อมูลไว้ 20000 ข้อมูล ดังภาพประกอบ 4-4 จากนั้นจึงทำการถ่ายข้อมูลไปยังเครื่องคอมพิวเตอร์เพื่อนำข้อมูลไปวิเคราะห์

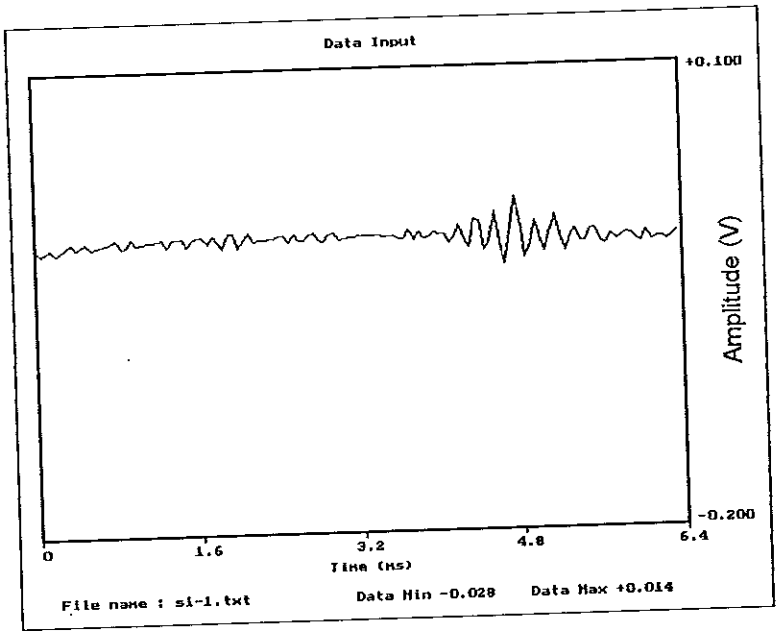


ภาพประกอบ 4-3 แสดงการวัดสัญญาณการสั่นสะเทือนของเครื่องยนต์

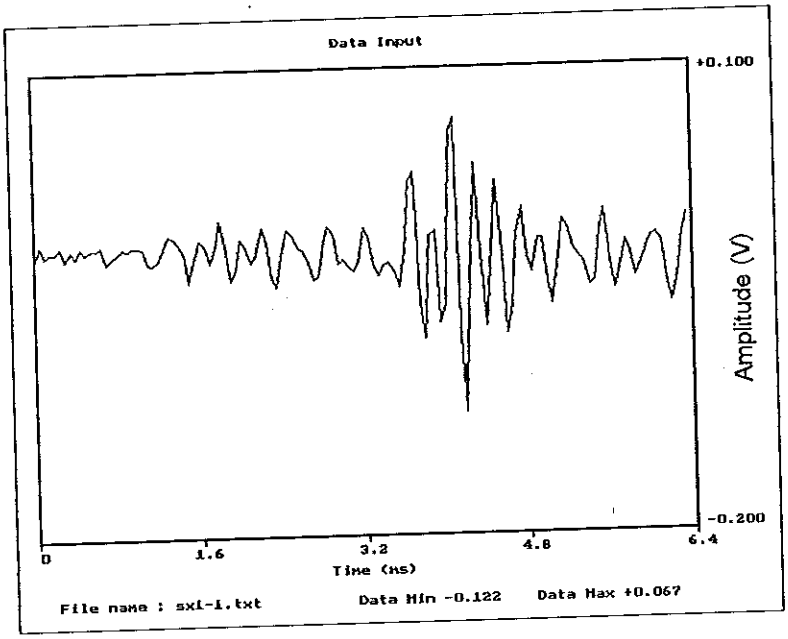


ภาพประกอบ 4-4 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่บันทึกจากเครื่องบันทึกสัญญาณ

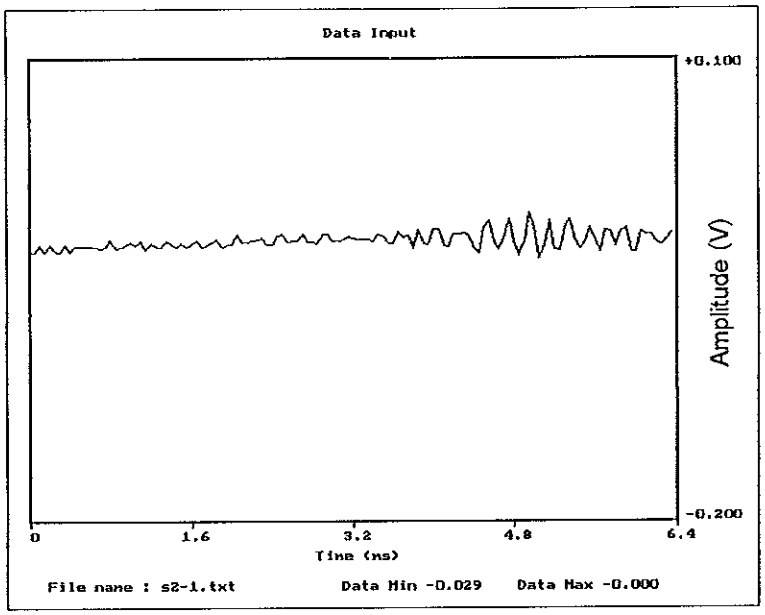
จากนั้นนำสัญญาณที่ได้ไปวิเคราะห์โดยการหาช่วงของข้อมูลที่ต้องการได้เป็นข้อมูลจำนวน 128 จุด หรือใช้เวลา 6.4 มิลลิวินาที จากตำแหน่งเริ่มต้นของการจุดระเบิด โดยทำการจับสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่ความเร็วรอบ 1800, 2000, 2200, 2400 และ 2600 รอบ/นาที เมื่อเริ่มทำการจับสัญญาณของเครื่องยนต์ในสภาพปกติจะตั้งจังหวะการจุดระเบิดของเครื่องยนต์ไว้ก่อนหน้า TDC 20 องศา และ ก่อนทำการจับสัญญาณของเครื่องยนต์ที่เกิดการน็อคจะตั้งจังหวะการจุดระเบิดของเครื่องยนต์เพื่อจับสัญญาณการน็อคของเครื่องยนต์ไว้ก่อนหน้า TDC 30 องศา โดยในแต่ละกรณีจะทำการจับสัญญาณอย่างละ 5 ครั้ง ตัวอย่างของสัญญาณการสะเทือนแสดงได้ดังภาพประกอบ 4-5 ถึงภาพประกอบ 4-14



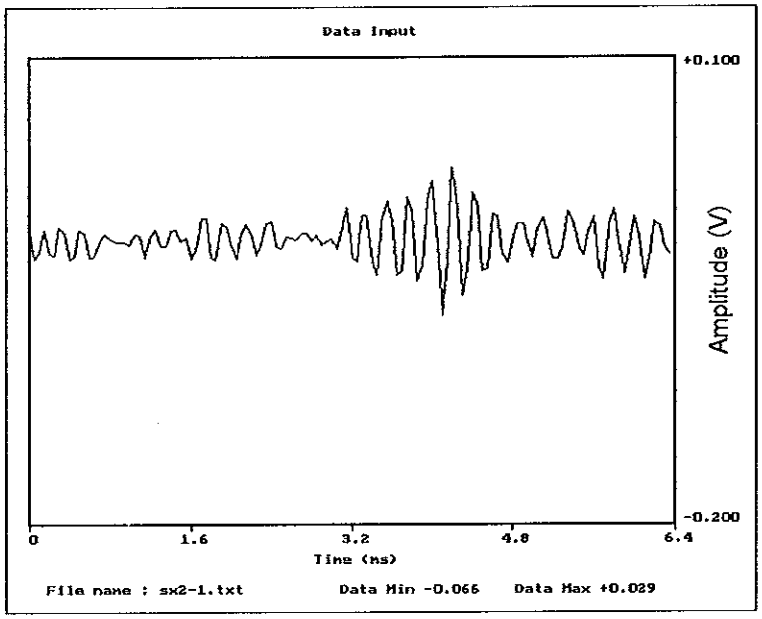
ภาพประกอบ 4-5 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์ปกติที่ความเร็วรอบ 1800 รอบ/นาที



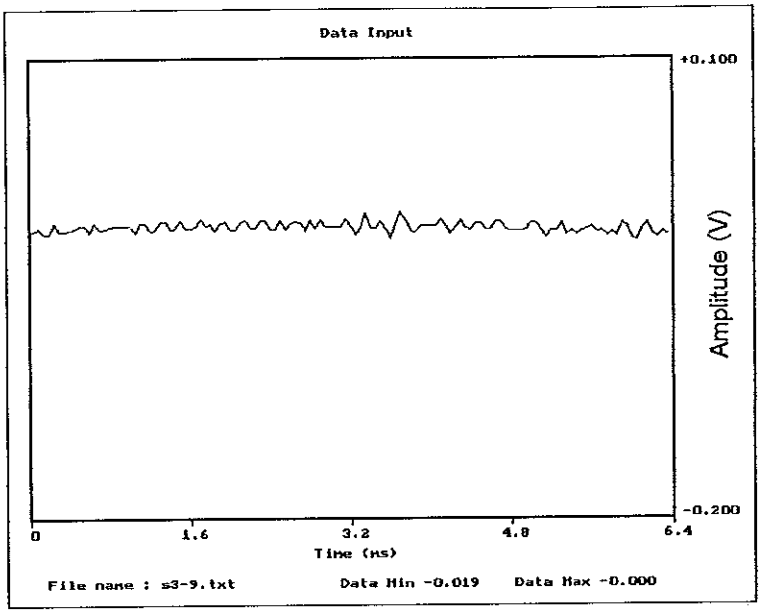
ภาพประกอบ 4-6 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อค
ที่ความเร็วรอบ 1800 รอบ/นาที



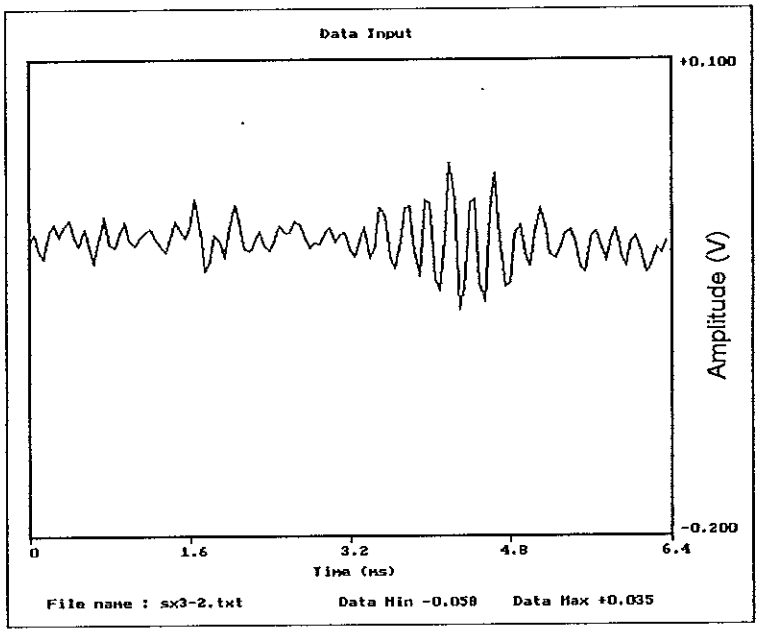
ภาพประกอบ 4-7 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์ปกติที่ความเร็วรอบ 2000 รอบ/นาที



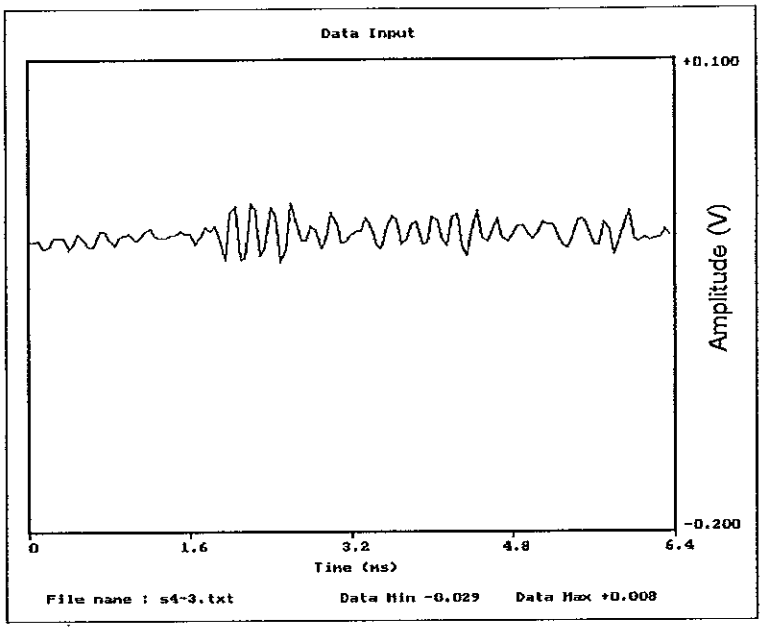
ภาพประกอบ 4-8 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อค
ที่ความเร็วรอบ 2000 รอบ/นาที



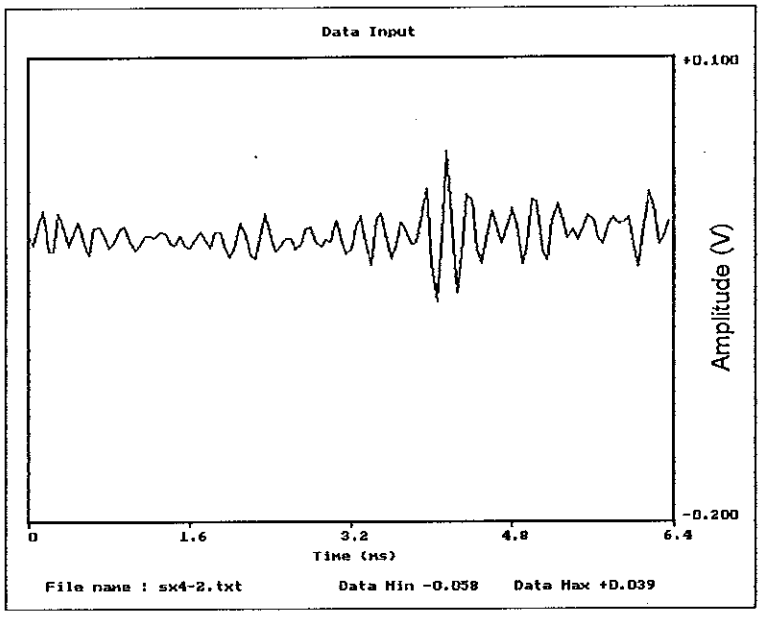
ภาพประกอบ 4-9 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์ปกติที่ความเร็วรอบ 2200 รอบ/นาที



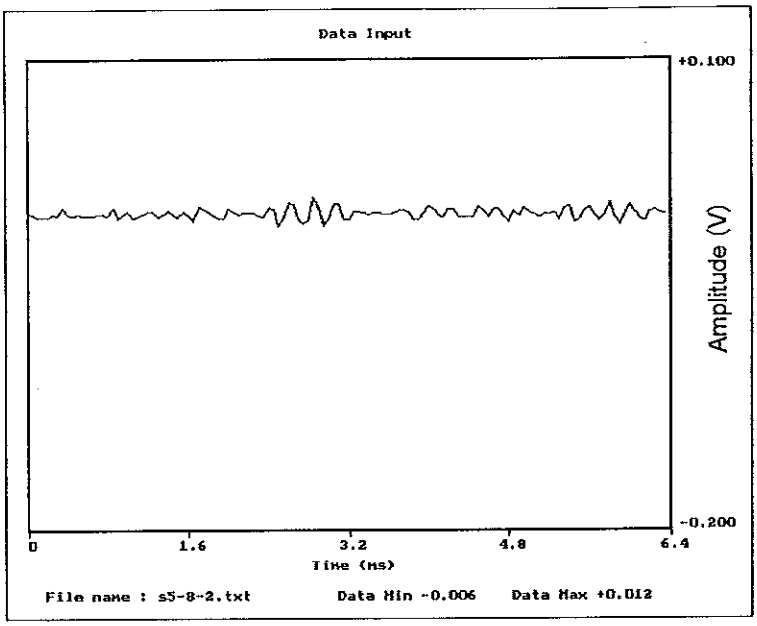
ภาพประกอบ 4-10 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อค
ที่ความเร็วรอบ 2200 รอบ/นาที



