รายการอ้างอิง

[1] Eugene I. Rivin, Passive Vibration Isolation. New York: The American Society of Mechanical Engineers, 2003.

[2] ฤทธิกิติ์ ประไพพิชิต. การศึกษาแบบจำลองการสั่นสะเทือนโดยวิธีการวิเคราะห์โมดัลเชิงทดลอง. วิทยานิพนธ์ปริญญามหาบัณฑิต, ภาควิชาวิศวกรรมเครื่องกล บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย ,2544.

[3] ราเชน จามน้อยพรหม. การศึกษาการสั่นสะเทือนของรถบรรทุกเล็ก. วิทยานิพนธ์ปริญญามหาบัณฑิต, ภาควิชาวิศวกรรมเครื่องกล บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย ,2546.

[4] Yunhe Yu, Nagi G. Naganathan , Rao V. Dukkipati. A Literature review of automotive vehicle engine mounting system. Mechanical and Machine Theory 36 (2001): 123-142

[5] B.L.Bolton – Knight. Engine Mount: Analytical method to reduce noise and vibration, Vibration and noise in Motor Vehicles, pp.25-34 The Institution of Mechanical Engineers, 1972.

[6] ธนู ฉุยฉาย. การสั่นสะเทือนเชิงกล. พิมพ์ครั้งที่ 2. กรุงเทพมหานคร: สมาคมส่งเสริมเทคโนโลยี (ไทย - ญี่ปุ่น), 2544.

[7] จักร จันทลักขณา. รูปร่างการสั่นกับการแก้ปัญหาความสั่นสะเทือนและการออกแบบทางวิศวกรรม (2). ใน เทคนิค222, หน้า 114-122. (กรุงเทพมหานคร: สำนักพิมพ์เอ็มแอนด์อี, 2546.

[8] D.J. Ewins, Modal Testing. second edition. London: Research studies press LTD, 2000.

[9] S.Braun, D.Ervin, and S.S. Rao, Encyclopedia of vibration London: Academic Press, 2001.

[10] Ole DØssing, Structural Testing Part II: Modal Analysis and Simulation. Denmark: Bruel&Kjaer, 1988.

[11] Sevend Gade, Henrik Herlufsen and Hans Konstantin-Hansen. How to Determine the Modal Parameters of Simple Structure. Application Note, Denmark: Bruel&Kjaer, 1994.

[12] จักร จันทลักขณา. รูปร่างการสั่นกับการแก้ปัญหาความสั่นสะเทือนและการออกแบบทางวิศวกรรม (1). ใน เทคนิค221, หน้า 122-127. กรุงเทพมหานคร: สำนักพิมพ์เอ็มแอนด์อี, 2546.

[13] Cyril M. Harris, Shock and Vibration Handbook. fourth edition. (New York: McGraw-Hill Companies Inc., 1996.

[14] Ole DØssing, Structural Testing Part I: Mechanical Mobility Measurements. Denmark: Bruel&Kjaer, 1988

[15] Jose Javier Almendros Gomez, Experimental methods for determining moment of inertia of small UAVS. Institute of technology Linköpings, June 2001.

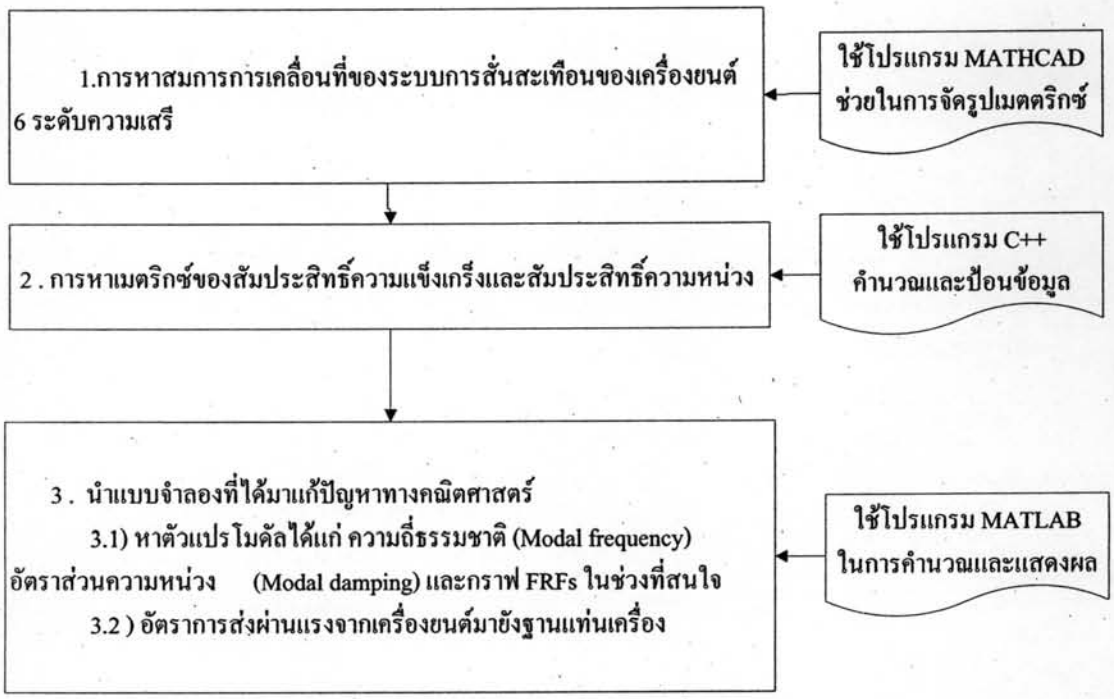
[16] P.A.R de Schrijver, Experimental modal analysis of tyre measurement tower, Dynamics and central technology group, Department mechanical engineer, Eindhoven University of Technology, Eindhoven, August , 2005

ภาคผนวก

ภาคผนวก ก

การใช้โปรแกรมคอมพิวเตอร์ในการศึกษาการสั่นสะเทือนของเครื่องยนต์

ขั้นตอนการแก้ปัญหาของการศึกษาการสั่นสะเทือนการสั่นสะเทือนของเครื่องยนต์สามารถแสดงได้ดังนี้



รูปที่ ก.1 ขั้นตอนการใช้โปรแกรมสำเร็จรูปมาใช้แก้ปัญหา

ภาคผนวกส่วนนี้จะแสดงรายละเอียดโปรแกรม 2 ส่วนคือ การใช้ โปรแกรม C++ และ MATLAB ซึ่งมีรายละเอียดดังนี้

1) โปรแกรม C++ .ใช้สำหรับการหาเมทริกซ์ของสัมประสิทธิ์ความแข็งเกร็งและสัมประสิทธิ์ความหน่วง โดยโปรแกรมจะสั่งให้ป้อนค่าตัวแปรซึ่งเป็นคุณสมบัติของแท่นเครื่องทั้ง 3แท่น ที่ตำแหน่งแท่นเครื่องต่าง ๆ พร้อมรวมรายละเอียดประกอบ ได้แก่ เวกเตอร์แสดงตำแหน่งและมุมของแท่นเครื่องที่ทำกับกรอบอ้างอิงที่กำหนด

