KTH Aeronautical
and Vehicle Engineering

Miscellaneous Exercises in MATLAB

Ulf Carlsson

Stockholm 2009

Preface to the 2009 edition

This edition is prepared for the course SD1000 Perspectives on Vehicle Engineering, the academic year 2009/2010. Compared to the previous edition

- a couple of errors have been corrected,
- 6 new "Level 1"-exercises have been added,
- some of the exercises have been slightly modified.

Stockholm, 2009-08-11

Ulf Carlsson

Preface

This collection of MATLAB™ exercises has been developed for use in the course 4B1052 Perspective on Vehicle Engineering, for the first year students, at the Vehicle Engineering programme at KTH, Stockholm, Sweden. The main material used in the course is "An Introduction to MATLAB" by David F Griffiths. This material can be downloaded from the homepage of the mathematics department at the University of Dundee, http://www.maths.dundee.ac.uk/. The aim is to exercise the student's skills in using MATLAB so that they can use it as a tool for learning and understanding the material taught in the courses they meet during their studies at KTH. A milestone in becoming a skilled MATLAB user is to reach the level where the student can search, find and use information useful for implementing a code on her/his own. The best way to reach this level is to start working with a tutorial focussing on the basic elements of MATLAB. Then exercise these steps with exercises with gradually increasing complexity and finally to start using MATLAB as a supplementary tool in other courses.

The collection starts with exercises practising simple expressions and arithmetic operators, how to create vectors and matrices, the colon operator, the built-in functions and plotting. Then the focus is turned to reading and writing data files, script files (M-files), function M-files, for – while loops, if – then constructs and animation. Then, finally, a couple of exercises on creating Graphic User Interfaces (GUI) are given.

Another aim with this collection of exercises is to demonstrate how MATLAB is implemented to solve engineering problems. There is no doubt that the best way to learn mathematics and programming at a technical university is to use the methods for solving engineering problems. For this reason a number of engineering problems are provided in the collection. Several of these applied engineering exercises use concepts and methods that the students will meet during their later studies. Since these concepts and methods have to be introduced and explained in a qualitative way, some problem formulations are fairly long. This will, no doubt, prove a valuable experience for the students in the problem solving process: find relevant information – formulate a model – implement the model in a MATLAB code and finally – evaluate the model for specific parameter values in order to solve a realistic engineering problem.

The examples are formulated in the spirit of the CDIO (Conceive Design Implement and Operate products and systems) concept. CDIO is a joint project between Chalmers, LiU (Linköpings Universitet), MIT (Massachusett's Institute of Technology) and KTH (Kungl Tekniska Högskolan) that aims to develop new ways of teaching engineering science.

Answers and solutions are provided for most problems. Note that there are several possible ways to solve a problem and implement a code. The solutions provided herein are by no means claimed to be optimal in any way.

As course examination each student has to solve a problem, formulated as an engineering project, using MATLAB. Up to days date approximately 10 different examination problems have been developed.

This material may be downloaded and used freely in its present form by anyone who wishes. Any changes of its contents are not accepted. Please notify the author of all errors found and possible ways to improve the material. The material is developed in MATLAB 6.

Stockholm 2003-08-28

Ulf Carlsson
MWL, Dept of Vehicle Engineering, KTH

Recommendations for reading and learning

This collection of exercises is used in the MATLAB part of the course 4B1052 Perspektiv på Farkosttekniken, as a complement to the basic material "An Introduction to MATLAB" by David F Griffiths. In this course the examination of the MATLAB part is performed at three levels.

The aims of the 1st level is to provide the student with skills enabling use of MATLAB as
- a calculator,
- a tool for simpler processing of numbers stored in vectors and matrices,
- a tool for plotting graphs of scientific formulae and
- code the instructions in a simple MATLAB script file.

In addition to the aims for level 1 the aims of the 2nd level is to provide skills sufficient for
- more advanced matrix computations such as eigen-values and eigen-vectors and
- manipulating subvectors and submatrices.

Finally level 3 adds
- developing codes including `for` loops and logical `if` statements,
- `function` subroutines and
- animation.

The exercises in this collection are labelled Level 1–3 corresponding to this list.

To fulfil the requirements for the examination levels we recommend to follow the learning procedure given below.

1) Depending on your programming skills you should start working 10 – 20 hours with the basic material "An Introduction to MATLAB" by David F Griffiths.
2) Exercise your skills using some of the recommended (see course home page) exercises labelled "Level 1".
3) If necessary, select some additional "Level 1"-exercises.
4) If you do not want to go for level 2 perform your examination exercise for "Level 1".
5) Repeat step 2)-4) for "Level 2".
6) Repeat step 2)-4) for "Level 3".

It should be noted that the step from level 1 to level 2 is not very big. For this reason we strongly recommend everyone to at least give level 2 a try before choosing to examine level 1.

Stockholm 2004-08-12

Ulf Carlsson
MWL, KTH Aeronautical and Vehicle Engineering
e-mail: ulfc@kth.se

EXERCISES

1 (Level 1)
Evaluate the following MATLAB statements:
a) `9-(3+5)*6/4`,
b) `9-3+5*6/4`,
c) `0.5+4.1-3^2*2`,
d) `0.5+4.1-3^(2*2)`.

2 (Level 1)
Evaluate the following expressions:
a) $8+3\dfrac{2+3}{4}$,
b) $-5+7\dfrac{1+4(1+3)}{3(2+5)}$,
c) $3+\dfrac{2(3+4)^2}{3(1-3)^2}\dfrac{(1+3)^3}{8}$.

3 (Level 1)
Evaluate the following expressions:
a) $3\cdot(-4)+(-2)\cdot(1-3)\cdot\dfrac{1}{4}$,
b) $-(-3)-\dfrac{1}{5}\cdot\dfrac{\frac{2}{3}(1-\frac{5}{4})}{(4-\frac{1}{3})(4+\frac{1}{3})}$,
c) $(5-\frac{4}{3})^2\cdot\frac{9}{8}-\sqrt{(5+3)(5-3)}$.

4 (Level 1)
Evaluate the following expressions:
a) $2.6\cdot10^3\cdot3.3\cdot10^4\cdot7\cdot10^{-5}$,
b) $\dfrac{3.7\cdot10^3}{7.1\cdot10^{-5}\cdot2.1\cdot10^2}\cdot5.3\cdot10^2$.

5 (Level 1)
Evaluate the following expressions:
a) $3\dfrac{2+2}{2-2}$,
b) $3\dfrac{2-2}{2-2}$.

What is the difference between the answers obtained in a) and b) respectively?

6 (Level 1)
Evaluate the following expressions:
a) $(1+i)+(2-i)$,
b) $(1+i)(2-i)$,
c) $\dfrac{1+i}{2-i}$,
d) $(1+i)(1-i)$,

7 (Level 1)
Evaluate the following expressions:
a) $y = 3\sin\dfrac{\pi}{4}$,
b) $y = 3\cos(\dfrac{\pi}{4}+\dfrac{\pi}{2})$,
c) $y = 3\tan\dfrac{\pi}{4}$,
d) $y = 3\cot(\dfrac{\pi}{4}+\dfrac{\pi}{2})$.

8 (Level 1)
Evaluate the following expressions:
a) $\sin\dfrac{\pi}{4}$,
b) cos 1.6,
c) arcsin 1,
d) arctan 0.5.

9 (Level 1)
Evaluate the following expressions:
a) e^1,
b) $\ln 2$,
c) $2.5\ln(1.5)$,
d) $\log(2)$,
e) $10\cdot\log 4.5$.

10 (Level 1)
Evaluate the following expressions:
a) $y = \sqrt{5\sin(\pi/3)}$,
b) $y = e^{2+\sqrt{3}}$,
c) $y = \ln(7+e^2)$,
d) $y = \log e^3$,
e) $y = \arcsin\dfrac{e^{i\pi/2}-e^{-i\pi/2}}{2i}$. (How can this expression be reformulated?)

11 (Level 1)
Calculate the angles, v, in degrees if,
a) $\sin v = 0.9$,
b) $\cos v = -0.42$,
c) $\tan v = -0.13$.
d) Are there other possible solutions to these equations?

12 (Level 1)
The following code is intended to calculate the radius r of a circle from its circumference L.

```
radius.m
L = input('Enter the circumference L: ');
r = L/2*pi
```

Use MATLAB Editor to create a new M-file, type these statements and save the file with the name `radius.m`.

Unfortunately the code contains an error. Analyse the code and correct it. Finally, try it on, for example for $L = 2\pi$.

13 (Level 1)
The strength of sound is measured in *sound pressure level*, L_p [dB], defined as,

$$L_p = 10 \cdot \log \frac{\tilde{p}^2}{p_{ref}^2} \text{ [dB]}, \quad p_{ref} = 2 \cdot 10^{-5} \text{ Pa.}$$

where log is the base 10 logarithm, the rms (root mean square) sound pressure, $\tilde{p}$ [Pa], is the measured output from the microphone and p_{ref} is a reference sound pressure. A definition of the root mean square value is given below in exercise 52.
Calculate the sound pressure level for the following measured rms sound pressures.
a) A normal conversation, $\tilde{p} = 35 \cdot 10^{-3}$ Pa.
b) A noisy workshop, $\tilde{p} = 1.1$ Pa.
c) A jet during take-off, $\tilde{p} = 200$ Pa.

14 (Level 1)
The following MATLAB code is supposed to evaluate the expression $z = x + y$ for,
a) $x = 5$, $y = 2$, and for
b) $x = y = 5$,

```
% xplusy.m
clear
% Part a)
x = 5;
y = 2;
z = x + y
% Part b)
x = y;
y = 5;
z = x + y
```

Use the MATLAB Editor to create the M-file and execute it. Locate the error and correct it.

15 (Level 1)

Two equally strong but uncorrelated sound sources do not, as one may think, cause a doubling of the sound pressure level. To calculate the total sound pressure level, $L_{p,tot}$, caused by N uncorrelated sound sources contributing with individual sound pressure levels, $L_{p,n}$, the following formula is applicable,

$$L_{p,tot} = 10 \cdot \log \sum_{n=1}^{N} 10^{L_{p,n}/10} \text{ [dB]}.$$

Calculate the total sound pressure level in the following cases.

a) Two simultaneously running engines with equal sound pressure levels at 85 dB.

b) Four jets, each with sound pressure level 140 dB, starting at the same time.

16 (Level 1)

Create the following vectors in MATLAB.

a) $a = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$,

b) $b = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$,

c) $c = \begin{bmatrix} 2 & 3 & 4 & \cdots & 25 \end{bmatrix}$,

d) $d = \begin{bmatrix} 20 & 18 & \cdots & -18 & -20 \end{bmatrix}^T$,

e) $e = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ \vdots \\ 100 \end{bmatrix}$.

17 (Level 1)

Evaluate the vector y in the following cases:

a) $y = \sin x$, $x = \begin{bmatrix} \frac{10\pi}{200} & \frac{20\pi}{200} & \frac{30\pi}{200} & \cdots & 10\pi \end{bmatrix}^T$.

What is the value of component 24 of y?

b) $y = \sin \sqrt{x}$, $x = \frac{1}{\pi}\begin{bmatrix} 1 & 2 & 3 & \cdots & 1000 \end{bmatrix}^T$.

What is the value of component 524 of y?

18 (Level 1)

The equation $x^3 + x - 1 = 0$, is a polynomial equation.

a) Use the MATLAB Help browser to find the command that solves, i.e. calculates the roots of, a polynomial equation.

b) Solve the polynomial equation given above.

19 (Level 1)
Build a time vector t with 100 equidistant components on the interval [0, 1], that is $t(1) = 0$ and $t(100) = 1$.

a) What is the value of the 50th component of t?

b) What are the values of the components in the range 51 to 60?

c) Build a time vector t_1 containing every fifth component of t starting from $t(5)$. What is the value of the 17th component of this vector. Plot $\sin(2\pi t)$ and $\sin(2\pi t_1)$ as functions of t and t_1 in a diagram. What is the difference between the curves?

20 (Level 1)
Sound pressure values p_n [Pa] are measured at frequencies f_n [Hz]. The data points (f_n, p_n), $n = 1, 2, 3, \ldots N$, are stored on the binary data file `soundpress.bin`. The values are stored on format (precision) `'float32'` in the following sequence: $f_1, p_1, f_2, p_2, f_3, p_3, \ldots f_N, p_N$.

a) Use the MATLAB Help browser to find out how to use commands `fopen`, `fread` and `fclose` to read the data file `soundpress.bin`. Read the data into MATLAB.

b) Use a MATLAB command to find the number of data values in the file.

c) Separate the data into one vector `f` with all frequencies f_n and one vector `p` with all sound pressures p_n.

d) Plot the sound pressure p versus the frequency f. Add axis labels with proper quantities and units.

21 (Level 1)
In a mechanics laboratory exercise you measure the position s [m] of a car at time t [s]. The measurement data is available on the MATLAB data file `meclab.mat`.

a) Use a MATLAB command and read the data into the MATLAB Workspace.

b) Plot the car's position s versus the time t. Use markers instead of a line. Add axis labels with proper quantities and units.

From basic mechanics it is known that the relationship between position s and acceleration a [m/s^2] is,

$$a = \frac{d^2 s}{dt^2}. \qquad (1)$$

If the acceleration is constant the position s can be shown to be (integrate Equation (1) twice with respect to time),

$$s = a \cdot \frac{t^2}{2} + v_0 \cdot t + s_0, \qquad (2)$$

where s_0 [m] and v_0 [m/s] are the position and speed at time $t = 0$ respectively.

c) Use the MATLAB Help and identify the command that fits a polynomial function to a set of data points. Assume that the car's acceleration is constant and find the coefficients in Equation (2). What is the value of acceleration?

d) Use the MATLAB Help and identify the command to evaluate the polynomial at prescribed time values. What is the position at time $t = 0$ and $t = 2.3$ s?

e) Compare the estimated polynomial relationship with the original measured data by plotting them in the same diagram. Note that you can use a higher time-resolution in the polynomial plot.

f) Store the polynomial coefficients determined in c) on a MATLAB data file named `accpoly.mat`.

22 (Level 1)

a) Build the function in Figure 1.

$$y(x) = \begin{cases} \sin x, & 0 \le x < \pi \\ \text{repeat for}, & \pi \le \text{x} \le 6\pi \end{cases}$$

(**Hint**: Construct a half period sine and use it as a building block.)

b) Find a simpler alternative method to build the function.

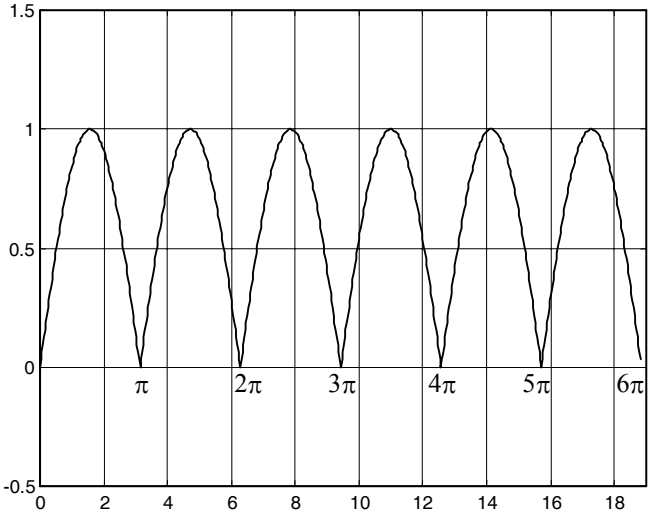


Figure 1 Rectified sine function.

23 (Level 1)

Plot the function you built in exercise 22. Try to get the same axis scaling as in Figure 1.

24 (Level 1)
Build the periodic triangle wave, in Figure 1, defined by,

$$y = \begin{cases} 1 - x, & 0 \le x < 1 \\ x - 1, & 1 \le x < 2 \\ \text{repeat for} & 2 \le x \le 6 \end{cases}.$$

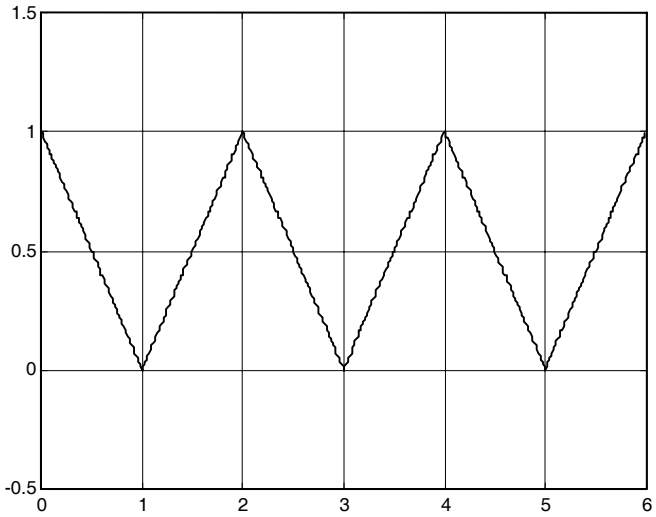


Figure 1 Periodic triangle wave.

25 (Level 1)
Plot the rectified sine function from exercise 22 in a multi-plot together with the triangle wave from exercise 24. Use solid line for sine and dashed line for triangle.

26 (Level 1)
Quick estimates of solutions to transcendent equations can be found graphically by plotting suitable functions and locate their intersections.

Find approximate values for the three lowest non-zero positive solutions to the equation, $x = \cot x$, by plotting the functions in a multi-plot and zooming the area around the intersections.

27 (Level 1)
Enter the vector u, $u = \begin{bmatrix} 1 & 2 & 3 & \cdots & 10 \end{bmatrix}$, and calculate,

a) the scalar product, $u \cdot u$,
b) the product, $u^T \cdot u$,
c) the scalar product using the command `dot`,
d) the element-wise product, `u.*u`,

28 (Level 1)
Define an equidistant vector x with first value 0, last value 5 and increment 0.05. Evaluate the following functions.

a) $y = 10 - x$. What is the value of the 17th component of y?
b) $y = 10 - x - \sin x$. What is the value of the 17th component of y?
c) $y = x^2 - 3x + 1$. What is the value of the 17th component of y?
d) $f(x) = \sin(x)/x$. What is the value of the 17th component of f?

29 (Level 1)
Modulation is important in many fields of technology. Two interacting sinuousoidal signals, one with low and one with high frequency, produce an amplitude modulated signal. Enter the vector, $x = [1 \quad 2 \quad 3 \quad \cdots \quad 1000]\frac{2\pi}{100}$, and plot the amplitude modulated function, $y = \sin(4x)(1.8 + \sin(0.1x))$.

(If you have a computer with a sound card, you can listen to the modulated signal by using the command `sound(y,1000)`.)

30 (Level 1)
In mechanics the work performed by a force is defined as the scalar product between the force, **F**, and its displacement, **s**. Both **F** and **s** are in general vectors with components in all three spatial (room) directions *xyz*. Calculate the work performed in the following cases.
a) When the force is, **F** = [100 0 0] N, and the displacement is, **s** = [0 3 2] m.
b) When the force is, **F** = [50 30 -10] N, and the displacement is, **s** = [1 2 –0.5] m.

31 (Level 1)
The following code is intended to evaluate the function, $y(x) = x\sin x$ for values of x given in the vector `x`:

```
x = (0:1/100:1)*10*pi;
y = x*sin(x);
```

Check the code and correct it if necessary.

32 (Level 1)

The following code is intended to evaluate the function, $y(x) = \frac{1}{1+\sin^2 x}$ for values of x given in the vector `x`.

```
x = (0:1/100:1)*10*pi;
y = 1/(1+sin(x).^2);
```

What is wrong with the code? Correct the code.

33 (Level 1)
Estimate the real-valued roots to the polynomial equation, $x^4 + 5x^3 - 64x^2 - x + 225 = 0$, using a graphical technique.

34 (Level 1)
In acoustics a silencer is used as a means to reduce the acoustic energy radiated from for instance engine exhaust outlets. The effect of inserting a silencer can be specified in terms of the transmission loss, D_{TL} [dB], which is defined as,

$$D_{TL} = 10 \cdot \log \frac{W_{in}}{W_{out}},$$

where W_{in} and W_{out} are the incoming and transmitted (passing) acoustic power respectively. A reactive silencer is in principle a sudden area expansion followed by a sudden area contraction, see Figure 1.

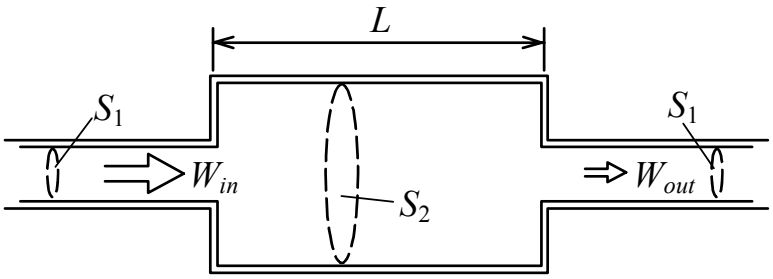


Figure 1 An expansion chamber is simple realisation of a reactive silencer.

For low frequencies, f, where only plane acoustic waves propagates in the pipe (with cross section area S_1) the following formula can be used to calculate the transmission loss for a silencer with cross section area S_2 and length L,

$$D_{TL}(f) = 10 \cdot \log(1 + (\frac{S_1}{2S_2} - \frac{S_2}{2S_1})^2 \sin^2(\frac{2\pi f}{c} L)),$$

where c is the speed of sound (for air at normal temperature and pressure conditions (NTP), c = 340 m/s).

In a specific case the pipe has cross section area $0.5 \cdot 10^{-4}$ m^2, the silencer cross section area $5 \cdot 10^{-4}$ m^2 and length 1 m. Plot the transmission loss as a function of frequency for frequencies in the interval $f \in [0,500]$ Hz. Make sure the frequency resolution is sufficiently high.

35 (Level 1)

a) Define an equidistant frequency vector f with the command

```
f = linspace(-1,1,200);
```

What is the value of the 41st component of the frequency vector?

b) Introduce a vector for the angular frequency, $\omega = 2\pi f$.
Evaluate the complex-valued receptance function h defined as a function of ω by,

$$h(\omega) = \frac{1}{\omega - 1 - 0{,}5i} - \frac{1}{\omega + 1 - 0{,}5i},$$

at angular frequency values given by the components of ω. What is the value of the 41st component of $h(\omega)$?

c) Find the maximum absolute value of $h(\omega)$ and the corresponding value of the frequency f.

d) Find the frequency interval $f \in [f_{min}, f_{max}]$ for which the absolute value of the receptance function is larger than 1.9.

e) Plot the receptance function h in three different formats:

(i) as a CoQuad plot, i.e. the real and imaginary parts as functions of the angular frequency ω.

(ii) as a Bode plot, i.e. the absolute value and phase angle (degrees) as functions of the angular frequency ω.

(iii) as a Nyquist or Argand plot, i.e. the imaginary part as a function of the real part.

36 (Level 1)
Enter the matrices,

a) $a = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix}$,

b) $b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$,

c) $c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$

d) $d = \begin{bmatrix} 1 & 0 & 1 & 0 & 2 & 2 \\ 0 & 1 & 0 & 1 & 2 & 2 \\ 1 & 0 & 3 & 3 & 3 & 1 \\ 0 & 1 & 3 & 3 & 3 & 1 \\ 2 & 2 & 3 & 3 & 3 & 1 \\ 2 & 2 & 1 & 1 & 1 & 4 \end{bmatrix}$

37 (Level 1)
Enter the matrices, $\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$, $\mathbf{B} = \frac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 2 & -2 & -1 \\ 1 & 2 & -2 \end{bmatrix}$, and the vector $\mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$,
and calculate,
a) **Ab**,
b) **Bb**,
c) **AB**,
d) **BA**.

A scaling matrix changes the length of the vector it is multiplied with. A rotation matrix rotates the vector it is multiplied with.

a) Is **A** a scaling matrix or a rotation matrix?
b) Is **B** a scaling matrix or a rotation matrix?

38 (Level 1)
Build the vector, $x = \{2\ 1\ 3\ 2\}$.

a) Calculate, $\mathbf{A} = \mathbf{x}^{\mathrm{T}}\mathbf{x}$, where T means transpose.

b) What is the sum of the third element in the second row and the second element in the third row of **A**?

c) Define the submatrix $\mathbf{B} = \begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix}$, where a_{kl} is the *l-th* element in row *k* of **A**.

d) Calculate the sum of the elements in the third row of **A**?

39 (Level 1)
Solve the equations for **x**.

a) $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

b) $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} = \begin{bmatrix} 3 & 0 & 1 \\ 2 & 2 & 2 \\ 0 & 1 & 3 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$.

40 (Level 1)
Solve the following systems of complex-valued linear equations.

a) $\begin{cases} x_1 + x_2 = 1 + i \\ x_1 + 2x_2 = 2 + i \end{cases}$,

b) $\begin{cases} x_1 + 2ix_2 = 1 + i \\ (1-i)x_1 + 2x_2 = 2 + i \end{cases}$,

c) $\begin{cases} x_1 + 3ix_2 - 2(1-i)x_3 = 2 - i \\ x_1 + 2x_2 - 3ix_3 = -i \\ x_1 + (i+1)x_2 + 2x_3 = 2 \end{cases}$.