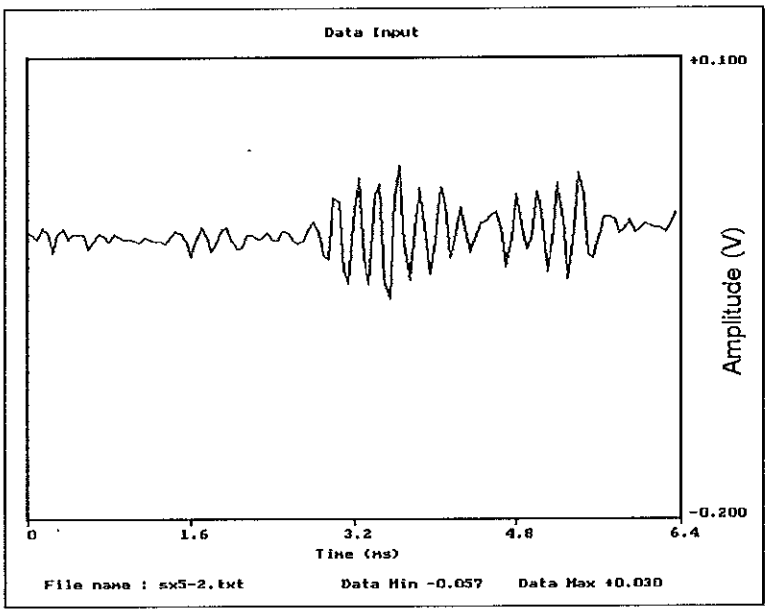
ภาพประกอบ 4-11 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์ปกติที่ความเร็วรอบ 2400 รอบ/นาที



ภาพประกอบ 4-12 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อค
ที่ความเร็วรอบ 2400 รอบ/นาที



ภาพประกอบ 4-13 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์ปกติที่ความเร็วรอบ 2600 รอบ/นาที

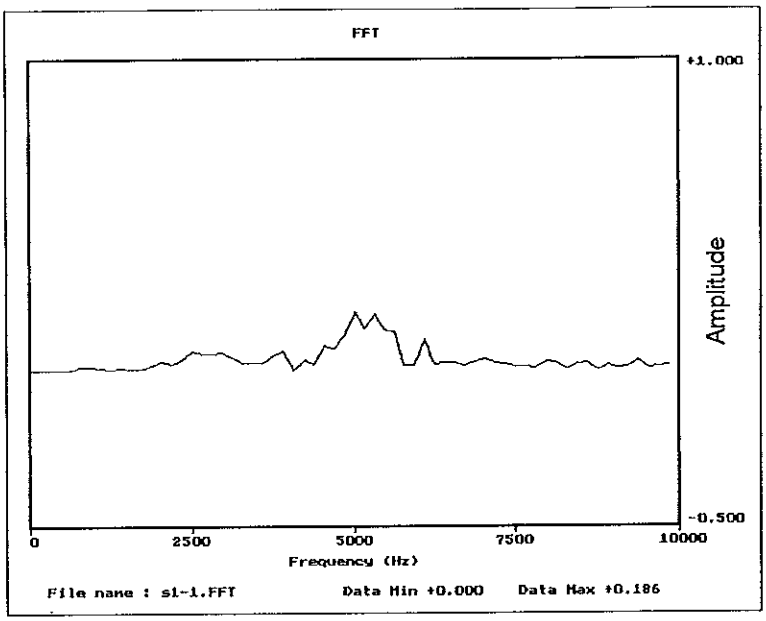


ภาพประกอบ 4-14 แสดงสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อค
ที่ความเร็วรอบ 2600 รอบ/นาที

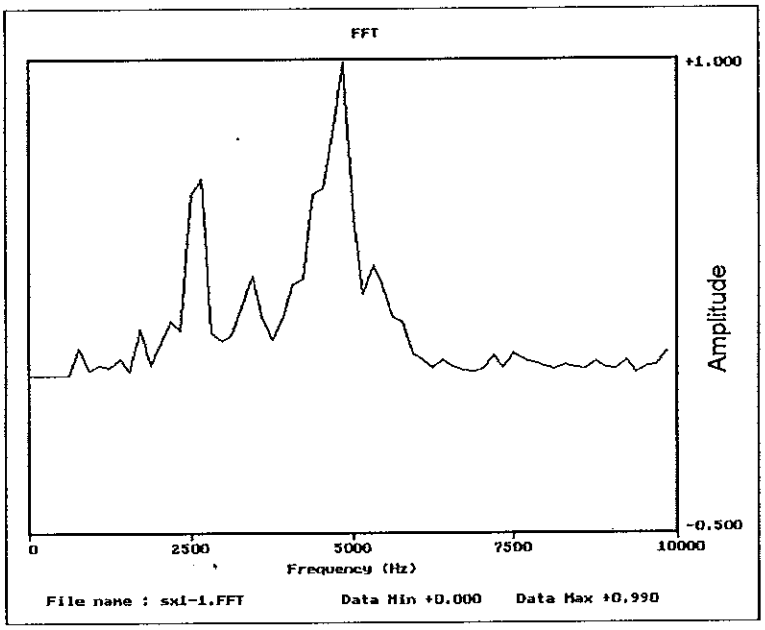
3. ผลการวิเคราะห์สัญญาณโดยการใช้ FFT

การวิเคราะห์สัญญาณโดยการใช้ FFT เป็นการนำสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่จับมาได้นำมาจำนวน 128 จุด หรือเป็นเวลา 6.4 มิลลิวินาที จากตำแหน่งเริ่มต้นของการจุดระเบิด แล้วนำข้อมูลนั้นไปทำการวิเคราะห์ด้วยโปรแกรมFFT เพื่อจะแสดงข้อมูลในโดเมนความถี่ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ในกรณีปกติ และในกรณีที่มีการน็อค โดยทำการแปลงสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่ความเร็วรอบ 1800, 2000, 2200, 2400 และ 2600 รอบ/นาที ในแต่ละกรณีจะมีข้อมูลจำนวน 5 ชุด

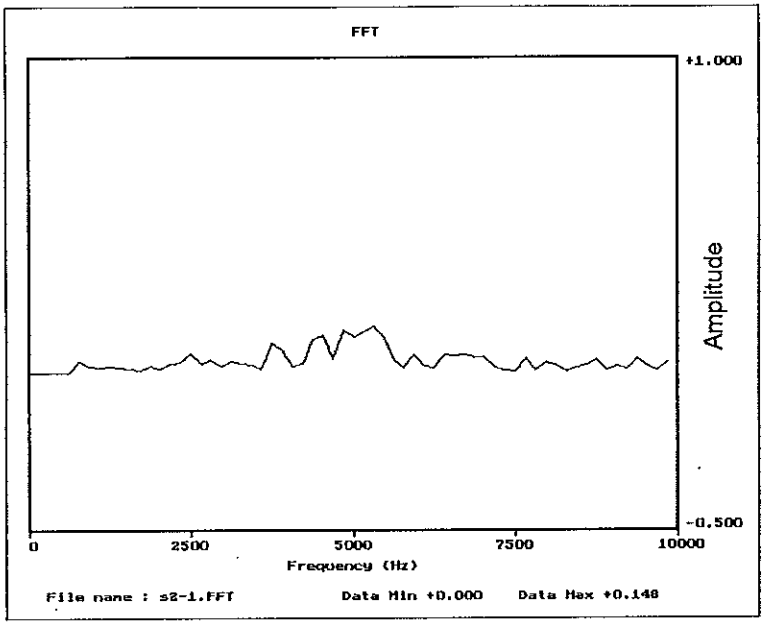
ผลการทำFFTจะได้เป็นค่าแอมพลิจูดของความถี่จาก 0 เฮิรตซ์ ถึง 10 กิโลเฮิรตซ์ ดังตัวอย่างในภาพประกอบ 4-16 ถึง 4-25



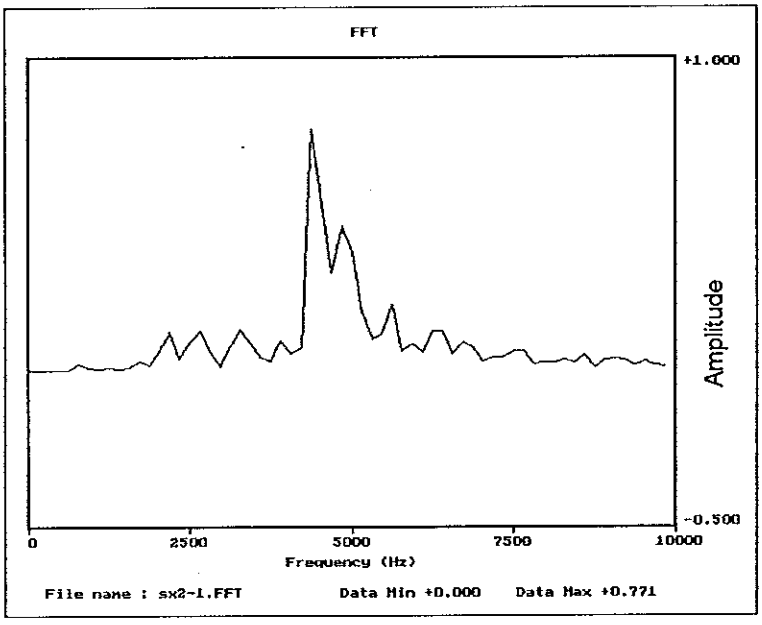
ภาพประกอบ 4-15 แสดงแอมพลิจูดจากการทำ FFT ของสัญญาณการสั่นสะเทือน
ของเครื่องยนต์ปกติที่ความเร็วรอบ 1800 รอบ/นาที



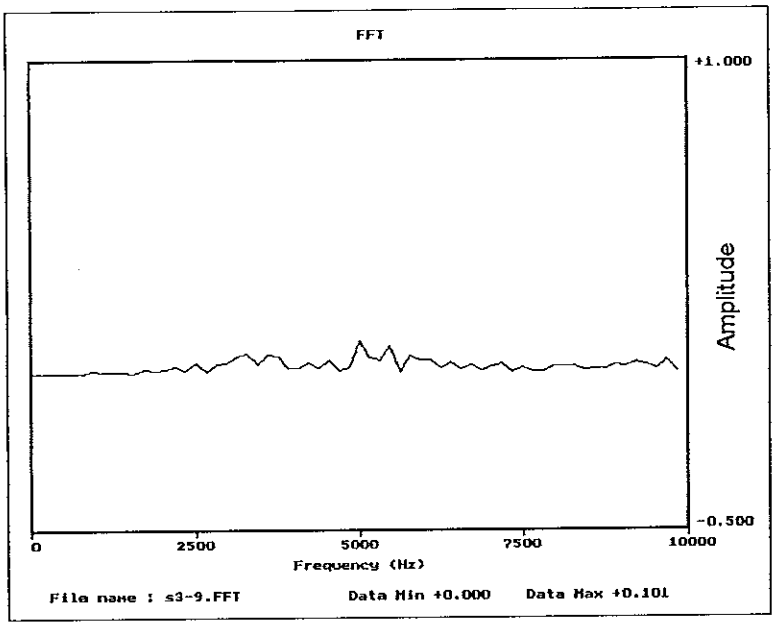
ภาพประกอบ 4-16 แสดงแอมพลิจูดจากการทำ FFTของสัญญาณการสั่นสะเทือน
ของเครื่องยนต์ที่เกิดการน็อคที่ความเร็วรอบ 1800 รอบ/นาที



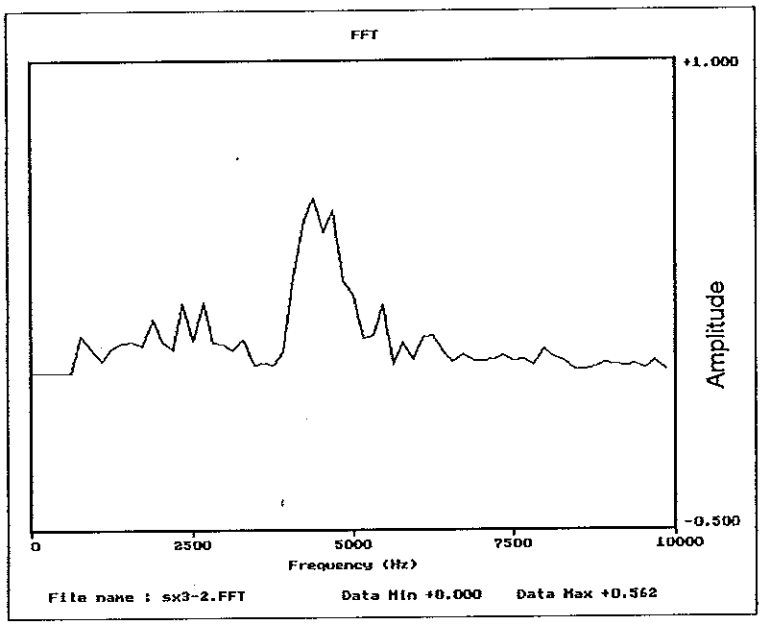
ภาพประกอบ 4-17 แสดงแอมพลิจูดจากการทำ FFT ของสัญญาณการสั่นสะเทือน
ของเครื่องยนต์ปกติที่ความเร็วรอบ 2000 รอบ/นาที



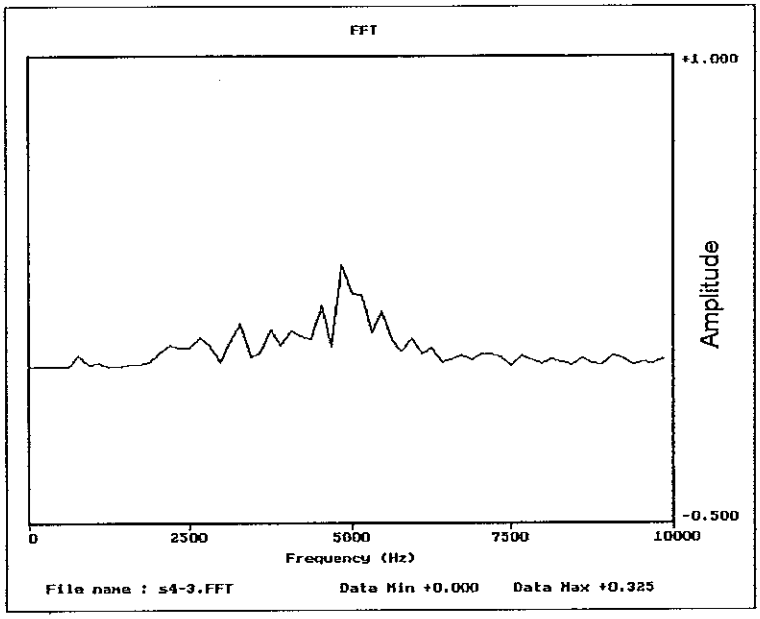
ภาพประกอบ 4-18 แสดงแอมพลิจูดจากการทำ FFTของสัญญาณการสั่นสะเทือน
ของเครื่องยนต์ที่เกิดการน็อคที่ความเร็วรอบ 2000 รอบ/นาที



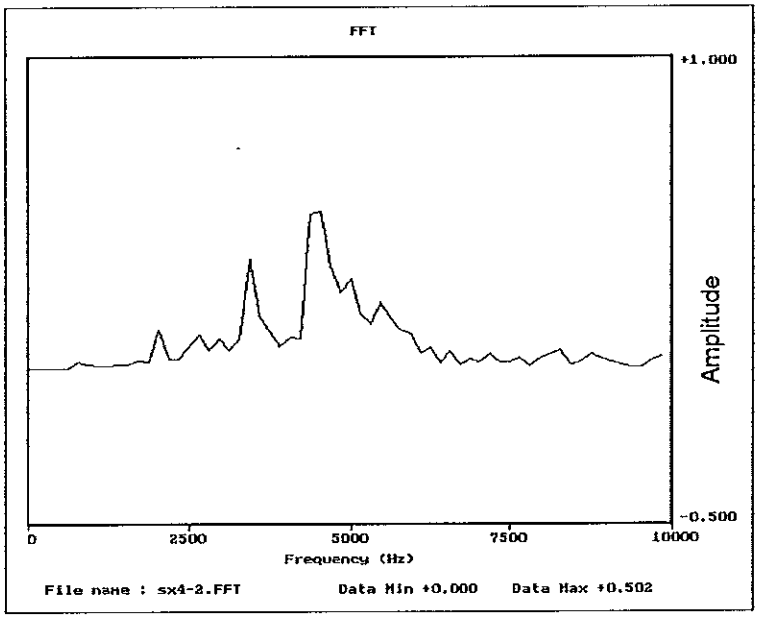
ภาพประกอบ 4-19 แสดงแอมพลิจูดจากการทำ FFT ของสัญญาณการสั่นสะเทือน
ของเครื่องยนต์ปกติที่ความเร็วรอบ 2200 รอบ/นาที