รายละเอียดโปรแกรม

```
////////////////////////////////////////////////////////////

// This Program use to compute Mass, Stiffness and Damping Matrix  //

////////////////////////////////////////////////////////////

#include <stdio.h>
#include <conio.h>
#include <math.h>

FILE *fp1,*fp2,*fp3;

/*Declare variable*/
const double  PI = 4.0*atan(1.0); // Define PI
const double  Ixx = 4.45, // Inertia Matrix's coefficient
                    Iyy = 5.27, // Edit here
                    Izz = .14.05
                    m   = 142.3.0;
double  k[3][3]; // Stiffness coefficient in Principle axis
double  c[3][3]; // Damping coefficient in Principle axis
double  p[3][3]; // Mounting position from Center of mass of Body
long    Be[3][9]; // Direction of cosine
double  co[3][9]; // Value of cosine
double  kxx[3],kxy[3],kxz[3],kyy[3],kyz[3],kzz[3]; // Stiffness coeficient in Orthogonal axis
double    sum_kxx = 0.0,sum_kyy = 0.0,
          sum_kzz = 0.0,sum_kxy = 0.0,
          sum_kxz = 0.0,sum_kyz = 0.0;
```

```
double  cxx[3],cxy[3],cxz[3],cyy[3],cyz[3],czz[3]; // Damping coefficint in Orthogonal
axis
double  sum_cxx = 0.0,sum_cyy = 0.0,
        sum_czz = 0.0,sum_cxy = 0.0,
        sum_cxz = 0.0,sum_cyz = 0.0;
double  c11,c12,c13,c14,c15,c16,c22, // Damping Matrix's coefficient
        c23,c24,c25,c26,c33,c34,c35,
        c36,c44,c45,c46,c55,c56,c66;
double  k11,k12,k13,k14,k15,k16,k22, // Stiffness Matrix's coefficient
        k23,k24,k25,k26,k33,k34,k35,
        k36,k44,k45,k46,k55,k56,k66;
/*End declare variable*/

/*Declare prototype*/
void Open_File();
void Set_Param();
void Compute_Stiffness_Component();
void Compute_Damping_Component();
void Compute_Stiffness_Matrix();
void Compute_Damping_Matrix();
int  Covert_To_Code(char a);
double Cal_Elements1(double temp1[],double temp2[],char a,char b);
double Cal_Elements2(double temp1[],double temp2[],
                                    double temp3[],double temp4[],
                                    char a,char b,char c,char d,
                                    char e,char f,char g,char h);
void Show_Matrix();
void Write_File();
/*End declare prototype*/
//-------------------------------------------------------------------------------//
int main()
```

```
{
        printf("This program is to compute Damping matrix and Stiffness matrix\n\n");

        Open_File();
        printf("##### BEGIN PROGRAM #####\n");
        Set_Param();
        Compute_Stiffness_Component();
        Compute_Damping_Component();
        Compute_Stiffness_Matrix();
        Compute_Damping_Matrix();
        Show_Matrix();
        Write_File();
        printf("\n##### END OF PROGRAM #####\n\n");

        fclose(fp1);
        fclose(fp2);
        fclose(fp3);
        getch();

        return 0;
}
//-------------------------------------------------------------------------------//
void Open_File()
{
        fp1 = fopen("C:\\mass.txt","w+t");
        fp2 = fopen("C:\\stiff.txt","w+t");
        fp3 = fopen("C:\\damp.txt","w+t");

        if( fp1 == NULL )
                printf("Cannot Create File mass.txt!!\n");
```

```
        else
                printf("Can Create File mass.txt\n");
        if( fp2 == NULL )
                printf("Cannot Create File stiff.txt!!\n");
        else
                printf("Can Create File stiff.txt\n");
        if( fp3 == NULL )
                printf("Cannot Create File damp.txt!!\n");
        else
                printf("Can Create File damp.txt\n");
        return;
}
//-------------------------------------------------------------------------------//
void Set_Param()
{
        printf("#####Please enter Stiffness coefficient#####\n\n");

        for( int i = 0 ; i < 3 ; i++ )
        {
                for( int j = 0 ; j < 3 ; j++ )
                {
                        switch (j)
                        {
                                case 0 :

                                        printf("kx%d :",i+1);
                                        scanf("%lf",&k[i][j]);
                                        break;
                                case 1 :
                                        printf("ky%d :",i+1);
                                        scanf("%lf",&k[i][j]);
```

```
                    break;

                default :

                    printf("kz%d :",i+1);
                    scanf("%lf",&k[i][j]);
            }
        }
    }

    /*for(int i = 0 ; i < 3 ; i++ )
    {
        for(int j = 0 ; j < 3 ; j++ )
            printf("%.1f\n",k[i][j]);
    }*/

    printf("\n");
    printf("#####Please enter Damping coefficient#####\n\n");

    for( int i = 0 ; i < 3 ; i++ )
    {
        for( int j = 0 ; j < 3 ; j++ )
        {
            switch (j)
            {
                case 0 :

                    printf("cx%d :",i+1);
                    scanf("%lf",&c[i][j]);
                    break;
```

```
                case 1 :

                        printf("cy%d :",i+1);
                        scanf("%lf",&c[i][j]);
                        break;

                default :

                        printf("cz%d :",i+1);
                        scanf("%lf",&c[i][j]);
            }
        }
}

printf("\n");
printf("#####Please enter Position vector#####\n\n");

for( int i = 0 ; i < 3 ; i++ )
{
        for( int j = 0 ; j < 3 ; j++ )
        {
            switch (j)
            {
                case 0 :

                        printf("px%d :",i+1);
                        scanf("%lf",&p[i][j]);
                        break;
                case 1 :
                        printf("py%d :",i+1);
                        scanf("%lf",&p[i][j]);
```

```
                    break;

                default :
                    printf("pz%d :",i+1);
                    scanf("%lf",&p[i][j]);
            }
        }
}
printf("\n");
printf("#####Please enter Direction of cosine of each principle axis#####\n\n");
printf("##### Please Enter in degree (Integers) #####");

for( int i = 0 ; i < 3 ; i++ )
{
    for( int j = 0 ; j < 9 ; j++ )
    {
        switch (j)
        {
            case 0 :
                printf("coxX%d :",i+1);
                scanf("%ld",&Be[i][j]);
                break;
            case 1 :
                printf("coxY%d :",i+1);
                scanf("%ld",&Be[i][j]);
                break;
            case 2 :

                printf("coxZ%d :",i+1);
                scanf("%ld",&Be[i][j]);
                break;
```

```
                case 3 :
                        printf("coyX%d :",i+1);
                        scanf("%ld",&Be[i][j]);
                        break;
                case 4 :
                        printf("coyY%d :",i+1);
                        scanf("%ld",&Be[i][j]);
                        break;
                case 5 :
                        printf("coyZ%d :",i+1);
                        scanf("%ld",&Be[i][j]);
                        break;
                case 6 :
                        printf("cozX%d :",i+1);
                        scanf("%ld",&Be[i][j]);
                        break;
                case 7 :
                        printf("cozY%d :",i+1);
                        scanf("%ld",&Be[i][j]);
                        break;
                default :
                        printf("cozZ%d :",i+1);
                        scanf("%ld",&Be[i][j]);
            }
            co[i][j] = double(cos((PI/180.0)*double(Be[i][j])));

        }
    }

    return;
}
```

```
//-------------------------------------------------------------------------------//
void Compute_Stiffness_Component()
{
        for( int i = 0 ; i < 3 ; i++ )
        {
                /*Find Stiffness of each component*/
                kxx[i] = k[i][0]*co[i][0]*co[i][0] + k[i][1]*co[i][3]*co[i][3] +
k[i][2]*co[i][6]*co[i][6];
                kyy[i] = k[i][0]*co[i][1]*co[i][1] + k[i][1]*co[i][4]*co[i][4] +
k[i][2]*co[i][7]*co[i][7];
                kzz[i] = k[i][0]*co[i][2]*co[i][2] + k[i][1]*co[i][5]*co[i][5] +
k[i][2]*co[i][8]*co[i][8];
                kxy[i] = k[i][0]*co[i][0]*co[i][1] + k[i][1]*co[i][3]*co[i][4] +
k[i][2]*co[i][6]*co[i][7];
                kxz[i] = k[i][0]*co[i][0]*co[i][2] + k[i][1]*co[i][3]*co[i][5] +
k[i][2]*co[i][6]*co[i][8];
                kyz[i] = k[i][0]*co[i][1]*co[i][2] + k[i][1]*co[i][4]*co[i][5] +
k[i][2]*co[i][7]*co[i][8];
                /*End Find Stiffness of each component*/

                /*Find summation of Stiffness component*/
                sum_kxx = kxx[i] + sum_kxx;
                sum_kyy = kyy[i] + sum_kyy;
                sum_kzz = kzz[i] + sum_kzz;
                sum_kxy = kxy[i] + sum_kxy;
                sum_kxz = kxz[i] + sum_kxz;
                sum_kyz = kyz[i] + sum_kyz;
                /*End Find summation of Stiffness component*/
        }
        return;
}
```

```
//-------------------------------------------------------------------------------//
void Compute_Damping_Component()
{
        for( int i = 0 ; i < 3 ; i++ )
        {
                /*Find Damping of each component*/
                cxx[i] = c[i][0]*co[i][0]*co[i][0] + c[i][1]*co[i][3]*co[i][3] +
c[i][2]*co[i][6]*co[i][6];
                cyy[i] = c[i][0]*co[i][1]*co[i][1] + c[i][1]*co[i][4]*co[i][4] +
c[i][2]*co[i][7]*co[i][7];
                czz[i] = c[i][0]*co[i][2]*co[i][2] + c[i][1]*co[i][5]*co[i][5] +
c[i][2]*co[i][8]*co[i][8];
                cxy[i] = c[i][0]*co[i][0]*co[i][1] + c[i][1]*co[i][3]*co[i][4] +
c[i][2]*co[i][6]*co[i][7];
                cxz[i] = c[i][0]*co[i][0]*co[i][2] + c[i][1]*co[i][3]*co[i][5] +
c[i][2]*co[i][6]*co[i][8];
                cyz[i] = c[i][0]*co[i][1]*co[i][2] + c[i][1]*co[i][4]*co[i][5] +
c[i][2]*co[i][7]*co[i][8];
                /*End Find Damping of each component*/

                /*Find summation of Damping component*/
                sum_cxx = cxx[i] + sum_cxx;
                sum_cyy = cyy[i] + sum_cyy;
                sum_czz = czz[i] + sum_czz;
                sum_cxy = cxy[i] + sum_cxy;
                sum_cxz = cxz[i] + sum_cxz;
                sum_cyz = cyz[i] + sum_cyz;
                /*End Find summation of Damping component*/
        }
        return;
}
```

```
//-------------------------------------------------------------------------------//
int Convert_To_Code(char a)
{
        int status = 0;
        switch(a)
        {
                case 'x' : {}
                case 'X' :

                        status = 0;
                        break;

                case 'y' : {}
                case 'Y' :

                        status = 1;
                        break;

                default :

                        status = 2;
        }

        return status;
}
//-------------------------------------------------------------------------------//
double Cal_Elements1(double temp1[],double temp2[],char a,char b)
{
        double result = 0.0;
        int code1 = 0,code2 = 0;
```

```
        code1 = Convert_To_Code(a);
        code2 = Convert_To_Code(b);
        for( int i = 0 ; i < 3 ; i++ )
                result = (temp1[i]*p[i][code1]-temp2[i]*p[i][code2]) + result;
        return result;
}
//-------------------------------------------------------------------------------//
double Cal_Elements2(double temp1[],double temp2[],
                                        double temp3[],double temp4[],
                                        char a,char b,char c,char d,
                                        char e,char f,char g,char h)
{
        double result = 0.0;
        int code1 = 0,code2 = 0,code3 = 0,code4 = 0,
                code5 = 0,code6 = 0,code7 = 0,code8 = 0;
        code1 = Convert_To_Code(a);
        code2 = Convert_To_Code(b);
        code3 = Convert_To_Code(c);
        code4 = Convert_To_Code(d);
        code5 = Convert_To_Code(e);
        code6 = Convert_To_Code(f);
        code7 = Convert_To_Code(g);
        code8 = Convert_To_Code(h);
        for(int i = 0 ; i < 3 ; i++ )
                result =  (temp1[i]*p[i][code1]*p[i][code2] +
temp2[i]*p[i][code3]*p[i][code4] -
                                        temp3[i]*p[i][code5]*p[i][code6] -
temp4[i]*p[i][code7]*p[i][code8]) +
                                        result;
        return result;
}
```

```
//-------------------------------------------------------------------------------//
void Compute_Stiffness_Matrix()
{
        k11 = sum_kxx;
        k12 = sum_kxy;
        k13 = sum_kxz;
        k14 = Cal_Elements1(kxz,kxy,'y','z');
        k15 = Cal_Elements1(kxx,kxz,'z','x');
        k16 = Cal_Elements1(kxy,kxx,'x','y');
        k22 = sum_kyy;
        k23 = sum_kyz;
        k24 = Cal_Elements1(kyz,kyy,'y','z');
        k25 = Cal_Elements1(kxy,kyz,'z','x');
        k26 = Cal_Elements1(kyy,kxy,'x','y');
        k33 = sum_kzz;
        k34 = Cal_Elements1(kzz,kyz,'y','z');
        k35 = Cal_Elements1(kxz,kzz,'z','x');
        k36 = Cal_Elements1(kyz,kxz,'x','y');
        k44 = Cal_Elements2(kyy,kzz,kyz,kyz,'z','z','y','y','y','z','y','z');
        k45 = Cal_Elements2(kxz,kyz,kzz,kxy,'y','z','x','z','x','y','z','z');
        k46 = Cal_Elements2(kxy,kyz,kyy,kxz,'y','z','x','y','x','z','y','y');
        k55 = Cal_Elements2(kxx,kzz,kxz,kxz,'z','z','x','x','x','z','x','z');
        k56 = Cal_Elements2(kxy,kxz,kxx,kyz,'x','z','x','y','y','z','x','x');
        k66 = Cal_Elements2(kxx,kyy,kxy,kxy,'y','y','x','x','x','y','x','y');

        return;
}
//-------------------------------------------------------------------------------//
void Compute_Damping_Matrix()
{
        c11 = sum_cxx;
```

```
        c12 = sum_cxy;
        c13 = sum_cxz;
        c14 = Cal_Elements1(cxz,cxy,'y','z');
        c15 = Cal_Elements1(cxx,cxz,'z','x');
        c16 = Cal_Elements1(cxy,cxx,'x','y');
        c22 = sum_cyy;
        c23 = sum_cyz;
        c24 = Cal_Elements1(cyz,cyy,'y','z');
        c25 = Cal_Elements1(cxy,cyz,'z','x');
        c26 = Cal_Elements1(cyy,cxy,'x','y');
        c33 = sum_czz;
        c34 = Cal_Elements1(czz,cyz,'y','z');
        c35 = Cal_Elements1(cxz,czz,'z','x');
        c36 = Cal_Elements1(cyz,cxz,'x','y');
        c44 = Cal_Elements2(cyy,czz,cyz,cyz,'z','z','y','y','y','z','y','z');
        c45 = Cal_Elements2(cxz,cyz,czz,cxy,'y','z','x','z','x','y','z','z');
        c46 = Cal_Elements2(cxy,cyz,cyy,cxz,'y','z','x','y','x','z','y','y');
        c55 = Cal_Elements2(cxx,czz,cxz,cxz,'z','z','x','x','x','z','x','z');
        c56 = Cal_Elements2(cxy,cxz,cxx,cyz,'x','z','x','y','y','z','x','x');
        c66 = Cal_Elements2(cxx,cyy,cxy,cxy,'y','y','x','x','x','y','x','y');

        return;
}
//-------------------------------------------------------------------------//
void Show_Matrix()
{
        /*Show Mass-Inertia Matrix*/
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",m,0.0,0.0,0.0,0.0,0.0);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",0.0,m,0.0,0.0,0.0,0.0);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",0.0,0.0,m,0.0,0.0,0.0);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",0.0,0.0,0.0,Ixx,0.0,0.0);
```

```
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",0.0,0.0,0.0,0.0,Iyy,0.0);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",0.0,0.0,0.0,0.0,0.0,Izz);
        /*End Mass-Inertia Matrix*/

        printf("\n\n");

        /*Show Stiffness Matrix*/
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",k11,k12,k13,k14,k15,k16);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",k12,k22,k23,k24,k25,k26);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",k13,k23,k33,k34,k35,k36);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",k14,k24,k34,k44,k45,k46);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",k15,k25,k35,k45,k55,k56);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",k16,k26,k36,k46,k56,k66);
        /*End Stiffness Matrix*/

        printf("\n\n");

        /*Show Damping Matrix*/
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",c11,c12,c13,c14,c15,c16);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",c12,c22,c23,c24,c25,c26);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",c13,c23,c33,c34,c35,c36);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",c14,c24,c34,c44,c45,c46);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",c15,c25,c35,c45,c55,c56);
        printf("%.1f    %.1f    %.1f    %.1f    %.1f    %.1f\n",c16,c26,c36,c46,c56,c66);
        /*End Damping Matrix*/

        return;
}
//-------------------------------------------------------------------------------//
void Write_File()
{
```

```
/*Write Mass Matrix to mass.txt*/
fprintf(fp1,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",m,0.0,0.0,0.0,0.0,0.0);
fprintf(fp1,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",0.0,m,0.0,0.0,0.0,0.0);
fprintf(fp1,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",0.0,0.0,m,0.0,0.0,0.0);
fprintf(fp1,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",0.0,0.0,0.0,Ixx,0.0,0.0);
fprintf(fp1,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",0.0,0.0,0.0,0.0,Iyy,0.0);
fprintf(fp1,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",0.0,0.0,0.0,0.0,0.0,Izz);
/*End Write Mass Matrix*/

/*Write Mass Stiffness to stiff.txt*/
fprintf(fp2,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",k11,k12,k13,k14,k15,k16);
fprintf(fp2,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",k12,k22,k23,k24,k25,k26);
fprintf(fp2,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",k13,k23,k33,k34,k35,k36);
fprintf(fp2,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",k14,k24,k34,k44,k45,k46);
fprintf(fp2,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",k15,k25,k35,k45,k55,k56);
fprintf(fp2,"%.1f %.1f %.1f %.1f %.1f
%.1f\n",k16,k26,k36,k46,k56,k66);
/*End Write Stiffness Matrix*/

/*Write Mass Damping to damp.txt*/
```

```
        fprintf(fp3,"%.1f        %.1f    %.1f    %.1f    %.1f
        %.1f\n",c11,c12,c13,c14,c15,c16);
        fprintf(fp3,"%.1f        %.1f    %.1f    %.1f    %.1f
        %.1f\n",c12,c22,c23,c24,c25,c26);
        fprintf(fp3,"%.1f        %.1f    %.1f    %.1f    %.1f
        %.1f\n",c13,c23,c33,c34,c35,c36);
        fprintf(fp3,"%.1f        %.1f    %.1f    %.1f    %.1f
        %.1f\n",c14,c24,c34,c44,c45,c46);
        fprintf(fp3,"%.1f        %.1f    %.1f    %.1f    %.1f
        %.1f\n",c15,c25,c35,c45,c55,c56);
        fprintf(fp3,"%.1f        %.1f    %.1f    %.1f    %.1f
        %.1f\n",c16,c26,c36,c46,c56,c66);
        /*End Write Damping Matrix*/

        return;
}
//---------------------------------------------------------------------------
```