41 (Level 2)
An often met eigenvalue problem can be written on the form, $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$, where **I** is the identity matrix. A generalized version of this is given by,

$$\mathbf{Ax} - \lambda\mathbf{Bx} = 0 \text{ or } (\mathbf{A} - \lambda\mathbf{B})\mathbf{x} = 0, \qquad (1)$$

where **A** and **B** are $n \times n$ matrices and λ is a scalar.

a) Use the MATLAB Help browser to find a suitable MATLAB command for solving eigenvalue problems of the kind given in (1).

Find the solutions to (1), i.e. the eigenvalues and eigenvectors, in the following cases:

b) $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$,

c) $\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4 & -1 \\ 1 & -1 & 2 \end{bmatrix}$.

42 (Level 2)

A linear system of equations on the form (compare with exercise 41 when **B** is the identity matrix),

$$\mathbf{Ax} - \lambda\mathbf{x} = 0 \text{ or } (\mathbf{A} - \lambda)\mathbf{x} = 0, \qquad (1)$$

where **A** is an $n \times n$ matrix and λ is a scalar, is an example of an eigenvalue problem. Eigenvalue problems of this type have non-trivial solutions for n specific values of the scalar λ. These solutions are denoted eigen-vectors and the corresponding values of λ are the eigenvalues.

Solve the eigenvalue problem given by (1), that is determine the eigenvalues and eigenvectors, for the following matrices **A**:

a) $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$,

b) $\mathbf{A} = \begin{bmatrix} 2 & 2 \\ 1 & 2 \end{bmatrix}$,

c) $\mathbf{A} = \begin{bmatrix} 2 & 1-i \\ 1 & 2i \end{bmatrix}$,

d) $\mathbf{A} = \begin{bmatrix} 1 & 1 & -2 \\ 2 & 2 & 1 \\ -1 & 2 & 3 \end{bmatrix}$.

43 (Level 2)

The solution vectors to eigenvalue problems given by,

$$(\mathbf{A} - \lambda\mathbf{B})\mathbf{x} = 0, \qquad (1)$$

have useful orthogonality properties. Suppose

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4 & -1 \\ 1 & -1 & 2 \end{bmatrix}.$$

a) Determine the solution vectors to the eigenvalue problem (1).

b) Let **V** be a matrix with columns equal to the solution vectors (eigenvectors) and calculate the matrix products, $\mathbf{R} = \mathbf{V}^{\mathrm{T}}\mathbf{AV}$ and $\mathbf{S} = \mathbf{V}^{\mathrm{T}}\mathbf{BV}$. What is the particular properties of **R** and **S**?

44 (Level 3)

Write `for`-loops that calculate the sums,

a) $\sum_{n=1}^{100} \frac{1}{n^2}$,

b) $\sum_{m=0}^{10} e^{-2\pi m}$,

45 (Level 2)

a) Create a vector of random numbers using command `rand(100,1)` .

b) Transform it to a random integer vector with components in the set [1, 2, 3, … 10]. (**Hint:** `fix`, `floor`, `round` and `ceil` are commands useful for this purpose.)

c) Calculate the number of components of this vector larger than 5.

46 (Level 3)

Solve the equation, $x = \tan x$, using the iterative technique suggested in paragraph 20.1 example 20.2 in "An Introduction to MATLAB" by D F Griffiths, i.e. let $x_{n+1} = \cot x_n$, $n = 0, 1, 2, \ldots$. Continue until the difference between two consequtive iterations are smaller than 10^{-4}.

47 (Level 3)

Write an M-file `countval.m` that counts the number of components in a vector y with values in the intervals [0, 1],]1, 2],]2, 3] and]3, 4]. Try the code on the following data: `x = (0:1/100:1)*2*pi ; y = 2+2*sin(x).`

48 (Level 1)

A mechanical single degree of freedom system consists of a point mass, a spring and a damper. The mass is allowed to move in the x-direction only (single degree of freedom). The spring and damper are both assumed to be massless. The equation of motion for the system, see Figure 1, reads,

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = F\,, \tag{1}$$

where m is the system mass, k is the system stiffness or spring constant and c is the system damping constant. The system displacement is given by the coordinate x and the external force acting on the system is F. When the system parameters m, k and c are independent of time t, Equation (1) is a linear, ordinary differential equation of second order with constant coefficients.

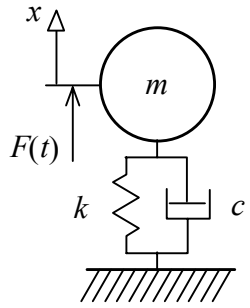


Figure 1 A mechanical single degree of freedom system.

The system displacement when the force is varying sinuousoidally in time with frequency f,

$$F(t) = \hat{F}\sin(2\pi f t) = \hat{F}\sin(\omega t) \tag{2}$$

where $\hat{F}$ is the size or amplitude of the force and ω is the angular frequency, is of great practical importance. For this particular time dependence the so called $i\omega$-method may be used to solve the equation of motion. The $i\omega$-method assumes that the displacement x and the force F are replaced with new complex variables $\mathbf{x}$ and $\mathbf{F}$,

$$\mathbf{F}(t) = \hat{F}(\cos(\omega t) + i\sin(\omega t)) = \hat{F}e^{i\omega t} \text{ hence } F(t) = \mathrm{Im}(\mathbf{F}(t)) \tag{3a}$$

$$\mathbf{x}(t) = \hat{x}(\cos(\omega t) + i\sin(\omega t)) = \hat{x}e^{i\omega t} \text{ hence } x(t) = \mathrm{Im}(\mathbf{x}(t))\,. \tag{3b}$$

If **x** and **F** are inserted in equation 1 the time derivatives are replaced with multiplication with the factor $i\omega$ and the differential equation is transformed to a algebraic equation,

$$(i\omega)^2 m\hat{x} + i\omega c\hat{x} + k\hat{x} = \hat{F}, \tag{4}$$

with the solution,

$$\hat{x} = \frac{\hat{F}}{-\omega^2 m + i\omega c + k}. \tag{5}$$

Usually the solution is given in terms of the ratio between the complex displacement **x** and the complex force **F**, the so called frequency response function,

$$\frac{\mathbf{x}(t)}{\mathbf{F}(t)} = \frac{\hat{x}e^{i\omega t}}{\hat{F}e^{i\omega t}} = \frac{\hat{x}}{\hat{F}} = \frac{1}{-\omega^2 m + i\omega c + k}. \tag{6}$$

Suppose the system parameters are: m = 1kg, c = 1Ns/m and k = 10000 N/m.

a) Introduce a frequency vector f with 500 equidistant components on the interval [0, 30] Hz. What is the value of the 265th component of f?

b) Calculate the frequency response function given by (6) as a function of frequency in frequency points given by the components of f. What is the value of the 265th component of the frequency response function ?

Obviously the frequency response function is complex-valued. The physically important properties of a complex-valued entity are its magnitude or absolute value and its phase angle. The absolute value of the frequency response function tells you the magnitude of the displacement amplitude relative to the force amplitude. For example, a value larger than 1 implies that the displacement amplitude is larger than the force amplitude. The phase angle, on the other hand, gives the time shift or lag between the displacement and force. The phase angle ϕ can be calculated as,

$$\phi = \arctan\frac{\mathrm{Im}(\hat{x}/\hat{F})}{\mathrm{Re}(\hat{x}/\hat{F})}. \tag{7}$$

Using the absolute value and the phase angle the frequency response function given in 6 can be written as,

$$\frac{\hat{x}}{\hat{F}} = \left|\frac{\hat{x}}{\hat{F}}\right| e^{i\phi}. \tag{8}$$

c) Calculate the magnitude and phase angle of the frequency response function as functions of frequency at frequency values given by the components of f. Plot the results in two separate diagrams. What is the magnitude of the 265th component of the frequency response function ? What is the phase (in degrees) of the 265th component of the frequency response function ?

d) Determine the maximum magnitude of the calculated vector, $\hat{x}/\hat{F}$. At what frequency value is the largest value located? What is the phase at this frequency?

The frequency response function provides information on the system as a function of the frequency of the external force. Suppose the force has a specific (fixed) frequency f how can the displacement $x(t)$ be calculated ? Using (6) and (3b) yields,

$$\mathbf{x}(t) = \frac{\mathbf{x}(t)}{\mathbf{F}(t)} \cdot \mathbf{F}(t) = \left|\frac{\hat{x}}{\hat{F}}\right| e^{i\phi} \cdot \hat{F} e^{i\omega t} = \left|\frac{\hat{x}}{\hat{F}}\right| \hat{F} e^{i(\omega t + \phi)} . \tag{9}$$

Finally the real displacement is obtained according to (3b) as,

$$x(t) = \mathrm{Im}(\mathbf{x}(t)) = \mathrm{Im}(\left|\frac{\hat{x}}{\hat{F}}\right| \hat{F} e^{i(\omega t + \phi)}) = \left|\frac{\hat{x}}{\hat{F}}\right| \hat{F} \sin(\omega t + \phi) .$$

e) Define a time vector with 500 equidistant components on the interval [0, 1] s. Suppose the force amplitude is 10 N and that the frequency is equal to the 265th component of the frequency vector. Evaluate the displacement, *x*, as a function of time at times given by the components of the time vector *t*. Plot the force and displacement as functions of time in two separate diagrams. How large is the time shift between the force and the displacement?

49 (Level 3)

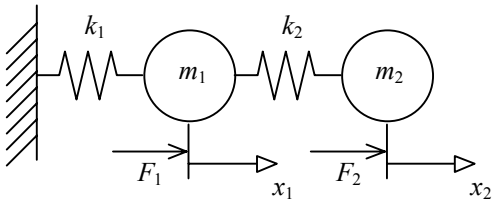


Figure 1 Mechanical 2-degree of freedom system excited by external force.

The displacement $\mathbf{x} = [x_1\ x_2]^{\mathrm{T}}$ of a linear mechanical 2-degree of freedom system, see Figure 1, excited with an external force, $\mathbf{F} = [F_1\ F_2]^{\mathrm{T}}$, is governed by a system of two equations of motion,

$$\begin{cases} m_1 \dfrac{d^2 x_1}{dt^2} + (k_1 + k_2) x_1 - k_2 x_2 = F_1 \\ -k_2 x_1 + m_2 \dfrac{d^2 x_2}{dt^2} + k_2 x_2 = F_2 \end{cases}$$

or in matrix-vector notation,

$$\mathbf{M}\frac{d^2\mathbf{x}}{dt^2} + \mathbf{K}\mathbf{x} = \mathbf{F}, \quad \mathbf{M} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}$$

Assume the external force is harmonic, i e sinuousoidal,

$$\mathbf{F}(t) = \begin{bmatrix} \hat{F}_1 \sin(2\pi f t) \\ \hat{F}_2 \sin(2\pi f t) \end{bmatrix} = \begin{bmatrix} \hat{F}_1 \\ \hat{F}_2 \end{bmatrix} \sin(\omega t) = \hat{\mathbf{F}} \sin(\omega t), \quad \omega = 2\pi f ,$$

where $\hat{F}_1$ and $\hat{F}_2$ are force amplitudes. Since the system is linear it can be shown that the displacement caused by this harmonic force is also harmonic with the same frequency,

$$\mathbf{x}(t) = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} \sin(\omega t) = \hat{\mathbf{x}} \sin(\omega t) ,$$

and that the acceleration vector is,

$$\frac{d^2\mathbf{x}}{dt^2} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} \frac{d^2(\sin(\omega t))}{dt^2} = -\omega^2 \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} \sin(\omega t) = -\omega^2 \mathbf{x} .$$

The governing equation for the motion of the harmonically excited system can then be written,

$$-\omega^2 \mathbf{M}\hat{\mathbf{x}} \sin(\omega t) + \mathbf{K}\hat{\mathbf{x}} \sin(\omega t) = \hat{\mathbf{F}} \sin(\omega t) ,$$

or since $\sin(\omega t)$ is not identical to zero,

$$(-\omega^2 \mathbf{M} + \mathbf{K})\hat{\mathbf{x}} = \hat{\mathbf{F}} \quad \text{or} \quad \mathbf{A}(\omega)\hat{\mathbf{x}} = \hat{\mathbf{F}} .$$

Thus the coefficient matrix is dependent on the excitation frequency.

In a particular case the system parameters have the values; m_1 = 1 kg, m_2 = 2 kg, k_1 = 100 kN/m and k_2 = 50 kN/m. The external force vector has the amplitude F_1 = 10 N and F_2 = 10 N for all frequencies.

a) Use MATLAB to calculate the displacement vector for all frequencies in the frequency vector $f \in$ [0.1 , 100] Hz using frequency step 0.1 Hz.

b) Plot the absolute value of the displacement components in a lin-log diagram.

c) Determine the frequency where the absolute value of the displacement for mass 1, i.e. x_1, is maximum.

d) Determine the displacement amplitude vector $\hat{\mathbf{x}}$ at the frequency determined in c). Normalize, i.e. multiply with a constant factor, the amplitude vector so that its first component becomes 1.

50 (Level 2)

Consider the mechanical 2 degree of freedom system in exercise 49. For harmonic excitation forces, $\mathbf{F}(t) = \hat{\mathbf{F}} \sin(2\pi f t)$, the equations of motion governing the motion **x** of the system can be written,

$$(-\omega^2 \mathbf{M} + \mathbf{K})\hat{\mathbf{x}} = \hat{\mathbf{F}} , \qquad (1)$$

where $\mathbf{M} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}$, $\mathbf{K} = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}$, $\hat{\mathbf{x}} = \begin{Bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{Bmatrix}$, $\hat{\mathbf{F}} = \begin{Bmatrix} \hat{F}_1 \\ \hat{F}_2 \end{Bmatrix}$ and $\omega = 2\pi f$.

In the absence of exciting forces, i e **F** = **0**, the equations of motion (1) is an eigenvalue problem,

$$(-\omega^2 \mathbf{M} + \mathbf{K})\hat{\mathbf{x}} = \mathbf{0} . \qquad (2)$$

The eigenvalues are interpreted as particular values of the angular frequency squared for which the homogenous (**F** = 0) equations of motions have non-trivial solutions. These frequency values are denoted eigen-frequencies.

Suppose m_1 = 1 kg, m_2 = 2 kg, k_1 = 100 kN/m and k_2 = 50 kN/m.

a) Solve the eigenvalue problem given by (2) for parameter values as specified above. Calculate the corresponding eigenfrequencies *f*. What is the value of the second eigenfrequency?

b) Normalize the second eigenvector, i.e. the second column vector of the eigenvector matrix, so that its first component becomes 1. What are the values of the components of the normalized second eigenvector?

c) Consider the maximum displacement components calculated for the same system in exercise 49 parts c) and d). Compare the frequency determined in 49 c) with the second eigenfrequencies determined in part a) above. Also compare the normalized second eigenvector with the normalized maximum displacement vector determined in 49 d) by calculating the relative difference between their components. Discuss the physical interpretation of the eigenvectors.

51 (Level 1)

The transfer function *h* for a linear system describes the relation between the input signal to and output signal from the system. For a particular system the transfer function is defined as,

$$h(\omega) = \frac{1}{\omega - 1 - 0{,}5i} - \frac{1}{\omega + 1 - 0{,}5i},$$

where ω is a complex-valued angular frequency.

a) Evaluate *h* on the rectangular domain given by,
Re(ω) ∈ [-2 , 2] and Im(ω) ∈ [-0,5 , 1,5].
Use an equidistant mesh with 40 points along the real axis and 20 points along the imaginary axis. What is the value of ω in point 24 along the real axis and point 14 along the imaginary axis? What is the corresponding value of the transfer function *h*?

b) Use the command `surf` to create a 3-dimensional graph of the transfer function.
(i) Plot the real part of *h* as a function of ω.
(ii) Plot the imaginary part of *h* as a function of ω.
(iii) Plot the base 10 logarithm of the absolute value of *h* as a function of ω.

52 (Level 1)

A signal is a physical quantity varying with time. The sound pressure *p*, for instance, is he time varying pressure obtained when the average pressure, i e the static pressure p_0 , is subtracted from the total pressure p_{tot},

$$p(t) = p_{tot}(t) - p_0 . \qquad (1)$$

Calculating the strength of the sound by simply taking the average of the sound pressure gives strength 0 (from equation 1) which is not a useful result. For this reason the *root mean square* or *rms* value of the signal is introduced as a measure of the signal strength. In terms of the sound pressure, $\widetilde{p}$, it is defined as,

$$\widetilde{p} = \sqrt{\frac{1}{T}\int_0^T p^2(t)\,dt} \qquad (2)$$

where *T*, the integration time, is the duration of the signal or the measurement time.

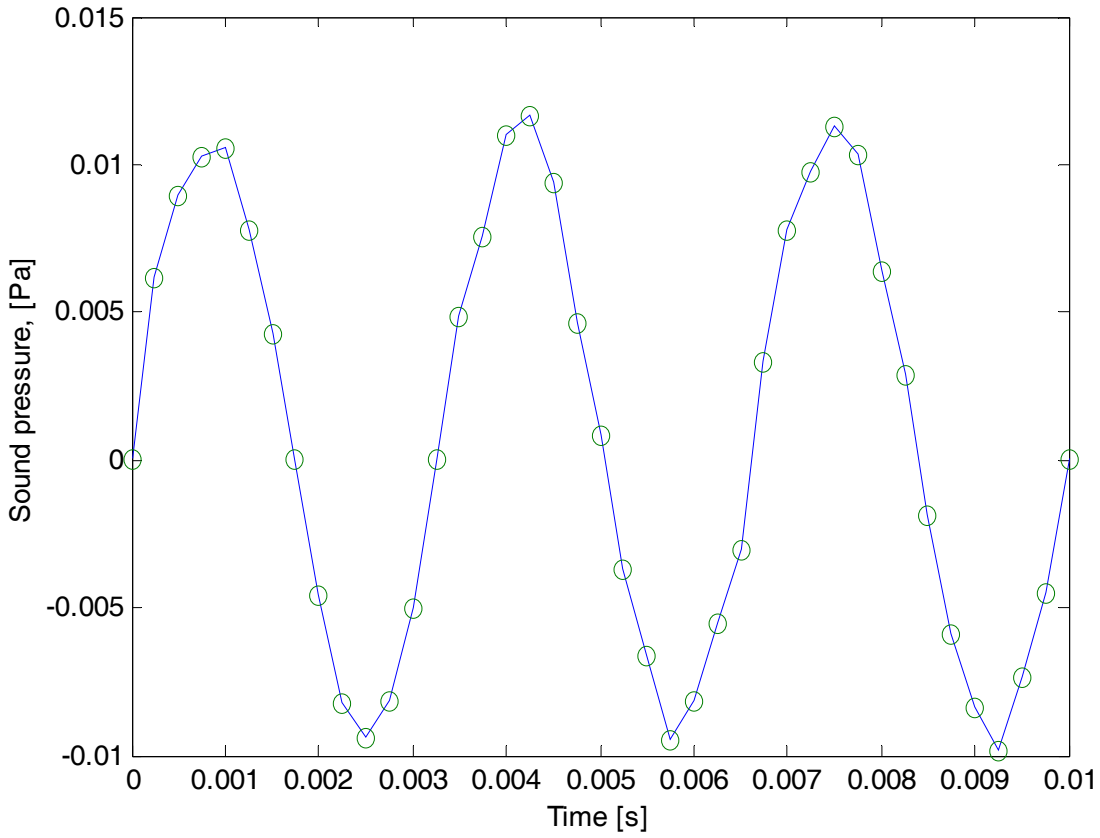


Figure 1 In a digital measurement instrument the continuous signal is sampled at discrete times with equidistant time intervals Δt.

Modern digital measurement instruments measures or samples, the signal at discrete equidistant times t_n, $n = 1, 2, 3, \ldots N$, see figure 1. The integral in definition (2), i e the area of the surface between the squared sound pressure curve and the time axis, is then estimated with the sum,

$$\frac{1}{T}\int_0^T p^2 dt \approx \frac{1}{N\Delta t}\sum_{n=1}^{N} p_n^2 \Delta t = \frac{1}{N}\sum_{n=1}^{N} p_n^2 , \tag{3}$$

where Δt is the time increment between two samples, N is the number of samples taken during the measurement time T and p_n is the sound pressure sampled at time t_n , see figure 2. Hence, the rms value is estimated as,

$$\tilde{p} \approx \sqrt{\frac{1}{N}\sum_{n=1}^{N} p_n^2} . \tag{4}$$

The static atmospheric pressure is normally $1 \cdot 10^5$ Pa, whereas the rms value of an ordinary, not too high, sound pressure is in the order of 0.01 Pa. A harmonic, i.e. purely sinuousoidal, pressure with frequency 100 Hz and peak sound pressure 0.01 Pa can be written as,

$$p_{tot}(t) = p_0 + p(t) = 1 \cdot 10^5 + 0.01\sin(2\pi 100 t) \tag{5}$$

An exact evaluation according to the definition gives, when the integration time becomes large, the rms value, $\tilde{p} = 0.01/\sqrt{2}$.

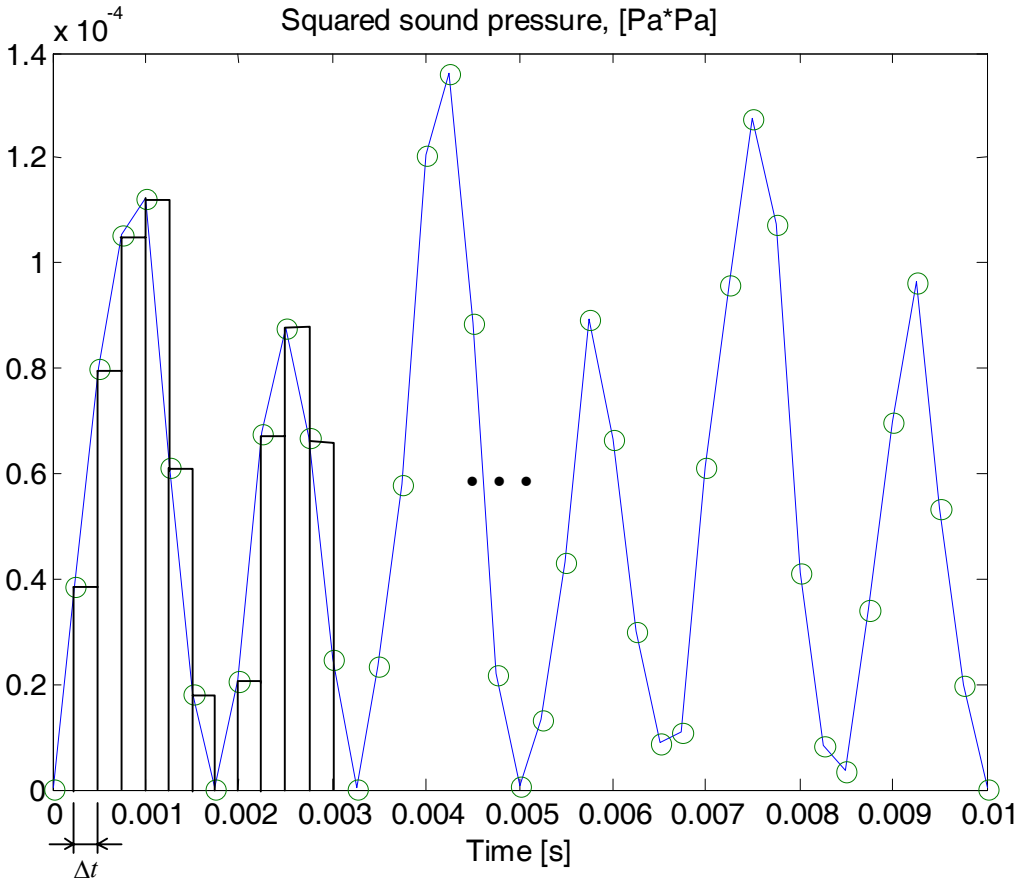


Figure 2 The area of the surface between the squared signal and the time axis is estimated with a sum.

a) Sample the pressure p_{tot} given by (5) at equidistant times t_n = (0, 0.0025 , 0.005, … 0.5) s. Plot a single period of the sound pressure p as function of time t. The period T is the inverse of the frequency. Use Equation (4) and calculate the rms value of the sound pressure p. Also calculate the relative error in the estimated rms value.

A method to increase the accuracy of digital measurements is to increase the sampling rate, i.e. decrease the time increment between the samples.

b) Sample the pressure at equidistant times t_n = (0, 0.00025 , 0.0005, … 0.5) s. Plot a single period of the sound pressure as function of time t. Calculate the rms value of the sound pressure and the relative error. Compare with results in a).

53 (Level 1)

The sound pressure level in dB is defined,

$$L_p = 10 \cdot \log \frac{\tilde{p}^2}{(2 \cdot 10^{-5})^2}, \qquad (1)$$

where $\tilde{p}$ is the root mean square value (see the definition in exercise 52) of the sound pressure.

a) Write an M-file that reads the rms sound pressure from the keyboard (Hint: Use the command `input`.) and calculates the sound pressure level in dB. What is the sound pressure level when the rms sound pressure is 0.1 Pa?