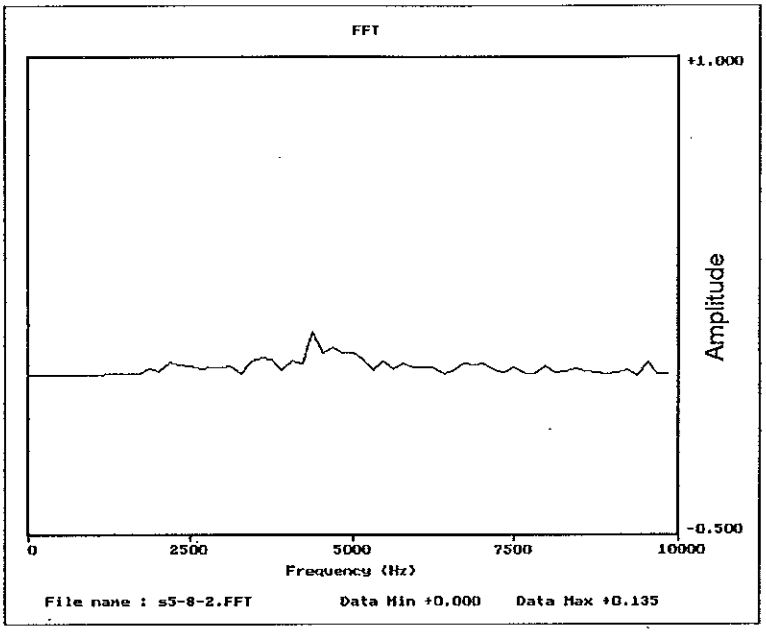
ภาพประกอบ 4-20 แสดงแอมพลิจูดจากการทำ FFTของสัญญาณการสั่นสะเทือน
ของเครื่องยนต์ที่เกิดการน็อคที่ความเร็วรอบ 2200 รอบ/นาที



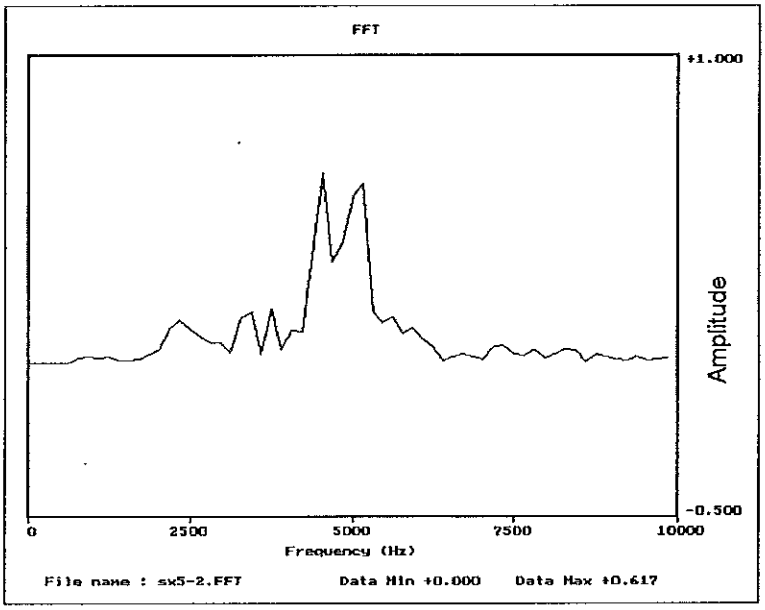
ภาพประกอบ 4-21 แสดงแอมพลิจูดจากการทำ FFT ของสัญญาณการสั่นสะเทือน
ของเครื่องยนต์ปกติที่ความเร็วรอบ 2400 รอบ/นาที



ภาพประกอบ 4-22 แสดงแอมพลิจูดจากการทำ FFTของสัญญาณการสั่นสะเทือน
ของเครื่องยนต์ที่เกิดการน็อคที่ความเร็วรอบ 2400 รอบ/นาที



ภาพประกอบ 4-23 แสดงแอมพลิจูดจากการทำ FFT ของสัญญาณการสั่นสะเทือน
ของเครื่องยนต์ปกติที่ความเร็วรอบ 2600 รอบ/นาที



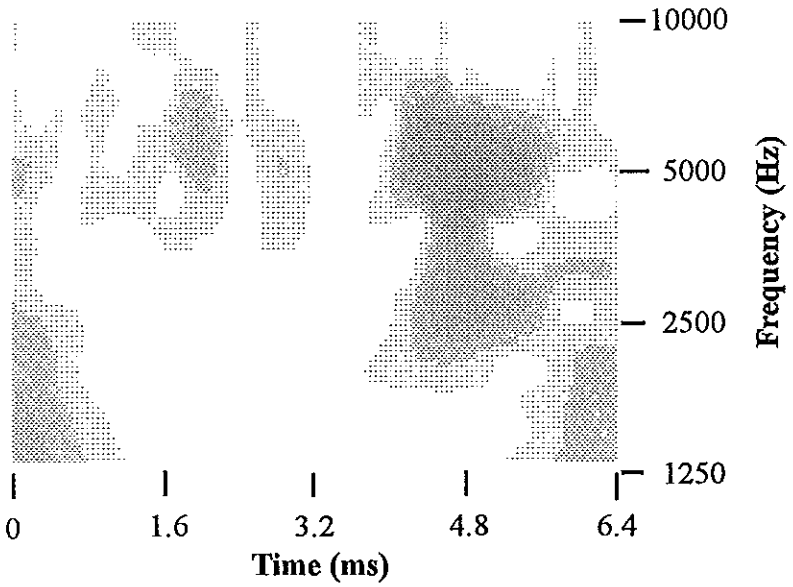
ภาพประกอบ 4-24 แสดงแอมพลิจูดจากการทำ FFTของสัญญาณการสั่นสะเทือน
ของเครื่องยนต์ที่เกิดการน็อคที่ความเร็วรอบ 2600 รอบ/นาที

ผลการวิเคราะห์โดยใช้ FFT นำมาพิจารณาเลือกช่วงความถี่ที่มีแอมพลิจูดของสเปคตรัมจากสัญญาณการสั่นสะเทือนที่เกิดการน็อคพบว่ามีค่ามากกว่าสัญญาณการสั่นสะเทือนที่ปกติในช่วงความถี่ 4,375 เฮิรตซ์ ถึง 6,250 เฮิรตซ์ ดังนั้นจะนำความถี่ในช่วงนี้มาใช้ในการวิเคราะห์สัญญาณต่อไป

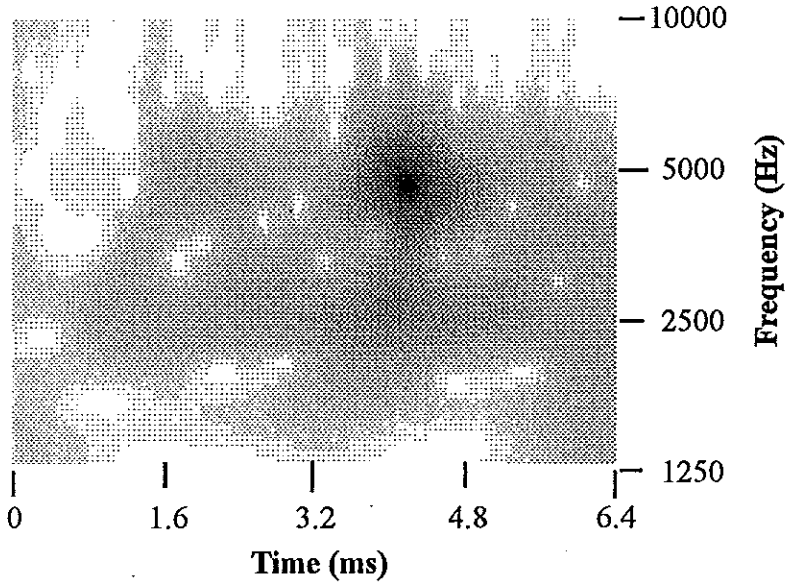
4. ผลการวิเคราะห์สัญญาณโดยใช้ DWT

การวิเคราะห์สัญญาณโดยวิธี DWT เป็นการเขียนโปรแกรมมาทำการวิเคราะห์สัญญาณการสั่นสะเทือน เพื่อจะแสดงข้อมูลในโดเมนความถี่ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ในกรณีปกติ และในกรณีที่มีการน็อค โดยทำการแปลงสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่ความเร็วรอบ 1800, 2000, 2200, 2400 และ 2600 รอบ/นาที ในกรณีจะมีข้อมูลจำนวน 5 ชุด

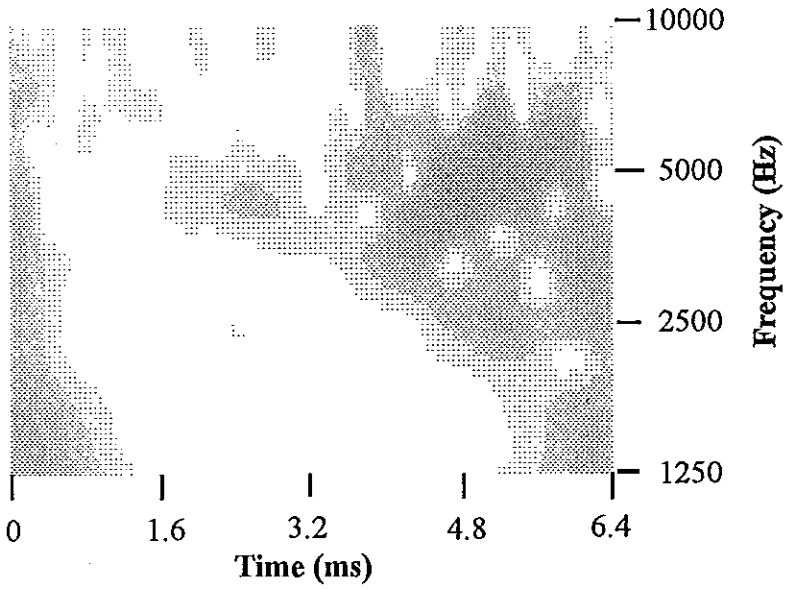
ผลที่ได้จะได้เป็นค่าแอมพลิจูดในรูปแบบของระดับสีเทา(gray level) ที่มีสีดำเป็นค่ามาก สีขาวมีค่าน้อย ของข้อมูลที่นำมาแปลงในช่วงความถี่ที่ต้องการระหว่าง 1,250 เฮิรตซ์ ถึง 10,000 เฮิรตซ์ ดังตัวอย่างในภาพประกอบ 4-27 ถึง 4-36



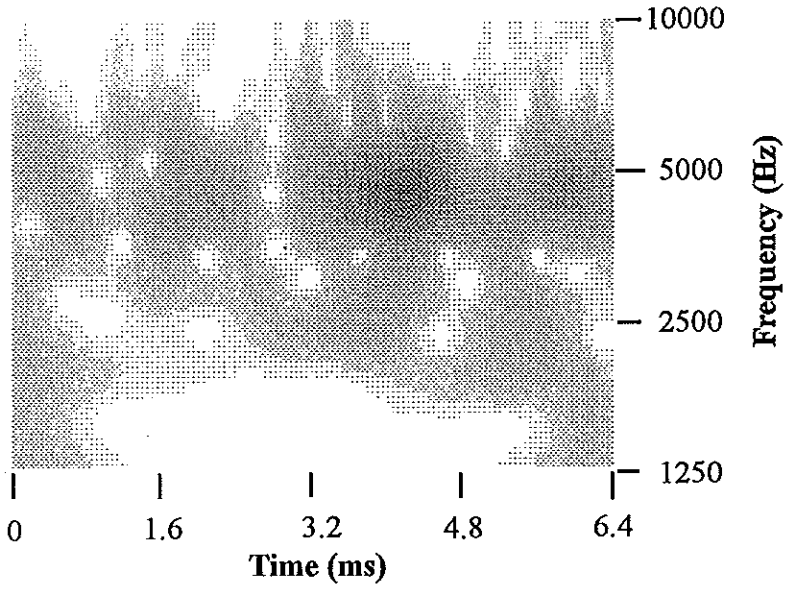
ภาพประกอบ 4-25 แสดงแอมพลิจูดจากการทำ DWT ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่ปกติที่ความเร็วรอบ 1800 รอบ/นาที



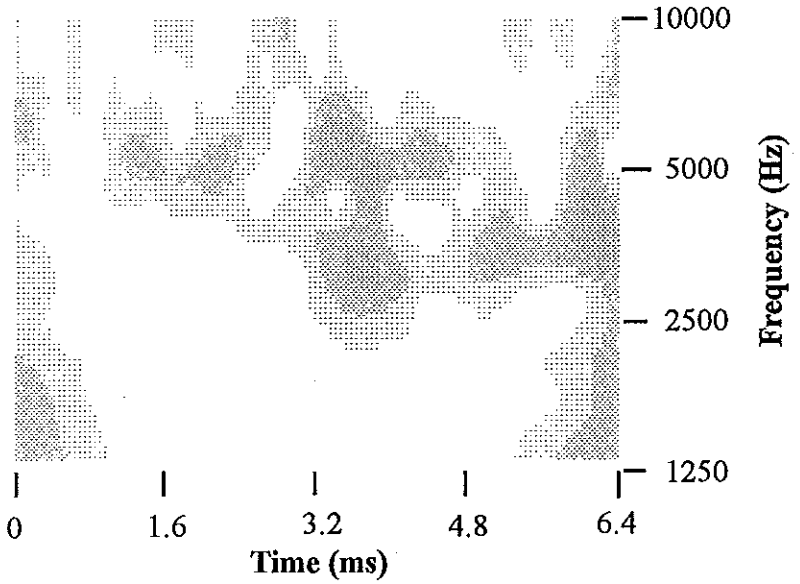
ภาพประกอบ 4-26 แสดงแอมพลิจูดจากการทำ DWT ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อคที่ความเร็วรอบ 1800 รอบ/นาที



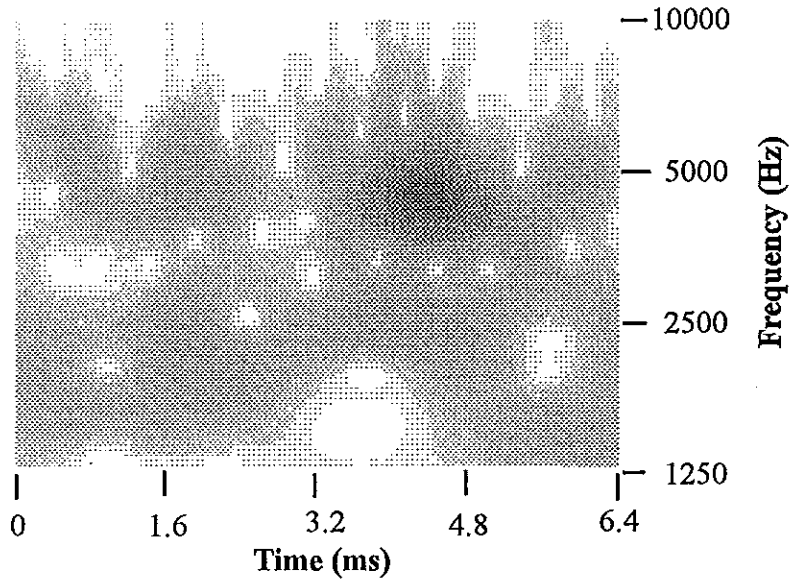
ภาพประกอบ 4-27 แสดงแอมพลิจูดจากการทำ DWT ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่ปกติที่ความเร็วรอบ 2000 รอบ/นาที



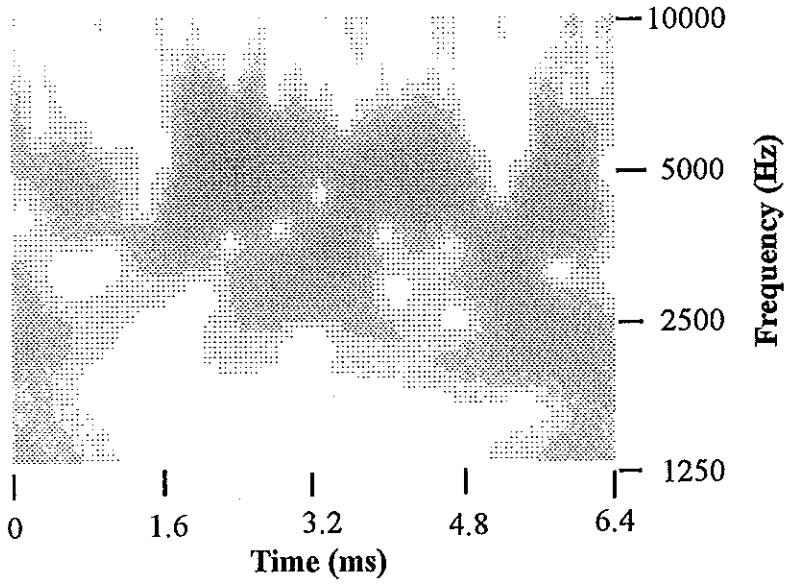
ภาพประกอบ 4-28 แสดงแอมพลิจูดจากการทำ DWT ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อคที่ความเร็วรอบ 2000 รอบ/นาที