2) โปรแกรม MATLAB ใช้ในการคำนวณหาตัวแปรโมดัลแล ะฟังก์ชั่นตอบสนองเชิงความถี่ ที่สนใจและสร้างแบบจำลองของอัตราการส่งผ่านแรงจากเครื่องยนต์มายังฐานแท่นเครื่อง

2.1) การใช้โปรแกรม MATLAB สำหรับการแสดงกราฟของ FRFs ที่สนใจ โดยจะเริ่มต้นจาก การโหลดค่าเมทริกซ์ของ Mass Stiff และ Damp ที่ได้จากส่วนที่ 1 และใช้โปรแกรมที่เขียนขึ้นใน MATLAB โดยใช้คำสั่งของ Control Toolbox โดยใช้รูปแบบของฟังก์ถ่ายโอน ซึ่งให้ผลเป็นรูปแบบเดียวกับที่ได้ ในฟังก์ชั่นตอบสนองเชิงความถี่ จะทำให้ง่ายต่อการเขียนโปรแกรมเพื่อหาค่า FRFs โดยตรง

รายละเอียดโปรแกรม

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                         %
%    This pogram find Frequency response function of                      %
%       Vibration of Engine in 6 Degree of freedom                        %
%          by using State space model approach                            %
%                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Load data mass stiffness and damping matrix from drive C:%
load c:\mass.txt;
load c:\damp.txt;
load c:\stiff.txt;
%End Load data mass stiffness and damping matrix from drive C:%

%Define Matrix of State space A B C D matrix%
M = mass;
C = damp;
K = stiff;
null = 0*eye(6,6);
   %Find Natural frequency of system%
   om = sqrt(eig(inv(M)*K));
   f  = (1/(2*pi))*om;
   %End Find Natural frequency of system%
A = [null eye(6,6);-inv(M)*K -inv(M)*C];
B = [null;inv(M)];
J = [eye(6,6) null];
D = null;
%End Define Matrix of State space A B C D matrix%
```

```
%Build State space model of system%
sys = ss(A,B,J,D);
set(sys,'outputname',{'Xc' 'Yc' 'Zc' 'Alfa' 'Beta' 'Gamma'});
set(sys,'inputname',{'Fx' 'Fy' 'Fz' 'Mx' 'My' 'Mz'});
set(sys,'statename',{'Xc' 'Yc' 'Zc' 'Alfa' 'Beta' 'Gamma' 'Xdotc' 'Ydotc' 'Zdotc' 'Alfadot'
'Betadot' 'Gammadot'});
syso = tf(sys);
%End build State space model of system%

%Define Frequency response function%
H11 = syso(1,1);
H12 = syso(1,2);
H13 = syso(1,3);
H14 = syso(1,4);
H15 = syso(1,5);
H16 = syso(1,6);
H21 = syso(2,1);
H22 = syso(2,2);
H23 = syso(2,3);
H24 = syso(2,4);
H25 = syso(2,5);
H26 = syso(2,6);
H31 = syso(3,1);
H32 = syso(3,2);
H33 = syso(3,3);
H34 = syso(3,4);
H35 = syso(3,5);
H36 = syso(3,6);
H41 = syso(4,1);
H42 = syso(4,2);
H43 = syso(4,3);
```

```
H44 = syso(4,4);
H45 = syso(4,5);
H46 = syso(4,6);
H51 = syso(5,1);
H52 = syso(5,2);
H53 = syso(5,3);
H54 = syso(5,4);
H55 = syso(5,5);
H56 = syso(5,6);
H61 = syso(6,1);
H62 = syso(6,2);
H63 = syso(6,3);
H64 = syso(6,4);
H65 = syso(6,5);
H66 = syso(6,6);
%End Define Frequency response function%

%Define Frequency response function matrix%
Hmat = [H11 H12 H13 H14 H15 H16;
    H21 H22 H23 H24 H25 H26;
    H31 H32 H33 H34 H35 H36;
    H41 H42 H43 H44 H45 H46;
    H51 H52 H53 H54 H55 H56;
    H61 H62 H63 H64 H65 H66];
I = [0 0 0 0 0 1]';
Y = Hmat*I;
%End Define Frequency response function matrix%

%Find Bode plot of system%
bode(Y(1));
figure;
```

```
bode(Y(2));
figure;
bode(Y(3));
figure;
bode(Y(4));
figure;
bode(Y(5));
figure;
bode(Y(6));
%End Find Bode plot of system%

%This algorithm uses to compute damping ratio%
rt = eig(A);
re_tmp = abs(real(rt));
re = [re_tmp(1,1);re_tmp(3,1);re_tmp(5,1);re_tmp(7,1);re_tmp(9,1);re_tmp(11,1)];
rtcon = conj(rt);
ww_tmp = rt.*rtcon;
ww =
[ww_tmp(1,1);ww_tmp(3,1);ww_tmp(5,1);ww_tmp(7,1);ww_tmp(9,1);ww_tmp(11,1)];
w = sqrt(ww);
ft = (1/(2*pi))*w;
damp = re./w;
%End algorithm for computing damping ratio%
```

2.2) การจำลองอัตราการส่งผ่านแรงไปยังฐานแท่นเครื่องโดยใช้ Simulink ใน MATLAB โดยจะต้องป้อนค่าข้อมูลต่างของแท่นเครื่องทั้ง 3 ในส่วนของตัวโปรแกรมและโหลดค่าของเมตริกซ์ของ Mass Stiff และ Damp เหมือนกับการใช้งานในโปรแกรมส่วน 2.1)

รายละเอียดโปรแกรม

```
function Forces =  Compute_Forces (State_Pos,State_Vel,Cosine1 ,Cosine2,Cosine3
,Pos1 ,Pos2 ,Pos3)
    % This block supports an embeddable subset of the MATLAB language.
    % See the help menu for details.

    d2r = pi/180.0;%Factor covert degree to radian

    %Poj1 represent Position of X.
    %Poj2 represent Position of Y.
    %Poj3 represent Position of Z.

    Po11 = Pos1(1);
    Po12 = Pos1(2);
    Po13 = Pos1(3);
    Po21 = Pos2(1);
    Po22 = Pos2(2);
    Po23 = Pos2(3);
    Po31 = Pos3(1);
    Po32 = Pos3(2);
    Po33 = Pos3(3);

    %Sp1 or Sv1 represent of xc
    %Sp2 or Sv2 represent of yc
    %Sp3 or Sv3 represent of zc
    %Sp4 or Sv4 represent of alfa
    %Sp5 or Sv5 represent of beta
    %Sp6 or Sv6 represent of gamma

    Sp1 = State_Pos(1);
```

```
Sp2 = State_Pos(2);
Sp3 = State_Pos(3);
Sp4 = State_Pos(4);
Sp5 = State_Pos(5);
Sp6 = State_Pos(6);

Sv1 = State_Vel(1);
Sv2 = State_Vel(2);
Sv3 = State_Vel(3);
Sv4 = State_Vel(4);
Sv5 = State_Vel(5);
Sv6 = State_Vel(6);

%coj1 represent p<X
%coj2 represent p<Y
%coj3 represent p<Z
%coj4 represent q<X
%coj5 represent q<Y
%coj6 represent q<Z
%coj7 represent r<X
%coj8 represent r<Y
%coj9 represent r<Z

co11 = cos(d2r*Cosine1(1));
co12 = cos(d2r*Cosine1(2));
co13 = cos(d2r*Cosine1(3));
co14 = cos(d2r*Cosine1(4));
co15 = cos(d2r*Cosine1(5));
co16 = cos(d2r*Cosine1(6));
co17 = cos(d2r*Cosine1(7));
co18 = cos(d2r*Cosine1(8));
```

```
co19 = cos(d2r*Cosine1(9));

co21 = cos(d2r*Cosine2(1));
co22 = cos(d2r*Cosine2(2));
co23 = cos(d2r*Cosine2(3));
co24 = cos(d2r*Cosine2(4));
co25 = cos(d2r*Cosine2(5));
co26 = cos(d2r*Cosine2(6));
co27 = cos(d2r*Cosine2(7));
co28 = cos(d2r*Cosine2(8));
co29 = cos(d2r*Cosine2(9));

co31 = cos(d2r*Cosine3(1));
co32 = cos(d2r*Cosine3(2));
co33 = cos(d2r*Cosine3(3));
co34 = cos(d2r*Cosine3(4));
co35 = cos(d2r*Cosine3(5));
co36 = cos(d2r*Cosine3(6));
co37 = cos(d2r*Cosine3(7));
co38 = cos(d2r*Cosine3(8));
co39 = cos(d2r*Cosine3(9));

%Edit Stiffness of matrix in Principle axis here%

kp1 = 14509.8;
kq1 = 10657.5;
kr1 = 881255.5;

kp2 = 14509.8;
kq2 = 10657.5;
kr2 = 1018910.3;
```

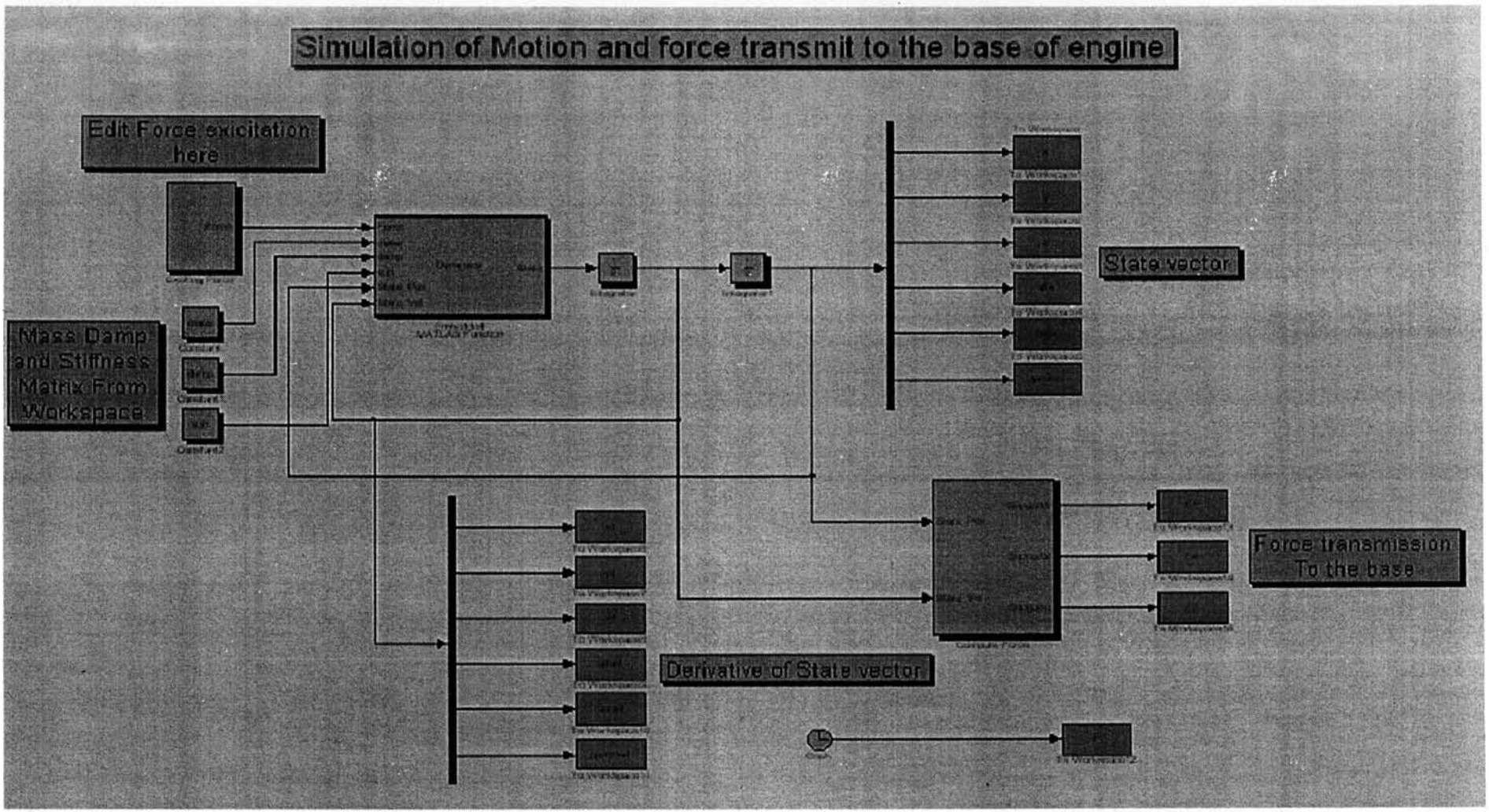
```
kp3 = 3514.1;
kq3 = 40300.7;
kr3 = 3982.;

K1 = [kp1 0 0;0 kq1 0;0 0 kr1];
K2 = [kp2 0 0;0 kq2 0;0 0 kr2];
K3 = [kp3 0 0;0 kq3 0;0 0 kr3];

%End Edit Stiffness of matrix in principle axis%

%Edit Damping of matrix in Principle axis%

cp1 = 108.8;
cq1 = 103.5;
cr1 = 914.5;

cp2 = 110.59;
cq2 = 95.69;
cr2 = 768.6;

cp3 = 40.34;
cq3 = 85.4;
cr3 = 70.94;

C1 = [cp1 0 0;0 cq1 0;0 0 cr1];
C2 = [cp2 0 0;0 cq2 0;0 0 cr2];
C3 = [cp3 0 0;0 cq3 0;0 0 cr3];