Usually the sound pressure varies with the frequency (tone) of the sound. Suppose the rms sound pressures for different frequencies are collected in a sound pressure vector *p* and that the corresponding frequency values are collected in the frequency vector *f*. The sound pressure level in dB at a specific frequency can then be calculated according to (1) where *p* is the rms sound pressure at the specific frequency.

The binary file `soundpress.bin` contains sound pressure data measured in the exhaust system of a Diesel engine. The data is stored on the floating point format `'float32'` in the sequence: $f_1, p_1, f_2, p_2, f_3, p_3, \ldots$.

b) Read the data in the binary file soundpress.bin and calculate the sound pressure level as a function of frequency. Use a dialog box to specify the name and path of the file. First use the command `uigetfile` then use `fopen`, `fread` and finally `fclose`. Modify the code so that it calculates the sound pressure level as a function of frequency, i.e. it evaluates the sound pressure level for all sound pressure values in the vector *p*.

c) Modify the code so that it locates the maximum value of the sound pressure level and the frequency for which it occurs. What is the maximum level and at what frequency does it occur?

d) Plot the sound pressure level as a function of frequency in a lin-lin diagram. Do not forget to assign proper labels to the axis.

The total sound pressure level including the energy at all frequencies is defined as,

$$L_{p,tot} = 10 \cdot \log \frac{\sum_n p_n^2}{(2 \cdot 10^{-5})^2}, \qquad (2)$$

where p_n is the *n*th component of the sound pressure vector *p*.

e) Modify the M-file so that it also calculates the total sound pressure level for the sound pressure vector *p*. What is the total measured sound pressure level for the Diesel engine?

f) Use the commands `text` and `num2str` to add text giving the maximum and total sound pressure levels to the diagram.

54 (Level 3)
Modify the M-file written in the previous exercise so that the calculations of the sound pressure level, its maximum and its total value are calculated by a separate function M-file.

55 (Level 3)
Frequency analysers used in acoustics and vibrations for measuring frequency spectra usually provide results in terms of narrow-band spectra. In a power spectrum the energy of a physical entity is measured as a function of frequency. The sound power spectrum, for instance, is the square of the rms sound pressure as a function of frequency, $\tilde{p}^2(f)$.

a) Design a function M-file `yband = band(fl,fu,f,y)` that sums the energy of the lines within the interval [fl, fu].

b) Design a function M-file `[throctf,throcty] = throct(f,y)` that converts the narrow-band spectrum defined by the frequency vector f and the power spectrum y to a third octave band spectrum and labels each band with its center frequency. The band limits are given in the table below:

Band number	Center frequency [Hz]	Lower limit [Hz]	Upper limit [Hz]
1	1,25	1,12	1,41
2	1,6	1,41	1,78
3	2	1,78	2,24
4	2,5	2,24	2,82
5	3,15	2,82	3,55
6	4	3,55	4,47
7	5	4,47	5,62
8	6,3	5,62	7,08
9	8	7,08	8,91
10	10	8,91	11,2
11	12,5	11,2	14,1
12	16	14,1	17,8
13	20	17,8	22,4
14	25	22,4	28,2
15	31,5	28,2	35,5
16	40	35,5	44,7
17	50	44,7	56,2
18	63	56,2	70,8
19	80	70,8	89,1
20	100	89,1	112
21	125	112	141
22	160	141	178
23	200	178	224
24	250	224	282
25	315	282	355
26	400	355	447
27	500	447	562
28	630	562	708
29	800	708	891
30	1000	891	1120
31	1250	1120	1410
32	1600	1410	1780
33	2000	1780	2240
34	2500	2240	2820
35	3150	2820	3550
36	4000	3550	4470
37	5000	4470	5620
38	6300	5620	7080
39	8000	7080	8910
40	10000	8910	11200

c) Try the codes developed in a) and b) on the sound pressure data stored in the binary file `soundpress.bin` that contains sound pressure data measured in the exhaust system of a Diesel engine. The data is stored on the floating point format `'float32'` in the sequence: $f_1, p_1, f_2, p_2, f_3, p_3, \ldots$.

(i) Calculate the sound pressure level, L_p, defined as,

$$L_p = 10 \cdot \log \frac{\widetilde{p}^2}{(2 \cdot 10^{-5})^2}.$$

(ii) Plot the spectrum, i e the sound pressure level as function of frequency in a graph.

56 (Level 3)
The standard trigonometric functions require angular input values in radians.

a) Write a function M-file that converts an angle, specified in radians, to degrees.

b) Write a function M-file that calculates the inverse tangent (arcustangent) to an input argument and then calls the function M-file written in a) to convert the resulting angle from radians to degrees.

c) Try the written code on the following input values,
(i) $x = 1$,
(ii) $x = 1/0$,
(iii) $x = 0.5$.

57 (Level 3)
Vibration measurement data, i e acceleration data, are stored in 5 different files: `acc1.dat`, `acc2.dat`, … and `acc5.dat`. Suppose the number assigned to the variable $n \in [1, 2, 3, 4, 5]$ determines the name of the file that is to be read, `acc`**`n`**`.dat`.

a) Write MATLAB commands that, for a fixed numeric value of `n`, assigns the correct file name to string variable `filename`.

If a file name is to be assigned repeatedly several times it may be worthwhile to write a function procedure that performs the task whenever called.

b) Write a function M-file,

```
function filename = name(n)
% n - integer in range [1,5]
%
...
```

that produces the correct file name string as output variable `filename`.

58 (Level 1)
The displacement field, w, of a vibrating rectangular membrane with fixed edges, see Figure 1, can be approximated with a sum of characteristic components (vibration modes),

$$w(x,y) = c_{11} \sin\frac{\pi x}{L_x} \cdot \sin\frac{\pi y}{L_y} + c_{12} \sin\frac{\pi x}{L_x} \cdot \sin\frac{2\pi y}{L_y} + \cdots + c_{mn} \sin\frac{n\pi x}{L_x} \cdot \sin\frac{m\pi y}{L_y} + \cdots,$$

where L_x and L_y are the length of the edges and c_{mn} is a coefficient specifying the contribution from mode mn to the displacement field.

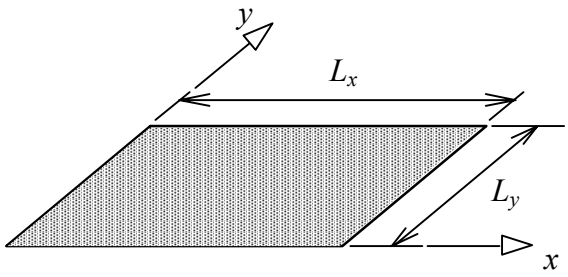


Figure 1 Rectangular membrane.

Consider a membrane with dimensions $1 \times \sqrt{2}$ m^2. Plot the following vibration modes using the commands `mesh` and `contour`. Make sure the resolution of the mesh is sufficiently high.

a) $\sin(\pi x) \cdot \sin \dfrac{\pi y}{\sqrt{2}}$,

b) $\sin(2\pi x) \cdot \sin \dfrac{\pi y}{\sqrt{2}}$,

c) $\sin(3\pi x) \cdot \sin \dfrac{2\pi y}{\sqrt{2}}$.

59 (Level 3)

Vibration testing aims to test whether or not a product can operate in its supposed environment. During the test the test object is mounted to a shaker, see Figure 1, used to enforce a motion specified in the test schedule as a power density spectrum for the acceleration.

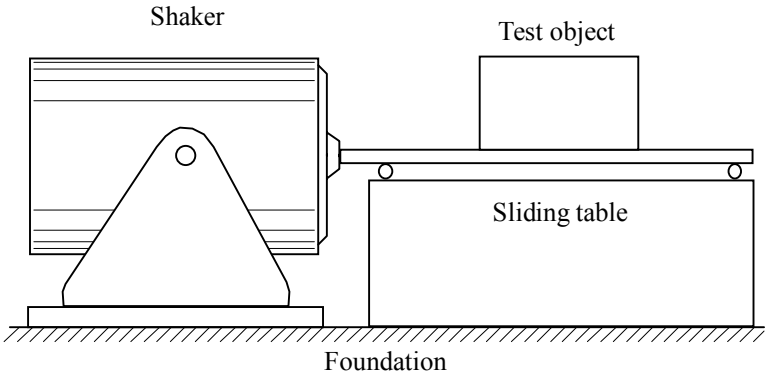


Figure 1 In a vibration test the test object is mounted to a shaker that enforces a predefined motion to the test object.

Suppose a microelectronic system developed for aeronautic applications is subject to a vibration test aiming to investigate its ability to operate in a vibrating environment. The vibration environment is specified in terms of a acceleration power density spectrum, i.e. the mean square value, see exercise 52, of the acceleration is given as function of frequency, see Figure 2.

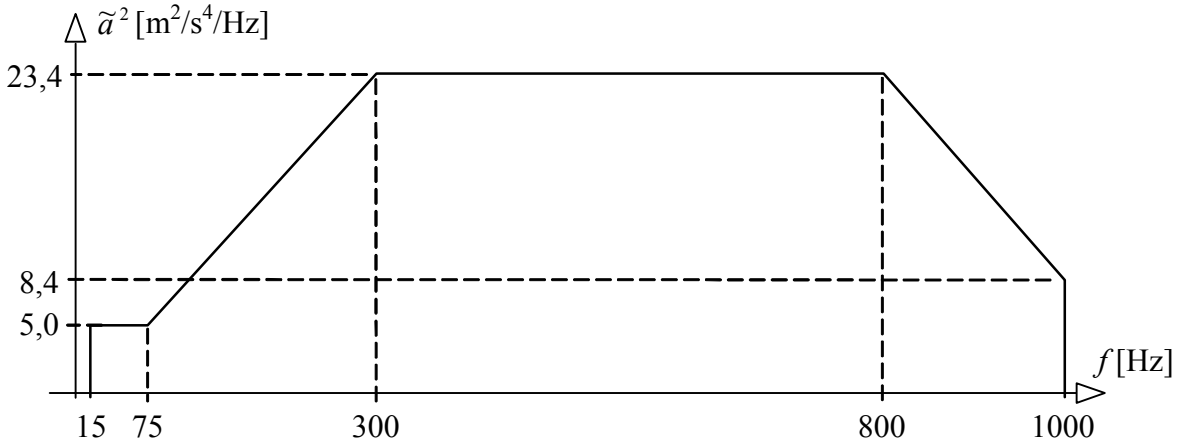


Figure 2 Acceleration power density spectrum used during test.

A parameter of interest in vibration testing is the total energy supplied to the test object. Since the energy of a system vibrating at a specific frequency is proportional to the square of the acceleration amplitude, the total vibration energy is proportional to the acceleration amplitude squares summed over all frequencies. Mathematically then the total energy is proportional to the area of the surface below the acceleration power density spectrum, i.e. the integral of the power density spectrum,

$$\overline{W} \propto \int_{f_{\min}}^{f_{\max}} \tilde{a}^2 df \,.$$

In MATLAB numeric integration or quadrature is performed using the command `quad` . For information on the use of `quad` see the MATLAB Help browser.

a) The `quad` command requires the integrand to be specified as a function providing a vectorial output y containing integrand values for the vectorial input x. Write a function M-file that calculates the values of an arbitrary piece-wise linear function, see for instance the power density spectrum function in figure 2 above.

b) Design a MATLAB code for calculating the integral of a piece-wise linear power density spectrum. Use MATLAB function for numeric integration `quad` and the function developed in part a).

c) Try the code on the spectrum in Figure 2.

60 (Level 3)

Consider a structure consisting of three rods, denoted 1, 2 and 3, mounted to a rigid base according to the figure 1. Assume the rods have lengths L_1, L_2 and L_3, cross section areas A_1, A_2 and A_3 and that they are made from materials with Young's moduli E_1, E_2 and E_3.

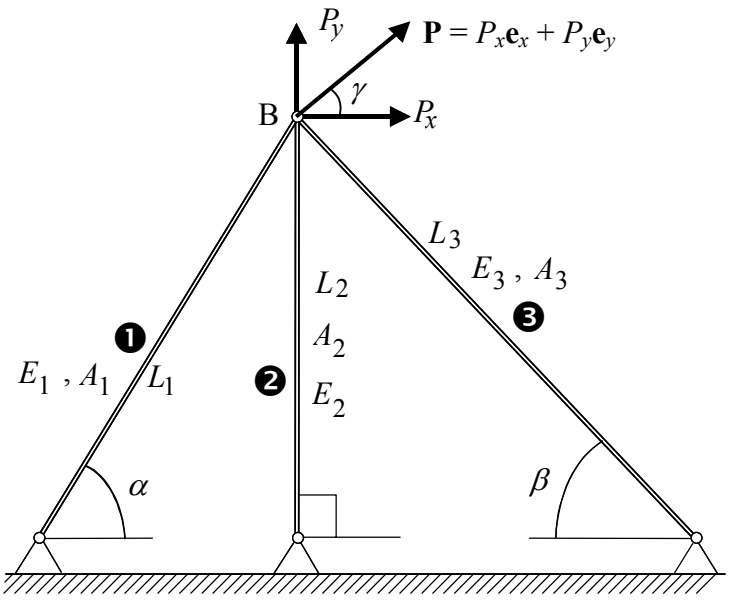


Figure 1 Structure built-up from three rods. An external force, $\mathbf{P} = P_x\mathbf{e}_x + P_y\mathbf{e}_y$, is applied to the junction, node B.

The structure is subjected to an external load **P**, with components P_x and P_y, which causes the structure to deform. The displacement vector $\boldsymbol{\delta}$, with components δ_x and δ_y, in the loading point B is related to the load **P** by the matrix equation,

$$\mathbf{P} = \mathbf{K} \cdot \boldsymbol{\delta} ,$$

or in component form,

$$\begin{Bmatrix} P_x \\ P_y \end{Bmatrix} = \begin{bmatrix} k_1 \cos^2\beta + k_3 \sin^2\beta & k_1 \cos\alpha\cos\beta - k_3 \sin\beta\cos\beta \\ k_1 \cos\alpha\cos\beta - k_3 \sin\beta\cos\beta & k_1 \cos^2\alpha + k_2 + k_3 \cos^2\beta \end{bmatrix} . \begin{Bmatrix} \delta_x \\ \delta_y \end{Bmatrix},$$

where, $k_1 = E_1A_1 / L_1$, $k_2 = E_2A_2 / L_2$ and $k_3 = E_3A_3 / L_3$, are the longitudinal stiffnesses of the individual rods. The matrix **K**, which relates the displacement to the load, is denoted the stiffness matrix.

a) Write an M-file that calculates the displacement vector $\boldsymbol{\delta}$ at point B for a known force with components P_x and P_y.

Suppose the following data is valid in a particular case,
$L_1 = 11$ m, $L_2 = 10$ m, $L_3 = 14$ m, $A_1 = 0.001$ m^2, $A_2 = 0.001$ m^2, $A_3 = 0.001$ m^2,
$E_1 = E_2 = E_2 = 73$ GN/m^2 (Aluminium) and $|\mathbf{P}| = 10000$ N

b) Use the M-file developed in part a) to calculate the displacement at point B:
   (i) when the load applied to point B is horizontal and
   (ii) when the load applied to point B is vertical.

The code can be made more effective by performing repetitive calculations in a function m-file.

c) Write a function M-file `displ = invK_p(gamma,P,L)` that for a given set of input data, `gamma` (γ), `P` (P) and `L` (L_1, L_2 and L_3), calculates the stiffness matrix **K**,

the right-hand side vector **P** and finally solves the system of equations for the node displacement `displ` (δ). Validate the code by repeating the calculations performed in part b).

d) Suppose the load is applied to point B at angle γ with respect to the horizontal axis see Figure 1. Modify the m-file so that it calculates the displacement components δ_x and δ_y as functions of the load angle γ in the interval $\gamma \in [0, 2\pi]$. Plot the displacement components as functions of the load angle γ in a lin-lin diagram. For which γ values is the horizontal displacement component zero?

61 (Level 3)

Vibrating machines are important sources of noise in vehicles and buildings. The noise can either be transmitted directly from the machine as noise radiated from its surfaces or indirectly as vibrations via the machine mounts. Successful noise abatement in vehicles therefore requires knowledge information on the amount of acoustic energy transmitted via the various transmission parts. Noise reducing actions is then aimed for the most important of the transmission parts. In this exercise method to determine the vibration energy transmitted from a machine to its foundation is demonstrated.

Suppose a machine is vibrating with a dominant frequency at 90 Hz. The vibrations is transmitted to the foundation via four mounts and then finally radiated as sound from the vibrating foundation. The vibration energy transmitted via a mounting can be determined by simultaneously measuring the force, F, and the velocity, v, in the mounting. In practise the velocity is measured indirectly by measuring the acceleration, see Figure 1.

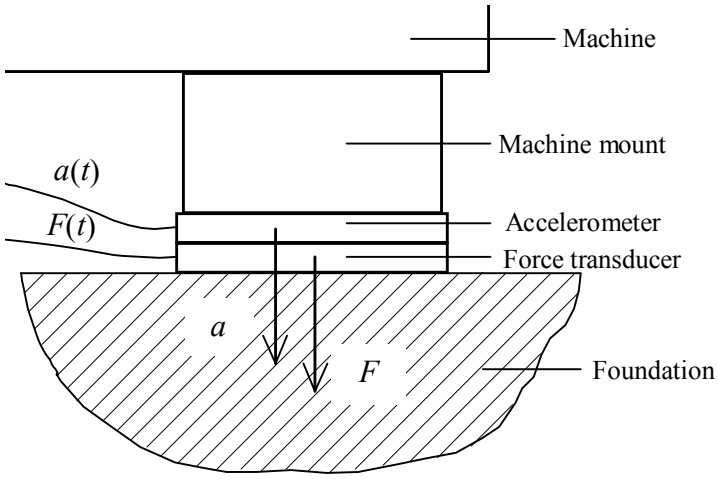


Figure 1 In each of the four mounting points the force and the velocity (acceleration) is measured using force transducers and accelerometers.

When the vibrations are dominated by a single frequency the following formula may be used to calculate the mechanical power transmitted over the mounting,

$$\overline{W} = \frac{1}{2}\mathrm{Re}(\mathbf{Fv}^*),$$

where $\overline{W}$ is the time average of the power, **F** is the complex force, **v** is the complex velocity and * denotes complex conjugation. When a single frequency dominates the measured force and acceleration signals can be written,

$$F(t) = \hat{F}\cos(2\pi t + \phi_F) = \mathrm{Re}(\hat{F}e^{i(2\pi ft+\phi_F)}),$$

$$a(t) = \hat{a}\cos(2\pi t + \phi_v) = \mathrm{Re}(\hat{a}e^{i(2\pi ft+\phi_v)})\,.$$

The complex force and acceleration are then introduced as,

$$\mathbf{F}(t) = \hat{F}e^{i(2\pi ft+\phi_F)},$$

$$\mathbf{a}(t) = \hat{a}e^{i(2\pi ft+\phi_a)},$$

where ^ denotes amplitude or peak value, f is the frequency and ϕ_F and ϕ_a are phase angles for the force and acceleration.

a) Write a function M-file, `W = powcalc(a,F,f)` , that calculates the mechanical power transmitted via a single mounting for a specified acceleration, force and frequency.

b) Develop a MATLAB code that calculates the total mechanical power transmitted over N mounting points.

c) Try the code on a case where vibration energy at 90 Hz is transmitted via four points with measured data according to table 1 below.

Table 1 Measured acceleration and force data in four mounting points.

Point	**1**		**2**		**3**		**4**	
Force	Amplitude [N]	Phase [degrees]	Amplitude [N]	Phase [degrees]	Amplitude [N]	Phase [degrees]	Amplitude [N]	Phase [degrees]
	13	0	17	33	8	70	21	85
Acc	Amplitude [N]	Phase [degrees]	Amplitude [N]	Phase [degrees]	Amplitude [N]	Phase [degrees]	Amplitude [N]	Phase [degrees]
	1,4	192	1,1	160	2,1	212	3,4	263

62 (Level 3)

The stress state in a loaded solid body is given by the stress matrix **S**. If a cut, with normal **n,** is made in the body (Figure 1) then the following stresses are found on the cut surface.

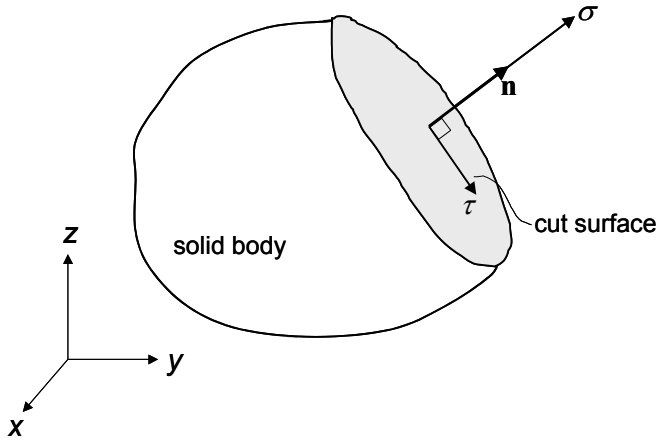


Figure 1 Stresses in a solid body.

Stress in the direction of **n**, i.e.the normal stress,

$$\sigma = \mathbf{n}^{\mathrm{T}}\mathbf{S}\mathbf{n}\,. \qquad (1)$$

Stress perpendicular to **n**, i.e.the shear stress,

$$\tau = (|\mathbf{Sn}|^2 - \sigma^2)^{1/2}. \qquad (2)$$

This means that σ and τ will be different for each **n**. For a given plane the stresses σ and τ can be plotted as a point in a diagram with σ on the x-axis and τ on the y-axis. Let,

$$\mathbf{S} = \begin{bmatrix} 10 & 4 & 8 \\ 4 & 3 & 1 \\ 8 & 1 & 25 \end{bmatrix} \text{ MPa.}$$

In this exercise we shall use randomly oriented cuts.

a) The easiest way to achieve a random normal to the cut $\mathbf{n} = [\, n_x \;\; n_y \;\; n_z \,]^T$ is the following. Write a function M-file `snorm = surfnorm` that determines n_x, n_y and n_z randomly in the interval]-1, 1[. Then scale the length of the normal vector so that its length, i.e. its absolute value, is 1.

b) Write a function M-file `[nstress,shear] = stress(S,n)` that determines a random normal direction **n** by calling the function M-file `snorm` and then calculates the normal stress σ (`nstress`) and the shear stress τ (`shear`) for **n** by using Equations (1) and (2).

c) Write a code `nrandstress.m` that, for a specified stress matrix **S**, calculates the normal stress (nstressn) and shear (shearn) for N random cuts. Save the corresponding shear and normal stress values in two vectors. Plot the shear as a function of the normal stress in a diagram with σ on the x-axis and τ on the y-axis. Enforce equal scaling on the axis. Also, use `*` as the plot symbol (marker). Execute the code for $N = 500$. The final plot will show the possible combinations of σ and τ that can be found by cutting in different orientations. What is the shape of the graph?

d) Modify the code to find the eigenvalues and eigenvectors of **S**. Calculate the normal stress and shear for **n** being equal to the three eigenvectors. Comment on the results. Try to find out how the eigen-values relate to the plot in c).

e) On an unloaded surface of a body all components in a row and the corresponding column of **S** is zero. Pick a row and the corresponding column in **S** given above, and set the components to zero. Calculate the eigenvalues and plot the $\sigma(\tau)$ graph for this case.

f) Redo a) and b) using another symmetric stress matrix $\mathbf{S} = \mathbf{S}^T$, for example,

$$\mathbf{S} = \begin{bmatrix} 10 & 2 & 0 \\ 2 & 10 & 1 \\ 0 & 1 & 10 \end{bmatrix} \text{ MPa.}$$

63 (Level 3)

Parametric or semi-empiric models are often used to describe physical phenomena. A parametric model is usually based on a theoretical model that prescribes a relationship between two variables on a certain mathematical form. For example, a parametric model of a linear relationship between variables x and y can be written,

$$y = c_1 x + c_2 \,,$$

where the model parameters c_1 and c_2 are constants.

In physics the model parameters are often determined experimentally. In principle two independent measurements are needed to determine the parameters. In practice, however, the influence of measurement errors must be reduced by using more than two independent measurements. A least squares approximation of the model parameters can then be estimated by solving an overdetermined system of equations,

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{Bmatrix}.$$

Suppose that you have N independent measured data pairs (x_n, y_n), $n = 1, 2, 3 \ldots, N$, stored on floating point format `'float32'` in a binary file `xydata.bin`. The storing sequence is $x_1, y_1, x_2, y_2, x_3, y_3, \ldots$.

a) Write a M-file that reads the data file, calls for a function M-file that solves the overdetermined system for the model parameters and finally plots the measured points in a graph together with the estimated model curve.

b) Try the code on the data given in `xydata.bin`.

64 (Level 3)

Transducers are used in mechanical measurements for transforming a mechanical quantity to a measurable electrical quantity. Important examples are strain gauges for strain or deformation measurements, force transducers, accelerometers and microphones. In order to use a transducer the relation between the mechanical input quantity and the electrical output quantity must be known.

Consider a force transducer that transforms an input force, F [N], to an output voltage, U [V]. In its linear range the relation between the input F and the output U is assumed to be,

$$U = CF + U_0,$$

where C [V/N] is the transducer sensitivity and U_0 is a bias voltage.