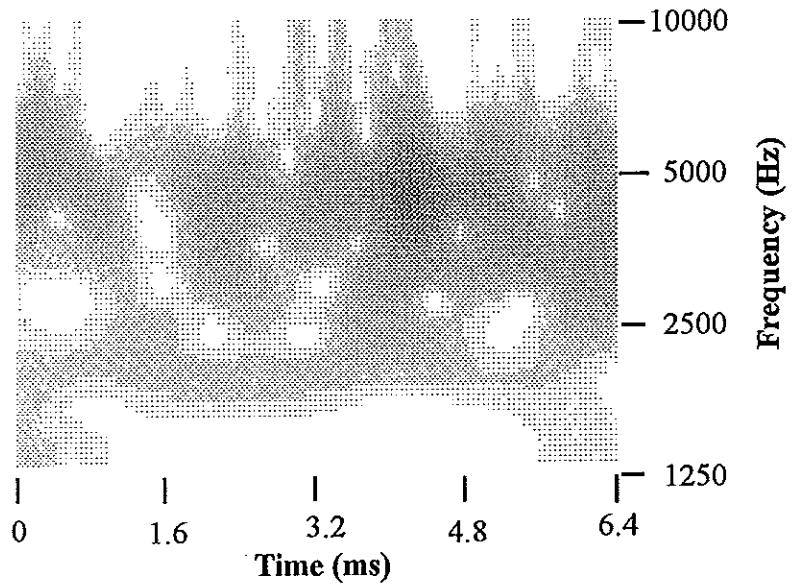
ภาพประกอบ 4-29 แสดงแอมพลิจูดจากการทำ DWT ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่ปกติที่ความเร็วรอบ 2200 รอบ/นาที



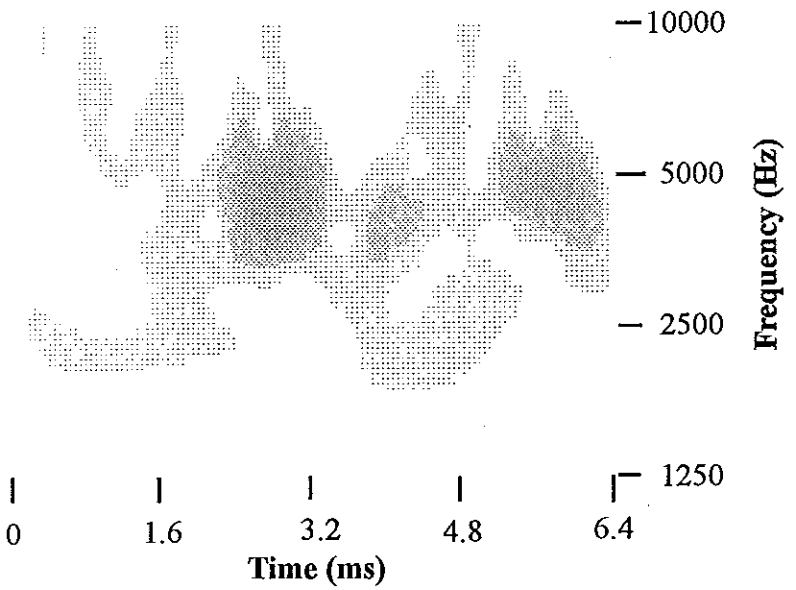
ภาพประกอบ 4-30 แสดงแอมพลิจูดจากการทำ DWT ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อคที่ความเร็วรอบ 2200 รอบ/นาที



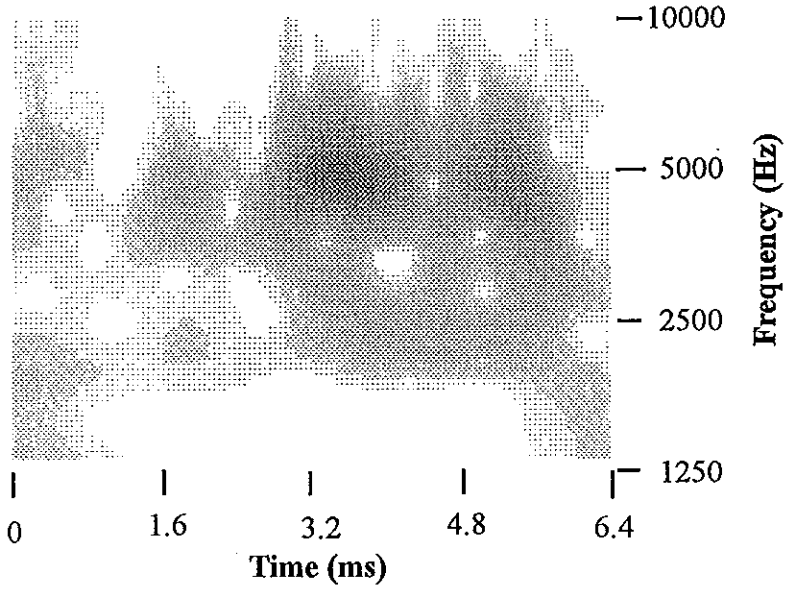
ภาพประกอบ 4-31 แสดงแอมพลิจูดจากการทำ DWT ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่ปกติที่ความเร็วรอบ 2400 รอบ/นาที



ภาพประกอบ 4-32 แสดงแอมพลิจูดจากการทำ DWT ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อคที่ความเร็วรอบ 2400 รอบ/นาที



ภาพประกอบ 4-33 แสดงแอมพลิจูดจากการทำ DWT ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่ปกติที่ความเร็วรอบ 2600 รอบ/นาที



ภาพประกอบ 4-34 แสดงแอมพลิจูดจากการทำ DWT ของสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อคที่ความเร็วรอบ 2600 รอบ/นาที

ผลการวิเคราะห์จากวิธี DWT พบว่าความถี่ในช่วง 4,352 เฮิรตซ์ ถึง 6,372 เฮิรตซ์ มีค่าแตกต่างกันที่สามารถแยกความแตกต่างระหว่างสัญญาณการน็อคและสัญญาณปกติได้ จึงได้นำค่าแอมพลิจูดของสัญญาณจากวิธี DWT ในช่วงความถี่นี้ ไปคำนวณต่อไปด้วยวิธีการหาค่าเฉลี่ยต่อไป

4. ผลการวิเคราะห์โดยการหาค่าเฉลี่ยจากแอมพลิจูดของ DWT

การวิเคราะห์เป็นการนำค่าแอมพลิจูดของสัญญาณจากวิธี DWT ในช่วง ความถี่ 4,352 เฮิรตซ์ ถึง 6,372 เฮิรตซ์ มาหาค่าเฉลี่ยตามสมการต่อไปนี้

$$\bar{x} = [\sum_{i=1}^{N} x_i \ ]/N$$

เมื่อ กำหนดให้

$\bar{x}$ คือ ค่าเฉลี่ย

x_i คือ แอมพลิจูดของสัญญาณจากวิธี DWT

N คือ จำนวนข้อมูล

หาค่าเฉลี่ยของสัญญาณที่ทำการทดลองของเครื่องยนต์ที่ความเร็วรอบ 1800, 2000, 2200, 2400 และ 2600 รอบ/นาที เมื่อเครื่องยนต์อยู่ในสภาพปกติและ เครื่องยนต์เกิดการน็อค โดยในแต่ละกรณีจะหาค่าเฉลี่ยอย่างละ 5 ครั้ง ได้ผลดังตารางที่ 4-1

ตาราง 4-1 แสดงค่าเฉลี่ยที่ได้จากแอมพลิจูดของ DWT

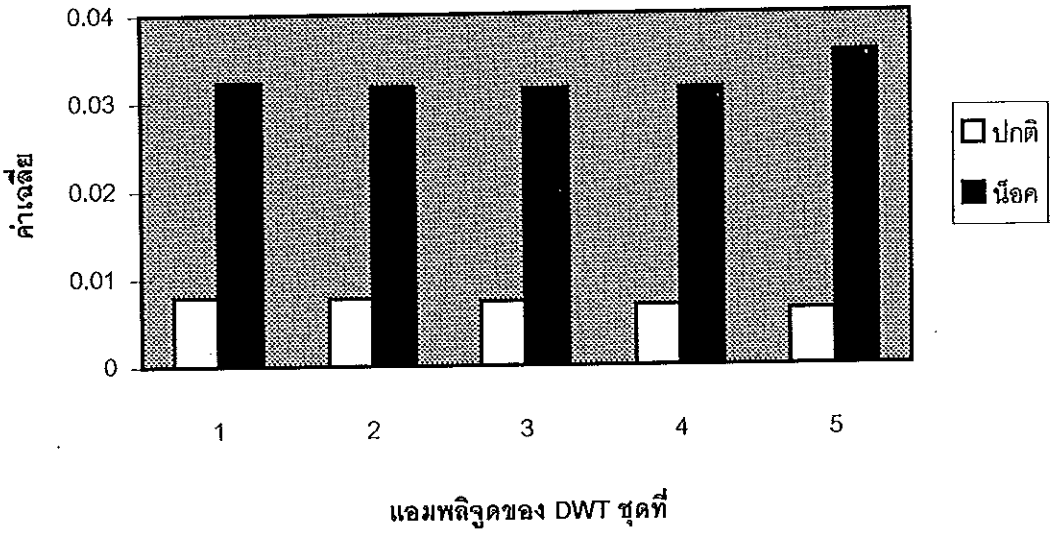
การวัดสัญญาณเครื่องยนต์		ค่าเฉลี่ยจากแอมพลิจูดของ DWT	
ความเร็วรอบ(รอบ/นาที)	ครั้งที่	ปกติ	น็อค
1800	1	0.007921	0.032327
	2	0.007812	0.031776
	3	0.00739	0.031528
	4	0.006871	0.031528
	5	0.006376	0.035649
2000	1	0.007656	0.033628
	2	0.006811	0.027011

ตาราง 4-1 (ต่อ)

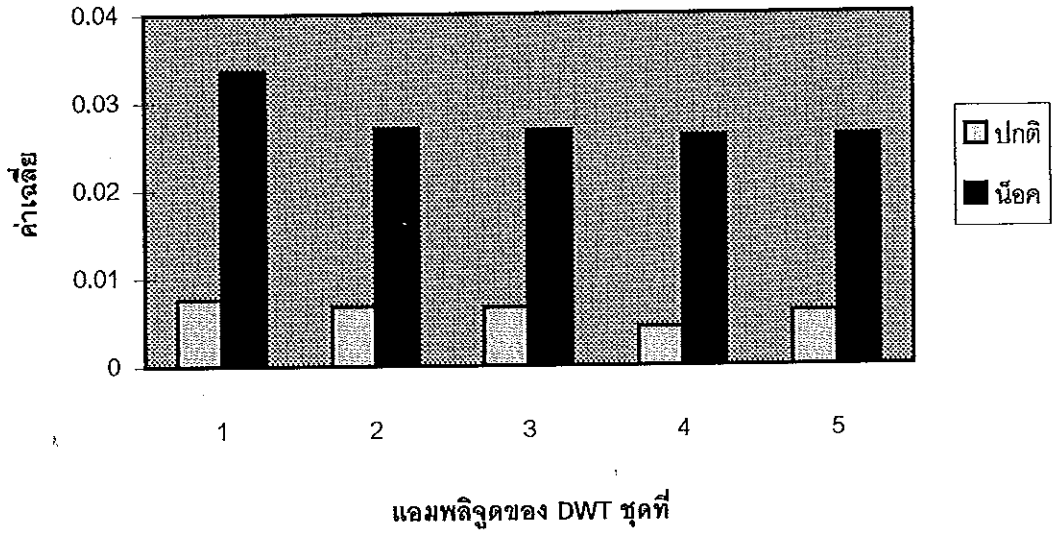
การวัดสัญญาณเครื่องยนต์		ค่าเฉลี่ยจากแอมพลิจูดของ DWT	
ความเร็วรอบ(รอบ/นาที)	ครั้งที่	ปกติ	น็อค
2000	3	0.006666	0.02678
	4	0.00446	0.026132
	5	0.006141	0.026162
2200	1	0.004123	0.024363
	2	0.003917	0.023433
	3	0.003608	0.019288
	4	0.003843	0.019295
	5	0.003336	0.021029
2400	1	0.012793	0.019462
	2	0.013777	0.025291
	3	0.011053	0.021294
	4	0.013376	0.02252
	5	0.014337	0.024072
2600	1	0.006025	0.019684
	2	0.004403	0.026001
	3	0.003499	0.025426
	4	0.003769	0.024367
	5	0.004043	0.025522

ตาราง 4-1 แสดงค่าเฉลี่ยที่ได้จากแอมพลิจูดของ DWT

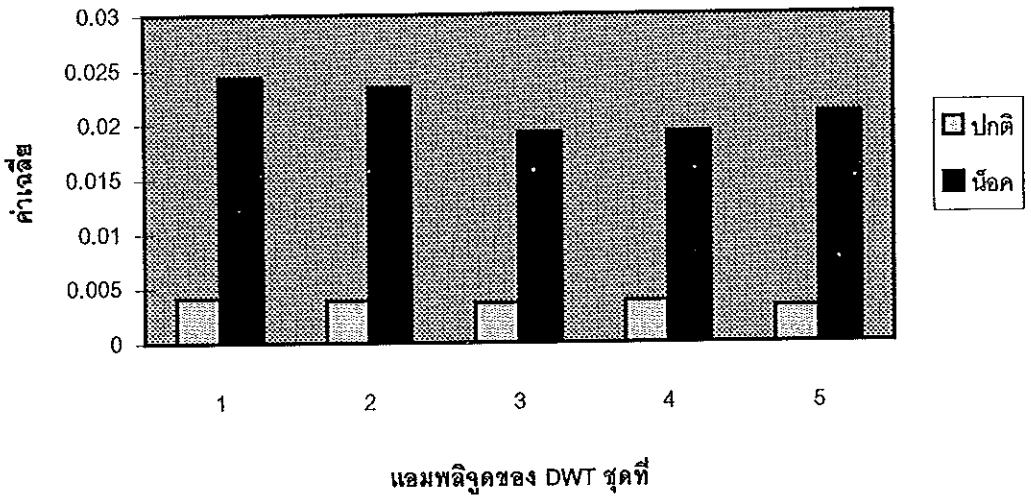
เมื่อนำค่าเฉลี่ยจากตาราง 4-1 มาวาดกราฟแสดงการเปรียบเทียบค่าเฉลี่ย DWT ระหว่างสัญญาณปกติและสัญญาณที่มีการน็อคที่ความเร็วรอบ 1800 , 2000, 2200, 2400 และ 2600 รอบ/นาที จะได้ผลดังภาพประกอบที่ 4-35 ถึง 4-39ตามลำดับ



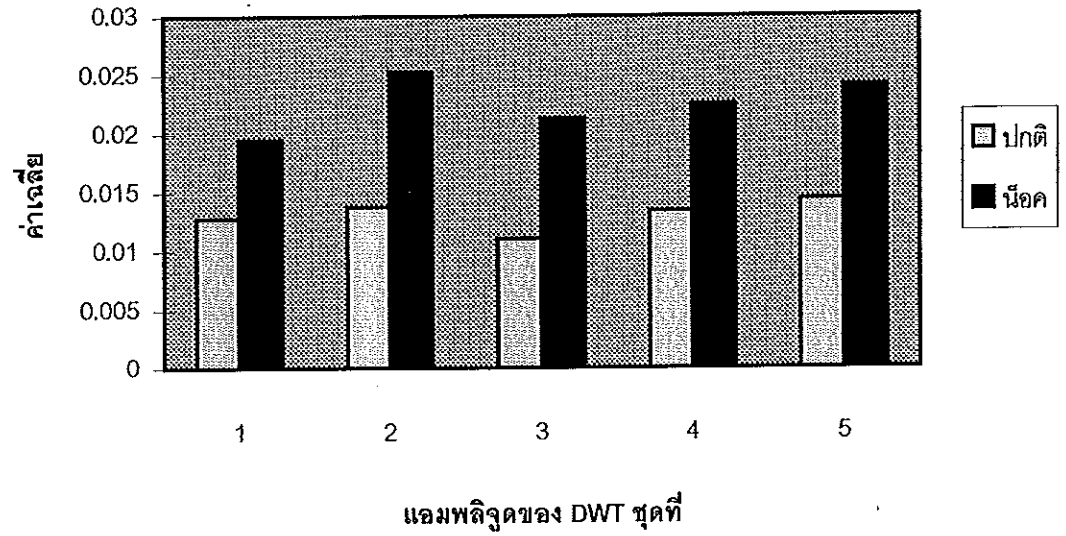
ภาพประกอบ 4-35 แสดงการเปรียบเทียบค่าเฉลี่ย DWT ระหว่างสัญญาณปกติและสัญญาณที่มีการน็อคที่ความเร็วรอบ 1800 รอบ/นาที