%End Edit Stiffness of matrix in Principle axis%
```

```
%Define Transformation matrix%

U1 = [co11 co12 co13;co14 co15 co16;co17 co18 co19];
U2 = [co21 co22 co23;co24 co25 co26;co27 co28 co29];
U3 = [co31 co32 co33;co34 co35 co36;co37 co38 co39];

%End define transformation matrix%

%Find K orthogonal from K principal axis%

Kor1 = U1'*K1*U1;
Kor2 = U2'*K2*U2;
Kor3 = U3'*K3*U3;

%End Find K orthogonal from K principal axis%

%Find C orthogonal from C principal axis%

Cor1 = U1'*C1*U1;
Cor2 = U2'*C2*U2;
Cor3 = U3'*C3*U3;

%End Find C orthogonal from C principal axis%

%Define Displacement of each support%

dx1 = Sp1 - Po12*Sp6 + Po13*Sp5;
dy1 = Sp2 - Po13*Sp4 + Po11*Sp6;
dz1 = Sp3 - Po11*Sp5 + Po12*Sp4;

dx2 = Sp1 - Po22*Sp6 + Po23*Sp5;
```

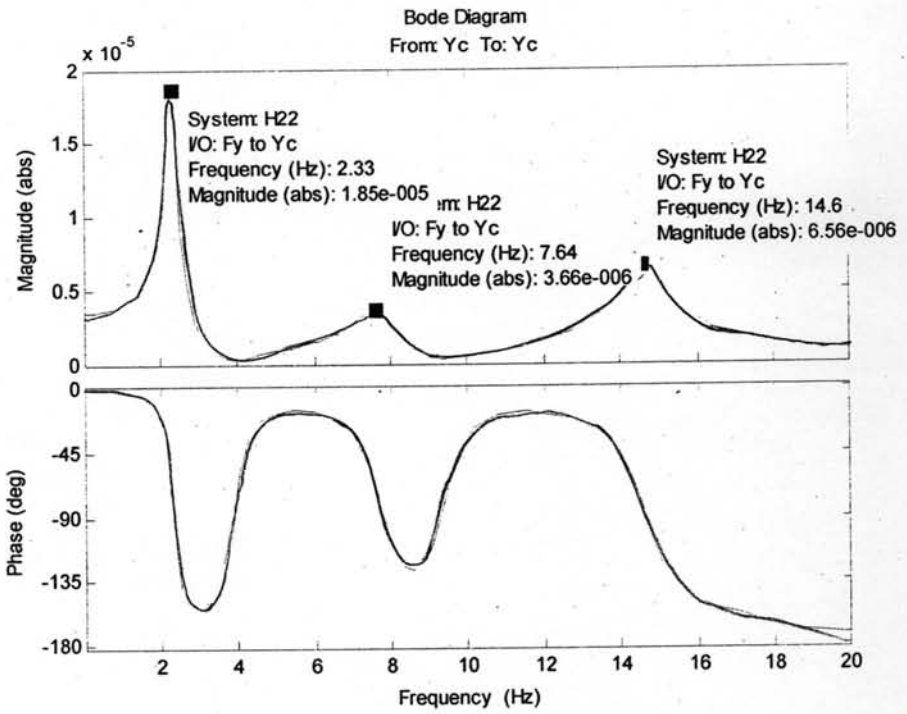
```
dy2 = Sp2 - Po23*Sp4 + Po21*Sp6;
dz2 = Sp3 - Po21*Sp5 + Po22*Sp4;

dx3 = Sp1 - Po32*Sp6 + Po33*Sp5;
dy3 = Sp2 - Po33*Sp4 + Po31*Sp6;
dz3 = Sp3 - Po31*Sp5 + Po32*Sp4;

D1 = [dx1;dy1;dz1];
D2 = [dx2;dy2;dz2];
D3 = [dx3;dy3;dz3];

%End Define Displace ment of each support%

%Define Velocity of each support%

vex1 = Sv1 - Po12*Sv6 + Po13*Sv5;
vey1 = Sv2 - Po13*Sv4 + Po11*Sv6;
vez1 = Sv3 - Po11*Sv5 + Po12*Sv4;

vex2 = Sv1 - Po22*Sv6 + Po23*Sv5;
vey2 = Sv2 - Po23*Sv4 + Po21*Sv6;
vez2 = Sv3 - Po21*Sv5 + Po22*Sv4;

vex3 = Sv1 - Po32*Sv6 + Po33*Sv5;
vey3 = Sv2 - Po33*Sv4 + Po31*Sv6;
vez3 = Sv3 - Po31*Sv5 + Po32*Sv4;

Ve1 = [vex1;vey1;vez1];
Ve2 = [vex2;vey2;vez2];
Ve3 = [vex3;vey3;vez3];
```

```
%End Define Velocity of each support%

%Define force at each support in each component XYZ%

F1 = Kor1*D1 + Cor1*Ve1;
F2 = Kor2*D2 + Cor2*Ve2;
F3 = Kor3*D3 + Cor3*Ve3;

%End Define force at each support%

%Define Resultant force at each support%

Fr1 = sqrt(F1'*F1);
Fr2 = sqrt(F2'*F2);
Fr3 = sqrt(F3'*F3);