To determine the transducer relation, i.e.the model parameters C and U_0, a series of five independent measurements was performed. In each measurement a known force was applied to the transducer whose output voltage was measured with a voltmeter. The results are tabulated in table 1.

Table 1 Results from calibration measurements.

Force [N]	1	5	10	20	50
Voltage [V]	0.08	0.11	0.14	0.28	0.55

Determine the optimum transducer model in the least squares sense.

65 (Level 2)

Reduction of vibrations transmitted from a machine to a building is achieved by placing the machine on soft elastic mounts. The achieved reduction is highly dependent on the mount elasticity or stiffness. Therefore producers of mounts should provide measured dynamic stiffness data for the mounts. The dynamic stiffness, k, is defined as the ratio of the force amplitude, $\hat{F}$, over the displacement amplitude, $\hat{x}$, both as a functions of frequency,

$$k(f) = \frac{\hat{F}(f)}{\hat{x}(f)}.$$

The measurement set-up is explained in Figure 1 below.

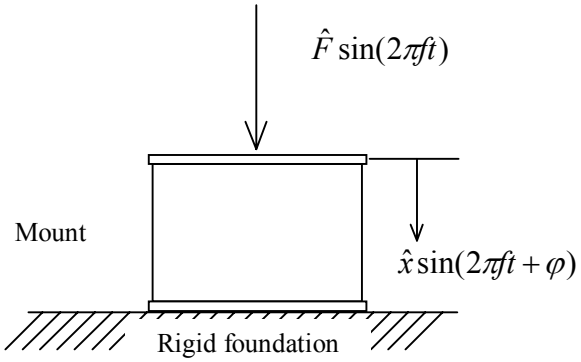


Figure 1 When the mount stiffness is measured the mount is fixed to a rigid foundation and loaded with a harmonic force $\hat{F}\sin(2\pi ft)$. The loading force and the corresponding spring deformation is measured and their amplitude ratio is the spring stiffness.

One specific producer uses a measurement system that measures the frequency-force-displacement triplets and stores them on a binary file `stiff.dat`. The storage precision is `'float32'` and the storage sequence is freq1-force1-disp1-freq2-force2-disp2-freq3-… .

a) Read the frequency-force-displacement triplets into a data vector.

b) Partition the data vector in a frequency vector, a force vector and a displacement vector.

c) Calculate the dynamic stiffness, *k*, as defined above and plot it as a function of frequency in a lin-log graph.

66 (Level 3)

The figure shows a mechanical single degree of freedom model used to simulate the motion of a car running on a road with a bump.

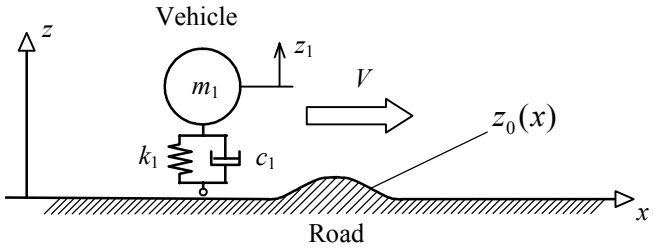


Figure 1 Mechanical single degree of freedom (SDOF) system for simulating a car running over a road bump.

The equations of motion for the mass and the spring relation yields the following second order differential equation governing the displacement *z* of the mass *m*.

$$m\frac{d^2z}{dt^2} + c\frac{dz}{dt} + kz = kz_0(t), \qquad (1)$$

where m is the mass, k is the spring constant, c is the damping constant and t is the time. The right-hand side of (1) is the spring force acting on the mass when the road surface profile is z_0 at time t.

MATLAB provides routines, e.g. `ode45` for solving this type of differential equations. Use of `ode45` requires the equation to be formulated as a first order equation. By introducing a vector y as,

$$\mathbf{y} = \begin{Bmatrix} \dfrac{dz}{dt} \\ z \end{Bmatrix}, \tag{2a}$$

and by using,

$$\frac{d\mathbf{y}}{dt} = \begin{Bmatrix} \dfrac{d^2z}{dt^2} \\ \dfrac{dz}{dt} \end{Bmatrix}, \tag{2b}$$

the second order equation (1) can be reformulated to the a first order system of differential equations. On matrix form this system reads,

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \frac{d\mathbf{y}}{dt} = \begin{Bmatrix} k/m \\ 0 \end{Bmatrix} \cdot z_0 + \begin{Bmatrix} c/m \\ 0 \end{Bmatrix} \cdot \frac{dz_0}{dt} - \begin{bmatrix} c/m & k/m \\ -1 & 0 \end{bmatrix} \cdot \mathbf{y}, \tag{3}$$

where $\dfrac{dz_0}{dt} = \{x = Vt\} = V\dfrac{dz_0}{dx}$.

The equation solving routine `ode45` requires the right-hand side of (3) to be calculated at time t in a function m-file specified in the parameter list. In this case, as can be seen in (3), this function must contain a calculation of the position of the car at time t, the road profile z_0 at the position of the car and finally the right-hand side of (3).

a) Suppose the road surface can be described by the following function,

$$z_0(x) = \hat{z}_0 \cdot f(x) = \hat{z}_0 \cdot \begin{cases} \cos^2(\pi x), & -0.5 \le x \le 0.5 \\ 0.001 \cdot random(-1,1), & x < -0.5 \vee x > 0.5 \end{cases}, \tag{4}$$

where *random*(-1,1) means a random number on the interval]-1, 1[. Write a function M-file `f = profile(x)` that calculates the road profile f as a function of the car position x according to (4).
Also, write a function M-file `fprim = profileprim(x)` that calculates the x-derivative of the road profile f as a function of the car position x according to (4). You may assume that the derivative of the random part of the profile is zero.

b) Write a function M-file `funcbump(t,y)` that calculates the right-hand side of (3). Note that the function output should be a column vector with the two components of the right-hand side vector.

In its ordinary use the equation solver `ode45` should be called as,

```
[T, displ] = ode45('funcbump',[tmin, tmax], ystart)
```

where `T` is the time vector for which the displacement `displ` have been calculated by the solver, t_{min} and t_{max} are the lower and upper time limits of the simulation time interval and

finally `ystart` are the starting conditions in terms of the displacement and its time derivative at `tmin`. If the solution is to be evaluated at specific time values the time interval specification above can be replaced with a time vector with components equal to the times for evaluation.

c) Write an M-file for simulating the motion of a car governed by Equation (1). Make use of the command `ode45` and the function M-files `funcbump`, `profile` and `profileprim`. Note that parameter values needed in more than one M-file can be made available by using `global` declaration.

Suppose the model parameters have the values: m = 250 kg, c = 200 Ns/m and k = 2000 N/m.

d) Simulate the motion of the car running at 50 km/h over a road profile with size, $\hat{z}_0$, equal to 0.3 m, for time in the interval [-5, 20] seconds. Assume starting conditions equal to 0. Plot the simulated displacement as function of time in a diagram.

67 (Level 3)
Animated movies can be created using sequential matrix multiplication. In this exercise you will create a triangle that is used to demonstrate animated sequences.

a) First use the `figure` and `axis` commands to create a figure window with fixed corners in the points {(-10,-10) (10,-10) (10,10) (-10,10)}.

Line figures or wire frames are drawings created by connecting the corner points of a geometrical object with straight lines. In MATLAB the command `line(x,y)` is designed to draw straight lines between the corner point coordinates defined by `x` and `y`.

b) A triangle has corner points (-5, -5), (5, -5) and (0, 5). Define a row vector `x` and a row vector `y` containing the x- and y-coordinates of the triangle corner points. Note that in order to create a closed triangle you have to end the line in the starting corner. Finally, use the command `line` to draw the triangle in the figure window. Make the triangle lines black.

Figure motion can be animated using basic matrix operations. To illustrate this define a corner point matrix with the x-coordinate vector as first row and the y-coordinates as second row.

Pure translation means that all corner points are moved along the same displacement vector. Hence, the displacement in x-direction should be added to all x-coordinates and the displacement in the y-direction should be added to all y-coordinates in the corner point matrix. To do this, first introduce a unit rigid body translation matrix having the same dimension as the corner point matrix and then a 2 by 2 point displacement matrix. The elements of the unit rigid body translation matrix are all 1 and the point displacement matrix is a diagonal matrix with the diagonal elements equal to the point displacements in x- and y-directions.

The corner point matrix of the translated matrix is calculated by multiplying the unit rigid body translation matrix from the left with the point displacement matrix and, finally, add the product to the original corner point matrix.

c) Translate the original triangle 2 units in the positive horizontal direction, i.e. in the positive x-direction.

(i) Define a unit rigid body translation matrix for the triangle.

(ii) Define a point displacement matrix displacing the point 2 units in the horizontal (x) direction and 0 units in the vertical (y) direction.
(iii) Calculate corner point matrix of the translated triangle.
(iv) Draw the translated triangle with blue colour.

Zooming a figure can be accomplished by multiplying the corner point matrix with a 2 by 2 diagonal scaling matrix (see exercise 37). Uniform, isotropic, scaling is obtained when the diagonal elements are equal. The elements on the diagonal represent the scaling factors with respect to the x- and y-directions. The corner point matrix of the zoomed figure is obtained by multiplying the original corner point matrix from the left with the scaling matrix.

d) Define an isotropic scaling matrix with scaling factor 1.2. Calculate the corner point matrix of the zoomed triangle and draw it using green colour.

Rotating a figure is performed using a rotation matrix (see exercise 37) defined as,

$$\mathbf{B} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix},$$

where ϕ is the rotation angle. Multiplying a vector with **B** from the left rotates the vector ϕ radians in the positive (counter-clock wise in a xy-diagram) direction. A figure is rotated by multiplying its corner point matrix from the left with a rotation matrix.

e) Define a rotation matrix rotating the figure 15 degrees in positive direction. Note that the trigonometric functions assume angles specified in radians. Calculate the corner point matrix for the rotated triangle and plot the rotated triangle using red colour.

68 (Level 3)
Motion sequences can be animated using matrix multiplication and graphic commands in a loop

a) Create the triangle **P** defined in exercise 67 and a matrix **B** that rotates the triangle 1 degree in positive, i.e. counter-clockwise direction.

b) Animate the triangle rotating 360 degrees in counter-clockwise direction.

69 (Level 3)
Consider the mechanical single degree of freedom system with mass m, spring constant k and damping constant c depicted in Figure 1.

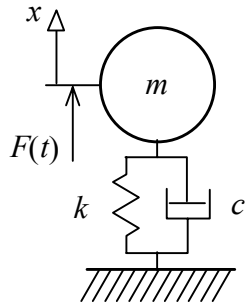


Figure 1 A mechanical single degree of freedom system.

For a sinuousoidal exciting force F oscillating at frequency f the displacement x as function of time t can be written,

$$x(t) = \frac{F}{\sqrt{(k-(2\pi f)^2 m)^2 + (2\pi f)^2 c^2}} \sin(2\pi f t + \arctan\frac{2\pi f c}{k-(2\pi f)^2 m}).$$

In a particular case the system parameters are:

m = 1 kg,
k = $1 \cdot 10^4$ N/m,
c = 1 Ns/m,
F = 50 N

The displacement was calculated at 500 equidistant frequency points on the interval [0,30] Hz

Write an M-file that animates the motion at the frequency given by the 265th frequency component. In the animation the system given in figure 1 should be animated.

70 (Level 3)

Consider the car running over a road bump in exercise 66. Write an M-file that animates the motion as function of time as the car passes the bump.

71 (Level 3)

The purpose of a Graphic User Interface (GUI) is to make it possible to use the program without detailed knowledge of the code. This exercise shows a step by step design of a simple GUI. In this exercise a GUI for calculating and plotting the Frequency Response Function, FRF, for a Single Degree Of Freedom system, SDOF, is developed.

A mechanical single degree of freedom system with mass m, damping c and spring constant k excited by an external force F is shown in Figure 1.

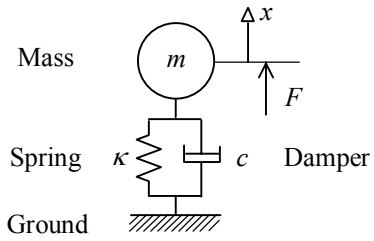


Figure 1 Mechanical single degree of freedom system.

The equation of motion for the system reads,

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = F. \quad (1)$$

Lots of valuable information on the system is provided in the frequency response function which can be calculated by assuming that the exciting force is harmonic, i.e. it can be written,

$$\mathbf{F}(t) = Fe^{i\omega t}, \quad (2)$$

where ω is the angular frequency. It can be shown that the displacement x caused by a force given by (2) is,

$$\mathbf{x}(t) = xe^{i\varphi}e^{i\omega t}, \quad (3)$$

where φ is the phase angle of the displacement relative to the exciting force. The frequency response function is defined as the ratio between the displacement **x** and the force **F**, that is,

$$\mathbf{H}(\omega) = \frac{\mathbf{x}(t)}{\mathbf{F}(t)} = \frac{x}{F} \cdot e^{i\varphi}. \quad (4)$$

After inserting (2) and (3) into (1) the frequency response function (4) becomes,

$$\mathbf{H}(\omega) = \frac{1}{k + i\omega c - \omega^2 m} = \frac{1}{\sqrt{(k-\omega^2 m)^2 + \omega^2 c^2}} e^{i \arctan \frac{ic}{k-\omega^2 m}}. \quad (5)$$

This expression is the frequency response function for the mechanical single degree of freedom system.

The first action taken when a GUI is created is to decide what is to be done and the graphical appearance of the interface on the screen. In this case the interface should contain one part where the system parameters are specified, one part with buttons for starting calculations and exiting the interface and finally one part for graphical presentation of the results. Figure 2 shows a schematic picture of one possible screen interface.

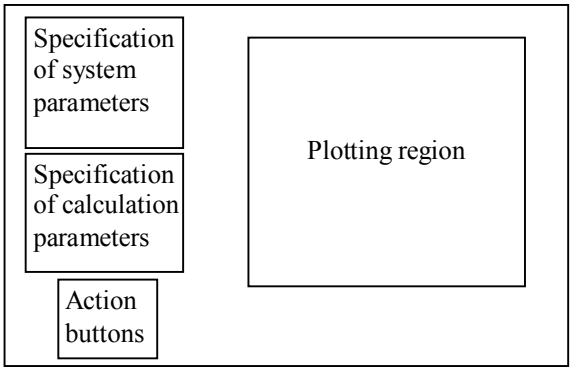


Figure 2 Schematic picture of suggested screen interface for calculating mechanical single degree of freedom systems.

There are several possible methods to implement a GUI. An often used method is based on recursive calling of a function M-file, meaning that the code consists of a function m-file that calls itself for performing various tasks. The basic idea of this programming technique is to use a function m-file that calls itself with an input string variable whose value controls the program flow. When a button in the user interface window is pushed the variable should be set to a specific value that when the function is called again allows the expected set of commands to be performed.

The first time the function is called immediately after starting the code. The user interface window, its text parts and various buttons must then be initialised. One way to solve this is to make use of the `nargin` command which, when used inside a function M-file returns the number of input arguments. Hence, at the first call, when there are no input parameters (no buttons in the user interface are pushed) and the user interface should be initialised, start with the following commands,

```
function Sdofgui(action)

% ...

if nargin < 1
            action = 'initialise';
end

% Here follows an "if...elseif...end" construct that controls % the
actions taken when the buttons in the user interface are % pushed.
The values of two strings may be compared with the
% command strcmp which returns the value 1, i.e. becomes
% true when the strings are equal.

if strcmp(action,'initialise')
```

```
            % Initialise user interface window
            ...
elseif strcmp(action,'ButtonAction1')
            % Commands executed when button number 1 is pushed:
            ...
end
```

a) Enter the commands above in a function m-file 'sdofgui.m'. Use the command `sdofFig = figure('position',[ ... ],...)` to create a interface window named `sdofFig` with suitable size, name and position etc.

Now it is time to initialise the first pushbutton. Pushbuttons of different kinds are created with the command

```
uicontrol('style','pushbutton','position',[...],'callback',...'set
of commands;',...)
```

The `callback` variable is a string variable that contains the MATLAB commands that will be executed when the button is pushed is entered.

b) Initialise a pushbutton named `closeBtn` which, when pushed, exits the program and closes the interface window.

The next step is to initialize the various parts of the interface window. In this GUI it is convenient to have one part where the model parameters, the mass *m*, the damping *c* and the spring constant *k* are entered. In a second part the frequency range for the FRF calculation must be entered. In a third part the calculations are performed and finally in a fourth part the calculated FRF is plotted in diagrams.

c) Begin with creating a header `MODEL PARAMETERS` placed at a suitable position in the interface window. This may be done with the command `uicontrol('style','text','string','MODEL ...';...)`.
Also create headers for entering frequency parameters, executing calculations and presenting results, placed at suitable positions in the interface. Name the headers `Frequency range:`, `Calculation:` and `Diagrams: Frequency Response Function`.

Below the header named `Model Parameters:` the numeric values of the model parameters should be entered from the keyboard. To accomplish this some initial default values of *m*, *c* and *k* must be defined.

d) Define initial default values of all model parameters in the interface initialization part. Also initialize values of the minimum frequency, the maximum frequency and the frequency increment for the FRF calculation. Then place a lead texts `'Mass m [kg]:'` etc at suitable locations. Finally continue in the same manner with initialization of the frequency range parameters.

Numeric values entered from the keyboard can be introduced by using an editable field. A field named `massField` where you can enter the mass from the keyboard during the execution is created with the command

```
massField = uicontrol('style','edit','string',num2str(m),'callback',
'sdofgui(''newmass'');'...)
```

Note that the numeric value is converted to a string variable using the `num2str` command. The change of value of the mass from the old to the new is accomplished by the `callback` command where the function `sdofgui` calls itself with the `action` variable set to the string '`newmass`'. The commands actually changing the value of the mass are then placed in the

```
elseif strcmp(action,'newmass')
...
elseif strcmp(action,…)
...
end
```

part of the program.

e) Change the program so that the model parameters and the frequency parameters can be changed from the keyboard. Note that in order to obtain a changed numeric value of the mass *m* you must first read the contents in the '`string`' parameter of the variable named `massField`. This is done with the `get(massField,'string')` command which will return a string which is the string equivalent to the numeric value of *m*. Then you must convert this string to its numeric equivalent using the `str2num` command. Note that both the `massField` variable and the mass *m* must be declared as `global`.

When all parameters have been updated to their correct values, the FRF calculation starts by pushing a button.

f) Initialise a pushbutton named `calcBtn` which when pushed calculates the frequency response function of the system according to the equations given above. Label the button with the string '`FRF`'.

Finally the calculated frequency response function should be plotted in a diagram. The plotting may be performed with a `Plot` button, but it is more convenient to introduce the plot commands in the same action part and directly after the calculation commands.

Since a frequency response function is complex valued two graphs are needed, one showing the absolute value and one showing the phase angle. Areas for plotting are created with the command `axes('position',...)` .

g) Create in the initialisation part two suitably positioned plotting areas named `absAxes` and `phaseAxes`. Add lines for plotting the absolute value and the phase to the action `FRFcalc` part. Note that the plotting areas are activated with the `subplot('...')` command.

72 (Level 3)

The following graphic user interface lets the user select a measured signal and plot in a lin-lin diagram. Add a pushbutton `Sound` to the graphic user interface so that the user can listen to the signal. Try the code on the data given in the files `gearvibr.bin` (containing acceleration signal measured on a gearbox), `railvibr.bin` (containing acceleration signal measured on a rail during a train passage), `heart_1.bin` (containing pressure signal

measured on a healthy heart) and `heart_2.bin` (containing pressure signal measured on a heart with a heart condition).

```
% sigplot.m
% GUI for plotting and listening to a user selected signal.
%
% Author: U Carlsson, 2001-08-23

% Create figure window for graph.

figWindow = figure('Name','Plot alternatives');

% Create file input button. Press 'File' calls the function 'firstplot'.

fileinpBtn = uicontrol('Style','pushbutton','string','File','position',...
             [5,395,40,20],'callback','[tdat,pdat,fsampl] = firstplot;');

% Create 'Sound' button.

fileinpBtn = uicontrol('Style','pushbutton','string','Sound','position',...
             [75,395,45,20],'callback','soundsc(pdat,fsampl);');

% Create exit pushbutton with red text.

exitBtn = uicontrol('Style','pushbutton','string','EXIT','position',...
          [510,395,40,20],'foregroundcolor', [1 0 0],'callback','close;');
% End of sigplot.m

function [tdat,pdat,fsampl] = firstplot
% Brings template for file selection. Reads selected filename and path and plots
% spectrum in a lin-lin diagram. Output data are frequency and pressure amplitude
% vectors, fdat and pdat.

% Author: U Carlsson, 2001-08-23

% Call MATLAB function 'uigetfile' that brings file selection template.

[filename,pathname] = uigetfile('*.bin','Select binary data-file:');

% Change directory. Open file for reading binary floating point numbers.
% Read data. Close file.

cd(pathname);
fid = fopen(filename,'rb');
data = fread(fid,'float32');
fclose(fid);

% Partition data vector into time and signal vectors.

tdat = data(1:2:length(data));
pdat = data(2:2:length(data));

% Calculate the sampling frequency for correct sound reproduction.

fsampl = 1/(tdat(2)-tdat(1));

% Plot pressure signal in a lin-lin diagram. Define suitable axis labels.

plot(tdat,pdat);
xlabel('Time, [s]');
ylabel('Signal amplitude');

% End of file firstplot.m
```

SOLUTIONS AND HINTS TO SOME EXERCISES

1

a) -3
b) 13.5000
c) -13.4000
d) -76.4000

2

a)
```
>> 8+3*(2+3)/4
ans =
      11.7500
```
b)
```
>> -5+7*(1+4*(1+3))/3/(2+5)
ans =
      0.6667
```
c) 68.3333

3

a) -11
b) 3.0021
c) 11.1250

4

a) $6.0060 \cdot 10^3$
b) $1.3152 \cdot 10^8$

5

a) `Inf`
b) `NaN`

6

All expressions in this exercise are complex-valued. The imaginary unit, *i*, is introduced in MATLAB as a predefined name `i` or `j`.

a) `>> (1+i)+(2-i)` (or equivalently `(1+j)+(2-j)`) gives
```
ans =
      3
```
b) 3.0000 + 1.0000i
c) 0.2000 + 0.6000i
d) 2

7

π is predefined in MATLAB as `pi`.

a)
```
>> y=3*sin(pi/4)
y =
    2.1213
```
b) -2.1213
c) 3.0000
d) -3.0000

8

a) 0.7071
b) -0.0292
c) 1.5708
d) 0.4636

9

a) 2.7183
b) 0.6931
c) 1.0137
d) 0.3010
e) 6.5321

10

Find the base 10 logarithm function using MATLAB Help browser.

a)
```
>> y = sqrt(5*sin(pi/3))
y =
    2.0809
```
b) 41.7647
c) 2.6665
d) 1.3029
e) 1.5708 (Expression is equivalent to arcsin(sin(π/2))).

11

Use the inverse trigonometric functions `asin`, `acos`, and `atan`. Remember that trigonometric equations may have several solutions. Use help facilities to find the functions domains and ranges in MATLAB.

a)
```
>> v = asin(0.9)*180/pi
v =
    64.158
```
b) 114.83
c) -7.4069
d) Yes, since the trigonometric functions are periodic there are an infinite number of roots. See basic mathematics textbooks. Also see MATLAB Help.

12

Corrected code:

```
% radius.m
L = input('Enter the circumference L: ');
r = L/2/pi       % Alternatively r = L/(2*pi)
```

Test run,

```
>> radius
Enter the circumference L: 2*pi
r =
     1
```

13

a)
```
>> pref=2e-5;
>> Lp=10*log10((35e-3/pref)^2)
Lp =
      64.8608
```

b) 94.8073

c) 140

14

Execution with erroneous code:

```
>> xplusy
z =
     7
z =
     7
```

Corrected code:

```
% xplusy.m
% Part a)
clear
x = 5;
y = 2;
z = x + y
% Part b)
x = 5;
y = x;
z = x + y
```

Execution with corrected code:

```
>> xplusy
z =
     7
z =
    10
```

15

a)
```
>> Lptot=10*log10(10^(85/10)+10^(85/10))
Lptot =
          88.0103
```

b) 146.0206

16

a) `a = [1 2 1]` , alternatively `a = [1,2,1]`

b) `b = [1;2;1]`

c) `c = 2:25;`

d) *T* denotes transposed matrix or vector.
`d = [20:-2:-20]';`

e) `e = [0.1:0.1:100]';`

17

a)
```
>> x=(1:200)'*10*pi/200;
>> y=sin(x);
>> y(24)
ans =
    -0.5878
```

b)
```
>> x = 1/pi*(1:1000)';
>> y = sin(sqrt(x));
>> y(524)
ans =
     0.3415
```

18

a) Use the keyword `roots` in the MATLAB Help browser.

b) The roots are found in two steps. First define the coefficient vector `c` defining the polynomial.

```
>> c = [1 0 1 -1];
```

Then use `roots` to find the polynomial roots:

```
>> x = roots(c)
x =
  -0.3412 + 1.1615i
  -0.3412 - 1.1615i
   0.6823
```

The answer is a vector with three roots, one real-valued and two complex-valued.