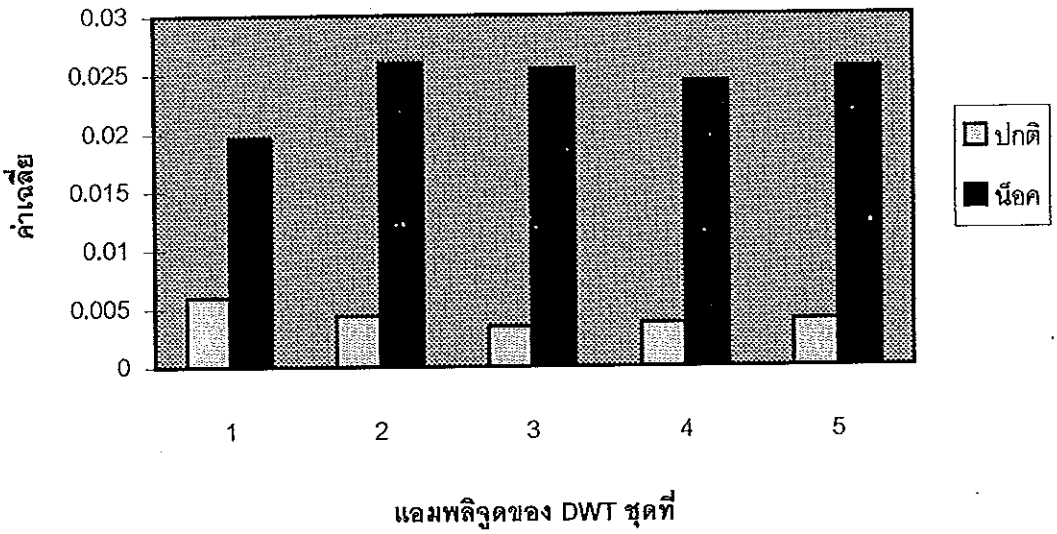
ภาพประกอบ 4-36 แสดงการเปรียบเทียบค่าเฉลี่ย DWT ระหว่างสัญญาณปกติและสัญญาณที่มีการน็อคที่ความเร็วรอบ 2000 รอบ/นาที



ภาพประกอบ 4-37 แสดงการเปรียบเทียบค่าเฉลี่ย DWT ระหว่างสัญญาณปกติและสัญญาณที่มีการน็อคที่ความเร็วรอบ 2200 รอบ/นาที

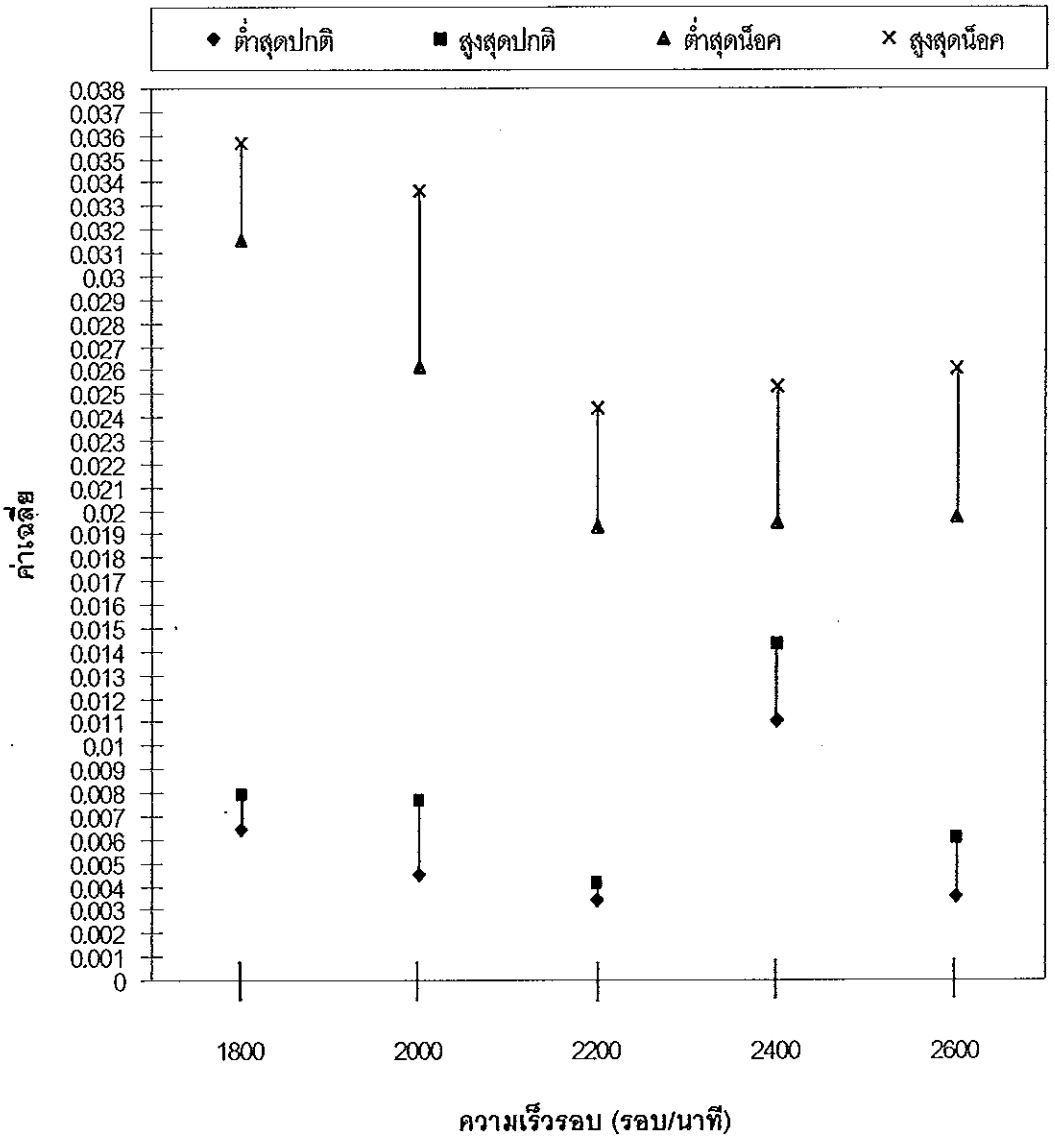


ภาพประกอบ 4-38 แสดงการเปรียบเทียบค่าเฉลี่ย DWT ระหว่างสัญญาณปกติและสัญญาณที่มีการน็อคที่ความเร็วรอบ 2400 รอบ/นาที



ภาพประกอบ 4-39 แสดงการเปรียบเทียบค่าเฉลี่ย DWT ระหว่างสัญญาณปกติและสัญญาณที่มีการน็อคที่ความเร็วรอบ 2600 รอบ/นาที

จากภาพประกอบ 4-35 ถึง 4-39 ผลที่ได้จะเห็นว่าค่าเฉลี่ย DWT ที่ได้จากสัญญาณที่มีการน็อคจะมีค่าสูงกว่า ค่าเฉลี่ย DWT ของสัญญาณปกติ ซึ่งสามารถนำมาตรวจหาสัญญาณสั่นสะเทือนที่มีการน็อคของเครื่องยนต์ได้ โดยเมื่อพิจารณาจากค่าเฉลี่ยสูงสุดและต่ำสุดทั้งจากสัญญาณปกติและสัญญาณที่มีการน็อค ในทุกความเร็วรอบที่จับสัญญาณมาวาดกราฟ ได้ดังภาพประกอบ 4-40



ภาพประกอบ 4-40 แสดงกราฟค่าเฉลี่ยจากแอมพลิจูดของ DWTสูงสุดและต่ำสุดทั้งจากสัญญาณปกติและสัญญาณที่มีการน็อค

จากกราฟในภาพประกอบ 4-40 จะได้ว่าค่าเฉลี่ย DWTของสัญญาณที่มีการน็อคจะมีค่ามากกว่าค่าเฉลี่ย DWT ของสัญญาณปกติโดย สามารถแยกได้จากค่าเฉลี่ยที่มีค่ามากกว่า 0.019

5. สรุป

ในการวิเคราะห์สัญญาณเพื่อตรวจสอบการน็อคของเครื่องยนต์โดยการใช้สัญญาณการสั่นสะเทือนนี้ ทำโดยการวัดสัญญาณการสัญญาณการสั่นสะเทือนของเครื่องยนต์เพื่อเก็บข้อมูลในโดเมนเวลาโดยเก็บสัญญาณการสั่นสะเทือนของเครื่องยนต์ปกติและสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อค ในความเร็วรอบ 1800 , 2000 , 2200 , 2400 และ 2600 รอบ/นาที แล้วแปลงข้อมูลให้เป็นข้อมูลในโดเมนความถี่โดยการใช้ FFT พบว่าความถี่ในช่วง4,375 เฮิรตซ์ ถึง 6,250 เฮิรตซ์ มีค่าที่ได้จากการแปลงในกรณีเครื่องยนต์เกิดการน็อคมากกว่ากรณีเครื่องยนต์ปกติ เมื่อนำไปวิเคราะห์โดยใช้ DWT พบว่าความถี่ 4,352 เฮิรตซ์ ถึง 6,372 เฮิรตซ์ มีค่าแตกต่างกันที่สามารถแยกความแตกต่างระหว่างสัญญาณที่มีการน็อคและสัญญาณปกติ เมื่อนำไปหาค่าเฉลี่ยของแอมพลิจูดจากวิธี DWT สามารถแยกสัญญาณการสั่นสะเทือนของเครื่องยนต์ที่เกิดการน็อคได้ที่ค่าเฉลี่ยที่มีค่ามากกว่า 0.019 ซึ่งตรงกับการตั้งสมมติฐานที่ว่าในการน็อคของเครื่องยนต์จะมีการปล่อยพลังงานออกมาทันทีทันใดของการจุดระเบิดตัวเอง เกิดเป็นคลื่นช็อค ซึ่งจะทำให้เกิดเสียงความถี่ธรรมชาติของกระบอกสูบเป็นเสียงของการน็อค โดยสังเกตุในช่วงความถี่ 4.3 กิโลเฮิรตซ์ ถึง 6.3 กิโลเฮิรตซ์ ทั้งนี้สามารถสรุปขั้นตอนการทำงานได้ดังนี้

1. ทำการจัดเตรียมและติดตั้งเครื่องมือการทดลอง
2. ทำการวัดสัญญาณการทดลองโดยใช้เครื่องจับสัญญาณ
3. ส่งข้อมูลการสั่นสะเทือนให้คอมพิวเตอร์
4. วิเคราะห์ผลสัญญาณโดยใช้วิธี FFT วิธี DWT และ
   การใช้ค่าเฉลี่ยจากแอมพลิจูดของ DWT
5. สรุปผลการวิเคราะห์สัญญาณ

6. คำเสนอแนะเพื่อดำเนินการต่อไป

จากขั้นตอนที่สรุปดังกล่าวจะเห็นว่ายังต้องมีการส่งข้อมูลผ่านระหว่างเครื่องบันทึกสัญญาณและคอมพิวเตอร์ในขั้นตอนที่ 3 ทำให้ต้องเสียเวลาเพิ่มขึ้น ซึ่งในขั้นตอนนี้สามารถรวมขั้นตอนที่ 2 และ ขั้นตอนที่ 3 เป็นขั้นตอนเดียวได้โดยใช้การบันทึกสัญญาณทางคอมพิวเตอร์โดยตรงผ่านทางแผงวงจรแปลงผันแอนะล็อกเป็นดิจิตอล (Analog to Digital converter card) จะทำให้สามารถประมวลผลสัญญาณการสั่นสะเทือนได้โดยตรง ซึ่งต้องพิจารณารายละเอียดการแปลง

สัญญาณแอนะล็อกให้เป็นดิจิตอลได้แก่การพิจารณาค่าเวลาการแปลงผัน(conversion time) เนื่องจากการแปลงสัญญาณแอนะล็อกให้เป็นดิจิตอลไม่ได้เกิดขึ้นโดยทันทีทันใดแต่ต้องใช้เวลาในการชักตัวอย่างสัญญาณเข้าและให้ผลลัพธ์เป็นสัญญาณดิจิตอลที่เป็นเลขฐานสอง(binary number) จึงต้องคำนึงถึงอัตราการชักตัวอย่าง ในการชักตัวอย่างสัญญาณให้สามารถเก็บข้อมูลในโดเมนความถี่ได้ โดยต้องมีอัตราการชักตัวอย่างของสัญญาณไม่น้อยกว่า 20000 ครั้งต่อวินาที และต้องมีความละเอียดของการเก็บสัญญาณได้อย่างน้อย 8 บิตเพื่อให้สามารถเก็บค่าข้อมูลที่ต่างกันได้เป็นค่า 256 ระดับ

อย่างไรก็ตามการวิเคราะห์ในขั้นตอนที่ 4 และ ขั้นตอนที่ 5 ต้องมีการสังเกตุเพื่อช่วยในการตัดสินด้วย จึงสามารถเพิ่มการสังเกตุโดยวิธีการทางคณิตศาสตร์อื่น ๆ ต่อไป เช่น การหาค่าจุดเริ่มเปลี่ยนเพื่อใช้ในการแยกสัญญาณปกติและสัญญาณน็อค โดยการหาการกระจายของความน่าจะเป็นของข้อมูลที่แตกต่างกัน จากข้อมูลที่ได้จากการแปลงให้อยู่ในโดเมนความถี่จะทำให้สามารถหาความแตกต่างของข้อมูลได้รวดเร็วขึ้น ดังนั้นขั้นตอนนี้เป็นแนวทางในการทำงานเพื่อนำไปใช้ศึกษาการวิเคราะห์เครื่องยนต์แก่ผู้สนใจต่อไป

บรรณานุกรม

ชูศรี วงศ์รัตนะ. 2525. เทคนิคการใช้สถิติเพื่อการวิจัย. โรงพิมพ์เจริญผล.

หลาบ รับศิริ. 2528. เครื่องยนต์เผาไหม้ภายใน. ฟิสิกส์เซ็นเตอร์ การพิมพ์.

สุจิตต์ สนองคุณ, มนตรี ขันกสิกรรม และ พิศาล ขำคม. 2531. ไฟฟ้ารถยนต์. มณีรัตน์การพิมพ์.

Les E. Atlas, Gary D. Bernard and Siva Bala Narayanan. 1996. "Applications of Time-Frequency Analysis to Signals from Manufacturing and Machine Monitoring Sensors" , Proceedings of The IEEE. (September 1996), 1319-1329

Bahman, Saminy and Giorgio, Rizzoni. 1996. "Mechanical Signature Analysis Using Time-Frequency Signal Processing: Application to Internal Combustion Engine Knock Detection" , Proceedings of The IEEE. (September 1996), 1330-1343

Daniel J. Inman. 1996. Engineering Vibration. Prentice Hall.

David S. Birkett. 1993. " Using Your PC for Function Analysis and Control " , The ComputerApplication Journal. (September 1993), 26-33

Martin, Vetterli. 1992. "Wavelets and Filter Banks: Theory and Design" , IEEE trans on Signal Processing. (September 1992), 2207-2232

Daubechies, Ingrid. 1992. Ten lectures on wavelets. Pennsylvania : Capital City Press

Mary Beth Ruskai., et al. 1992. Wavelets and their applications. Boston:Jones and Bartlett.

Hisakazu kikuchi., et al. 1992. " Fast Wavelet Transform and Its Application to Detecting Detonation " , IEEE Trans. Fundamentals. (August 1992) , 980-986

Willis, J.Tompkins and John, G.Webster. 1992. Interfacing Sensors To The IBM PC. Prentice-Hall.

Paul, M.Embree and Bruce, Kimble. 1991. C Language Algorithm for Digital Signal Processing. Prentice-Hall.

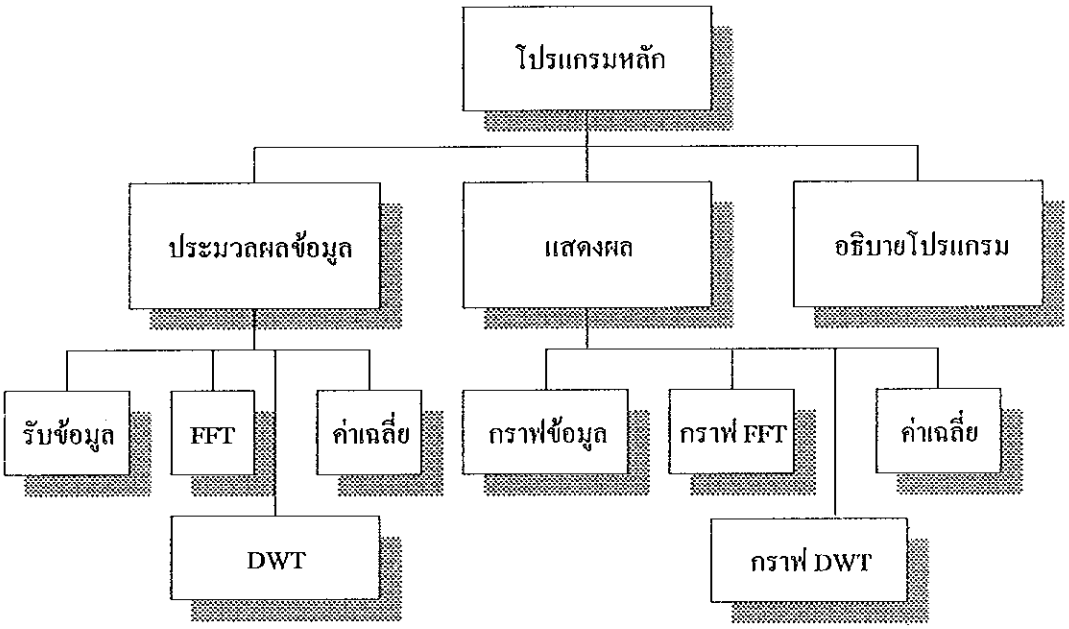
Heywood, B. John. 1988. Internal Combustion Engine Fundamentals. McGraw-Hill.