%End Define Resultant force at each support%
```

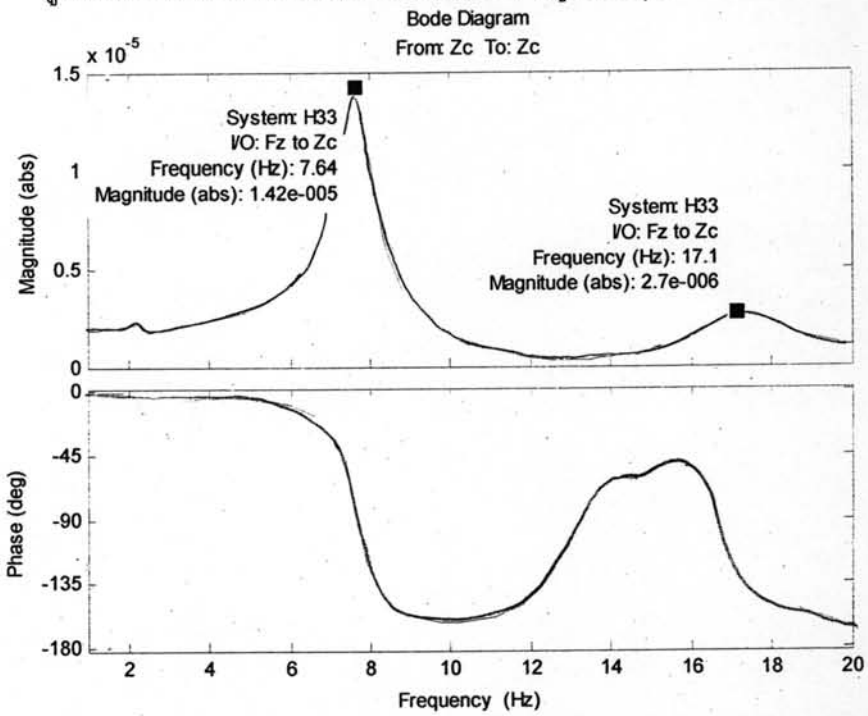
ส่วนข้างต้นเป็นส่วนหนึ่งของการจำลองเพื่อแก้ปัญหาสำหรับการหาอัตราการส่งผ่านแรงไปยังฐานแท่นเครื่อง สามารถแสดงภาพรวมของการจำลองการแก้ปัญหาได้ดังรูปต่อไปนี้



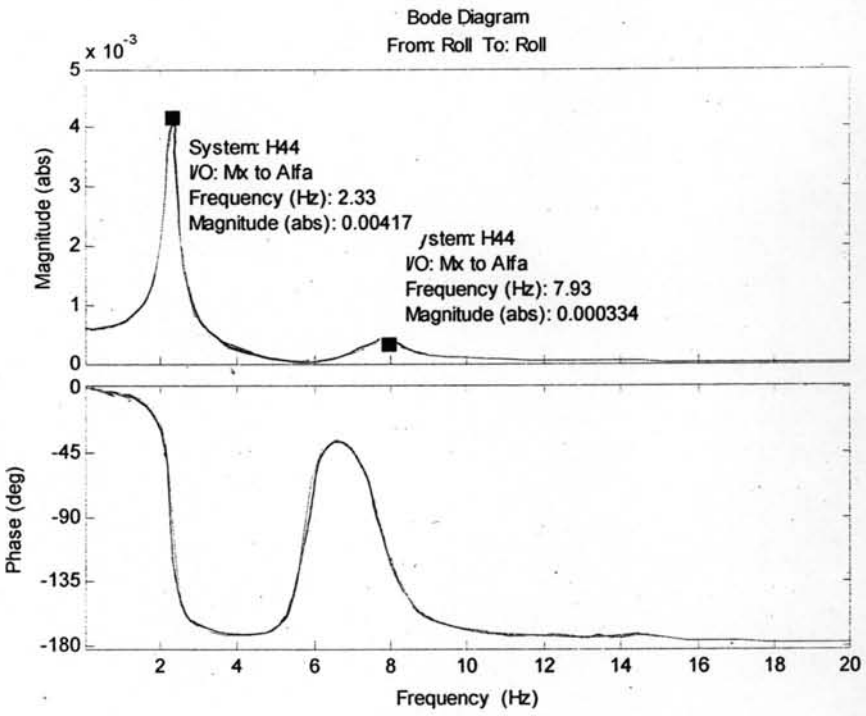
รูปที่ ก.2 รูปแบบการใช้ Simulink ในการแก้ปัญหา



รูปที่ ก.3 กราฟ FRFs ของผลตอบสนอง Y ที่ถูกกระตุ้นโดยแกน Y



รูปที่ ก.4 กราฟ FRFs ของผลตอบสนอง Z ที่ถูกกระตุ้นโดยแกน Z



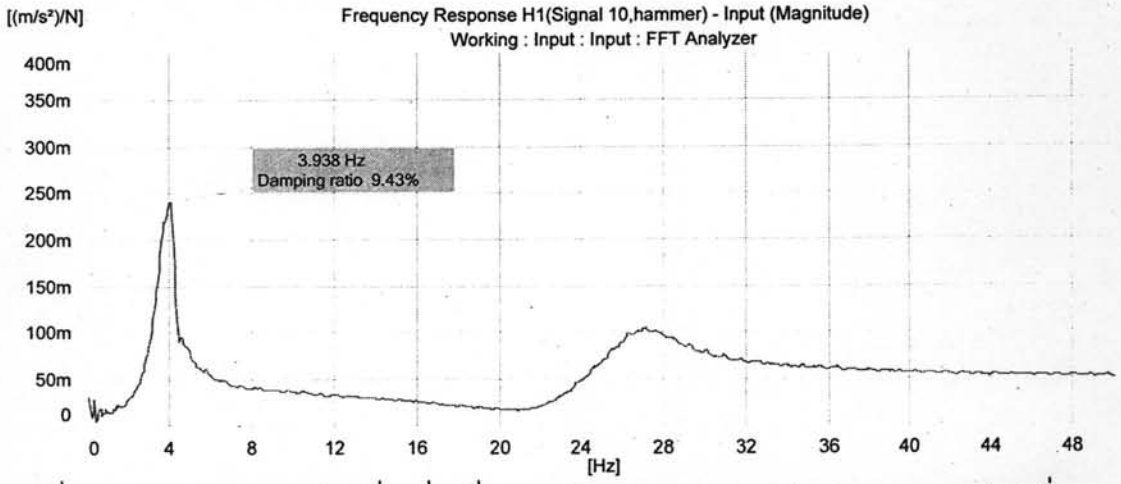
รูปที่ ก.5 กราฟ FRFs ของผลตอบสนอง Roll ที่ถูกกระตุ้นโดยรอบแกน M_x

ภาคผนวก ข

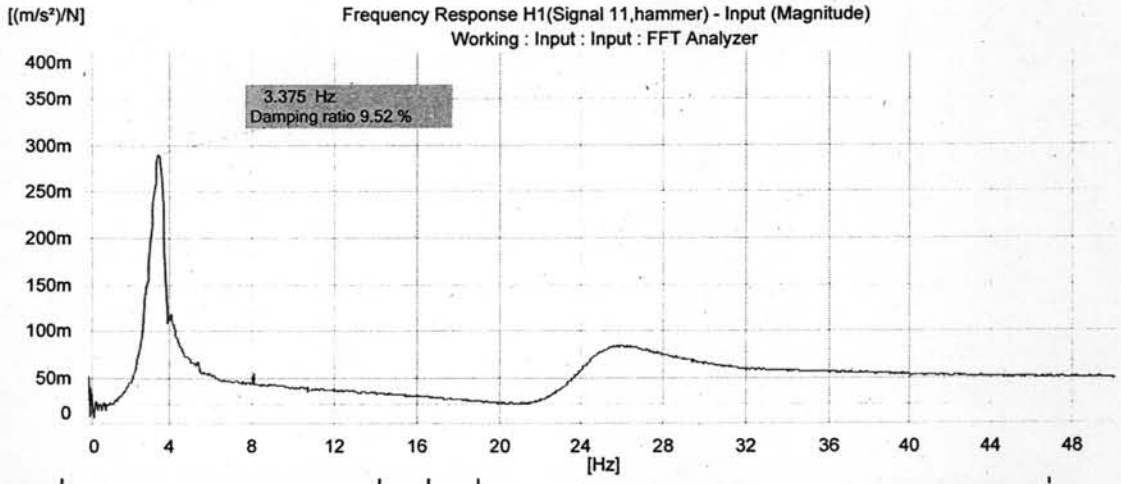
กราฟแสดงผลการทดสอบคุณสมบัติแท่นเครื่อง

ข้อมูลได้จากการทดสอบชุดทดสอบ โดยเครื่องมือ Pulse-Multi analyzer โดยใช้วิธี Impact excitation โดยจะเคาะชุดทดสอบ 3 ตำแหน่งโดยจะพยายามเน้นการเคาะผ่านแกนที่เป็นแกนหลัก หรือที่เรียกว่า Principles of elastic axes ของตัวแท่นเครื่อง

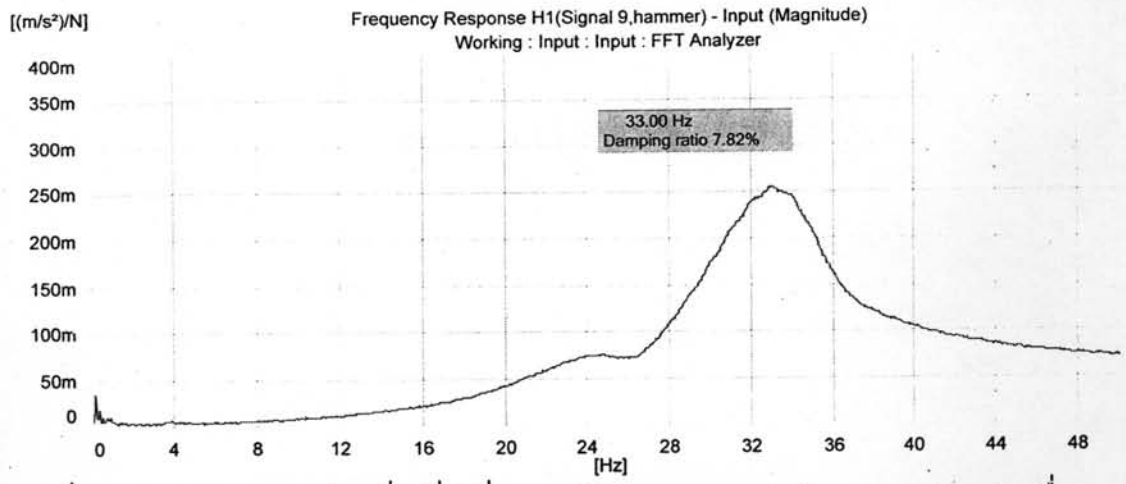
กราฟชุดที่ 1 จะแสดงคุณสมบัติของแท่นเครื่องที่ 1



รูปที่ ข.1 ผลการทดสอบแท่นเครื่องที่ 1 ที่ทดสอบโดยการเคาะขนานกับ แกน p ของแท่นเครื่อง

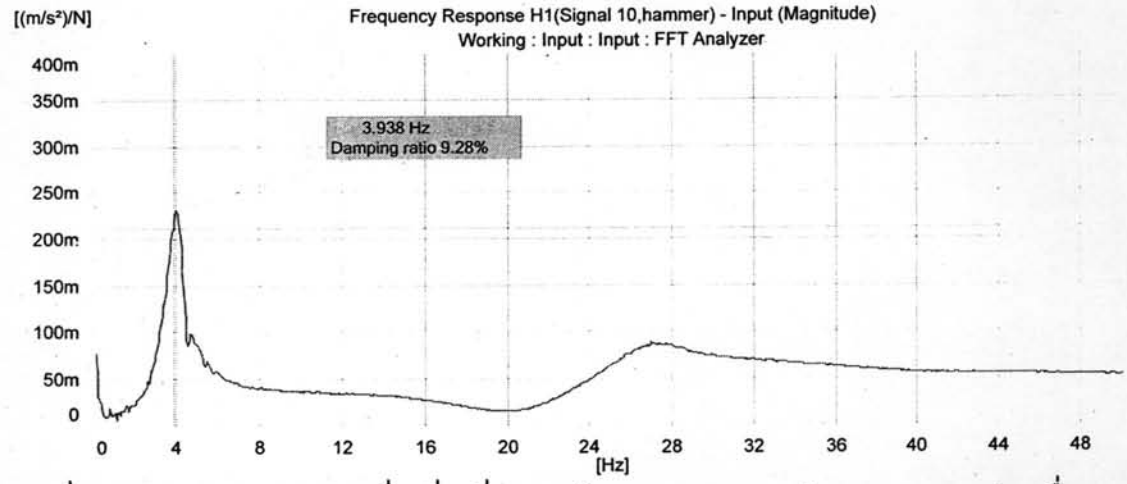


รูปที่ ข.2 ผลการทดสอบแท่นเครื่องที่ 1 ที่ทดสอบโดยการเคาะขนานกับ แกน q ของแท่นเครื่อง

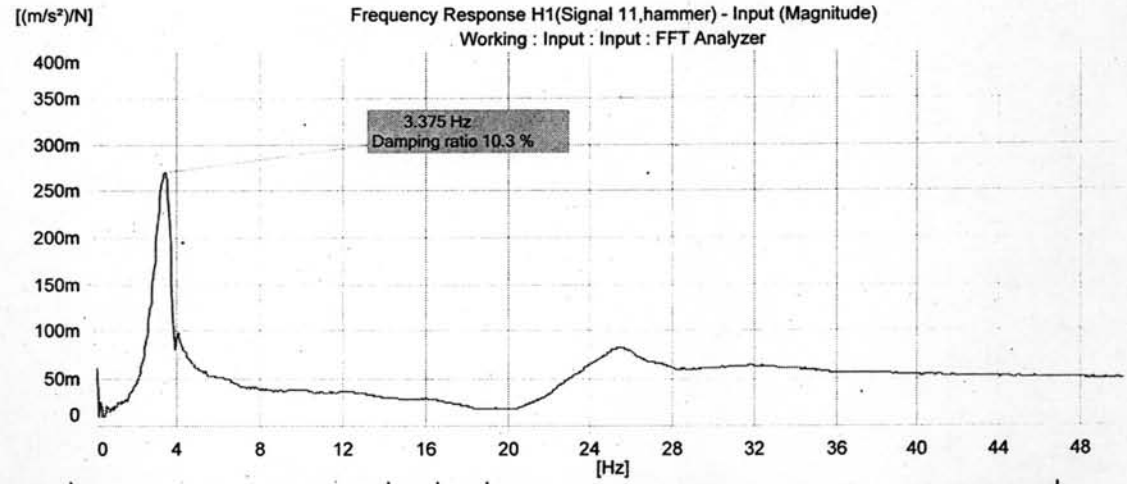


รูปที่ ข.3 ผลการทดสอบแท่นเครื่องที่ 1 ที่ทดสอบโดยการเคาะขนานกับ แกน r ของแท่นเครื่อง

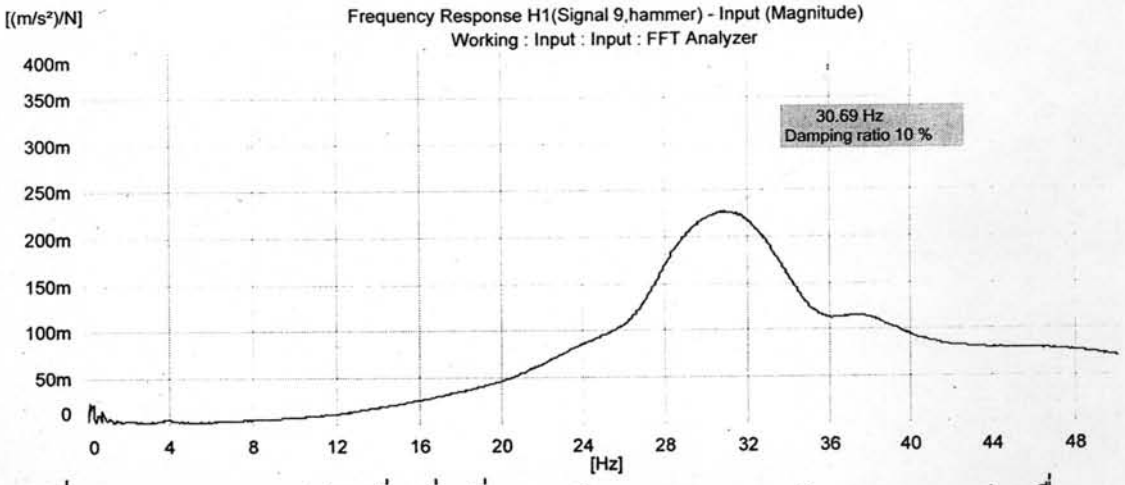
กราฟชุดที่ 2 จะแสดงคุณสมบัติของแท่นเครื่องที่ 2



รูปที่ ข.4 ผลการทดสอบแท่นเครื่องที่ 2 ที่ทดสอบโดยการเคาะขนานกับ แกน p ของแท่นเครื่อง