19

Build the time vector. There are several alternative solutions.

Alternative 1: `x = (0:1/99:1)';  % Gives a column vector`

Alternative 2: `x = linspace(0,1,100); % Gives a row vwctor`

a)
```
>> t(50)
ans =
    0.4949
```

b)
```
>> t(51:60)
ans =
    0.5051
    0.5152
    0.5253
    0.5354
    0.5455
    0.5556
    0.5657
    0.5758
    0.5859
    0.5960
```

c) The vector t_1 containing every fifth component of t is built by

```
t1 = t(5:5:length(t)); % Alt t1 = t(5:5:end);
```

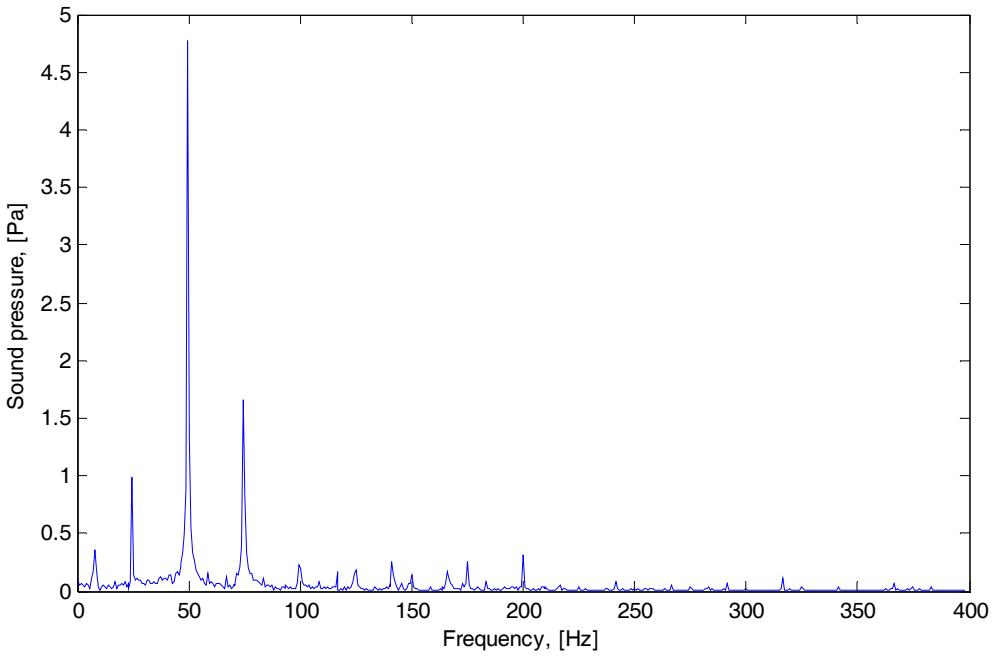
The value of the 17th component of this vector is,

```
>> t1(17)
ans =
    0.8485
```

The functions $\sin(2\pi t)$ and $\sin(2\pi t_1)$ are plotted with the command,

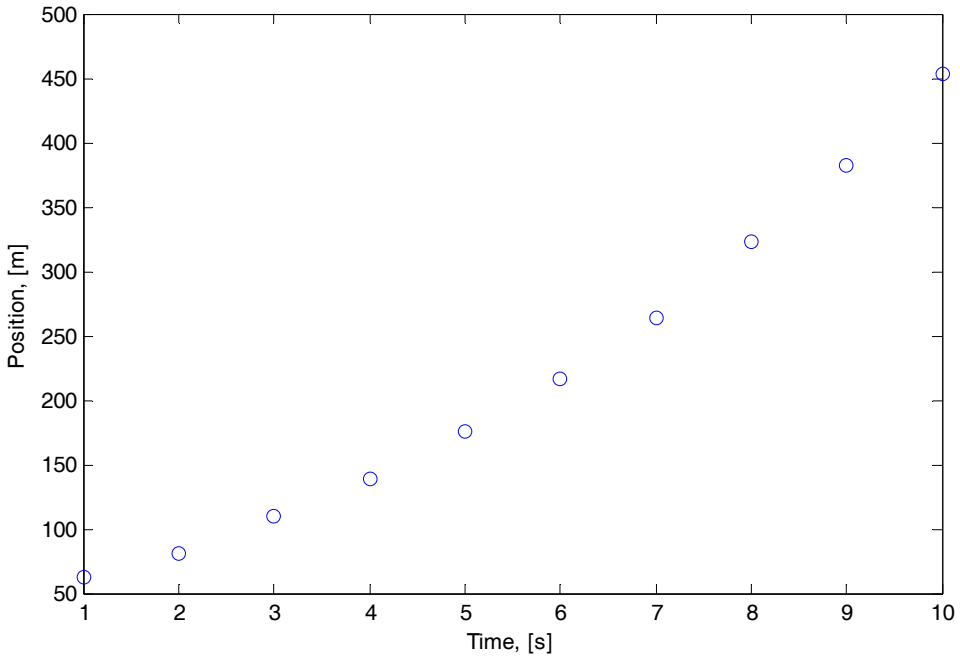
```
plot(t,sin(2*pi*t),t1,sin(2*pi*t1))
```

20
d)



21

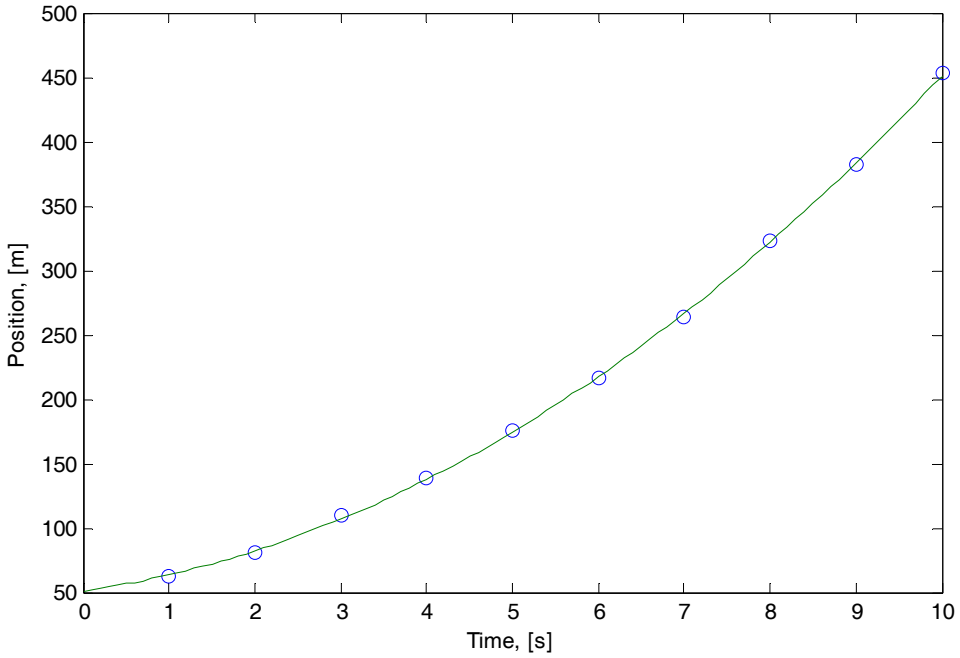
b)



c) $a = 3.0659$ [m/s^2], $v_0 = 9.3411$ [m/s], $s_0 = 51.1567$ [m].

d) Car position at $t = 0$ is 51.1567 m ≈ 51.2 m;
at $t = 2.3$ seconds, $s = 88.8598$ m ≈ 88.9 m.

e)



22

a) Start with building the first part of the function in the interval [0, π]. The following commands generate a column vector `x1` with 50 equidistantly spaced components on the interval [0, 49π/50] and a column vector `y1` with the corresponding sine values.

```
>> x1 = [0:49]'*pi/50;
>> y1 = sin(x1);
```

Use the vectors `x1` and `y1` as building blocks and build the rectified sine function.

```
>> x = [x1;x1+pi;x1+2*pi;x1+3*pi;x1+4*pi;x1+5*pi];
>> y = [y1;y1;y1;y1;y1;y1];
```

Plot the result in a graph.

```
>> plot(x,y)
```

Note that the end point of the interval, 49π/50, is selected so that no boundary points are doubled when the rectified sine is built.

b) A simpler solution is to use the absolute value of the sine function.

```
>> x1 = [0:1/50:1-1/50]'*pi;
>> x = [x1;x1+pi;x1+2*pi;x1+3*pi;x1+4*pi;x1+5*pi];
>> y = abs(sin(x));
```

23

Use the vectors `x` and `y` built in exercise 22. Use the command `axis` to change the plot axis. The command `grid` generates grid lines in the graph.

```
>> axis([0 6*pi -0.5 1.5])
>> grid
```

24

In the first alternative the curve is constructed using the same technique as in exercise 22.

```
>> x1 = [0 1];
>> y1 = [1 0];
>> x = [x1 x1+2 x1+4 6];
>> y = [y1 y1 y1 1];
>> plot(x,y);
```

An alternative method is simply to enter the *x*- and *y*-coordinates for the break points in two vectors.

```
>> x = [0 1 2 3 4 5 6];
>> y = [1 0 1 0 1 0 1];
>> plot(x,y)
```

25

Suppose the *x*- and *y*-coordinates from exercise 22 are denoted `X1` and `Y1` and those from exercise 24 `X2` and `Y2`. The following plot command will produce the required multi-plot:

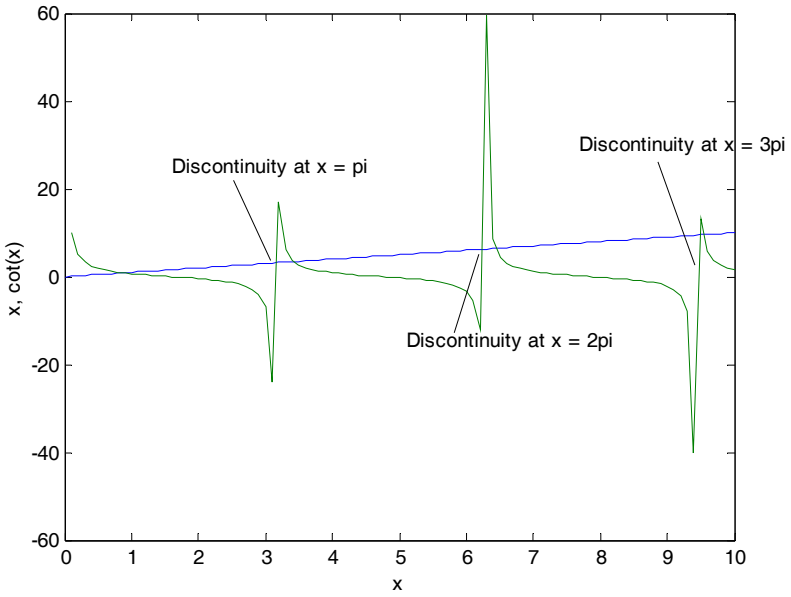
```
>> plot(X1,Y1,X2,Y2,'--')
```

26

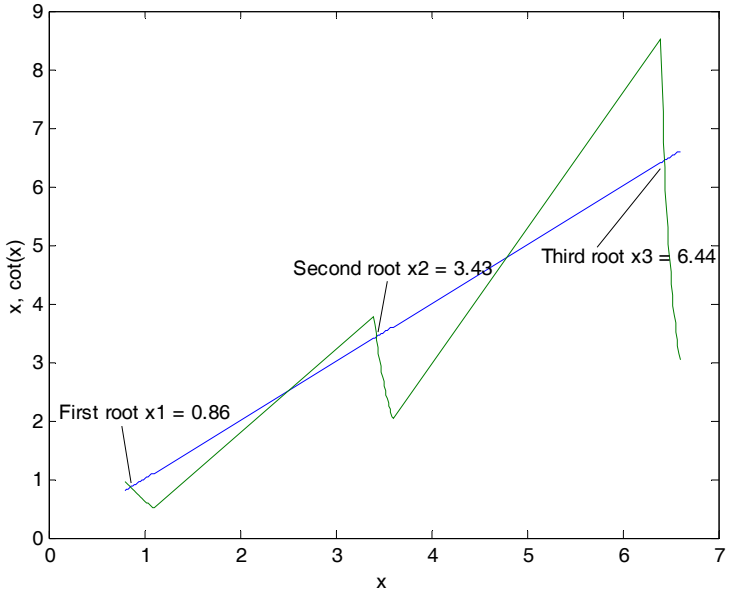
The first rough estimate of the roots can be obtained by looking at the function cot$x$ in a basic mathematics book or by plotting cot$x$ together with x in a diagram.

```
>> x1 = [0:0.1:10];
>> x2 = 1./tan(x1);
>> plot(x1,x1,x1,x2)
```

The plot of the two functions is given below. Clearly the three lowest positive roots are located around 1, 3.5 and 6.5. Note that the zero crossings in between these do not represent roots to the equation. These zero crossings are a consequence of the `plot` command connecting each point with a line. In reality the function has discontinuities at these points, approaching negative infinity from below and positive infinity from above.



The next step is to refine the plot around the roots and to get improved estimates using the zoom option. Which gives: $x_1 = 0.86$, $x_2 = 3.43$ and $x_3 = 6.44$.



27

Define the row vector u using the colon operator:

```
>> u = 1:10;
```

a) Calculate the scalar product:

```
>> u*u'
ans =
     385
```

b) Calculate the product:

```
>> u'*u
ans =
     1   2   3   4   5   6   7   8   9  10
     2   4   6   8  10  12  14  16  18  20
     3   6   9  12  15  18  21  24  27  30
     4   8  12  16  20  24  28  32  36  40
     5  10  15  20  25  30  35  40  45  50
     6  12  18  24  30  36  42  48  54  60
     7  14  21  28  35  42  49  56  63  70
     8  16  24  32  40  48  56  64  72  80
     9  18  27  36  45  54  63  72  81  90
    10  20  30  40  50  60  70  80  90 100
```

c) Calculate the scalar product with the command `dot`:

```
>> dot(u,u)
ans =
     385
```

d) Calculate the element-wise product:

```
>> u.*u
ans =
      1   4   9  16  25  36  49  64  81 100
```

28

a)
```
>> x = 0:0.05:5;
>> y = 10-x;
>> y(17)
ans =
     9.2000
```

b) 8.4826

c) 0.7600

d) 0.8967

29

Create the vector `x` with the *x*-values and vector `y` with the modulated signal. Finally plot the signal in a graph to view the results. Note that `.*` must be used when a component-wise multiplication is to be performed.

```
>> x = (1:1000)*2*pi/100;
>> y = sin(4*x).*(1.8+sin(0.1*x));
>> plot(x,y)
```

30

a)
```
>> F = [100 0 0];
>> s = [0 3 2];
>> A = F*s'
A =
     0
```

Comment: The work performed by the force is 0 since the displacement is orthogonal, i.e. perpendicular, to the force.

b) 115.

31

It is not possible to execute the given code, because the dimensions of the vectors `x` and `sin(x)` are incompatible. A correct code should use component-wise multiplication obtained with the `.*` operator.

```
x = (0:1/100:1)*10*pi;
y = x.*sin(x);
```

32

The matrix dimensions of the numerical constant 1 and the vector `(1+sin(x).^2)` are incompatible. A correct code should use component-wise division and reads,

```
x = (0:1/100:1)*10*pi;
y = 1./(1+sin(x).^2);
```

33

One possible graphical method is to plot the two functions, $y = x - 225$ and $y = x^4 + 5x^3 - 64x^2$, in the same diagram and find approximate solutions by zooming in the intersections. Estimates of the four real-valued roots are: x = -10.758, x = -1.803, x = 2.138 and x = 5.423.

34

```
>> f = linspace(0,500,251);
>> S1 = 0.5e-4;
>> S2 = 5e-4;
>> L = 1;
>> c = 340;
>> DTL = 10*log10(1+(S1/2/S2-S2/2/S1)^2*(sin(2*pi*f*L/c)).^2);
>> plot(f,DTL)
>> xlabel('Frequency, [Hz]');
>> ylabel('Transmission loss, [dB]');
```

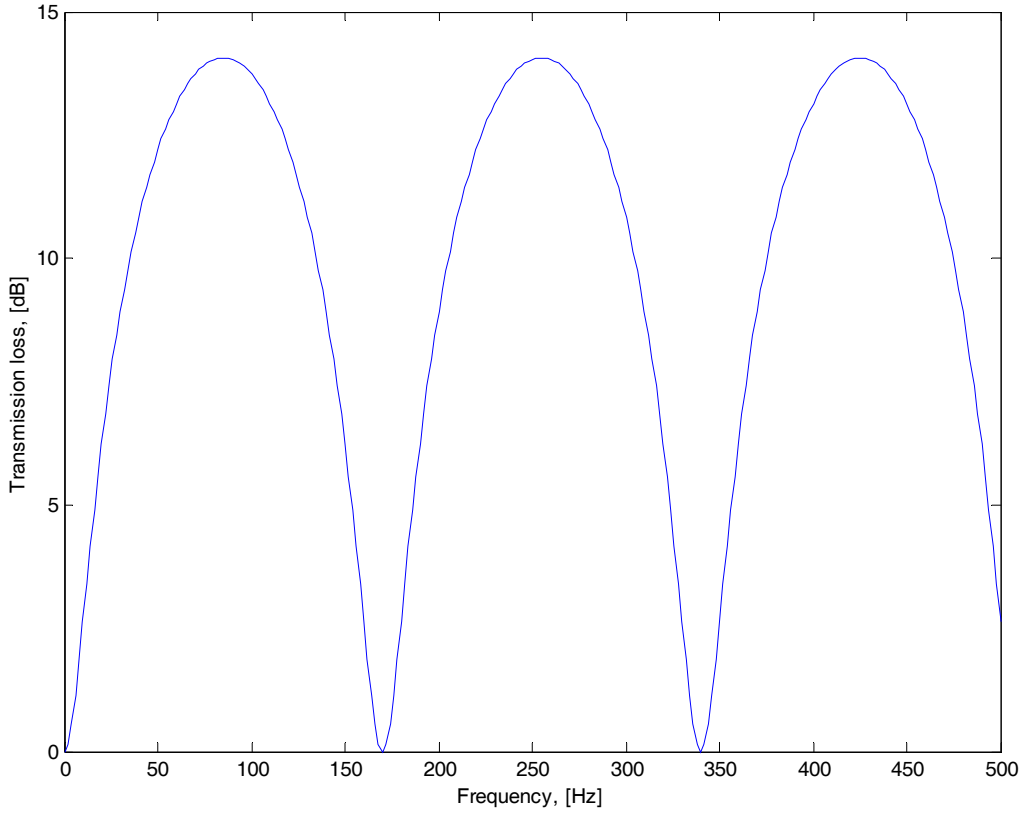
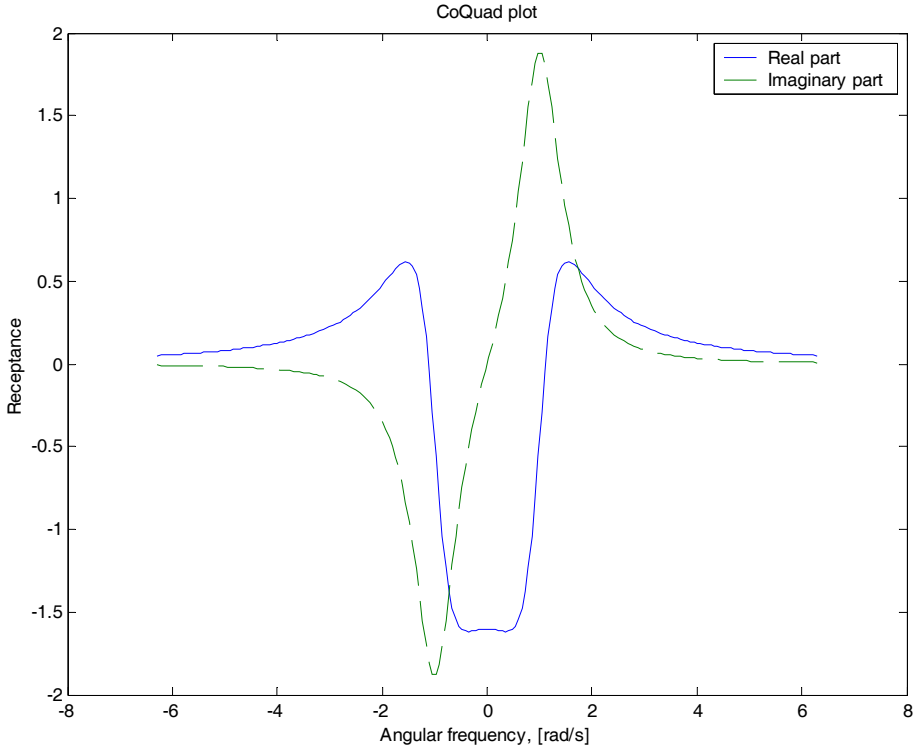


Figure Transmission loss as function of frequency for silencer in exercise 34.

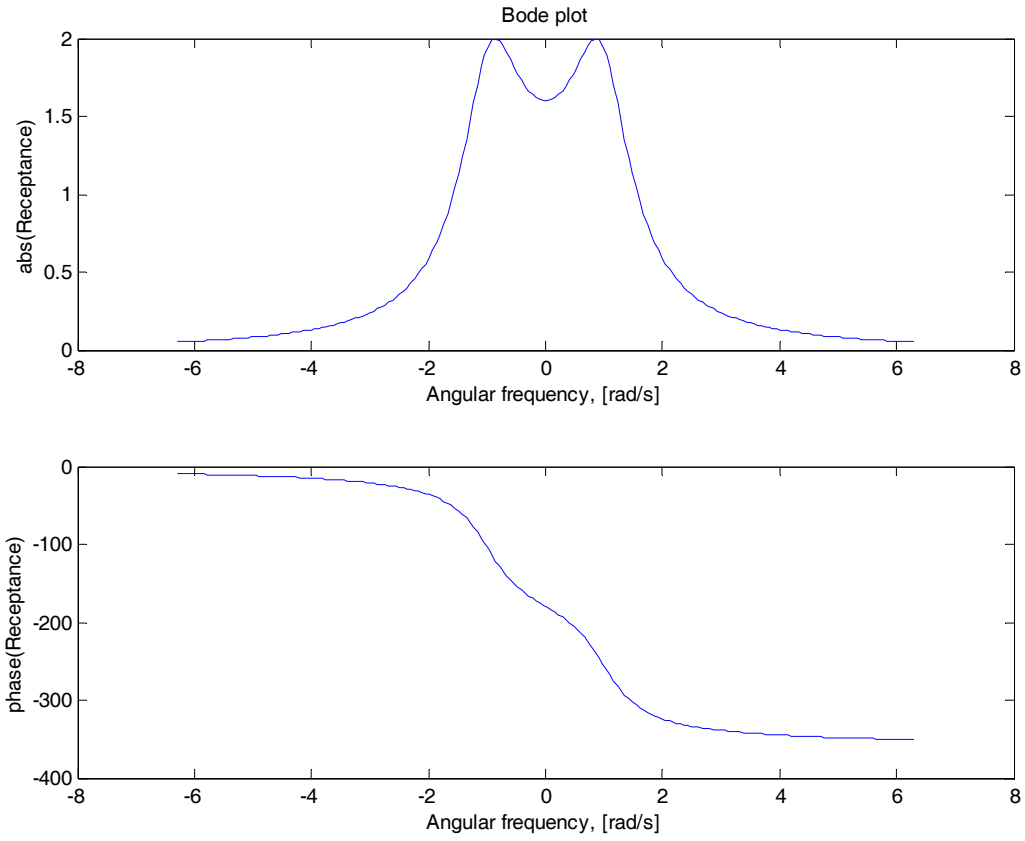
35

```
>> f=linspace(-1,1,200);
>> f=f';
>> om=2*pi*f;
>> h=1./(om-1-0.5*i)-1./(om+1-0.5*i);

>> f(41)
ans =
   -0.5980

>> h(41)
ans =
   0.1432 - 0.0418i

>> [maxvalue,maxlin]=max(abs(h))
maxvalue =
  1.9995
maxlin =
    87

>> f(maxlin)
ans =
    -0.1357
```