E. Ovan Brigham. 1974. The Fast Fourier Transform. Prentice-Hall.

ภาคผนวก

รายละเอียดโปรแกรมที่ใช้วิเคราะห์ข้อมูล

โปรแกรมที่เขียนขึ้นเพื่อทำการวิเคราะห์ข้อมูลโดยการรับแฟ้มข้อมูลการสั่นสะเทือนของเครื่องยนต์จากนั้นจะทำการประมวลผลด้วยวิธี FFT , DWT และการหาค่าเฉลี่ยจากแอมพลิจูดของDWT มีโครงสร้างการทำงานเป็นดังนี้



ภาพประกอบ ผ1 แสดงโครงสร้างการทำงานของโปรแกรมวิเคราะห์ข้อมูล

โปรแกรมวิเคราะห์ข้อมูลจะออกแบบให้เป็นส่วนโปรแกรมแยกกันและมีการเรียกใช้เชื่อมโยงกันตามส่วนการทำงานในโครงสร้างตามภาพประกอบ ผ1 ซึ่งสามารถแสดงส่วนของโปรแกรมหลักที่เรียกใช้ดังตาราง ผ1

ตาราง ผ1 แสดงส่วนของโปรแกรมหลักที่ใช้ในการทำงานของโปรแกรมวิเคราะห์ข้อมูล

การทำงาน	ชื่อและหน้าที่ส่วนโปรแกรมหลักที่เรียกใช้
รับข้อมูล	input_file รับชื่อแฟ้มข้อมูลเข้า input_string รับข้อความจากแป้นพิมพ์
การประมวลผล FFT	datacount นับจำนวนข้อมูลจากแฟ้มข้อมูล fexist ตรวจสอบการมีแฟ้มข้อมูลอยู่หรือไม่ readfile อ่านข้อมูลจากแฟ้มข้อมูลเก็บไว้ในหน่วยความจำ writefile เขียนข้อมูลจากหน่วยความจำลงแฟ้มข้อมูล log2 คำนวณค่าขนาดของ FFT ในรูปกำลังสองของสอง fft_main โปรแกรมควบคุมหลักในการทำ FFT fft หาค่า FFT ของข้อมูล
การประมวลผล DWT	datacount นับจำนวนข้อมูลจากแฟ้มข้อมูล fexist ตรวจสอบการมีแฟ้มข้อมูลอยู่หรือไม่ readfile อ่านข้อมูลจากแฟ้มข้อมูลเก็บไว้ในหน่วยความจำ dwritefile เขียนข้อมูลจากหน่วยความจำลงแฟ้มข้อมูล

ตาราง ผ1 (ต่อ)

การทำงาน	ชื่อและหน้าที่ส่วนโปรแกรมหลักที่เรียกใช้
การประมวลผล DWT	wavelet_main โปรแกรมควบคุมหลักในการทำDWT wavelet_transform หาค่า DWT ของข้อมูล
การหาค่าเฉลี่ย	mean_main โปรแกรมควบคุมหลักในการหาค่าเฉลี่ย average หาค่าเฉลี่ยของข้อมูล
กราฟข้อมูล	set_graph_2d กำหนดค่าเริ่มต้นในการแสดงกราฟ graph2d โปรแกรมหลักควบคุมการแสดงกราฟข้อมูล plot2d แสดงกราฟของข้อมูล
กราฟ FFT	set_graph_2d กำหนดค่าเริ่มต้นในการแสดงกราฟ graph2d โปรแกรมหลักควบคุมการแสดงกราฟข้อมูล plot2d แสดงกราฟของข้อมูล
กราฟ DWT	graph3d โปรแกรมหลักควบคุมการแสดงกราฟ DWT plot3d แสดงกราฟ DWT

ตาราง ผ1 (ต่อ)

การทำงาน	ชื่อและหน้าที่ส่วนโปรแกรมหลักที่เรียกใช้
แสดงค่าเฉลี่ย	pulldown_box แสดงกรอบรอบข้อมูล cprintf แสดงค่าข้อมูล
อธิบายโปรแกรม	say_about แสดงคำอธิบายโปรแกรม