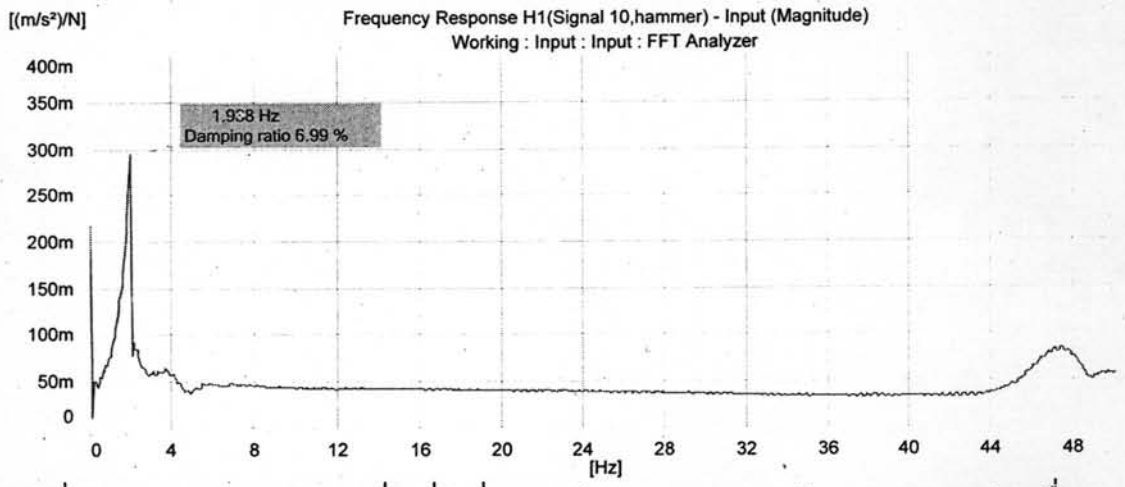


รูปที่ ข.5 ผลการทดสอบแท่นเครื่องที่ 2 ที่ทดสอบโดยการเคาะขนานกับ แกน q ของแท่นเครื่อง

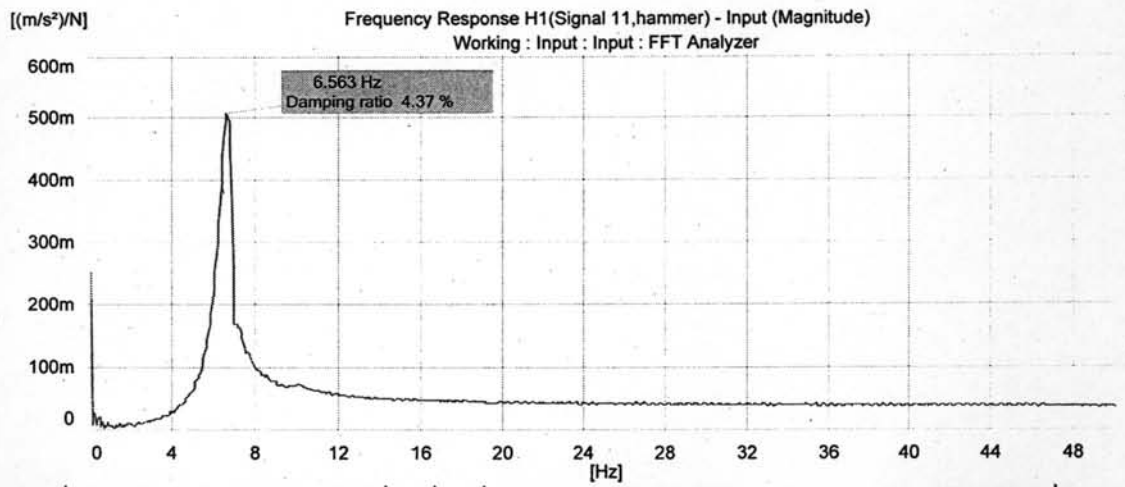


รูปที่ ข.6 ผลการทดสอบแท่นเครื่องที่ 2 ที่ทดสอบโดยการเคาะขนานกับ แกน r ของแท่นเครื่อง

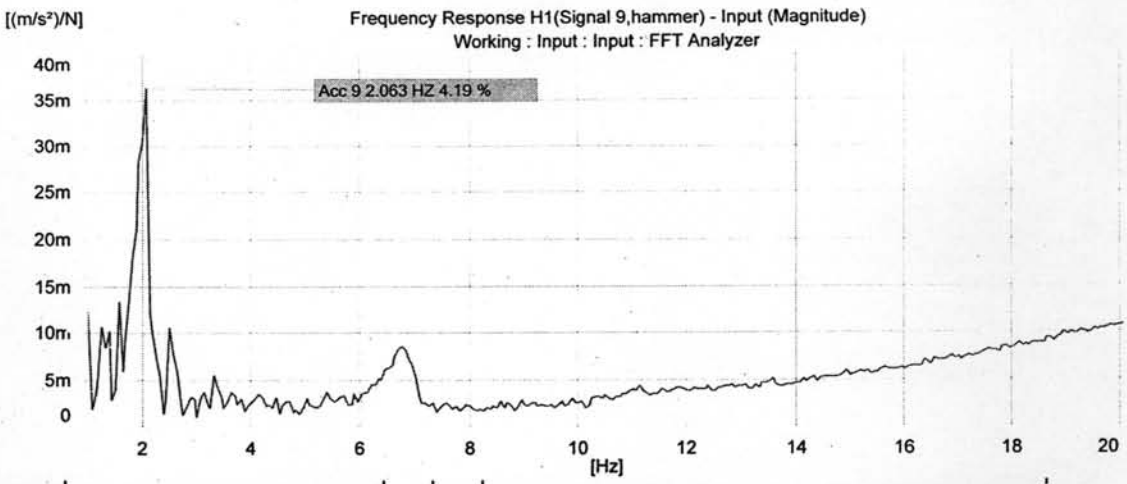
กราฟชุดที่ 3 จะแสดงคุณสมบัติของแท่นเครื่องที่ 3



รูปที่ ข.7 ผลการทดสอบแท่นเครื่องที่ 3 ที่ทดสอบโดยการเคาะขนานกับ แกน p ของแท่นเครื่อง



รูปที่ ข.8 ผลการทดสอบแท่นเครื่องที่ 3 ที่ทดสอบโดยการเคาะขนานกับ แกน q ของแท่นเครื่อง

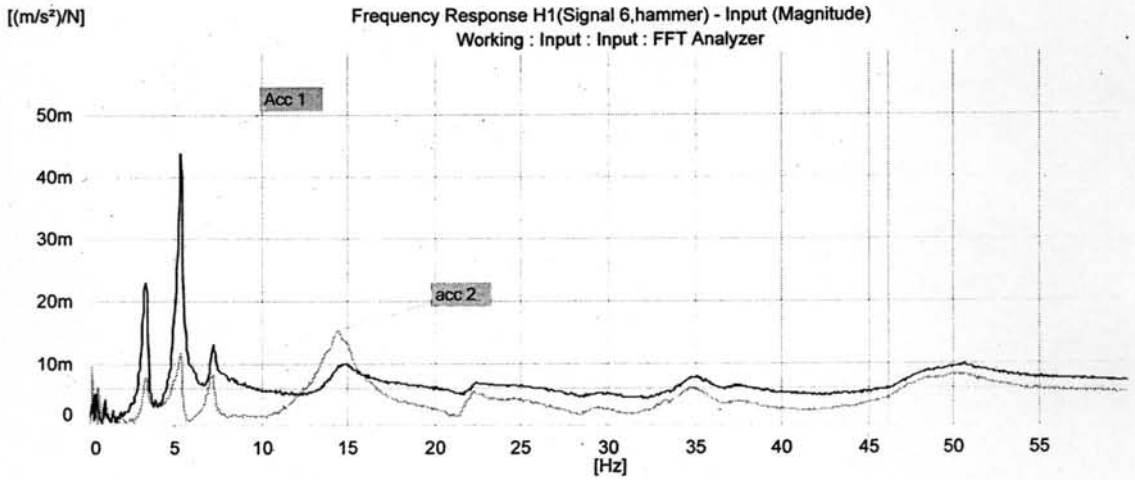


รูปที่ ข.9 ผลการทดสอบแท่นเครื่องที่ 3 ที่ทดสอบโดยการเคาะขนานกับ แกน r ของแท่นเครื่อง

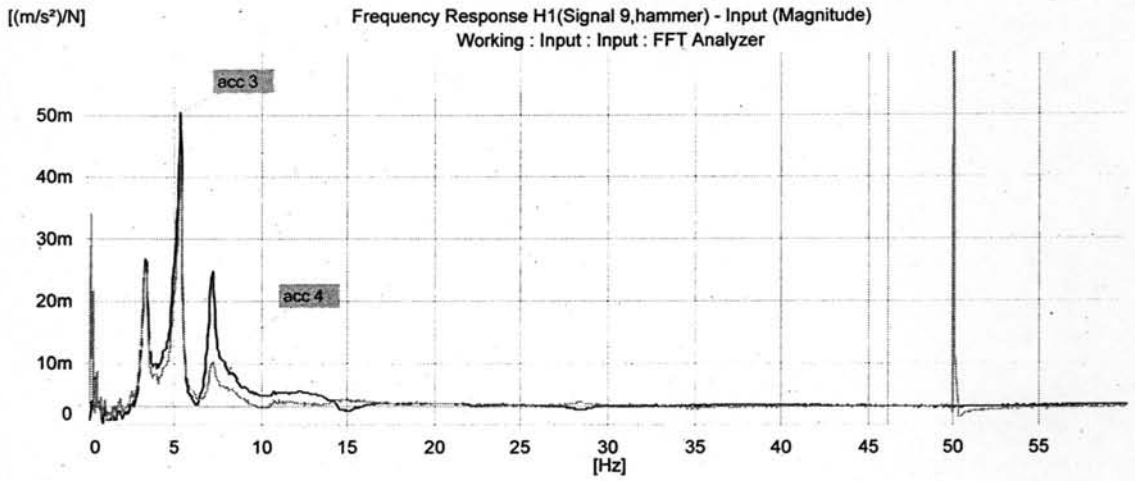
ภาคผนวก ค

ผลการทดสอบโดยวิธี Impact excitation

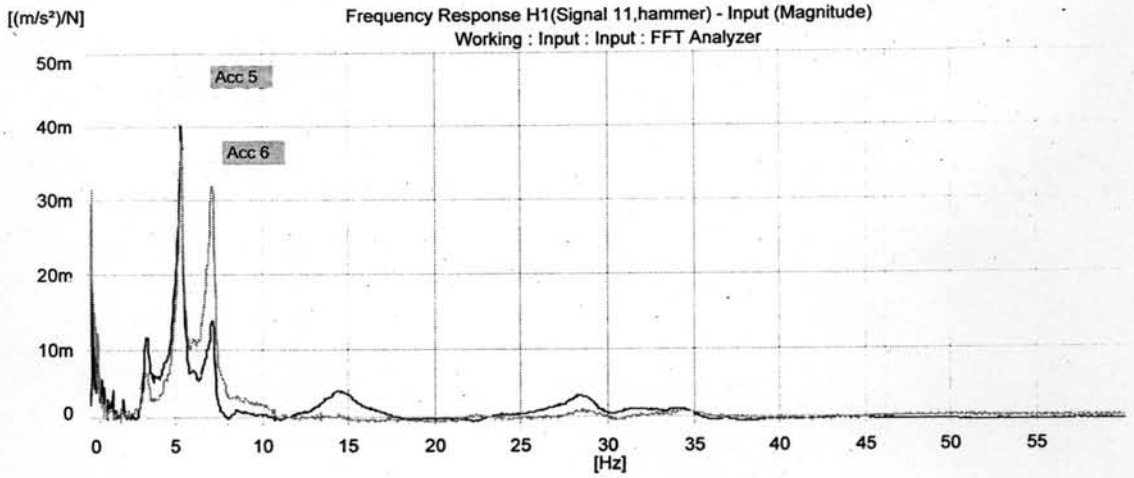
ผลจากการทดสอบเพื่อหาตัวแปรโมดัลในการศึกษาการสั่นสะเทือนของเครื่องยนต์ในวิธี Impact excitation ซึ่งสามารถผลได้ด้วยฟังก์ชั่นตอบสนองเชิงความถี่ (Frequency Response Functions ,.FRFs) โดยการติดตั้งตัวหยั่งสัญญาณความเร่ง (Accelerometers) บนจุดที่สนใจวัดค่าบนเครื่องยนต์จากนั้นกระตุ้นให้เกิดสัญญาณโดยใช้ Impact hammers ในทิศทางต่าง ๆ



(ก)

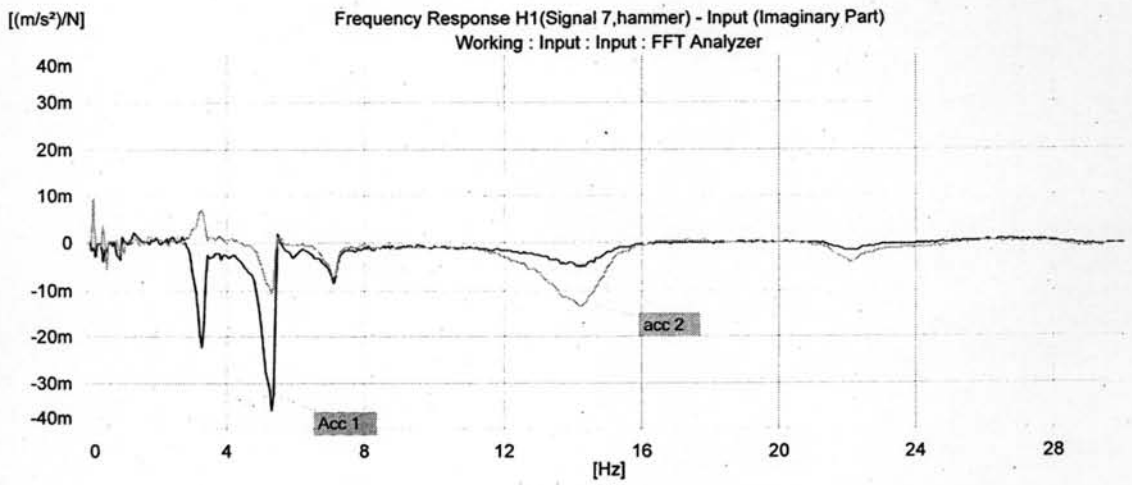


(ข)

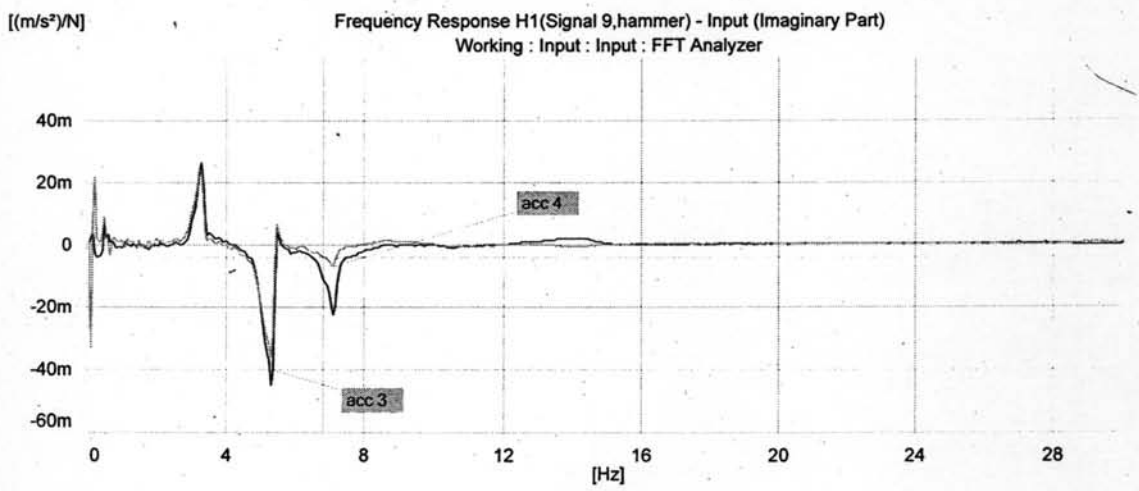


(ค)

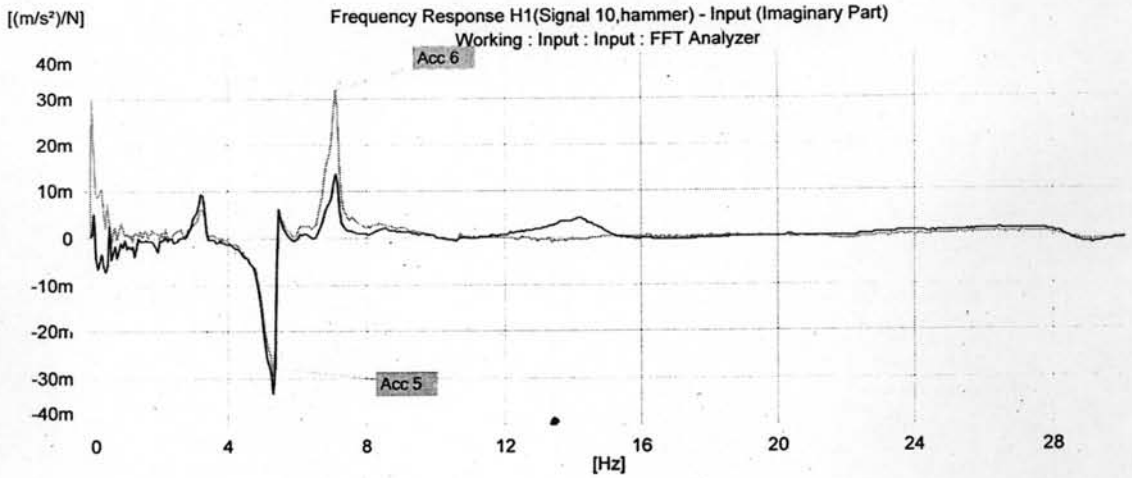
รูปที่ ค.1 ค่า Magnitude จากการเคาะบนเครื่องยนต์ที่ตำแหน่งที่ 1 (ก) จุดวัด 1และ2 (ข) จุดวัด 3และ4 (ค) จุดวัด 5และ6



(ก)

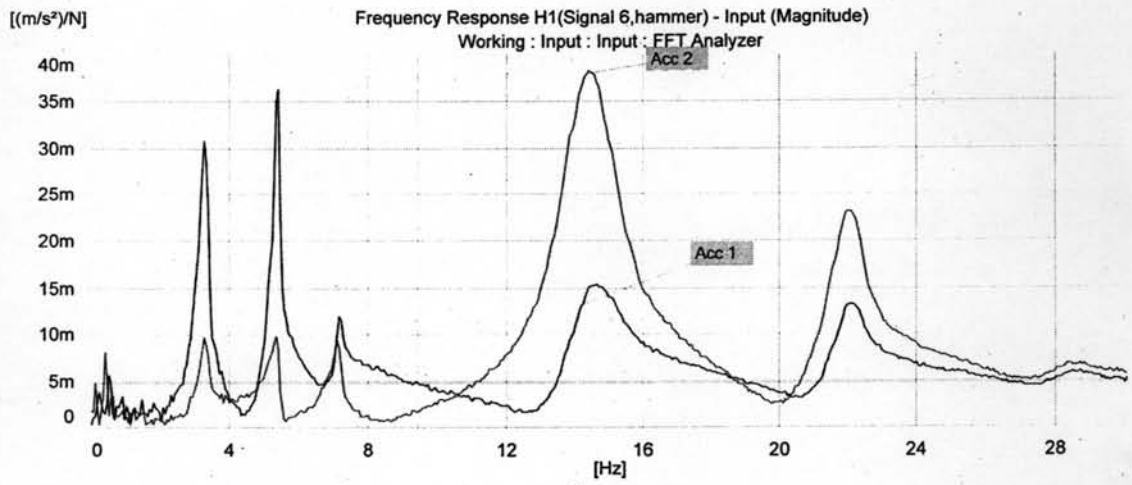


(ข)

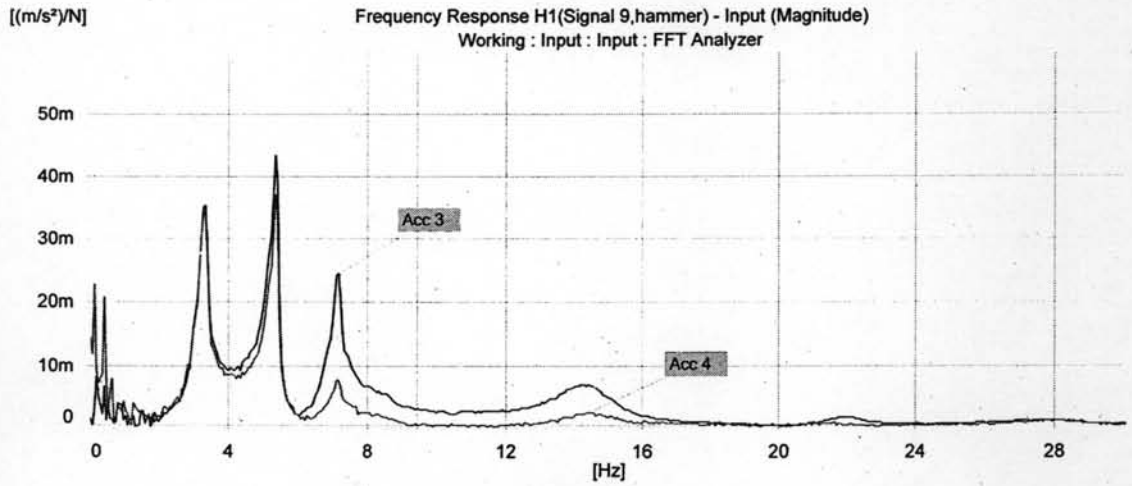


(ค)

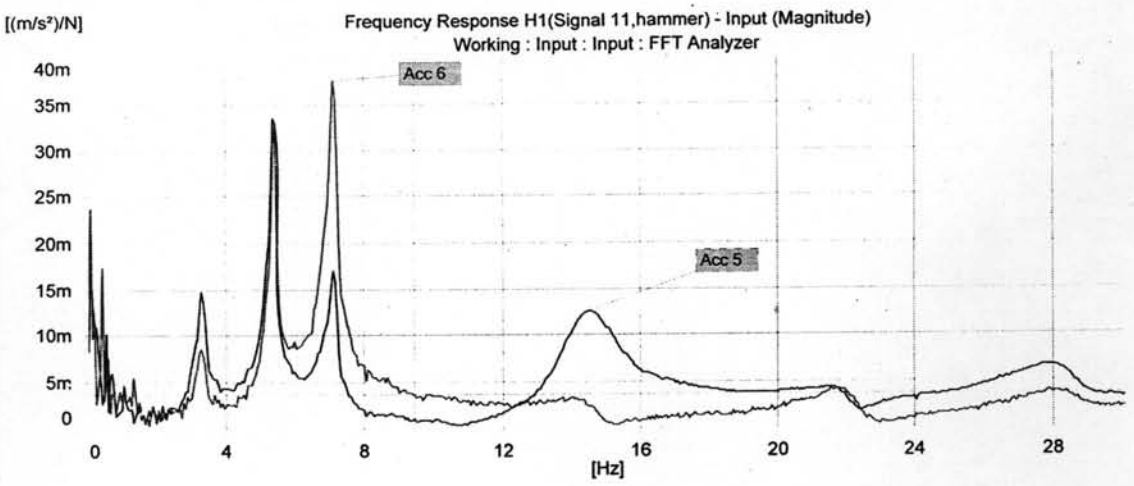