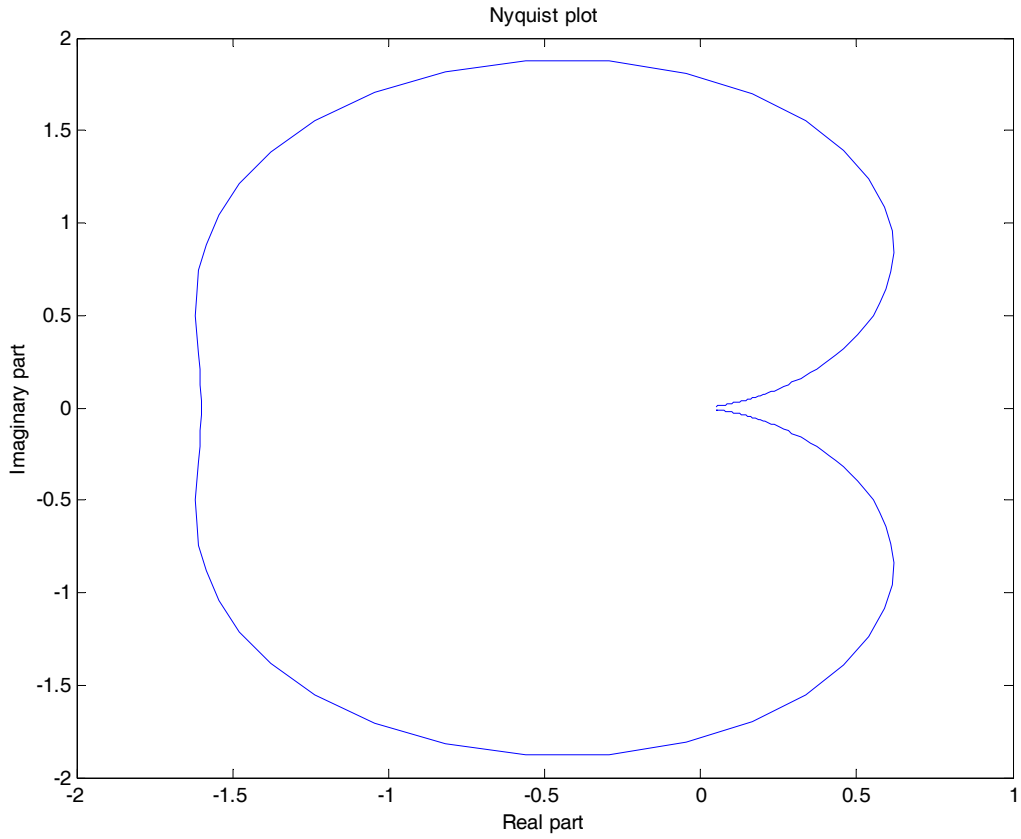
```
>> freqint=f(abs(h)>1.9)
freqint =
   -0.1558
   -0.1457
   -0.1357
   -0.1256
   -0.1156
   -0.1055
    0.1055
    0.1156
    0.1256
    0.1357
    0.1457
    0.1558

>> plot(om,real(h),om,imag(h),'--')
>> xlabel('Angular frequency, [rad/s]')
>> ylabel('Receptance')
>> legend('Real part','Imaginary part')
>> title('CoQuad plot')
```



```
>> phaseangle = 180/pi*unwrap(angle(h)); % Degrees
>> figure          % New figure window
>> subplot(211)
>> plot(om,abs(h))
>> xlabel('Angular frequency, [rad/s]')
>> ylabel('abs(Receptance)');
>> title('Bode plot')
>> subplot(212)
>> plot(om,phaseangle)
>> xlabel('Angular frequency, [rad/s]')
>> ylabel('phase(Receptance)');
```



```
>> subplot(111);  % Restore to full plot area
>> plot(real(h),imag(h));
>> xlabel('Real part')
>> ylabel('Imaginary part')
>> title('Nyquist plot')
```



36

a)
```
>> a = [1 2 1;0 2 1;0 0 3]
a =
   1     2     1
   0     2     1
   0     0     3
```

b) Can be solved using the identity matrix command `eye`.
```
>> b = eye(4)
b =
      1     0     0     0
      0     1     0     0
      0     0     1     0
      0     0     0     1
```

c)

The matrix can be partitioned to, $\begin{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \end{bmatrix}$.

```
>> c = [eye(2) zeros(2);zeros(2) ones(2)]
ans =

     1     0     0     0
     0     1     0     0
     0     0     1     1
     0     0     1     1
```

d) The matrix is most simply built by partitioning the matrix in 2×2 blocks and by introducing the following two by two matrices.

```
>> a = eye(2);
>> b = ones(2);
>> c = [3 1; 3 1];
>> e = [3 1; 1 4];
```

Finally these are combined to the requested matrix.

```
>> d = [a a 2*b; a 3*b c; 2*b c' e]
d =
     1     0     1     0     2     2
     0     1     0     1     2     2
     1     0     3     3     3     1
     0     1     3     3     3     1
     2     2     3     3     3     1
     2     2     1     1     1     4
```

37

```
>> A = [1 0 0;0 2 0;0 0 3];          % Alt  A = diag(1:3);
>> B = [2 1 2;2 -2 -1;1 2 -2]/3;
>> b = [0 1 0]';
```

a)
```
>> A*b
ans =
     0
     2
     0
```

b)
```
>> B*b
ans =
    0.3333
   -0.6667
    0.6667
```

c)
```
>> A*B
ans =
    0.6667    0.3333    0.6667
    1.3333   -1.3333   -0.6667
    1.0000    2.0000   -2.0000
```

d)
```
>> B*A
ans =
    0.6667    0.6667    2.0000
    0.6667   -1.3333   -1.0000
    0.3333    1.3333   -2.0000
```

Hence, matrix multiplication is not commutative.

e) **A** is a scaling matrix which changes the length but not the orientation of the vector. For instance the orientation of the vector **b** is not changed after multiplication with **A**. This is verified by calculating the direction cosine before and after the multiplication.

f) **B** is a rotation matrix which changes the orientation but not the length of the vector. For instance the length of **b** is not changed after multiplication with **B**.

38

```
>> x = [2 1 3 2];
```

a)
```
>> A = x'*x;
```

b)
```
>> A(2,3) + A(3,2)
ans =
     6
```

c)
```
>> B = A(2:3,2:3)
B =
     1     3
     3     9
```

d)
```
>> sum(A(3,:))
ans =
    24
```

39

a) Linear systems of equations are solved using the operator \.
Define the coefficient matrix and the right-hand side vector:

```
>> A = [2 1;1 2];
>> b = [1;1];
```

Solve the equation:

```
>> x = A\b
x =
    0.3333
    0.3333
```

b)
```
x =
    0.4286
   -1.1429
    0.7143
```

40

a) The system can be written, $\mathbf{Ax} = \mathbf{b}$, where,

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix},\ \mathbf{b} = \begin{Bmatrix} 1+i \\ 2+i \end{Bmatrix} \text{ and } \mathbf{x} = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix}.$$

```
>> A = [1 1;1 2];
>> b = [1+i;2+i];
>> x = A\b
x =
         0 + 1.0000i
    1.0000
```

b)
```
x =
   1.0000 + 2.0000i
  -0.5000
```
c)
```
x =
   1.6309 + 1.2416i
  -0.9664 - 0.4765i
   0.4295 + 0.1007i
```

41

a) The MATLAB command `eig` solves eigenvalue problems of the kind specified in equation (1).

b)
```
>> A=[2 1;1 2];
>> B=[2 -1;-1 2];
>> [v,d]=eig(A,B)
v =
   -0.4082    0.7071
    0.4082    0.7071
d =
    0.3333         0
         0    3.0000
```
c)
```
v =
    0.3466    0.2806   -0.3047
    0.4234   -0.3331    0.0382
    0.2607    0.0416    0.7452
d =
    0.2205         0         0
         0    0.5974         0
         0         0    1.2655
```

42

a) Let `v` be the eigenvector matrix with the eigenvectors in each column and `d` the eigenvalue matrix with the eigenvalues as diagonal elements.
```
>> A = [2 1;1 2];
>> [v,d] = eig(A)
v =
   -0.7071    0.7071
    0.7071    0.7071
d =
     1     0
     0     3
```
b) In this case the matrix **A** is non-symmetric.
```
v =
    0.8165   -0.8165
    0.5774    0.5774
d =
    3.4142         0
         0    0.5858
```

c) In this case the matrix **A** is non-symmetric and complex-valued.

```
v =
   0.9540             -0.4045 + 0.0364i
   0.2301 + 0.1922i    0.9138
d =
   2.4426 - 0.0398i         0
        0             -0.4426 + 2.0398i
```

d)

```
v =
   0.6848             -0.5181 + 0.2686i  -0.5181 - 0.2686i
  -0.5775             -0.0524 + 0.3631i  -0.0524 - 0.3631i
   0.4443              0.7245             0.7245
d =
  -1.1409                   0                  0
        0              3.5705 + 0.6317i        0
        0                   0             3.5705 - 0.6317i
```

43

Define the matrices **A** and **B**:

```
>> A = [2 -1 0;-1 2 -1;0 -1 2];
>> B = [4 -1 1;-1 4 -1;1 -1 2];
```

The command `[V,D] = eig(A,B)` returns the solution vectors as columns in the matrix `V` and the eigen-values as diagonal elements in the matrix `D`.

```
>> [V,D] = eig(A,B)
D =
    0.2205         0         0
         0    0.5974         0
         0         0    1.2655
V =
    0.3466    0.2806   -0.3047
    0.4234   -0.3331    0.0382
    0.2607    0.0416    0.7452
>> R=V'*A*V
R =
    0.2205    0.0000   -0.0000
    0.0000    0.5974    0.0000
   -0.0000    0.0000    1.2655
>> S=V'*B*V
S =
    1.0000   -0.0000   -0.0000
   -0.0000    1.0000    0.0000
   -0.0000    0.0000    1.0000
```

Hence both matrices **R** and **S** are diagonal. The eigenvector matrix **V** is said to be orthogonal with respect to **R** and **S**.

44

a) Begin with initializing the sum to 0, then for every n increase the sum with the value $1/n^2$.

```
>> s = 0;
>> for n = 1:100
s = s + 1/n^2;
end;
>> s
s =
    1.6350
```

b) 1.0019

45

a) A random column vector `x` with 100 components is created by,

```
x = rand(100,1);
```

b) The random number generator generates numbers in the range]0, 1[. Hence, a vector y with values in the specified set is obtained by multiplying with 10 and then rounding the components to the nearest integer above.

```
>> y = ceil(10*x);
```

c) The number of components larger than a specified value is obtained by using the `length` and `find` commands.

```
>> length(find(y>5))
ans =
    53
```

The `find` command extracts all components larger than 5 and the `length` command gives the number of components extracted by the `find` command. Note that your answer can be different because of the random nature of `x`.

46

$x = 0.8604$. Compare the result with exercise 26.

47

One possible code is

```
% countval.m
% Counts values of vector y in ranges.

% Ulf Carlsson. 2002-03-07.

clear;
x = (0:1/100:1)'*2*pi;
y = 2+2*sin(x);

number01 = 0;
number12 = 0;
number23 = 0;
number34 = 0;

for n = 1:length(y)
    if (y(n) >= 0 & y(n) <= 1)
        number01 = number01+1;
    elseif (y(n) > 1 & y(n) <= 2)
```

```
        number12 = number12+1;
    elseif (y(n) > 2 & y(n) <= 3)
        number23 = number23+1;
    elseif (y(n) > 3 & y(n) <= 4)
        number34 = number34+1;
    end;
end

number01
number12
number23
number34
```

% End of code

An alternative simpler code makes use of commands `max`, `size` and `find`. Replace the `for…if…` construction above with the following commands:

```
number01 = max(size(find(y>=0 & y<=1)));
number12 = max(size(find(y>1 & y<=2)));
number23 = max(size(find(y>2 & y<=3)));
number34 = max(size(find(y>3 & y<=4)));
```

The result of an execution is:

```
>> countval
number01 =
          33
number12 =
          18
number23 =
          17
number34 =
          33
```

48

A code solving the exercise is given in the following:

% Exercise48.m

```
% Ulf Carlsson. 2002-03-05.

% Initialise parameters

m = 1;
c = 1;
k = 1e4;
Ftopp = 10;         % Used in part e)
```

% Part a)

```
% Define column vector with equidistant frequency values on
% interval

f = (0:30/499:30)'; % Alt f = linspace(0,30,500)';
frequency = f(265)
```

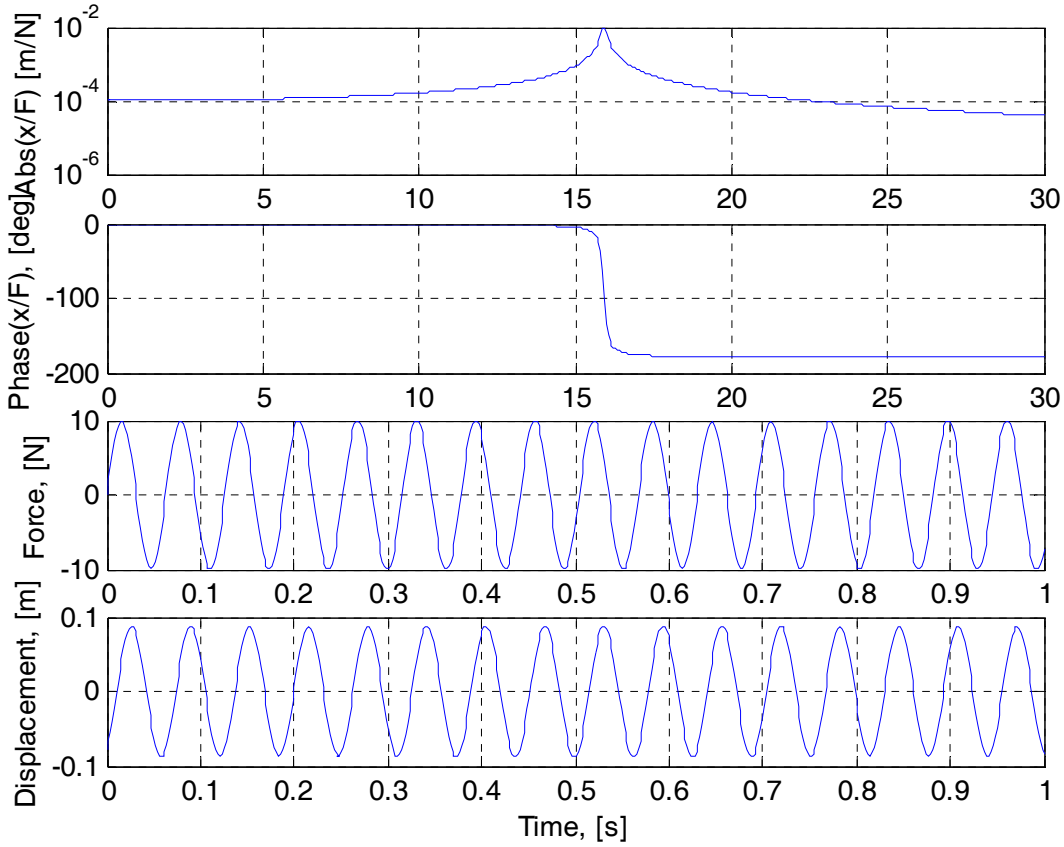
```
w = 2*pi*f;
w2 = w.^2;

% Part b)
% Calculate the frequency response function FRF in in points
% given by vector f

xoverF = 1 ./(-w2*m+i*w*c+k);
FRF = xoverF(265)

% Part c)
% Calculate absolute value and phase angle

sizexoverF = abs(xoverF);
phasexoverF = angle(xoverF);
absFRF = sizexoverF(265)
phaseFRF = 180/pi*phasexoverF(265)     % Phase angle in degrees

% Part d)
% Determine the largest absolute value and the corresponding
% frequency and phase.

[sizexoverFmax,linmax] = max(sizexoverF);
sizexoverFmax
frequencymax = f(linmax)
phasexoverFmax = 180/pi*phasexoverF(linmax)  % Phase in degrees

% Part e)
% Plot versus frequency in separate diagrams. Convert phase
% from radians to degrees in plot.

subplot(4,1,1);
semilogy(f,sizexoverF);
xlabel('Frequency, [Hz]');
ylabel('Abs(x/F) [m/N]');
grid;
subplot(4,1,2);
plot(f,180/pi*phasexoverF);
xlabel('Frequency, [Hz]');
ylabel('Phase(x/F), [deg]');
grid;

% Define equidistant time vector

t=linspace(0,1,500);

% Calculate displacement as function of time

x = sizexoverF(265)*Ftopp*sin(2*pi*f(265)*t+phasexoverF(265));
F = Ftopp*sin(2*pi*f(265)*t);

% Plot x(t) and F(t) in seprate diagrams

subplot(4,1,3);
plot(t,F);
xlabel('Time, [s]');
```

```
ylabel('Force, [N]');
grid;
subplot(4,1,4);
plot(t,x);
xlabel('Time, [s]');
ylabel('Displacement, [m]');
grid;

% Calculate time shift between force and displacement

lag = -phasexoverF(265)/2/pi/f(265)

% End of code
```

Running the code yields:

a) `frequency = 15.8717`
b) `FRF = 0.0042 - 0.0077i`
c) `absFRF = 0.0088`
`phaseFRF = -61.1652`
d) `sizexoverFmax = 0.0098`
`frequencymax = 15.9319`
`phasexoverFmax = -101.6180`
e) `lag = 0.0107`

The plot is shown below.



49

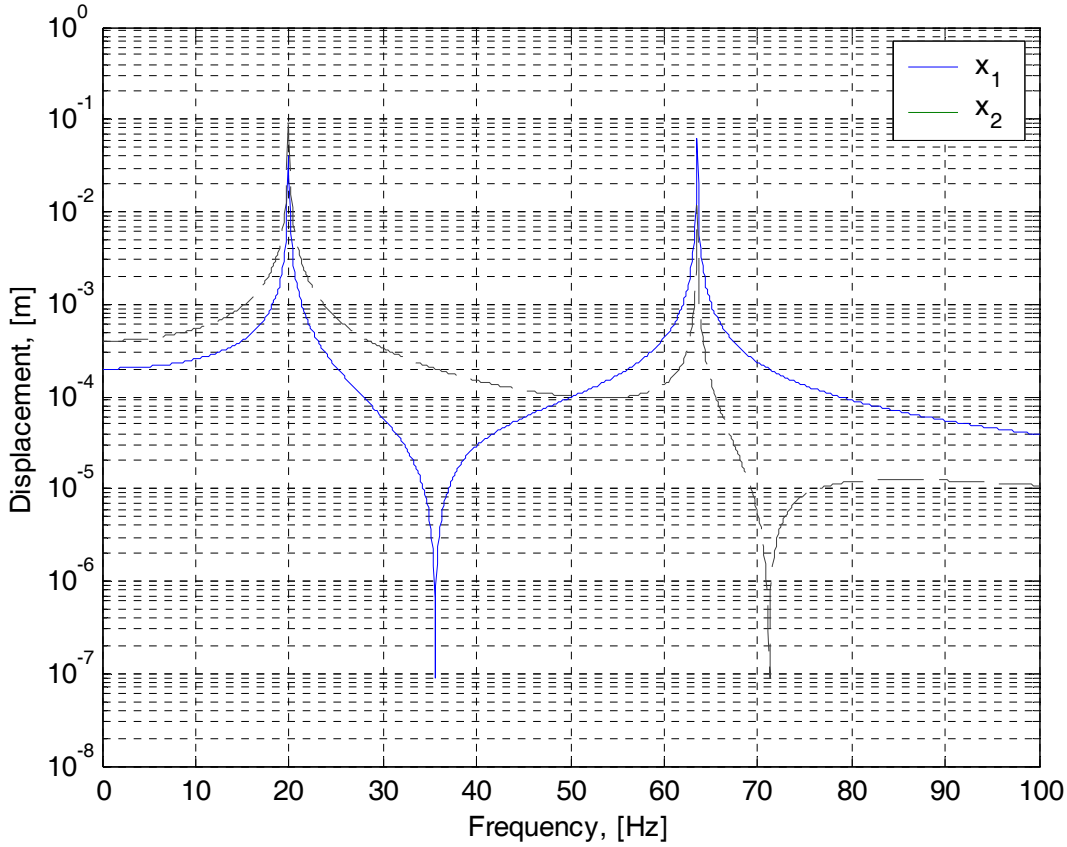
```
% Exercise49.m

% Ulf Carlsson. 2002-04-18.

% Initialise parameters

m1 = 1;
m2 = 2;
k1 = 1e5;
k2 = 5e4;
F1 = 10;
F2 = 10;

% Define column vectors and matrices

M = [m1 0;0 m2];
K = [k1+k2 -k2;-k2 k2];
F = [F1;F2];
f = linspace(0.1,100,1000); % Alt: f = (0.1:0.1:100);
om = 2*pi*f;
om2 = om.*om;

% Part a)
% Calculate the displacement amplitude as a function of frequency.

for n=1:length(f)
    A = -om2(n)*M+K;
    x = A\F;
    xtopp1(n) = x(1);
    xtopp2(n) = x(2);
end;

% Part b)
% Plot the absolute value of the displacement amplitude vector.

semilogy(f,abs(xtopp1),f,abs(xtopp2),'--')
xlabel('Frequency, [Hz]');
ylabel('Displacement, [m]');
grid;
legend('x_1','x_2');

% Part c,d)
% Determine the maximum displacement and
% the corresponding frequency.

[xmax,linmax] = max(abs(xtopp1));
fmax = f(linmax);
xtoppmax = [xtopp1(linmax),xtopp2(linmax)]';

% Normalize the maximum displacement vector so that the 1st
% component becomes 1.

xtoppmax = xtoppmax/xtoppmax(1);

% End of code
```

Running the program gives:

c) `fmax = 63.5000`

d)
```
xtoppmax =
    1.0000
   -0.1869
```

The graph produced in part b) is shown below.



50

```
% Exercise50.m

% Ulf Carlsson. 2002-04-17.

% Initialise parameters

m1 = 1;
m2 = 2;
k1 = 1e5;
k2 = 5e4;
F1 = 10;
F2 = 10;

% Define column vectors and matrices

M = [m1 0;0 m2];
K = [k1+k2 -k2;-k2 k2];
```

```
F = [F1;F2];

% Part a). Solve the eigenvalue problem.

[v,d] = eig(K,M);
feig = diag(sqrt(d))/2/pi;

% Part b). Normalize the second eigenvector, i.e. the second column
% of v, so that its 1st component becomes 1.

v(:,2) = v(:,2)/v(1,2);

% Part c). Compare the second eigenvector and the maximum
% displacement vector calculated in exercise 49 by calculating the
% relative difference between their components.
% xtoppmax is calculated in exercise 49.

reldiff = (v(:,2)-xtoppmax)./v(:,2);

% End of code
```

Running the program gives:

a) `feig =`

```
      19.9376
      63.5240
```

(compare with maximum frequency according to exercise 49, which was 63.5 Hz)

b) `v(:,2) =`

```
       1.0000
      -0.1861
```

Compare with results in exercise 43 part d).

c) `reldiff =`

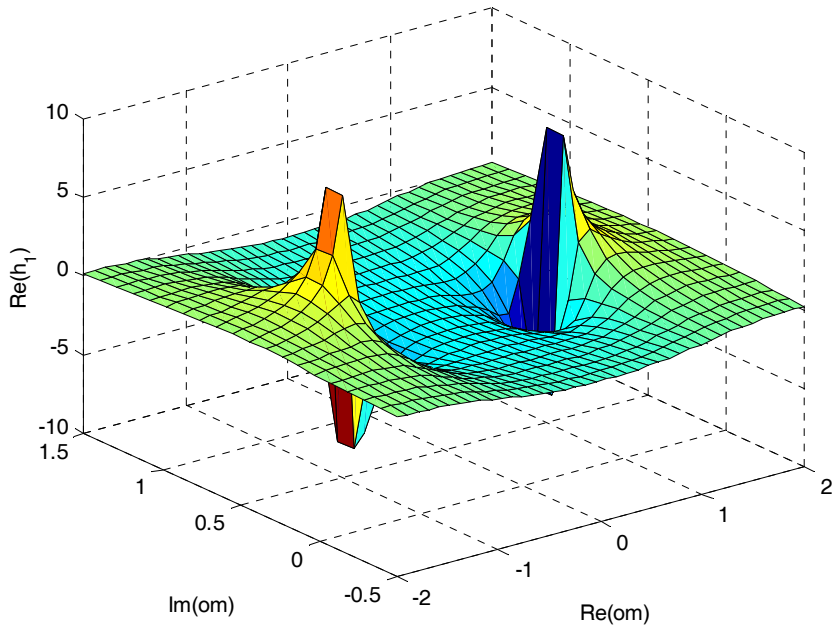
```
            0
      -0.0041
```

Hence, the relative difference is only 0.41 %. Obviously the eigenvalues and eigenvectors of a system provides information on frequencies that are easily excited to large motion amplitudes and the deformation shape for the system when it vibrates at these large amplitude conditions.