ตาราง ผ1 แสดงส่วนของโปรแกรมหลักที่ใช้ในการทำงานของโปรแกรมวิเคราะห์ข้อมูล

สำหรับรายละเอียดต้นฉบับทั้งหมดของโปรแกรมการวิเคราะห์ได้แสดงต่อไปนี้

รหัสต้นฉบับ (source code)

```
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <alloc.h>
#include <graphics.h>
#include <ctype.h>
#include <dos.h>
#include <bios.h>

/*************************************************************************/
/*                    D E F I N E      S E C T I O N                     */
/*************************************************************************/
/*    MENU SECTION    */
#define NMAIN           3
#define NBOTTOM         1
#define FFT_EXT         "FFT"
#define WT_MAG_EXT      "MAG"
#define WT_CUT_EXT      "CUT"
#define MEAN_EXT        "MEA"

/*    WAIT SECTION    */
#define YWAIT           12
#define XWAIT           25
#define CHWAITLEN       30
#define TIMER           0X1C
#define ON              1
#define OFF             0

/*    FFT SECTION    */
#define SAMPL_RATE      20000
#define MAX(a,b)        (((a) > (b)) ? (a) : (b))
#define MIN(a,b)        (((a) < (b)) ? (a) : (b))
#define ROUND(a)        (((a) < 0) ? (int)((a)-0.5) : (int)((a)+0.5))
```

```
/*  WAVELET SECTION  */
#define PHI              3.1415927
#define ROOT10OF2        1.071773463
#define SCALE_STEP       20
#define ROOT20OF2        1.035264924
#define SCR_DRIVER       VGA
#define SCR_MODE         VGAHI
#define VGAGRAY          256
#define BEGINSCALE       4
#define ENDSCALE         1

/*    MEAN  SECTION  */
#define FBEGIN           13
#define FEND             24

/*    GRAPH SECTION  */
#define GX_START         35
#define GY_START         450
#define WINWIDE          550
#define WINHIGH          370
#define GMIN_DATA        -0.2
#define GMAX_DATA        0.1
#define GMIN_FFT         -0.5
#define GMAX_FFT         1.0
#define GDRIVER_DIR      "C:\\TC\\BGI"

/*****************************************************************************/
/*                    D A T A   T Y P E   S E C T I O N                      */
/*****************************************************************************/
typedef struct {
   float  real;
   float  imag;
} COMPLEX;

typedef struct  {
 int x,y;
 int w,h;
} WINDOW;
```

```
typedef struct{
 double x,y;
 double w,h;
} GRAPH;

/*****************************************************************************/
/*                    P R O T O T Y P E     S E C T I O N                    */
/*****************************************************************************/
/* MENU SECTION */
void main_menu(void);
void textwindow(int x1,int y1,int x2,int y2,int fcolor,int bcolor);
void check_input(char input_key);
void printfxy(int x, int y, char *s, int color);
void top_menu(void);
void shown(int ,int ,char *);
void pulldown(void);
void pulldown_box(int x1,int y1,int x2,int y2,int fcolor,int bcolor);
void writexy(int x, int y, char *s, int attr);
void clear(void);
void pulldown_sel(void);
void select_pull(void);
void write_head(void);
void write_bottom(void);
int  input_file(void);
void say_about(void);
void add_file_ext(char *fin, char *fout, char *ext);
int  input_string(char *message, int namelenmax);

/* WAIT SECTION */
int  error_msg(char *msg);
void interrupt (*oldtimer)();
void interrupt newtimer();
void wait_on(void);
void wait_off(void);
void soundbeep(void);
void set_cursor(int status);
```

```
/* FFT SECTION */
void fft(COMPLEX *x, int fft_size);
void fft_main(void);
int  log2(unsigned int x);
int  datacount(char *fin);
int  fexist(char *fin);
int  readfile(char *fin, float *data, int npts);
void writefile(char *fout, float *data, int npts);

/* WAVELET SECTION */
void dwritefile(char *fout, float far **data, int npts, int nfreq);
void wavelet_transform(float *x, int npts, int beginscale, int endscale,
     float far **Smag);
void wavelet_main(void);

/* MEAN SECTION */
void  mean_main(void);
float average(float *data, int insize);

/* GRAPH SECTION */
void set_graph_2d(float far *data1, int datacount, int type);
void plot2d(float far *data, int type); /* type==0 data , type ==1 fft */
void graph2d(int type);
int  plot3d(int xsize, int ysize, float far **data);
void graph3d(void);

/*****************************************************************************/
/*              G LO BAL  VA R I A B L E   S E C T I O N                     */
/*****************************************************************************/
/* MENU SECTION */
char Filein[13];
int  Fexist=0;
int  Cnt=0;
char *Menu_down[NBOTTOM*2]={"ALT-X","-Exit"};
char *Menu_top[NMAIN*2]={"P","rocess","R","esult","A","bout"};
int  Menu_pos[NMAIN]={4,14,23};  /* calculate by item-size+3 to start position */
char Full_filein[13];
```

```
/* WAIT SECTION */
volatile unsigned char Chpos = 0;

/* MEAN SECTION */
int    Filelen;
int    Nfreq;
float  Mean=0;

/* GRAPH SECTION */
WINDOW  Mainwindow={20,40,WINWIDE,WINHIGH};
GRAPH   Maingraph={0,0,30000,1};
float  Datamin=0, Datamax=0;
char   Dispname[13];
int    Zerotrigger=0;

/*****************************************************************************/
/*                P R O G R A M        S E C T I O N                 */
/*****************************************************************************/
void main(void)
{
 char input_key;

 set_cursor(OFF);
 main_menu();
 Cnt=0;
 shown(Menu_pos[0],1,"Process");
 while((input_key=getch())!=45) {
  check_input(input_key);
 }
 textcolor(WHITE);
 textbackground(BLACK);
 clrscr();
 set_cursor(ON);
}
```

```
/* MENU SECTION */
void check_input(char input_key)
{
  switch(input_key) {
   case 13 :
    switch(Cnt) {
     case 0 :
      Fexist=0;
      if(input_file()) {
       strcpy(Filein, Full_filein);
       pulldown_box(15, 8, 65, 16, BLUE,WHITE);
       textbackground(WHITE);
       textcolor(RED);
       gotoxy(16,10);
       cprintf("  Process of FFT , Wavelet Transform and Mean  ");
       textcolor(WHITE);
       gotoxy(16,15);
       cprintf("              Please Wait");
       if((Fexist=fexist(Filein))==1) {
        wait_on();
        fft_main();
        wavelet_main();
        mean_main();
        wait_off();
        soundbeep();
        soundbeep();
        clear();
       }
       else {
        textbackground(WHITE);
        textcolor(RED);
        gotoxy(16,10);
        cprintf("     File Does not Exist : Please Check !  ");
        textcolor(WHITE);
        gotoxy(16,15);
        cprintf("          Press a key to Continue ");
        getch();
       }
       clear();
      }
      break;
```

```
 case 1 :
  if(Fexist) {
   select_pull();
   main_menu();
   writexy(Menu_pos[1],1,"Result",15);
  }
  break;
 case 2 :
  say_about();
 clear();
 break;
 }
 break;
case 33 :   /* ALT-F */
 Cnt=0;
 top_menu();
 shown(Menu_pos[0],1,"Process");
 clear();
 break;
case 68 :   /* F10 */
 Cnt=0;
 top_menu();
 shown(Menu_pos[0],1,"Process");
 break;
case 0x19 :  /* ALT-P */
 Cnt=0;
 top_menu();
 shown(Menu_pos[0],1,"Process");
 break;
case 0x13 :  /* ALT-R */
 Cnt=1;
 top_menu();
 shown(Menu_pos[1],1,"Result");
 clear();
 break;
case 0x1E :  /* ALT-A */
 Cnt=2;
 top_menu();
 shown(Menu_pos[2],1,"About");
 clear();
 break;
```

```
    case 77 :  /* Right */
     clear();
     if(Cnt>=NMAIN-1)
      Cnt=0;
     else
      Cnt++;
     switch(Cnt){
       case 0 : top_menu();
          writexy(Menu_pos[0],1,"Process",15);
        break;
       case 1 : top_menu();
          writexy(Menu_pos[1],1,"Result",15);
         break;
       case 2 : top_menu();
          writexy(Menu_pos[2],1,"About",15);
         break;
     }
     break;
    case 75 :  /* LEFT */
     clear();
     if(Cnt<=0)
      Cnt=NMAIN-1;
    else
      Cnt--;
     switch(Cnt){
      case 0 :
         top_menu();
          writexy(Menu_pos[0],1,"Process",15);
         break;
       case 1 :
          top_menu();
          writexy(Menu_pos[1],1,"Result",15);
          break;
       case 2 : top_menu();
          writexy(Menu_pos[2],1,"About",15);
          break;
     }
    break;
  }
}
```

```
void select_pull(void) {
 int pos=0;
 char input_key=1;

 pulldown();
 shown(Menu_pos[1]-1,3,"Data    Graph ");
 while(input_key!=27) {
  switch(input_key){
   case  27 :
    clear();
    break;
   case  13 :
    switch(pos)  {
     case 0 :
      graph2d(0);
      clear();
      break;
     case 1 :
      graph2d(1);
      clear();
      break;
     case 2 :
      graph3d();
      clear();
      break;
     case 3 :
      clear();
      top_menu();
      pulldown_box(15, 9, 65, 16, BLUE, WHITE);
      textbackground(WHITE);
      textcolor(RED);
      gotoxy(20,11);
      cprintf("      Mean of Data is %f   ",Mean);
      textcolor(BLUE);
      gotoxy(20,14);
      cprintf("       Press a key to continue");
      getch();
      clear();
      break;
    }
    break;
```

```
case 72 :   /* UP */
 if(pos<=0)
  pos=3;
 else
  pos--;
 switch(pos) {
  case 0 : pulldown();
   shown(Menu_pos[1]-1,3,"Data    Graph ");
   break;
  case 1 : pulldown();
   shown(Menu_pos[1]-1,4,"Fft     Graph ");
   break;
  case 2 : pulldown();
   shown(Menu_pos[1]-1,5,"Wavelet  Graph ");
   break;
  case 3 : pulldown();
   shown(Menu_pos[1]-1,6,"Mean    Value ");
   break;
 }
 break;
case 80 :   /* Down */
 if(pos>=3)
   pos=0;
  else
   pos++;
  switch(pos) {
   case 0 : pulldown();
    shown(Menu_pos[1]-1,3,"Data    Graph ");
    break;
   case 1 : pulldown();
    shown(Menu_pos[1]-1,4,"Fft     Graph ");
    break;
   case 2 : pulldown();
    shown(Menu_pos[1]-1,5,"Wavelet  Graph ");
    break;
   case 3 : pulldown();
    shown(Menu_pos[1]-1,6,"Mean    Value ");
    break;
  }
  break;
}
```

```
  if(input_key!=13)
   input_key=getch();
  else
   input_key=27;
 }
}

void main_menu(void) {
  clrscr();
  textwindow(1,2,80,24,LIGHTGRAY,BLUE);
  write_bottom();
  clreol();
  write_head();
  gotoxy(1,1);
  top_menu();
  textbackground(7);
}

void write_head(void) {
  gotoxy(20,2);
  textbackground(BLUE);
  textcolor(WHITE);
  cprintf(" Program Transform for Knock Combusion ");
}

void write_bottom(void) {
  int i=0;
  gotoxy(1,25);
  while(i<NBOTTOM*2){
   if(i%2!=1){
    textbackground(7);     textcolor(RED);
    cprintf(" %s",Menu_down[i]);
   }
   else{
    textbackground(7);
    textcolor(BLACK);
    cprintf("%s",Menu_down[i]);
   }
   i++;
  }
}
```

```
void textwindow(int x1,int y1,int x2,int y2,int fcolor,int bcolor) {
 int x,y;

 textcolor(fcolor);
 textbackground(bcolor);
 for(y=y1+1;y<y2;y++) {
  for(x=x1+1;x<x2;x++) {
    gotoxy(x,y);
    putch(0x20);
   }
  gotoxy(x1,y); putch(0xb3);
  gotoxy(x2,y); putch(0xb3);
 }
 gotoxy(x1+1,y1);
 for(x=x1+1;x<x2;x++)
  putch(0xcd);
 gotoxy(x1+1,y2);
 for(x=x1+1;x<x2;x++)
  putch(0xc4);
 gotoxy(x1,y1); putch(0xd5);
 gotoxy(x2,y1); putch(0xb8);
 gotoxy(x1,y2); putch(0xc0);
 gotoxy(x2,y2); putch(0xd9);
}

void shown(int a,int b ,char *s) {
 gotoxy(a,b);
 textbackground(BLACK);
 textcolor(WHITE);
 cprintf("%s",s);
}

void top_menu(void) {
 int i;
 gotoxy(1,1);
 i=0;
 while(i<NMAIN*2){
  if(i%2!=1){
   textbackground(7);    textcolor(RED);
   cprintf("  %s",Menu_top[i]);
  }
```

```
  else {
   textbackground(7);
   textcolor(BLACK);
   cprintf("%s",Menu_top[i]);
  }
  i++;
  clreol();
 }
}

void pulldown(void) {
 pulldown_box(Menu_pos[1]-2,2,Menu_pos[1]-2+16,7,BLACK,LIGHTGRAY);
 writexy(Menu_pos[1]-1,3,"D",116);
 writexy(Menu_pos[1]-1+1,3,"ata",112);
 writexy(Menu_pos[1]-1+9,3,"Graph",112);
 writexy(Menu_pos[1]-1,4,"F",116);
 writexy(Menu_pos[1]-1+1,4,"ft",112);
 writexy(Menu_pos[1]-1+9,4,"Graph",112);
 writexy(Menu_pos[1]-1,5,"W",116);
 writexy(Menu_pos[1]-1+1,5,"avelet",112);
 writexy(Menu_pos[1]-1+9,5,"Graph",112);
 writexy(Menu_pos[1]-1,6,"M",116);
 writexy(Menu_pos[1]-1+1,6,"ean",112);
 writexy(Menu_pos[1]-1+9,6,"Value",112);
}

void pulldown_box(int x1,int y1,int x2,int y2,int fcolor,int bcolor) {
 int x,y;

 textcolor(fcolor);
 textbackground(bcolor);
 for(y=y1+1;y<y2;y++) {
  for(x=x1+1;x<x2;x++) {
   gotoxy(x,y);
   putch(0x20); /* Put Space character to clear */
  }
  gotoxy(x1,y); putch(0xb3);/* Vertical line */
  gotoxy(x2,y); putch(0xb3);
 }
```

```
 gotoxy(x1+1,y1);
 for(x=x1+1;x<x2;x++)
   putch(0xc4);          /* Horizontal line */
 gotoxy(x1+1,y2);
 for(x=x1+1;x<x2;x++)
   putch(0xc4);          /* Horizontal line */
 gotoxy(x1,y1); putch(0xda); /*  Corner line  */
 gotoxy(x2,y1); putch(0xbf);
 gotoxy(x1,y2); putch(0xc0);
 gotoxy(x2,y2); putch(0xd9);
}

void writexy(int x, int y, char *s, int attr) {
 gotoxy(x, y);
 textattr(attr);
 cprintf("%s", s);
}

void clear(void) {
 int i;

 textwindow(2,2,78,20,BLUE,BLUE);
 gotoxy(2,2);
 textcolor(LIGHTGRAY);
 for(i=0;i<78;i++)
   cprintf("0");
 write_head();
}

void printfxy(int x, int y, char *s, int color) {
 gotoxy(x,y);
 textcolor(color);
 cprintf("%s",s);
}
```

```
int input_file(void) {
 char fin[13];
 int haveinput=0;

 pulldown_box(9,9,71,11,BLUE,WHITE);
 printfxy(10,10, "Enter File Name to Analaze : ", BLUE);
 textcolor(BLACK);
 set_cursor(ON);
 if((haveinput=input_string(fin, 13))==1)
  strcpy(Full_filein, fin);
 clear();
 set_cursor(OFF);
 return(haveinput);
}

int error_msg(char *msg) {
 int xpos;

 xpos=(80-(strlen(msg)+29))/2;
 if(xpos < 0 ) {
  msg[51]='\0';
  xpos=0;
 }
 textcolor(RED);
 textbackground(WHITE);
 gotoxy(1,25);
 cprintf("                    ");
 cprintf("                    ");
 gotoxy(xpos,25);
 cprintf("%s",msg);
 cprintf(" - Press a key to continue - ");
 getch();
 gotoxy(1,25);
 cprintf("                    ");
 cprintf("                    ");
 write_bottom();
}
```

```
void say_about() {
 pulldown_box(15, 9, 65, 16, BLUE, WHITE);
 textbackground(WHITE);
 textcolor(BLUE);
 gotoxy(20,10);
 cprintf(" This is Knock Combustion Check Program ");
 gotoxy(20,12);
 cprintf(" for Analysis of Vibration Engine signal ");
 gotoxy(20,14);
 cprintf(" Written by Paijit Kochakornjarupong  ");
 getch();
}

void add_file_ext(char *fin, char *fout, char *ext) {
  char ch;
  int i;

  if(strlen(fin)!=0) {
   i=0;
   do {
    ch=fin[i];
    if(ch=='.')
     break;
    else
     if(ch=='\x0') {
      break;
     }
     else {
      fout[i]=fin[i];
      i++;
     }
   } while((i<8));
   if(i>0) {
    fout[i]='.';
    fout[i+1]=ext[0];
    fout[i+2]=ext[1];
    fout[i+3]=ext[2];
    fout[i+4]='\x0';
   }
  }
}
```

```
int input_string(char *message, int namelenmax)  {
 char   Temp[80], *input;
 int    i, ch, exitinput=0, inputcon=0, xpos, ypos;

 if( (input=(char *) malloc(namelenmax)) != NULL )  {
  strcpy(input, "");
  xpos=wherex()-1;
  ypos=wherey();
  i=0;
  do  {
   ch=getch();
   if(!ch)
    ch=getch();
   switch(ch) {
    case '\b' :
     inputcon=1;       /* BACK */
     if(i>0) {
      i--;
      sprintf(Temp," ");
      gotoxy(xpos+i+1,ypos);
      cprintf("%s", Temp);
      gotoxy(xpos+i+1,ypos);
     }
     break;
    case 27 :          /* ESC */
     inputcon=1;
     exitinput=1;
     i=0;
     break;
    case '\r' :        /* ENTER */
     inputcon=1;
     stpcpy(message, input);
     exitinput=1;
     break;
   }
   if( (ch<45 || ch>122)) {
    inputcon=0;
    continue;
   }
```

```
    else {
     if(!inputcon && i < namelenmax) {
      input[i]=ch;
      i++;
      input[i]='\0';
      gotoxy(xpos+i, ypos);
      cprintf("%c", ch);
     }
     else
      inputcon=0;
    }
   } while (exitinput!=1);
   if(i == 0)
    exitinput=0;
 }
 return(exitinput);
}
/* END MENU SECTION */

/* WAIT SECTION */
void set_cursor(int status) {
 if(status == ON) {
  _AH=0x01;
  _CL=0x0C;
  _CH=0x0A & 0xDF;
  geninterrupt(0x10);
 }
 if(status == OFF) {
  _AH=0x01;
  _CL=0x0D;
  _CH=0x0C | 0x20;
  geninterrupt(0x10);
 }
}

void wait_on(void) {
 oldtimer=getvect(TIMER);
 setvect(TIMER,newtimer);
}
```

```
void wait_off(void) {
 setvect(TIMER, oldtimer);
}

void soundbeep(void) {
 putch(7);
}

void interrupt newtimer() {
 static char ch=(char)176;

 gotoxy(XWAIT+Chpos, YWAIT);
 cprintf("%c", ch);
 Chpos++;
 if(Chpos==CHWAITLEN) {
  Chpos=0;
  ch=(ch==(char)176)?(char)177:(char)176;
 }
 if(oldtimer)
  (*oldtimer)();
}
/* END WAIT SECTION */

/* FFT SECTION */
void fft_main(void) {
 int       i, fft_size, insize;
 float far  *mag;
 float far  *data;
 double     tempflt;
 char       fin[13], fout[13];
 COMPLEX far *samp;

 strcpy(fin, Filein);
 insize=datacount(fin);
 if(insize > 0) {
  add_file_ext(fin, fout, FFT_EXT);
  data=(float far *)farcalloc(insize, sizeof(float));
  insize = readfile(fin, data, insize);

  mag = (float far *) farcalloc(insize, sizeof(float));
  samp = (COMPLEX far *) farcalloc(insize, sizeof(COMPLEX));
```

```
  if (!mag || !samp) {
   error_msg(" Error : FFT Memory allocation error.");
   exit(1);
  }

  for (i=0; i<insize; i++) {
   samp[i].real = data[i];
   samp[i].imag = 0;
   mag[i]=0;
  }

  fft_size = log2(insize);
  fft(samp,fft_size);
  for (i=0; i<insize; i++) {
   tempflt  = samp[i].real * samp[i].real;
   tempflt += samp[i].imag * samp[i].imag;
   mag[i] = sqrt(tempflt);
  }

  writefile(fout, mag, insize/2);
  farfree(samp);
  farfree(mag);
  farfree(data);
 }
}

void fft(COMPLEX *x, int fft_size) {
 COMPLEX *w, u, temp, tm;
 COMPLEX *xi, *xip, *xj, *wptr;
 int    i, j, k, l, le, windex, n=1;
 double  arg, w_real, w_imag, wrecur_real, wrecur_imag, wtemp_real;

 if(fft_size == 0)
  return;     /* if fft_size=0 is end */
 n = 1 << fft_size;
 le = n/2;
 w = (COMPLEX *) calloc(le-1,sizeof(COMPLEX));
 if(!w) {
   error_msg("Error : FFT can not allocate complex w array");
   exit(1);
 }
```

```
/* calculate the w values recursively */
arg = 2.0 * PHI / n;
wrecur_real = w_real = cos(arg);
wrecur_imag = w_imag = -sin(arg);
xj = w;
for (j = 1 ; j < le ; j++) {