รูปที่ ค.2 ค่าจินตภาพจากการเคาะบนเครื่องยนต์ที่ตำแหน่งที่ 1 (ก) จุดวัด 1และ2 (ข) จุดวัด 3และ4 (ค) จุดวัด 5และ6



(ก)

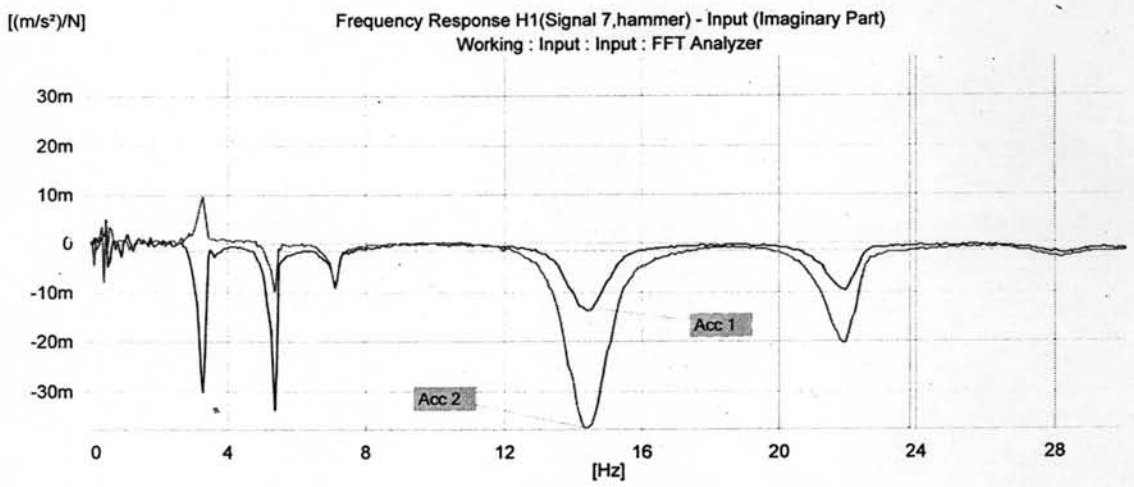


(ข)

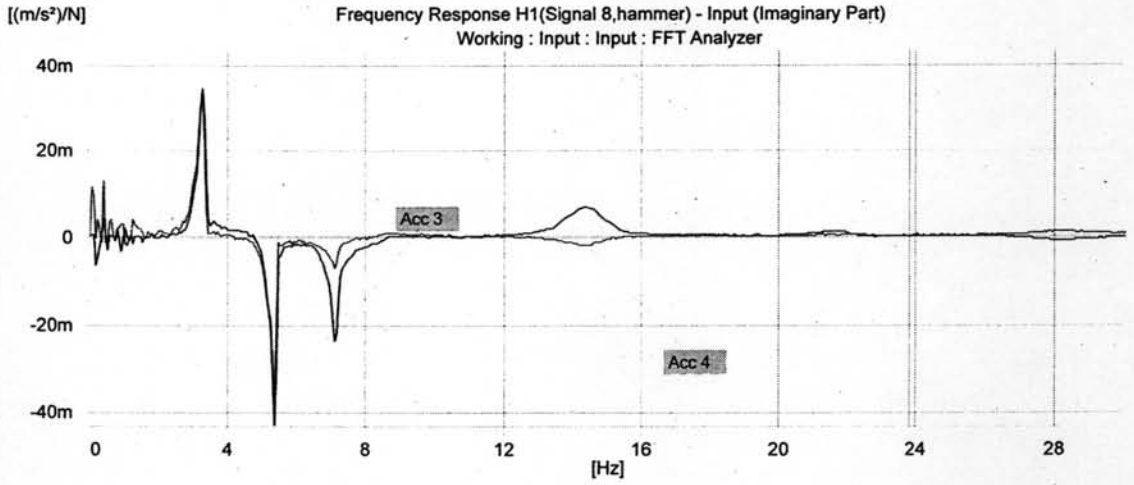


(ค)

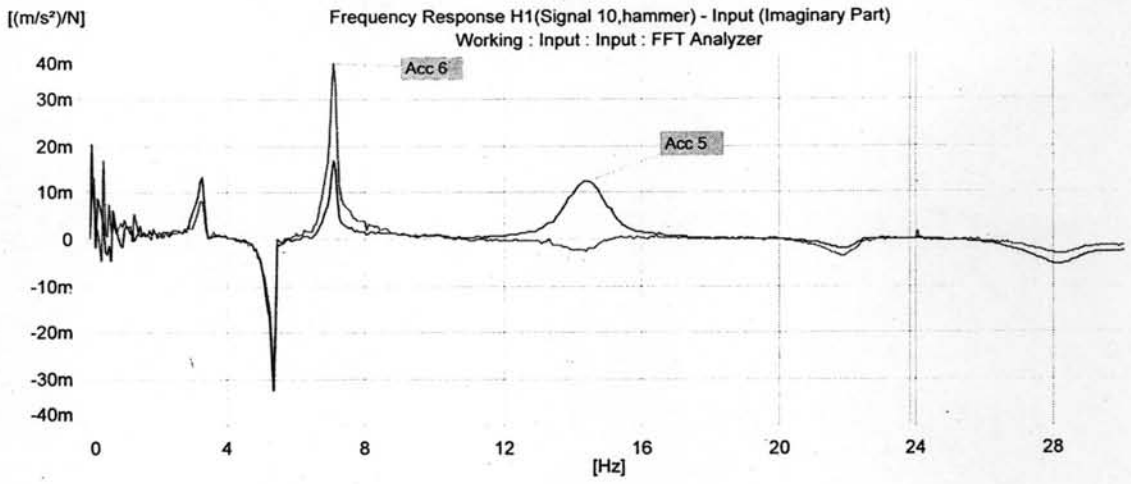
รูปที่ ค.3 ค่า Magnitude จากการเคาะบนเครื่องยนต์ที่ตำแหน่งที่ 2 (ก) จุดวัด 1และ 2 (ข) จุดวัด3 และ4 (ค) จุดวัด 5 และ 6



(ก)

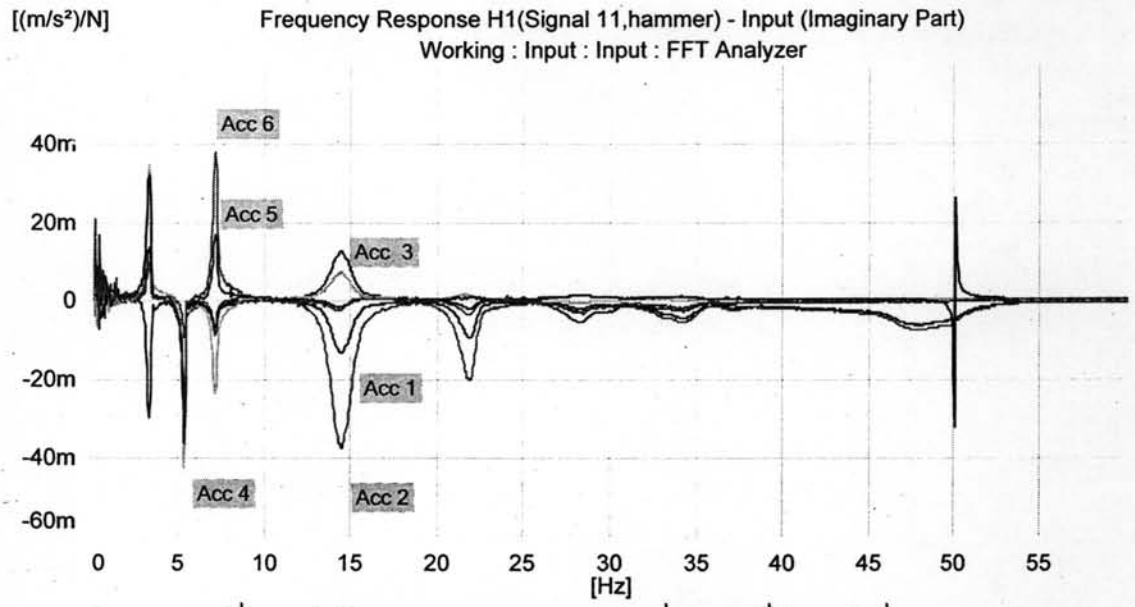


(ข)

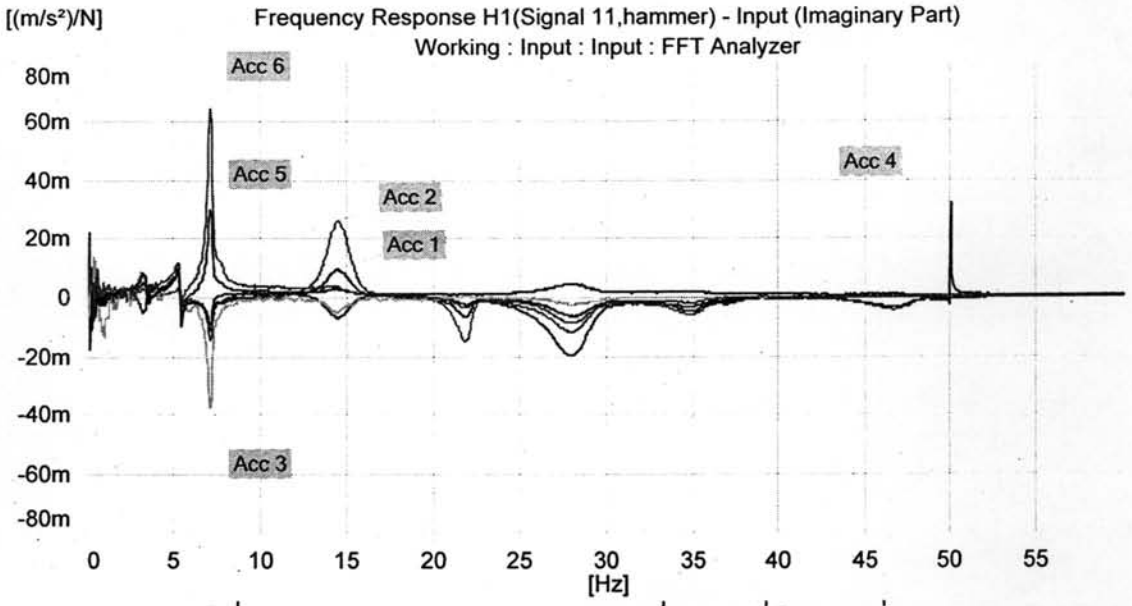


(ค)

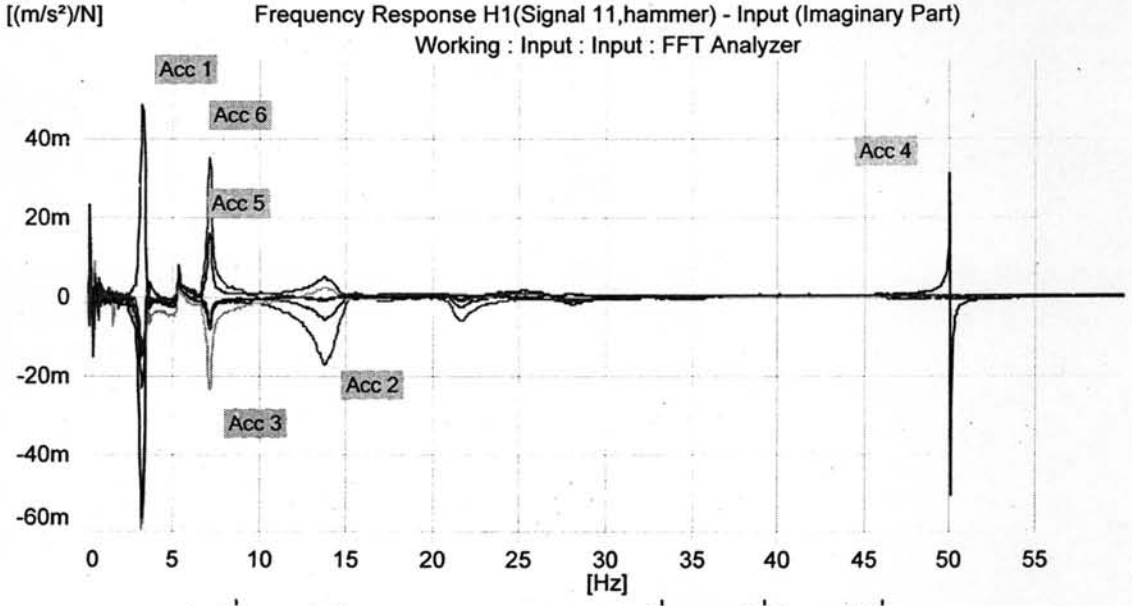
รูปที่ ค.4 ค่าจินตภาพจากการเคาะบนเครื่องยนต์ที่ตำแหน่งที่ 2 (ก) จุดวัด 1และ2 (ข) จุดวัด 3และ4 (ค) จุดวัด 5และ6



รูปที่ ค.5 ค่าจินตภาพจากการเคาะบนเครื่องยนต์ที่ตำแหน่งที่ 2



รูปที่ ค.6 ค่าจินตภาพจากการเคาะบนเครื่องยนต์ที่ตำแหน่งที่ 3



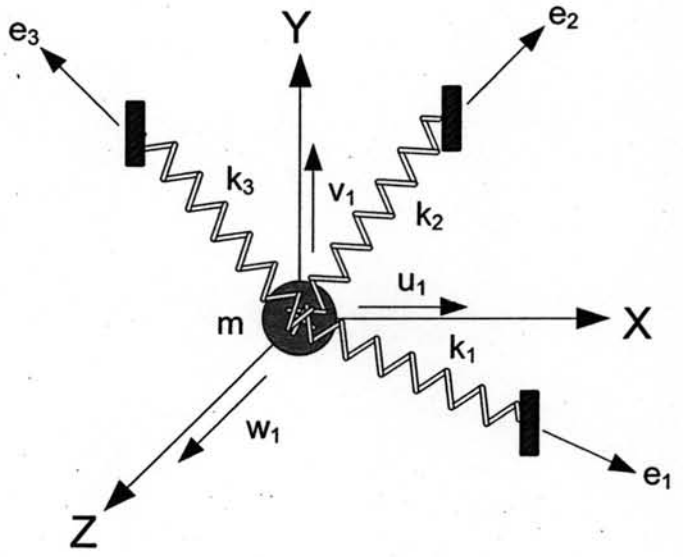
รูปที่ ค.7 ค่าจินตภาพจากการเคาะบนเครื่องยนต์ที่ตำแหน่งที่ 4

ภาคผนวก ง

การตรวจสอบโปรแกรมที่เขียนจากภาษา C++

เพื่อตรวจสอบความถูกต้องของโปรแกรม จึงนำโปรแกรมมาใช้แก้ปัญหาการสั่นสะเทือนที่ 3 องศาอิสระ มีรายละเอียดดังนี้

โจทย์ Small orthogonal translations u_1, v_1 and w_1 are to be used as displacement coordinates for the spring-suspended mass in the figure. Unit vectors in the direction of the springs are given as (where i, j, k denote unit vectors in the x, y and z directions, respectively) $e_1 = 0.8i - 0.6j, e_2 = 0.6j - 0.8k, e_3 = 0.6j + 0.8k$. Find natural Frequencies of the system



รูปที่ ง.1 แผนผังแรงอิสระของปัญหา

โดยการกำหนดค่าของ e1,e2 และ e3 ลงใน MATLAB ดังคำสั่งข้างล่าง