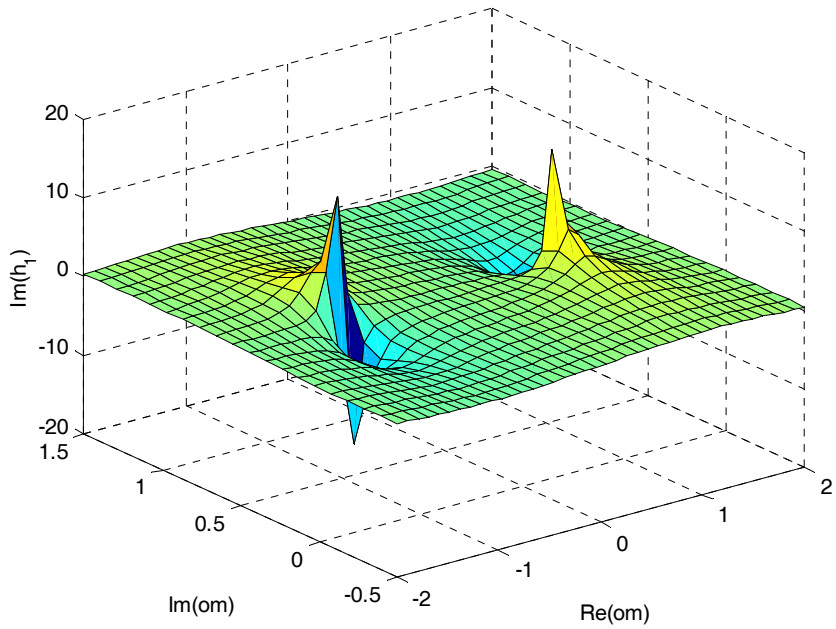
51

```
>> im=linspace(-0.5,1.5,20);
>> re=linspace(-2,2,40);
>> for n=1:40
>> for m=1:20
>> h1(n,m)=1./(re(n)+i*im(m)-1-0.5*i)-1./(re(n)+i*im(m)+1-0.5*i);
>> end;
>> end;

>> h1(24,14)
ans =
  -1.8581 - 0.4881i

>> re(24)+i*im(14)
ans =
   0.3590 + 0.8684i
```

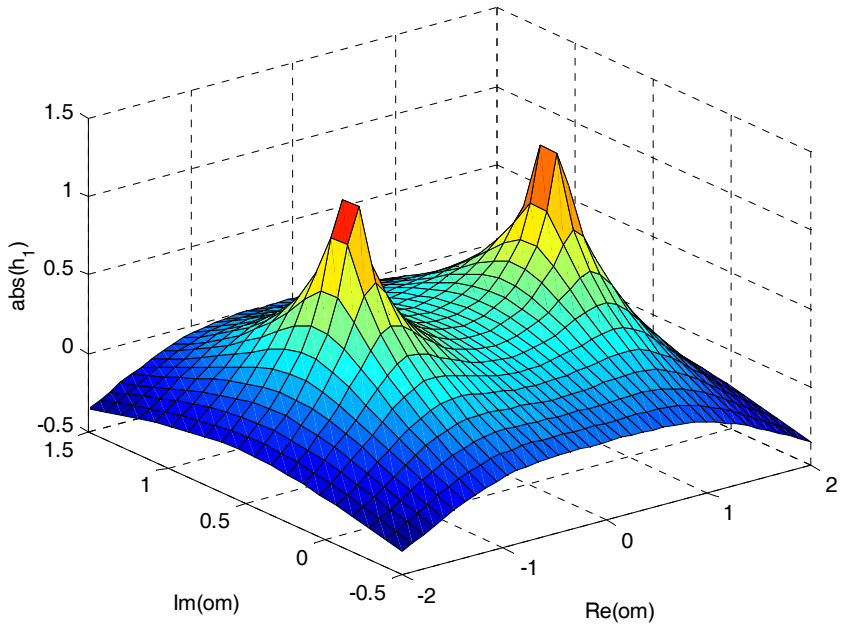
```
>> surf(re,im,real(conj(h1')));  % ' is conjugate transpose
                                 % Hence conj((...)') is transpose
>> xlabel('Re(om)');
>> ylabel('Im(om)');
>> zlabel('Re(h_1)');
```




```
>> surf(re,im,imag(conj(h1')))
>> xlabel('Re(om)');
>> ylabel('Im(om)');
>> zlabel('Im(h_1)');
```



```
>> surf(re,im,log10(abs(conj(h1'))))
>> xlabel('Re(om)');
>> ylabel('Im(om)');
>> zlabel('abs(h_1)');
```



52

The exercise is solved using the following MATLAB code:

```
% rmscalc.m
% Ulf Carlsson, MWL, KTH, 2002-08-09.

t = (0:0.0025:0.5);              % Equidistant sample points
f = 100;                         % Frequency
ptot = 1e5 + 0.01*sin(2*pi*100*t); % Calculate total pressure
p = ptot - mean(ptot);           % Calculate the sound pressure
T = 1/f;                         % The period of the sound pressure
tper = t(find(t<=T));     % Build a time vector with samples
                          % covering a single period

% Plot the sound pressure with the sample points overlayed as rings

plot(tper,p(1:length(tper)),tper,p(1:length(tper)),'o');

axis([0 max(tper) -0.012 0.012]);  % Set the axis scaling
xlabel('Time, t, [s]');            % Horizontal axis notation
ylabel('Sound pressure, p, [Pa]'); % Vertical axis notation
N = length(t);                     % Number of samples
prms = sqrt(1/N*sum(p.*p))       % Calculate and write the rms value
prmsex = 0.01/sqrt(2);           % Calculate the exact rms value
relerror = (prms-prmsex)/prmsex  % Calculate the relative error

% End of code
```

a) Running the code gives the following output:

```
>> rmscalc
prms =
    0.0071
relerror =
   -0.0025
```

Hence, the estimation is 0.25 percent lower than the correct value. The plot of a period of the sampled function is given in figure 1 below.

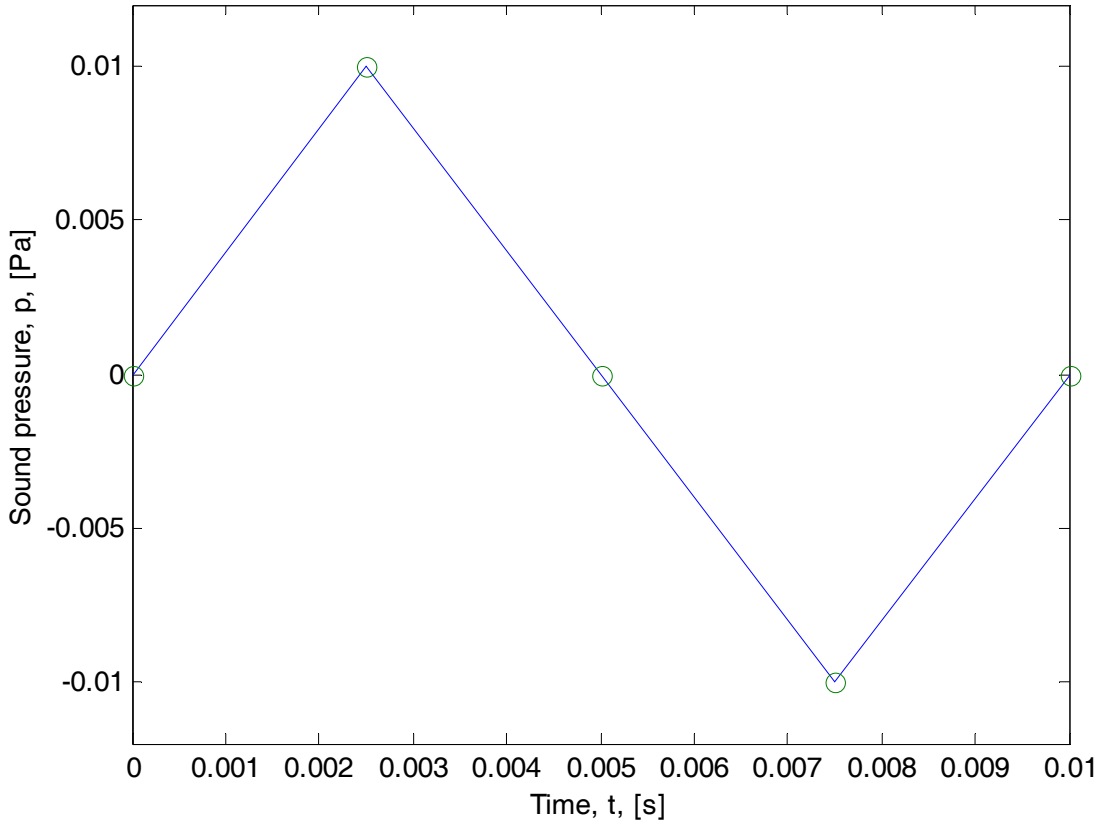


Figure 1 Sampled sound pressure with time increment 0.0025 s. Rings denote sample points.

b) Running the code with modified time increment between samples gives the following results:

```
>> rmscalc
prms =
        0.0071
relerror =
       -2.4991e-004
```

Hence the relative error is reduced a factor of 10 when the sampling rate is changed. The plot of a period of the sampled function is given in Figure 2 below.

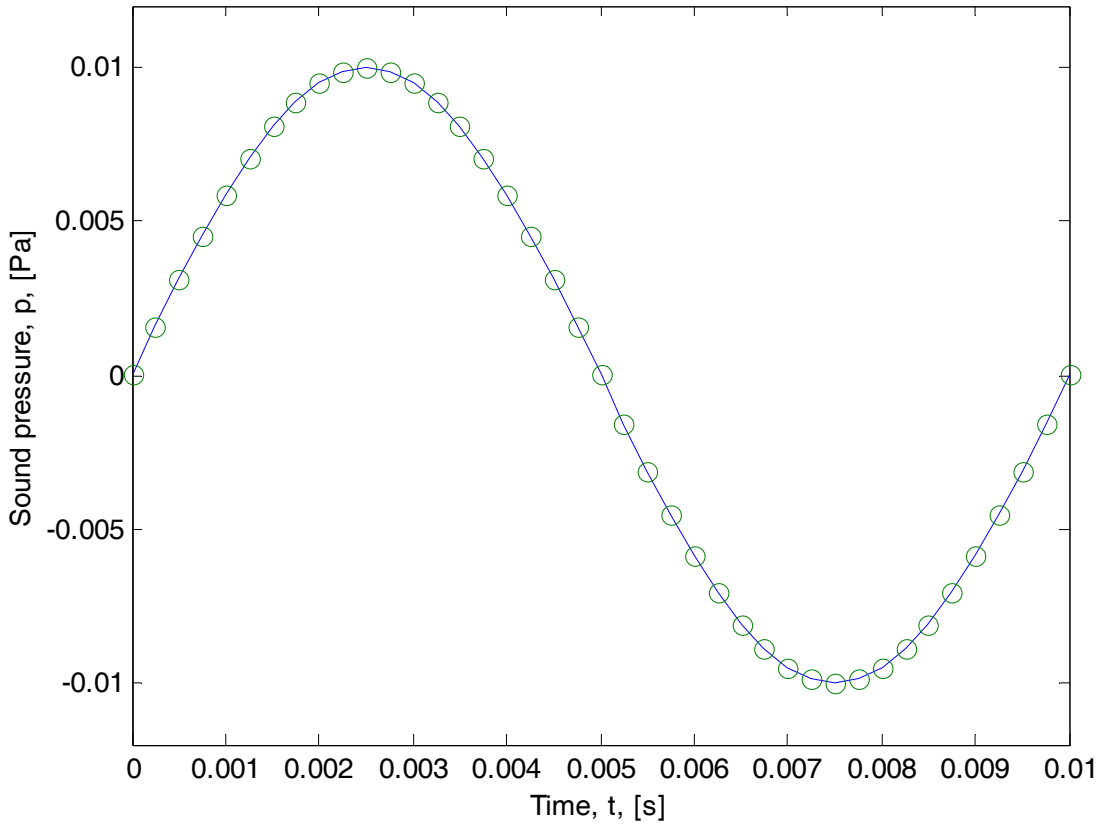


Figure 2 Sampled sound pressure with time increment 0.00025 s. Rings denote sample points.

53

a) The following code solves part b) – f) of the exercise

```
% splevel.m
% Program that calculates the total sound pressure level and
% the sound pressure level as function of frequency.
% Sound pressure data is stored on floating point format
% 'float32' in a binary file specified in a dialog box. The
% storage sequence is f1, p1, f2, p2, ... .

% Ulf Carlsson, 2002-03-07.

clear

% Read name and path of input data file.

[name,path] = uigetfile('*.bin',...
'Specify a binary input data-file');

% Open specified datafile and read binary data.

fid = fopen([path,name],'rb');
data = fread(fid,'float32');
fclose(fid);
```

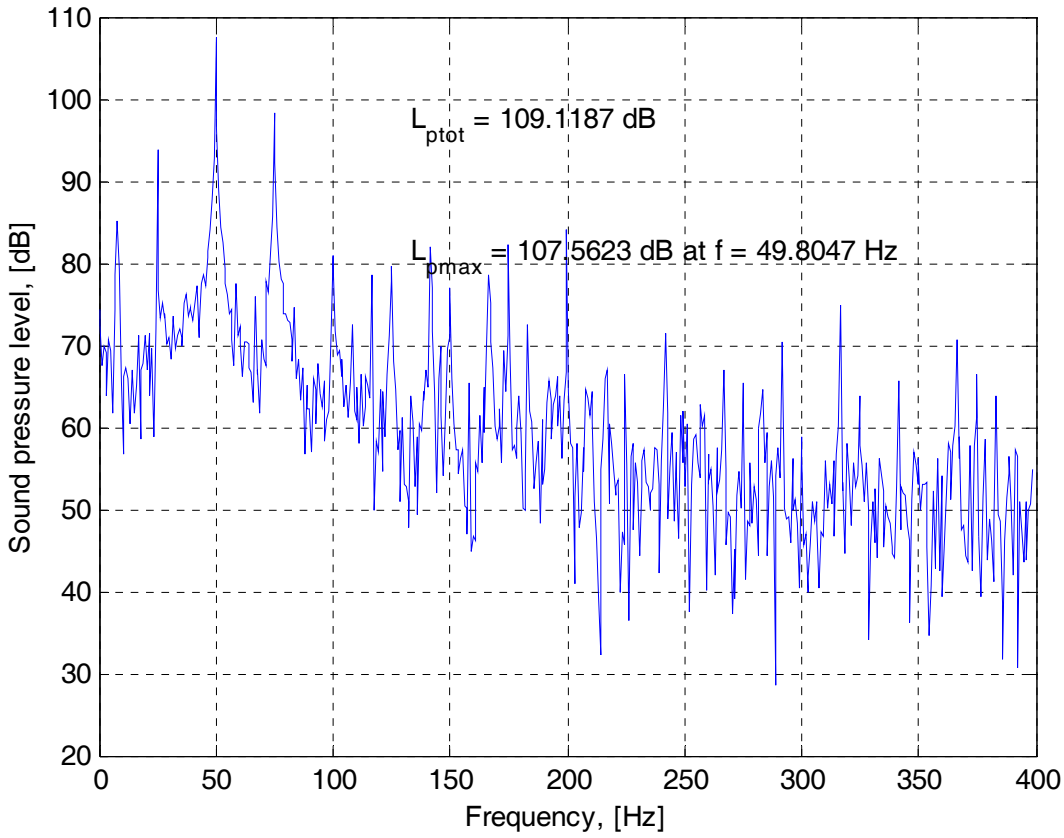
```
% Partition data into x and y variables.

f = data(1:2:end);
p = data(2:2:end);

% Calculate sound pressure level as function of frequency.

p2 = p.*p;
Lp = 10*log10(p2/(2e-5)^2);

% Localise the maximum value and its frequency.

[Lpmax,linmax] = max(Lp)
freqmax = f(linmax)

% Calculate the total sound pressure level.

p2sum = sum(p2);
Lptot = 10*log10(p2sum/(2e-5)^2)

% Plot the sound pressure level as function of frequency.

plot(f,Lp);
xlabel('Frequency, [Hz]');
ylabel('Sound pressure level, [dB]');
grid

% Add info to plot

text1 = text(f(length(f))/3,0.9*Lpmax,...
   ['L_p_t_o_t = ',num2str(Lptot),' dB']);
text2 = text(f(length(f))/3,0.75*Lpmax,...
    ['L_p_m_a_x = ',num2str(Lpmax),...
    ' dB at f = ',num2str(freqmax),' Hz']);
```

% End of code

Execution of the code:

```
>> splevel
Lpmax =
   107.5623
linmax =
    69
freqmax =
   49.8047
Lptot =
   109.1187
```

A plot of the calculated sound pressure level is given in the figure below.



54
The modified code is:

```
% splevelmod.m
% Program that  calculates the total sound pressure level and
% the sound pressure level as function of frequency.
% Sound pressure data is stored on floating point format
% 'float32' in a binary file specified in a dialog box.
% The storage sequence is f1, p1, f2, p2, ... .

% Ulf Carlsson, 2002-03-08.

clear

% Read name and path of input data file.

[name,path] = uigetfile('*.bin','Specify a binary input data file');

% Open specified datafile and read binary data.

fid = fopen([path,name],'rb');
data = fread(fid,'float32');
fclose(fid);

% Partition data into x and y variables.

f = data(1:2:length(data));
```

```
p = data(2:2:length(data));

% Call function m-file soundcalc.m for calculations

[fmax,Lpmax,Lptot,Lp] = soundcalc(f,p);
Lptot
fmax
Lpmax

% Plot the sound pressure level as function of frequency.

plot(f,Lp);
xlabel('Frequency, [Hz]');
ylabel('Sound pressure level, [dB]');
grid;

% Add info to plot

text1 = text(f(length(f))/3,0.9*Lpmax,...
    ['Lptot = ',num2str(Lptot),' dB']);
text2 = text(f(length(f))/3,0.75*Lpmax,...
    ['Lpmax = ',num2str(Lpmax),' dB at f = ',num2str(fmax),' Hz']);

% End of main program

function [xmax,ymax,ytot,Ly] = soundcalc(x,y);

% Function for calculating fmax, Lpmax, Lptot. Input data is y.

% Ulf Carlsson, 2003-03-08.

% Calculate sound pressure level as function of frequency.

y2 = y.*y;
Ly = 10*log10(y2/(2e-5)^2);

% Localise the maximum value and its frequency.

[ymax,linmax] = max(Ly);
xmax = x(linmax);

% Calculate the total sound pressure level.

y2sum = sum(y2);
ytot = 10*log10(y2sum/(2e-5)^2);

% End of function soundcalc
```

55

a)

```
function yband = band(fl,fu,f,y);

% Function m-file for energy summation of spectra.
% Input parameters:   fl - lower frequency band limit
%                     fu - upper frequency band limit
%                     f - current narrow-band frequency
%                     y - narrow band energy function
% Output: yband - total energy in band

% Ulf Carlsson, 2002-03-11.

% Determine limit lines for band within given limits.

fband = find(f > fl & f < fu);

% If no lines exist in band the sum is 0.

if isempty(fband)
   yband = 0;
   return;
end;

% Sum energy of lines inside band.

yband = sum(y(fband));

% Add half of the energy in lines within 1e-3 Hz from limits.

if abs(f(fband(1)-1)-fl) <= 1e-3
   yband = yband + y(fband(1)-1)/2;
end;    % Lower limit.
if abs(f(fband(length(fband))+1)-fu) <= 1e-3
   yband = yband + y(fband(length(fband))+1)/2;
end;    % Upper limit.

% End of function band
```

b)

```
function [throctf,throcty] = throct(f,y);

% Function m-file for summing narrowband spectra to third
% octave band spectra.
% Input parameters:   f - frequency vector
%                     y - energy vector
% Output parameters:  throctf - vector with center frequencies
%                     throcty - vector with band energies

% Ulf Carlsson, 2002-03-11.

% Define band limits.

limits = [1.12 1.41 1.78 2.24 2.82 3.55 4.47 5.62 7.08 ...
      8.91 11.2 14.1 17.8 22.4 28.2 35.5 44.7 56.2 70.8  ...
```

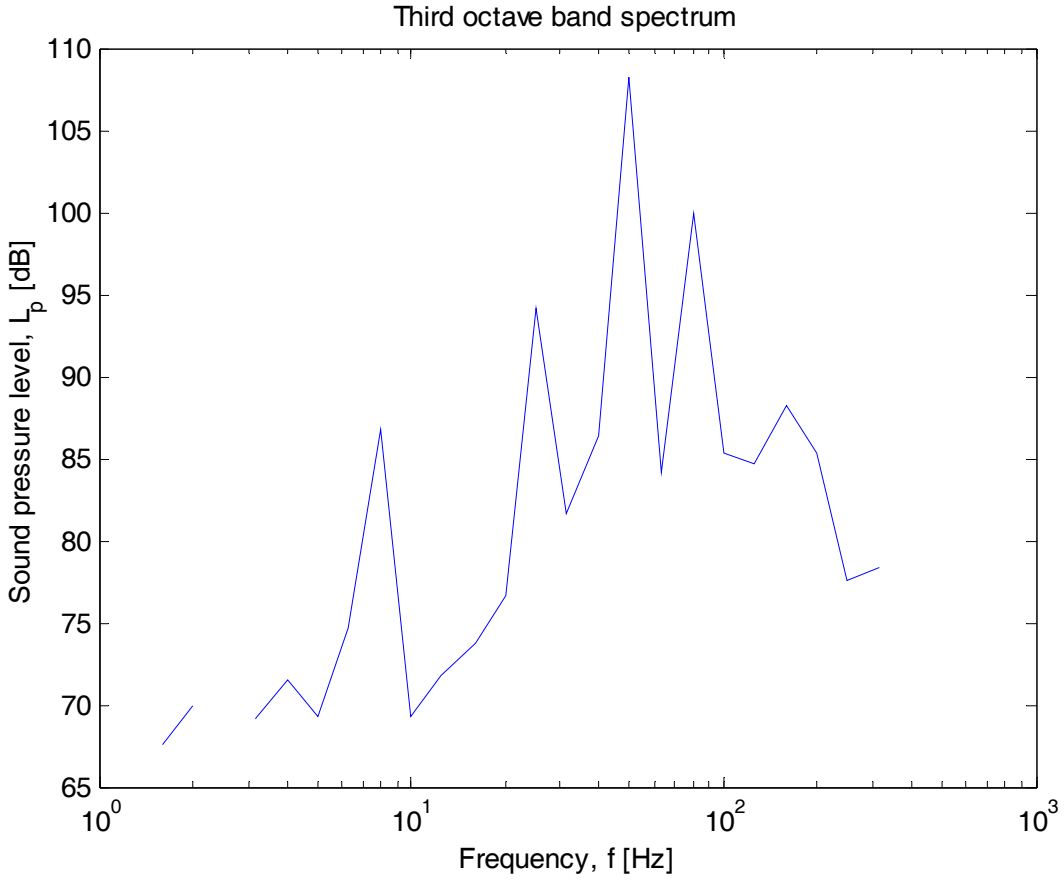
```
      89.1 112 141 178 224 282 355 447 562 708 891 1120 ...
      1410 1780 2240 2820 3550 4470 5620 7080 8910 11200 ...
      14100 17800 22400];
center = [1.25 1.6 2 2.5 3.15 4 5 6.3 8 10 12.5 16 20 25 ...
      31.5 40 50 63 80 100 125 160 200 250 315 400 500 630 ...
      800 1000 1250 1600 2000 2500 3150 4000 5000 6300 ...
      8000 10000 12500 16000 20000];

% Determine the first and last "full" band.

fmin = min(f);
fmax = max(f);
bandmin = min(find(limits > fmin));
bandmax = max(find(limits < fmax))-1;

% Loop through all bands.

for n = bandmin:bandmax
   throcty(n-bandmin+1) = band(limits(n),limits(n+1),f,y);
   throctf(n-bandmin+1) = center(n);
end;

% End of function throct
```

c) A code for testing the cods developed in a) and b) is given below.

```
% exercise55.m
% Program testing the codes developed in part a) and b).

% Ulf Carlsson, 2002-08-15.

clear;

% Read input sound pressure data from file.

[name,path] = uigetfile('*.bin',...
    'Specify a binary data input file with sound pressure data.');
fid = fopen([path,name],'rb');
data = fread(fid,'float32');
fclose(fid);

% Partition data

f = data(1:2:end);
y = data(2:2:end);

% Calculate the squared (power) data.

y2 = y.*y;

% Call function m-file throct

[F,Y] = throct(f,y2);

% Calculate and plot sound pressure level.
```

```
Lp = 10*log10(Y/(2e-5)^2);
semilogx(F,Lp);
title('Third octave band spectrum');
ylabel('Sound pressure level, L_p [dB]');
xlabel('Frequency, f [Hz]');

% End of code
```

The result when the code is used on the data in soundpress.bin is shown in the figure below.