 xj->real = (float)wrecur_real;
 xj->imag = (float)wrecur_imag;
 xj++;
 wtemp_real = wrecur_real*w_real - wrecur_imag*w_imag;
 wrecur_imag = wrecur_real*w_imag + wrecur_imag*w_real;
 wrecur_real = wtemp_real;
}

/* start fft */
le = n;
windex = 1;
for (l = 0 ; l < fft_size ; l++) {
 le = le/2;
 /* start calculate with no multiplies */
 for(i = 0 ; i < n ; i = i + 2*le) {
  xi = x + i;
  xip = xi + le;
  temp.real = xi->real + xip->real;
  temp.imag = xi->imag + xip->imag;
  xip->real = xi->real - xip->real;
  xip->imag = xi->imag - xip->imag;
  *xi = temp;
 }
 /* continue calculate remaining use stored w */
 wptr = w + windex - 1;
 for (j = 1 ; j < le ; j++) {
  u = *wptr;
  for (i = j ; i < n ; i = i + 2*le) {
   xi = x + i;
   xip = xi + le;
   temp.real = xi->real + xip->real;
   temp.imag = xi->imag + xip->imag;
   tm.real = xi->real - xip->real;
   tm.imag = xi->imag - xip->imag;
   xip->real = tm.real*u.real - tm.imag*u.imag;
```

```
      xip->imag = tm.real*u.imag + tm.imag*u.real;
      *xi = temp;
     }
     wptr = wptr + windex;
    }
    windex = 2*windex;
   }

   /* change position of data by bit reverse */
   j = 0;
   for (i = 1 ; i < (n-1) ; i++) {
    k = n/2;
    while(k <= j) {
     j = j - k;
     k = k/2;
    }
    j = j + k;
    if (i < j) {
     xi = x + i;
     xj = x + j;
     temp = *xj;
     *xj = *xi;
     *xi = temp;
    }
   }
}

int log2(unsigned int x) {
 unsigned int mask,i;

 if(x == 0)
  return(-1);          /* error size = 0 return -1   */
 x--;                  /* set max index              */
 for(mask = 1 , i = 0 ; ; mask *= 2 , i++) {
  if(x == 0)
   return(i);          /* end process                */
  x = x & (~mask); /* check bit                  */
 }
}
```

```
int readfile(char *fin, float *data, int npts) {
 int  i;
 FILE *fptr;
 int  nread=0;
 float value;

 nread=datacount(fin);
 nread=(nread>npts)?npts:nread;
 if(nread>0) {
  if((fptr=fopen(fin,"rt"))!=NULL) {
   i=0;
   while (i<nread) {
    fscanf(fptr, "%e", &value);
    data[i++]=value;
   }
   fclose(fptr);
  }
  else
   error_msg("Error : Readfile Open input file");
 }
 else
  error_msg("Error : Readfile File Not have data");
 return(nread);
}

void writefile(char *fout, float *data, int npts) {
 int  i;
 FILE *fptr;

 if((fptr=fopen(fout,"wt"))!=NULL) {
  for(i=0; i<npts; i++) {
   fprintf(fptr, "%e ", data[i]);
   fprintf(fptr, " \n");
  }
  fclose(fptr);
 }
 else
  error_msg("Error : Writefile open output file");
}
```

```
int datacount(char *fin) {
 FILE *fptr;
 int  count=0;
 float value;

 if((fptr=fopen(fin,"rt"))!=NULL) {
  while (fscanf(fptr, "%f", &value)!=0 && !feof(fptr))
   count++;
  fclose(fptr);
 }
 else {
  count=-1;
  error_msg("Error : Datacount Open input file");
 }
 return(count+1);
}

int fexist(char *fin) {
 int status=0;

 if(fopen(fin,"rt")!=NULL)
  status=1;
 return(status);
}
/* END FFT SECTION */

/* WAVELET SECTION */
void dwritefile(char *fout, float far **data, int npts, int nfreq) {
 int  i,j;
 FILE *fptr;
 if((fptr=fopen(fout,"wt"))!=NULL) {
  for(i=0; i<npts; i++) {
   for(j=0; j<nfreq; j++) {
    fprintf(fptr, "%f\n", data[i][j]);
   }
  }
  fclose(fptr);
 }
 else
  error_msg("Error : Dwritefile Open output file");
}
```

```
void wavelet_transform(float *x, int npts, int beginscale, int endscale,
                float far **Smag)
{
 int  b, i, j, t, sscale, escale, index=0;
 float far *sr;
 float far *si;
 float a, scos, ssin, delta;

 sr=(float far *) farcalloc( (beginscale-endscale)*SCALE_STEP+1 ,
    sizeof(float));
 si=(float far *) farcalloc( (beginscale-endscale)*SCALE_STEP+1 ,
    sizeof(float));
 for(i=0; i<(endscale-beginscale)*SCALE_STEP+1; i++) {
  sr[i]=0;
  si[i]=0;
 }
 sscale=beginscale*SCALE_STEP; /* end of step  | Low  Frequency */
 escale=endscale*SCALE_STEP;  /* begin of step | High Frequency */
 for(b=0; b<npts; b++){
  index=0;
  for(i=escale; i<sscale; i++) { /* Index 0 is High frequency */
   scos=0;
   ssin=0;
   a=pow(ROOT20OF2, i);
   for(t=0; t<npts; t++) {
    delta=(t-b)/a;
    scos+=cos(2*PHI*delta)*exp(-1*pow(delta,2)/2)*x[t];
    ssin+=sin(2*PHI*delta)*exp(-1*pow(delta,2)/2)*x[t];
   }
   scos/=sqrt(a);
   ssin/=sqrt(a);
   sr[index]=scos;
   si[index]=ssin;
   index++;
  }
  for (j = 0; j < (beginscale-endscale)*SCALE_STEP+1; j++)
   Smag[b][j] = sqrt(sr[j] * sr[j] + si[j] * si[j]);
 }
 farfree((float far *)sr);
 farfree((float far *)si);
}
```

```
void wavelet_main(void) {
 char fin[13], fmag[13];
 int  i, j, beginscale, endscale, npts, nfreq;
 float *x;
 float far **smag;

 strcpy(fin, Filein);
 add_file_ext(fin, fmag, WT_MAG_EXT);
 npts=datacount(fin);
 x = (float *) malloc( npts * sizeof(float));
 npts=readfile(fin, x, npts);
          /*                                  |   N    | */
          /* Frequency of Transform in term of | 1/2  * fs| */
          /* fs is Sampling rate                          */
 beginscale=BEGINSCALE; /* width  value scale  low  frequency  */
 endscale=ENDSCALE;     /* narrow value scale  high frequency  */

 nfreq=(beginscale-endscale)*SCALE_STEP+1;
 Filelen=npts;    /* set global for mean section use          */
 Nfreq=nfreq;    /* set global for mean section use           */

 smag=(float far **) farcalloc( npts , sizeof(float *));
 for(i=0;i<npts;i++) {
  smag[i] = (float far *) farcalloc( nfreq , sizeof(float));
  for(j=0;j<nfreq;j++)
    smag[i][j]=0;
 }

 wavelet_transform(x, npts, beginscale, endscale, smag);
 free(x);
 dwritefile(fmag, smag, npts, nfreq);

 for(i=npts-1;i>=0;i--)
  farfree((float far *)smag[i]);
 farfree(smag);
}
/* WAVELET SECTION */
```

```
/* MEAN SECTION */
void  mean_main(void) {
 char  fin[13], fout[13];
 int   i=0, index=0, j=0, npts, nfreq, fbegin, fend, nread;
 float far  *buffer;
 float far  *cut;
 float far  **data;
 float mean=0;

 add_file_ext(Filein, fin, WT_MAG_EXT);
 npts = Filelen;          /* data read */
 nfreq = Nfreq;           /* frequency scan  is   1 / a * sampling rate   */
 fbegin = FBEGIN;    /* start frequency |                    fbegin+20 | */
 fend = FEND;             /* end frequency   | a=(sqrt20of2)              | */
 nread = datacount(fin);
 buffer=(float far *)farcalloc(nread, sizeof(float));
 nread = readfile(fin, buffer, nread);
 data=(float far **)farcalloc(npts, sizeof(float far *));
 for(i=0; i<npts; i++)  {
  data[i]=(float far *)farcalloc(nfreq, sizeof(float));
  for(j=0; j<nfreq; j++)
    data[i][j]=buffer[i*nfreq+j];
 }
 cut=(float far *)farcalloc((fend-fbegin+1)*npts, sizeof(float));
 index=0;
 for(i=fbegin; i<(fend+1); i++)
  for(j=0; j<npts; j++)
    cut[index++]=data[j][i];
 mean=average(cut, index);
 strcpy(fout, fin);
 add_file_ext(fin, fout, WT_CUT_EXT);
 writefile(fout, cut, (fend-fbegin+1)*npts);
 add_file_ext(fin, fout, MEAN_EXT);
 writefile(fout, &mean, 1);
 Mean=mean;
 for(i=0; i<npts; i++)
  farfree((float far *)data[i]);
 farfree((float far *)data);
 farfree((float far *)cut);
 farfree((float far *)buffer);
}
```

```
float average(float *data, int insize) {
 int i;
 double sum=0.0;

 for(i=0; i<insize; i++)
  sum=sum+data[i];
 return((float)(sum/insize));
}

/* GRAPH SECTION */
void set_graph_2d(float far *data1, int datacount1, int type) {
 int  i=0;
 float value, datamin=0, datamax=0, datahigh, datawide;
 float expand_ratio=1.0; /* set hight of display by (yhigh+space_add)/yhigh */

 datamax=datamin=data1[0];     /*  Find datamax datamin  */
 for(i=0; i<datacount1; i++) {
  value=data1[i];
  if(datamax < value)
   datamax=value;
  if(datamin > value)
   datamin=value;
 }
 Datamax=datamax;              /* Max of Data for Display */
 Datamin=datamin;              /* Min of Data for Display */

 if(type==0) {                 /* Set Min , Max of Graph  */
  if(datamin>GMIN_DATA)
   datamin=GMIN_DATA;
  if(datamax < GMAX_DATA)
   datamax=GMAX_DATA;
 }
 else {
  if(datamin>GMIN_FFT)
   datamin=GMIN_FFT;
  if(datamax < GMAX_FFT)
   datamax=GMAX_FFT;
 }
 datahigh=datamax-datamin;
 datawide=datacount1;
```

```
 Maingraph.w=datawide;
 if(datahigh<=0)
  Maingraph.h=(double)1.0;
 else
  Maingraph.h=(double)(datahigh*expand_ratio);
 Maingraph.y=(double)datamin-((datahigh*expand_ratio-datahigh)/2.0);
}

void  plot2d(float far *data, int type)  {
 char strtemp[80];
 int  gdriver,gmode, fill[8];
 int  yzero=0, texthigh=0, textwide=0, i=0;
 int  lastxpos, lastypos, xpos, ypos;

 detectgraph(&gdriver, &gmode);
 initgraph(&gdriver, &gmode, GDRIVER_DIR);
 setwritemode(COPY_PUT);
 setfillstyle(SOLID_FILL,BLACK);
 setbkcolor(BLACK);
 setcolor(WHITE);
 rectangle(0,0,getmaxx(), getmaxy());
 rectangle(Mainwindow.x-1, Mainwindow.y-1,
       Mainwindow.x+Mainwindow.w, Mainwindow.y+Mainwindow.h);
 fill[0]=Mainwindow.x;
 fill[1]=Mainwindow.y;
 fill[2]=Mainwindow.x+Mainwindow.w-1;
 fill[3]=Mainwindow.y;
 fill[4]=Mainwindow.x+Mainwindow.w-1;
 fill[5]=Mainwindow.y+Mainwindow.h-1;
 fill[6]=Mainwindow.x;
 fill[7]=Mainwindow.y+Mainwindow.h-1;
 fillpoly(4,fill);

 setcolor(YELLOW);
 lastxpos=(int)Mainwindow.x;
 lastypos=(int)((1-((data[Maingraph.x]-Maingraph.y)/Maingraph.h))
      *Mainwindow.h+Mainwindow.y);
 if(Maingraph.w==1)
  line(lastxpos,lastypos,lastxpos+Mainwindow.w,lastypos);
 else  {
  for (i=Maingraph.x; i<Maingraph.x+Maingraph.w; i++)  {
```

```
  xpos=(int)(((i-Maingraph.x)/Maingraph.w)*Mainwindow.w+Mainwindow.x);
  ypos=(int)((1-((data[i]-Maingraph.y)/Maingraph.h))
        *Mainwindow.h+Mainwindow.y);
  if (xpos >= Mainwindow.x && xpos <= Mainwindow.x + Mainwindow.w &&
    ypos >= Mainwindow.y && ypos <= Mainwindow.y + Mainwindow.h )
   line(lastxpos,lastypos,xpos,ypos);
  lastxpos=xpos;
  lastypos=ypos;
 }
}

setcolor(WHITE);
if(Zerotrigger) {
 yzero=(int)((1+(Maingraph.y/Maingraph.h))*Mainwindow.h+Mainwindow.y);
 line(Mainwindow.x, yzero, Mainwindow.x+Mainwindow.w-1, yzero);
}

texthigh=textheight(" ");
textwide=textwidth(" ");
/* display detail of graph */
setcolor(LIGHTBLUE);
if(type ==0) {
 outtextxy(Mainwindow.x+Mainwindow.w/2-5*textwide,
       Mainwindow.y+Mainwindow.h+3*texthigh, "Sample No.");
 outtextxy(Mainwindow.x+Mainwindow.w/2-4*textwide,
      Mainwindow.y-3*texthigh, " Data Input ");
 setcolor(LIGHTMAGENTA);              /* display scale of graph */
 sprintf(strtemp,"%1.0f",Maingraph.x+Maingraph.w);
 outtextxy(Mainwindow.x+Mainwindow.w-textwide,
      Mainwindow.y+Mainwindow.h+texthigh, strtemp);
}
else {
 outtextxy(Mainwindow.x+Mainwindow.w/2-7*textwide,
       Mainwindow.y+Mainwindow.h+3*texthigh,"Frequency (Hz)");
 outtextxy(Mainwindow.x+Mainwindow.w/2-textwide,
       Mainwindow.y-3*texthigh, " FFT ");
 setcolor(LIGHTMAGENTA);
 sprintf(strtemp,"%1.0f",Maingraph.x+Maingraph.w/(Maingraph.w*2)*SAMPL_RATE);
 outtextxy(Mainwindow.x+Mainwindow.w-textwide,
      Mainwindow.y+Mainwindow.h+texthigh, strtemp);
}
```

```
 /* display scale x , y of graph */
 setcolor(LIGHTMAGENTA);
 sprintf(strtemp,"%1.0f",Maingraph.x);
 outtextxy(Mainwindow.x, Mainwindow.y+Mainwindow.h+texthigh,strtemp);
 sprintf(strtemp,"%+1.3f",Maingraph.y);
 outtextxy(Mainwindow.x+Mainwindow.w+textwide,
       Mainwindow.y+Mainwindow.h-texthigh, strtemp);
 sprintf(strtemp,"%2d",0);
 outtextxy(Mainwindow.x+Mainwindow.w+textwide, yzero-0.5*texthigh, strtemp);
 sprintf(strtemp,"%+1.3f",Maingraph.y+Maingraph.h);
 outtextxy(Mainwindow.x+Mainwindow.w+textwide,
       Mainwindow.y, strtemp);

 /* Display Detail File Name , Data Max, Data Min */
 sprintf(strtemp,"File name : %s  ",Dispname);
 outtextxy(GX_START,GY_START+1*texthigh, strtemp);
 sprintf(strtemp, "Data Min ");
 moveto(GX_START+250, GY_START+1*texthigh);
 outtext(strtemp);
 sprintf(strtemp, "%+1.3f", Datamin);
 outtext(strtemp);
 sprintf(strtemp, "Data Max ");
 moveto(GX_START+400, GY_START+1*texthigh);
 outtext(strtemp);
 sprintf(strtemp, "%+1.3f", Datamax);
 outtext(strtemp);

 getch();
 closegraph();
}

void graph2d(int type) {
 char fin[13];
 int  insize;
 float *data;

 if(type==0)                  /* type of data display   */
  strcpy(fin, Filein);          /*     0 is data       */
 else
  add_file_ext(Filein, fin, FFT_EXT); /*     1 is FFT        */
 insize=datacount(fin);
```

```
 data=(float *)calloc(insize, sizeof(float));
 insize = readfile(fin, data, insize);
 strcpy(Dispname, fin);
 set_graph_2d(data, insize, type);
 plot2d(data, type);
 free(data);
}

int plot3d(int xsize, int ysize, float far **ydata) {
 char  strtemp[80];
 int   i=0, j=0, gdriver=SCR_DRIVER, gmode=SCR_MODE, errorcode;
 int   ycolor=0, textwide, texthigh;
 int   zoom=3, xcur, ycur, xloop, yloop, xstart, ystart;
 int   xmax, ymax, nstep=16;  /* Use nstep to set display colors */
 float ydatamax=0 ,ydatamin=0 ,ystep=0;
 float value;                 /*        for quantize data       */
 double remain, ipart;        /*        for quantize data       */
 struct palettetype pal;
 initgraph(&gdriver, &gmode, GDRIVER_DIR);
 errorcode=graphresult();
 if(errorcode!= grOk)
  error_msg("Error : Plot3d Initial graphics error");
 else {
  xmax=getmaxx();
  ymax=getmaxy();
  xstart=(xmax-xsize*zoom)/2;
  ystart=(ymax-ysize*zoom)/2;
  if((xsize>0) && (ysize>0)) {
   getpalette(&pal);
   /* Set gray scale black is minimum and white is maximum */
   for(i=0; i<pal.size; i++)
    setrgbpalette(pal.colors[i], i*4, i*4, i*4);
   ydatamax=ydatamin=ydata[0][0];
   for(i=0; i<xsize ;i++) {
    for(j=0; j<ysize ;j++) {
     if(ydatamax < ydata[i][j])
      ydatamax=ydata[i][j];
     if(ydatamin > ydata[i][j])
      ydatamin=ydata[i][j];
    }
   }
```

```
nstep=16;  /* Show shade */
            /* change display nstep to VGAGREY if change mode */
ystep= (ydatamax - ydatamin) / nstep;
if(ystep==0)
 ystep=1;
floodfill(2,2, 0); /* Background BLACK=0 */
setcolor(0);
rectangle(xstart-1,ystart-1, xstart+xsize*zoom+1, ystart+ysize*zoom+1);
rectangle(xstart-2,ystart-2, xstart+xsize*zoom+2, ystart+ysize*zoom+2);
setcolor(getmaxcolor());
rectangle(xstart-3,ystart-3, xstart+xsize*zoom+3, ystart+ysize*zoom+3);
rectangle(0,0,xmax,ymax);
rectangle(1,1,xmax-1,ymax-1);
rectangle(2,2,xmax-2,ymax-2);
xcur=xstart;

texthigh=textheight(" ");
textwide=textwidth(" ");
outtextxy((xmax-19*textwide)/2,5*texthigh, " Wavelet Transform ");

sprintf(strtemp,"%1.0f",(float)SAMPL_RATE/2);
outtextxy(xstart+xsize*zoom+3+textwide, ystart-texthigh/2, strtemp);
line(xstart+xsize*zoom, ystart, xstart+xsize*zoom+textwide, ystart);
line(xstart+xsize*zoom, ystart+ysize*zoom/3, xstart+xsize*zoom+textwide,
   ystart+ysize*zoom/3);
line(xstart+xsize*zoom, ystart+ysize*zoom/3*2,
   xstart+xsize*zoom+textwide, ystart+ysize*zoom/3*2);
line(xstart+xsize*zoom, ystart+ysize*zoom, xstart+xsize*zoom+textwide,
   ystart+ysize*zoom);
sprintf(strtemp,"%1.0f",(float)SAMPL_RATE/4);
outtextxy(xstart+xsize*zoom+3+textwide,
      ystart+ysize*zoom/3-texthigh+texthigh/2, strtemp);
sprintf(strtemp,"%1.0f",(float)SAMPL_RATE/8);
outtextxy(xstart+xsize*zoom+3+textwide,
      ystart+2*ysize*zoom/3-texthigh+texthigh/2, strtemp);
sprintf(strtemp,"%1.0f",(float)SAMPL_RATE/16);
outtextxy(xstart+xsize*zoom+3+textwide,
      ystart+ysize*zoom-texthigh/2, strtemp);
outtextxy(xstart-textwide/2, ystart+ysize*zoom+3+texthigh, "0");
sprintf(strtemp,"%d",xsize/2);
```

```
outtextxy(xstart+xsize*zoom/2-textwide,
        ystart+ysize*zoom+3+texthigh, strtemp);
sprintf(strtemp,"%-d",xsize);
outtextxy(xstart+xsize*zoom-1.5*textwide,
        ystart+ysize*zoom+3+texthigh, strtemp);
line(xstart, ystart+ysize*zoom, xstart, ystart+ysize*zoom+texthigh);
line(xstart+xsize*zoom/2, ystart+ysize*zoom,
    xstart+xsize*zoom/2, ystart+ysize*zoom+texthigh);
line(xstart+xsize*zoom, ystart+ysize*zoom,
    xstart+xsize*zoom, ystart+ysize*zoom+texthigh);
outtextxy((xmax-10*textwide)/2, ystart+ysize*zoom+3+3*texthigh, "Sample no.");
settextstyle(DEFAULT_FONT, VERT_DIR, 1);
outtextxy(xstart-2*texthigh,ystart+ysize*zoom/2-7*textwide,
        "Frequency (Hz)");
for(i=0; i < xsize; i++) {
 ycur=ystart;
 for(j=0; j < ysize; j++) {
  value = (ydata[i][j] - ydatamin) / ystep;
  remain = modf(value, &ipart);
  if(remain!=0)
   ipart++;
  if((remain==0)&&(ipart==0))       /* Increace of Lower data */
   ipart++;
  ycolor = (nstep-1)-((int)ipart-1);   /*        reverse color        */
  if( ycolor > (VGAGRAY-1) || ycolor < 0) {
   setcolor(RED);
   outtextxy(10,440,"Color Error : Calculate of color not in range ");
   delay(10000);
   outtextxy(10,440,"                                    ");
   closegraph();
   return(-1);
  }
  for(xloop=0; xloop<zoom; xloop++)
   for(yloop=0; yloop<zoom; yloop++)
    putpixel(xcur+xloop, ycur+yloop, ycolor);
  ycur=ycur+zoom;
 }
 xcur=xcur+zoom;
}
getch();
}
```

```
  else {
   setcolor(RED);
   outtextxy(10,440,"Graph size error ");
   delay(1000);
   outtextxy(10,440,"                ");
  }
  closegraph();
 }
 return(0);
}

void graph3d(void) {
 char fin[13];
 int  i, j, index, nread, npts, nfreq;
 float far *x;
 float far **smag;

 add_file_ext(Filein, fin, WT_MAG_EXT);
 nread=datacount(fin);
 x = (float far *) farmalloc( nread * sizeof(float));
 nread=readfile(fin, x, nread);
 nfreq=(BEGINSCALE-ENDSCALE)*SCALE_STEP+1;
 npts=nread/nfreq;

 smag=(float far **) farcalloc( npts , sizeof(float *));
 index=0;
 for(i=0;i<npts;i++) {
  smag[i] = (float far *) farcalloc( nfreq , sizeof(float));
  for(j=0;j<nfreq;j++)
   smag[i][j]=x[index++];
 }

 plot3d(npts,nfreq, smag);

 for(i=npts-1;i>=0;i--)
  farfree((float far *)smag[i]);
 farfree(smag);
 farfree(x);
}
/* END GRAPH SECTION */
```

ประวัติผู้เขียน

ชื่อ นายไพจิตร กชกรจารุพงศ์

วัน เดือน ปีเกิด 29 พฤษภาคม 2512

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิทยาศาสตร์บัณฑิต (วิทยาการคอมพิวเตอร์)	มหาวิทยาลัยเกษตรศาสตร์	2533