```
e1 =
   0.8000   -0.6000         0
>> e2=[0 0.6 -0.8]
e2 =
         0    0.6000   -0.8000
```

```
>> e3=[0 0.6 0.8]
e3 =
      0    0.6000    0.8000
```

เนื่องจากโปรแกรมที่ใช้งานของภาษา C++ มีการป้อนค่ามุมเป็นองศาจึงต้องประยุกต์การใช้งาน โดยใช้ MATLAB เปลี่ยนค่าเวกเตอร์การกระจัดที่โจทย์ให้มาเป็นค่ามุมที่กระทำกับแกนหลัก X Y Z โดยใช้คำสั่งต่อไปนี้เพื่อคำนวณค่าของมุมที่ e1 ทำกับแกน X Y Z, e2 ทำกับแกน X Y Z และ e3 ทำกับแกน X Y Z โดย code ของ comang.m ตามรายละเอียดนี้

```
function y  = comang(u1,u2,u3);

vx1 = ([u1(1) u1(2) u1(3)]*[1 0 0]')/(sqrt([u1(1) u1(2) u1(3)]*[u1(1) u1(2) u1(3)]'));
ax1 = acos(vx1)*180/pi;
vy1 = ([u1(1) u1(2) u1(3)]*[0 1 0]')/(sqrt([u1(1) u1(2) u1(3)]*[u1(1) u1(2) u1(3)]'));
ay1 = acos(vy1)*180/pi;
vz1 = ([u1(1) u1(2) u1(3)]*[0 0 1]')/(sqrt([u1(1) u1(2) u1(3)]*[u1(1) u1(2) u1(3)]'));
az1 = acos(vz1)*180/pi;

vx2 = ([u2(1) u2(2) u2(3)]*[1 0 0]')/(sqrt([u2(1) u2(2) u2(3)]*[u2(1) u2(2) u2(3)]'));
ax2 = acos(vx2)*180/pi;
vy2 = ([u2(1) u2(2) u2(3)]*[0 1 0]')/(sqrt([u2(1) u2(2) u2(3)]*[u2(1) u2(2) u2(3)]'));
ay2 = acos(vy2)*180/pi;
vz2 = ([u2(1) u2(2) u2(3)]*[0 0 1]')/(sqrt([u2(1) u2(2) u2(3)]*[u2(1) u2(2) u2(3)]'));
az2 = acos(vz2)*180/pi;

vx3 = ([u3(1) u3(2) u3(3)]*[1 0 0]')/(sqrt([u3(1) u3(2) u3(3)]*[u3(1) u3(2) u3(3)]'));
ax3 = acos(vx3)*180/pi;
vy3 = ([u3(1) u3(2) u3(3)]*[0 1 0]')/(sqrt([u3(1) u3(2) u3(3)]*[u3(1) u3(2) u3(3)]'));
ay3 = acos(vy3)*180/pi;
vz3 = ([u3(1) u3(2) u3(3)]*[0 0 1]')/(sqrt([u3(1) u3(2) u3(3)]*[u3(1) u3(2) u3(3)]'));
```

```
az3 = acos(vz3)*180/pi;

a1 = [ax1;ay1;az1]
a2 = [ax2;ay2;az2]
a3 = [ax3;ay3;az3]

y = 'End of find angle';
```

โดยที่หน้าต่าง Command window ได้พิมพ์คำสั่งลงไปดังนี้

```
>> comang(e1,e2,e3)
a1 =
  36.8699
  126.8699
   90.0000

a2 =
   90.0000
   53.1301
  143.1301

a3 =
   90.0000
   53.1301
   36.8699
ans =

End of find angle
```

แล้วนำค่ามุมที่ได้ไปกรอกลงในภาษา C++ ที่อยู่ในภาคผนวก ก โดยโปรแกรมจะทำการคำนวณเมตริก Mass,Stiffness และ Damping

```
>> load c:\mass.txt
```

```
>> load c:\stiff.txt
>> mass

mass =

     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
>> stiff
stiff =
     640   -480     0     0     0     0
  -  480   1080     0     0     0     0
     0     0     1280  0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

โดยโปรแกรมได้คำนวณมาให้จะเป็นเมทริกซ์ 6x6 แต่เนื่องจากระบบเป็นแบบเมทริกซ์ 3x3 โดยได้ใช้คำสั่งของ MATLAB ดังนี้

```
>> M=mass(1:3,1:3)

M =

     1     0     0
     0     1     0
     0     0     1

>> K=stiff(1:3,1:3)
```

K =

	640	- 480	0
-	480	1080	0
	0	0	1280

และจะได้ค่า Eigenvalues ดังนี้

```
>> eigen=eig(inv(M)*K)

eigen =
 1.0e+003 *
   0.3320
   1.3880
   1.2800
```

(เฉลย $\omega^2_{1,2,3} = 0.322k/m, 1.28k/m, 1.39k/m$ จาก William Weaver , Jr. , VIBRATION PROBLEM IN ENGINEERING)

ประวัติผู้เขียนวิทยานิพนธ์

นายพรประเสริฐ พงษ์พานิช เกิดเมื่อวันที่ 5 เดือนกันยายน พุทธศักราช 2519 จังหวัดตรัง สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต จากภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ เมื่อปีการศึกษา 2541 และเข้าศึกษาต่อในระดับปริญญามหาบัณฑิต สาขาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2546