56

a) The function M-file `rad2degr.m` converts angular values in radians to degrees.

```
function d = rad2degr(r)

% rad2degr.m
% Function that converts angular values from radians to
% degrees.
% Input parameter:   r - angle measured in radians
% Output parameter:  d - angle measure in degrees

% Ulf Carlsson. 2002-03-06.

d = 180*r/pi;

% End of code
```

b) The function M-file `atandeg.m` provides arcustangent function in degrees.

```
function v = atandeg(x)

% Arcustangent function that provides resulting angles in degrees
% Input parameter:   x - angle measured in degrees
% Output parameter:  v - value of atan(x)

% Ulf Carlsson. 2002-03-06.

y = atan(x);

% Call function rad2degr for conversion from radians to degrees

v = rad2degr(y);

% End of code
```

c) Try the programs on different input values:

```
>> atandeg(1)
ans =
    45
>> atandeg(1/0)
ans =
    90
>> atandeg(0.5)
ans =
   26.5651
```

57

a) The following MATLAB command assigns the string `accn.dat`, where `n` is the current numeric value of `n`, to a string variable `filename`:

```
filename = ['acc',num2str(n),'.dat']
```

b)

```
function filename = name(n);
% Ulf Carlsson, 2002-04-18.

filename = ['acc',num2str(n),'.dat'];

% End of function filename
```

Running the function with input value of n = 3 yields.

```
>> filename = name(3)
filename =
       acc3.dat
```

58

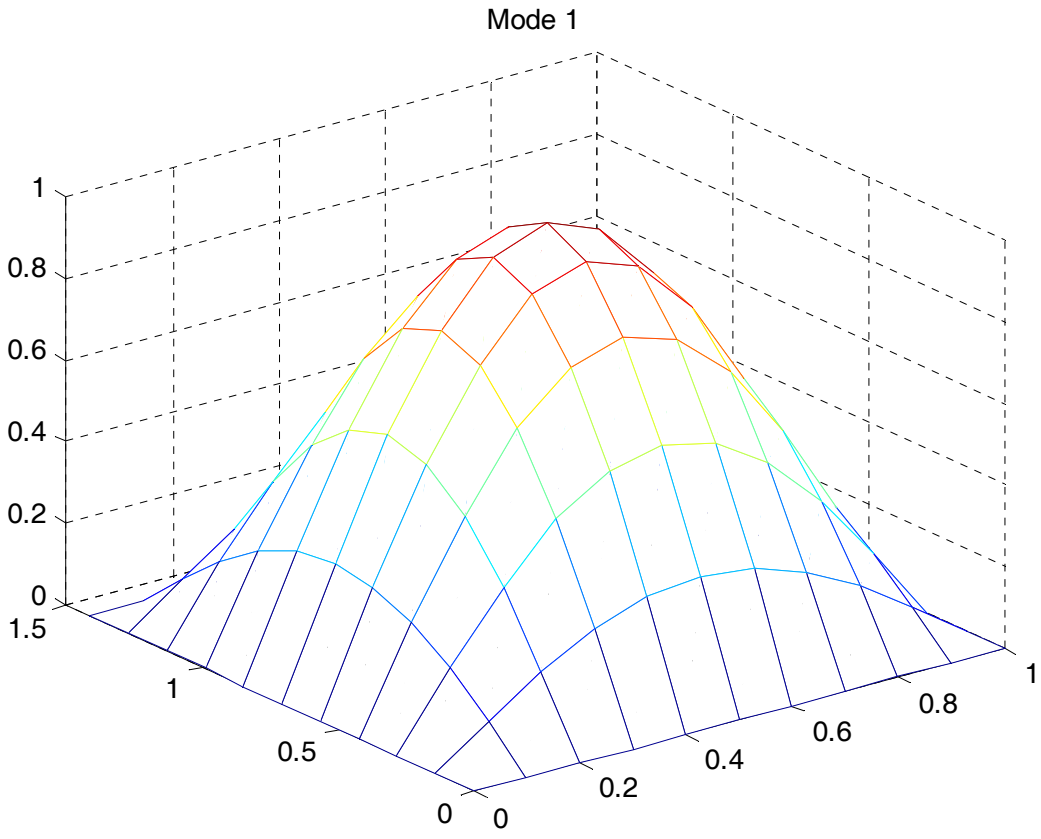
a) Use 11 mesh points in both, *x*- and *y*-directions.

```
>> x = (0:10)'*0.1;
>> y = (0:10)'*sqrt(2)/10;
```

Calculate the mode function *z* in the mesh defined by `x` and `y`. Finally plot the mode function in a 3-dimensional plot.

```
>> z = sin(pi*x)*sin(pi*y/sqrt(2))';
>> figure;
>> mesh(x,y,z); % contour(x,y,z) gives a contour plot
>> title('Mode 1');
```

The plot of the mode 1 is shown in the figure below.



b) and c) Modify the calculation of *z*.

59

a) One possible MATLAB implementation able to treat a function with any number of linear segments follows below:

```
function y = powerspec(x,X,Y);

% Evaluation of power density spectrum.
% Input data:   vector x
%               vector X - break points frequency
%               vector Y - break points power density
% Output data:  vector y

% Ulf Carlsson, 2002-08-19

global X Y;

% Find the lower interval limit for the minimum x value

m = min(find(X>min(x)))-1;

% Interpolate the y values

for n = 1:length(x)
    if (x(n)>=X(m) & x(n)<=X(m+1))
        y(n) = Y(m) + (Y(m+1)-Y(m))/(X(m+1)-X(m))*(x(n)-X(m));
    else
        m = m + 1;
        y(n) = Y(m) + (Y(m+1)-Y(m))/(X(m+1)-X(m))*(x(n)-X(m));
    end;
end

% End of code
```

b) A MATLAB code calling the function M-file `powerspec.m` follows below. The specific function specified in this case corresponds to the spectrum in Figure 2.

```
% powerintegr.m
% MATLAB code for integrating the energy below a
% power density spectrum. The power density function
% is assumed to be a series of break points connected
% with straight line elements.

% Ulf Carlsson, 2002-08-19

clear;
global X Y;

% Define break points which specifies the function.

X = [15;75;300;800;1000];   % Frequency values
Y = [5;5;23.4;23.4;8.4];    % Power values

% Plot function as check.

plot(X,Y);
```

```
title('Power Density Spectrum');
ylabel('Acceleration^2, [m^2/s^4/Hz]');
xlabel('Frequency, [Hz]');

% Use MATLABs integration (quadrature) function quad.
% Integrate between minimum frequency and maximum
% frequency.

W = quad(@powerspec,min(X),max(X))

% End of code
```

c) Executing the code developed in part a) and part b) gives:

```
>> powerintegr
W =
  1.8375e+004
```

60

a) The code given below calculates the displacement for a specified force:

```
% threerodab.m
% Calculation of the node displacement in a simple rod structure.
% Based on exercise suggested by Sören Östlund, Solid Mechanics, KTH

% 2002-11-13, Ulf Carlsson, MWL, KTH

% Initialise parameter data

clear;  % Clear workspace

% Young's moduli, cross-section areas and lengths:

E = [2e11;2e11;2e11];
A = [0.1e-2;0.1e-2;0.1e-2];
L = [11;10;14];

% Initialize force magnitude

Force = 1e4;

% Calculate column vector with direction angles for rods
% specified with respect to horizontal plane

alfa = asin(L(2)./L);

% Calculate the axial stiffness of the rods

k = E.*A./L;

% Initialise stiffness matrix coupling the node force with
% the node displacement.

k11 = k(1)*cos(alfa(3))^2+k(3)*sin(alfa(3))^2;
k12 = k(1)*cos(alfa(1))*cos(alfa(3))-k(3)*sin(alfa(3))*cos(alfa(3));
k22 = k(1)*cos(alfa(1))^2+k(2)+k(3)*cos(alfa(3))^2;
```

```
K = [k11 k12;k12 k22];

% Part b). Calculate the node displacement

% Horizontal force

P = [Force;0];
dispbi = K\P

% Vertical force

P = [0;Force];
dispbii = K\P
```

% End of program for part a-b)

b) Results obtained when executing the code developed in part a) are the following:

```
>> threerodab
dispbi =
  1.0e-003 *
    0.6218
    0.0380
dispbii =
  1.0e-003 *
    0.0380
    0.3340
```

c) A function m-file performing the specified tasks:

```
function displ = invK_p(gamma,Force,L,E,A)

% Function m-file that calculates the stiffness matrix
% for specified input data rod lengths, L, rod cross
% section areas, A, rod Young's moduli and finally for a
% specified force magnitude, Force, and force angle,
% gamma solves the system for the displacement, displ.

% Ulf Carlsson, MWL, KTH, 2002-11-21,

% Calculate column vector with direction angles for rods
% with respect to horizontal plane.

alfa = asin(L(2)./L);

% Calculate the axial stiffnesses of the rods

k = E.*A./L;   % column vector with axial stiffnesses for the rods

% Initialise stiffness matrix coupling the node force
% with the node displacement.

k11 = k(1)*cos(alfa(3))^2+k(3)*sin(alfa(3))^2;
k12 = k(1)*cos(alfa(1))*cos(alfa(3))-k(3)*sin(alfa(3))*cos(alfa(3));
k22 = k(1)*cos(alfa(1))^2+k(2)+k(3)*cos(alfa(3))^2;
K = [k11 k12;k12 k22];
```

```
% Calculate force vector and solve linear system of
% equations for node displacement.

P = Force*[cos(gamma);sin(gamma)];
displ = K\P;

% End of function displ
```

When the code is tested on the data given in part b) the results are:

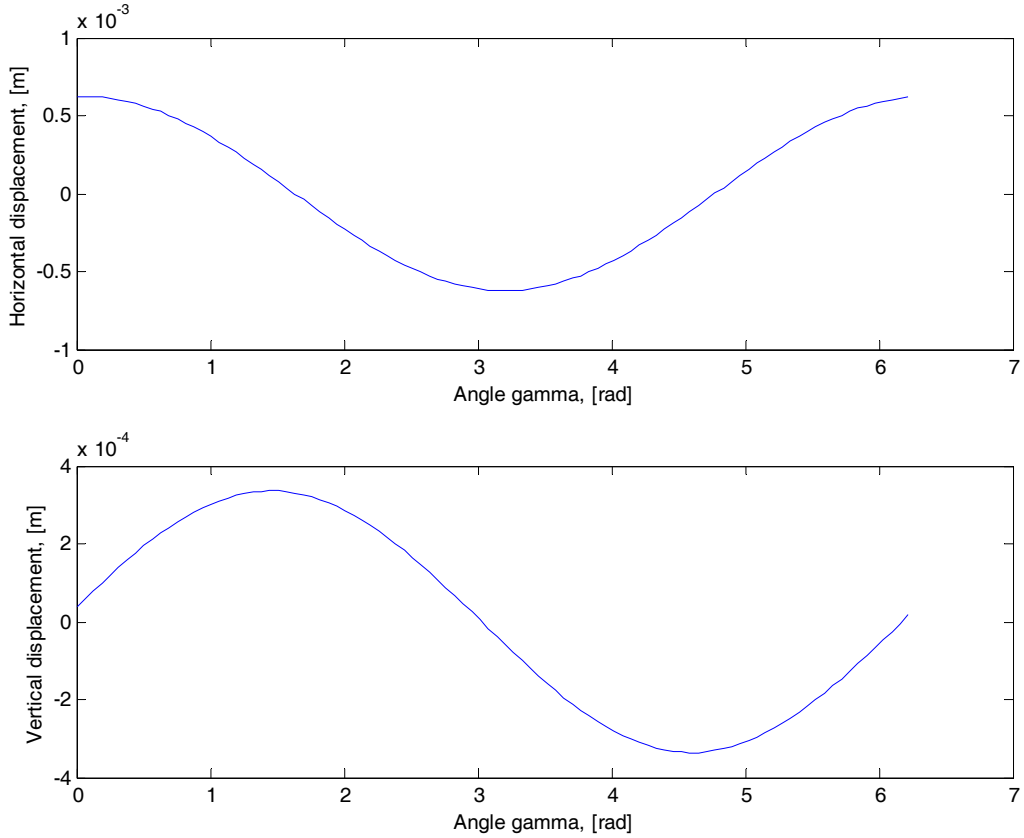
i)
```
>> gamma = 0;
>> disp = invK_p(gamma,Force,L,E,A)
disp =
  1.0e-003 *
    0.6218
    0.0380
```
ii)
```
>> gamma=pi/2;
>> disp=invK_p(gamma,Force,L,E,A)
disp =
  1.0e-003 *
    0.0380
    0.3340
```

d) A code for evaluating the node displacement as function of the force angle γ is:

```
% threerodd.m
% MATLAB code for calculating the displacement vector as
% function of the force angle gamma.

% 2002-11-21, Ulf Carlsson, MWL, KTH.

clear;

% Initialise parameter data,
% Young's moduli E, cross section areas A and lengths L.

E = [2e11;2e11;2e11];
A = [0.1e-2;0.1e-2;0.1e-2];
L = [11;10;14];

% Initialize force and force angle, gamma.

Force = 1e4;
gamma = [0:1/100:1-1/100]'*2*pi;

for n = 1:length(gamma)
    disp = invK_p(gamma(n),Force,L,E,A);
    dispx(n) = disp(1);  % Collect displacement components in
    dispy(n) = disp(2);  % separate vectors
end;

% Plot results
```

```
subplot(2,1,1)
plot(gamma,dispx)
xlabel('Angle gamma, [rad]');
ylabel('Horizontal displacement, [m]');

subplot(2,1,2)
plot(gamma,dispy)
xlabel('Angle gamma, [rad]');
ylabel('Vertical displacement, [m]');

% End of code
```

Execution of the code gives graphs showing the displacement components as functions of the force angle γ, see figure below. Obviously the displacement components are zero for two different force angles. The horizontal component is zero for angles equal to 1.63 and 4.77 radians. The vertical component is zero for 3.03 and 6.17 radians.



61

a) The following function M-file calculates the power transmitted over a single mount:

```
function W = powcalc(a,F,f);

% Function m-file calculating the power transmitted over a
% mount using measured force and acceleration data.
% Input data:
%      a - [peak phase(degrees)], measured acceleration at
%          frequency f, [m/s2]
%      F - [peak phase(degrees)], measured peak force at           %
%          frequency f, [N]
%      f - frequency, [Hz]
% Output data:
%      W - time average of transmitted mechanical power, [W]

% Ulf Carlsson, 2002-08-20

% Calculate complex force and acceleration

apeak = a(1);
fia = a(2);
Fpeak = F(1);
fiF = F(2);
acc = apeak*exp(i*fia*pi/180);
forc = Fpeak*exp(i*fiF*pi/180);

% Calculate complex velocity

v = acc/(i*2*pi*f);

% Calculate transmitted power via the coupling point.

W = real(forc*conj(v))/2

% End of code
```

b) The following code calls the code developed in part a) for the calculation of the power transmitted over N mounts.

```
% powertot.m

% MATLAB code for calculating and summing total mechanical
% power transmitted from a machine over N mounts.

% Ulf Carlsson, 2002-08-20

clear;

% Initialise data

Force = [13 0; 17 33; 8 70; 21 85];          % Amplitude and phase
acc = [1.4 192; 1.1 160; 2.1 212; 3.4 263];  % Amplitude and phase
freq = 90;
```

```
% Calculate power vector

for n = 1:length(acc)
    W(n) = powcalc(acc(n,:),Force(n,:),freq);
end;

% Calculate total transmitted power

Wtot = sum(W)
```

% End of code

c) Executing the code with input data according to table 1 gives,

```
>> powertot
W =
  -0.00334578419127
W =
   0.01320498309531
W =
   0.00914531959314
W =
   0.00220325620661
Wtot =
   0.02120777470379
```

62

a) The first step is to write a MATLAB function that generates random components of a unit surface normal vector.

```
function snorm = surfnorm

% Function m-file for generating random normal surface
% unit vectors.

% 2002-11-22, Ulf Carlsson, MWL, KTH

snorm = 1-2*rand(1,3)';                % Generate random components
snorm(:) = snorm(:)/norm(snorm(:));    % Normalise
```

% End of code

b) The second step is to write a MATLAB function that for a specified stress matrix calls the random normal vector generator and calculates the normal and shear stresses.

```
function [nstress,shear] = stress(S)

% Function m-file for calculating the normal stress and
% shear for a specified stress matrix S and a cut surface
% with normal n.

% 2002-11-22, Ulf Carlsson, MWL, KTH.
```

```
n = surfnorm;  % Generate random unit surface vector.
Sn = S*n;
nstress = n'*Sn;
shear = sqrt(Sn'*Sn-nstress^2);

% End of code
```

c) The next step is to write a MATLAB code that calculates and plots the relationship between the normal and shear stresses using a number *N* of, say 500, randomly oriented cuts.

```
% nrandstress.m
% S - Stress matrix [N/m2]
% N - Number of random cuts
% sigma - normal stress vector calculated for N random cuts
% tau - vector with shear values calculated for N random cuts
% MATLAB file that for a given stress matrix calculates and
% plots the normal stress and shear for N random cuts.

% 2002-11-25, Ulf Carlsson, MWL, KTH

clear;
% Initialise number of random cuts N and stress matrix S.
N = 500;
S = [10 2 0;2 10 1;0 1 10];
for n = 1:N
    [sigma(n),tau(n)] = stress(S);
end

% Plot normal stress as function of shear in graph with
% equal scaling and each point represented with a star '*'.
plot(sigma,tau,'*')
axis('equal')
xlabel('Normal stress, [MPa]');
ylabel('Shear stress, [MPa]');

% End of code
```

Testing the code on the matrix given produces the diagram given in part d) below, with exception for the three rings at zero shear stress.

d) The final step in developing the code is to include the calculation of the eigenvalues for the stress tensor.

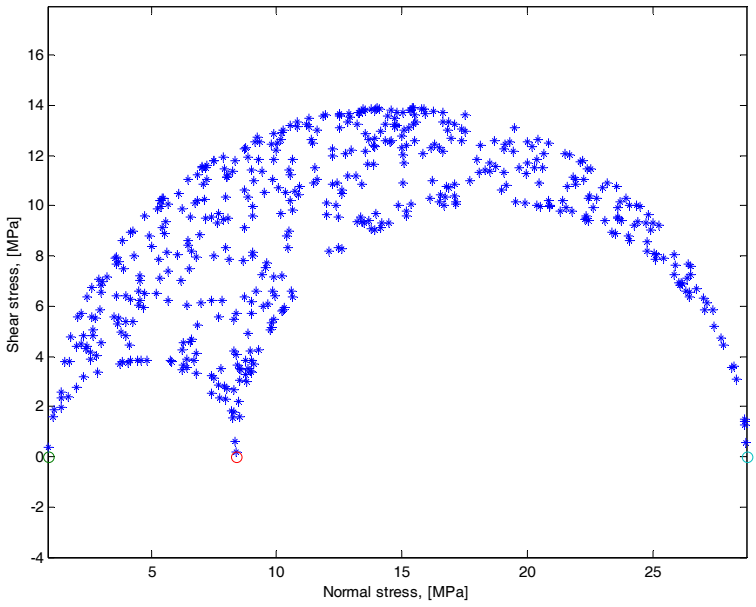
```
% nrandstress.m
% S - Stress matrix [N/m2]
% N - Number of random cuts
% sigma - normal stress vector calculated for N random cuts
% tau - vector with shear values calculated for N random cuts
% MATLAB file that for a given stress matrix calculates and
% plots the normal stress and shear for N random cuts.

% 2002-11-25, Ulf Carlsson, MWL, KTH
```

```
clear;
% Initialise number of random cuts N and stress matrix S.
N = 500;
S = [10 2 0;2 10 1;0 1 10];
for n = 1:N
    [sigma(n),tau(n)] = stress(S);
end

% Calculate the eigen-values (val) of the stress matrix
val = eig(S);

% Plot normal stress as function of shear in graph with
% equal scaling and each point represented with stars ('*').
% Add eigenvalues plotted along the normal stress axis.
% Represent the eigenvalues with rings ('o').

plot(sigma,tau,'*',val(1),0,'o',val(2),0,'o',val(3),0,'o')
axis('equal')
xlabel('Normal stress, [MPa]');
ylabel('Shear stress, [MPa]');

% End of code
```

The stress matrix eigenvalues are:

```
val =
    0.8782
    8.4067
   28.7151
```

Testing the code on the matrix given produce the diagram below.



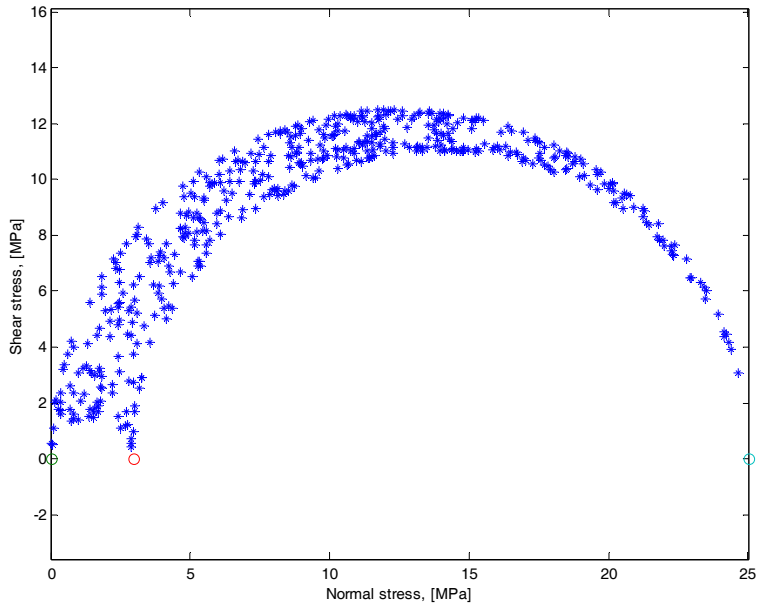
Try the code for the stress matrix representing an unloaded surface by letting,

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 8 \\ 0 & 1 & 25 \end{bmatrix} \text{ MPa.}$$

The stress matrix eigen-values are

```
val =
         0
    2.9546
   25.0454
```

The relation between the normal shear stresses is shown in the diagram below.



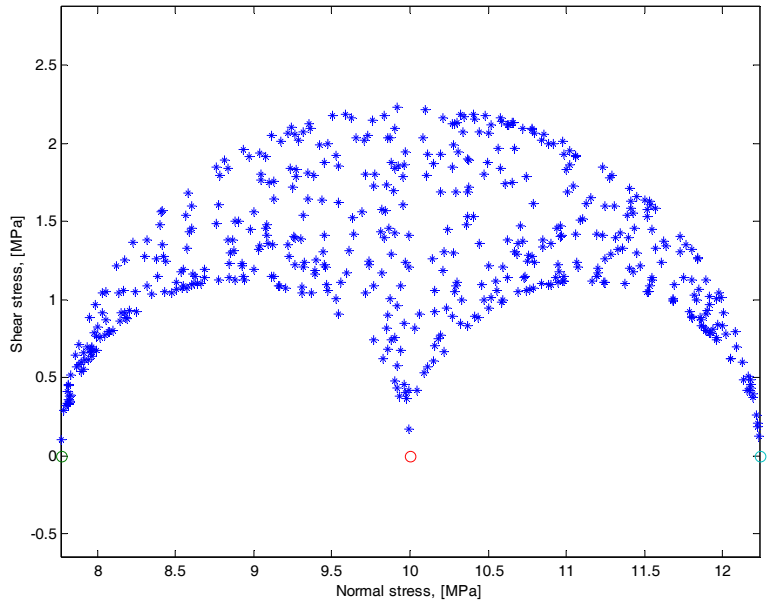
Finally try the code on the stress tensor given by,

$$\mathbf{S} = \begin{bmatrix} 10 & 2 & 0 \\ 2 & 10 & 1 \\ 0 & 1 & 10 \end{bmatrix} \text{ MPa.}$$

The stress matrix eigen-values are,

```
val =
    7.7639
   10.0000
   12.2361
```

The relationship between the normal and hear stresses is shown in the diagram below.



63

a) The code consist of two parts: the main program named `linmodel.m` and the function M-file `linreg.m`.

```
% linmodel.m
% m-file for least squares estimation of a linear relation
% y = c1*x + c2. Measured data (x,y) is provided in sequence
% x1, y1, x2, y2 ... on 'float32' format in binary file.

% Ulf Carlsson. 2002-03-07.

clear;

% Get name and path of input data file.

[name,path] = uigetfile('*.bin','Specify a binary data-file');

% Open specified datafile and read binary data.

fid = fopen([path,name],'rb');
data = fread(fid,'float32');
fclose(fid);

% Partition data into x and y variables.

x = data(1:2:length(data));
y = data(2:2:length(data));

% Call function m-file for model parameter estimation.

c = linreg(x,y)

% Plot characteristic model cuve together with measured data.
```

```
u = [min(x);max(x)];
v(1) = c(1)*u(1)+c(2);  % Calculate characteristic model
v(2) = c(1)*u(2)+c(2);
plot(u,v,x,y,'*');
xlabel('x-values');
ylabel('y-values');
legend('Linear model','Measured data');

% End of linmodel


function [c] = linreg(x,y)

% Function m-file for least squares estimation of linear model
% parameters on the form y = c1*x + c2. The calling program
% should provide input data in two vectors x and y.

% Ulf Carlsson. 2002-03-07.

% Assemble system matrix A.

A(:,1) = x;
A(:,2) = ones(length(x),1);

% Solve overdetermined system of equations.

c = A\y;

% End of linreg
```

b) The test data in file xydata.bin is generated from the linear model $y = 2x + 1$, i.e. c_1 and c_2 are 2 and 1 respectively.

```
>> linmodel
c =
    1.9815
    1.0283
```

64

This problem can be solved using the function `linreg.m` developed in exercise 63. Before the function is called the force and voltage values must be collected in two column vectors. Let `F` and `U` denote these two vectors.

```
>> F = [1 5 10 20 50]';
>> U = [0.08 0.11 0.14 0.28 0.55]';
>> [c] = linreg(F,V)
c =
    0.0098
    0.0632

>> y = c(1)*x+c(2);
>> plot(F,V,'*',x,y)
>> xlabel('Force, [N]')
>> ylabel('Voltage, [V]')
>> grid
>> legend('Measurement','Calculated');
```

Hence, the optimum linear model is,

$$U = 0.0098F + 0.0632 .$$

This function is plotted together with the measured data in the graph below.

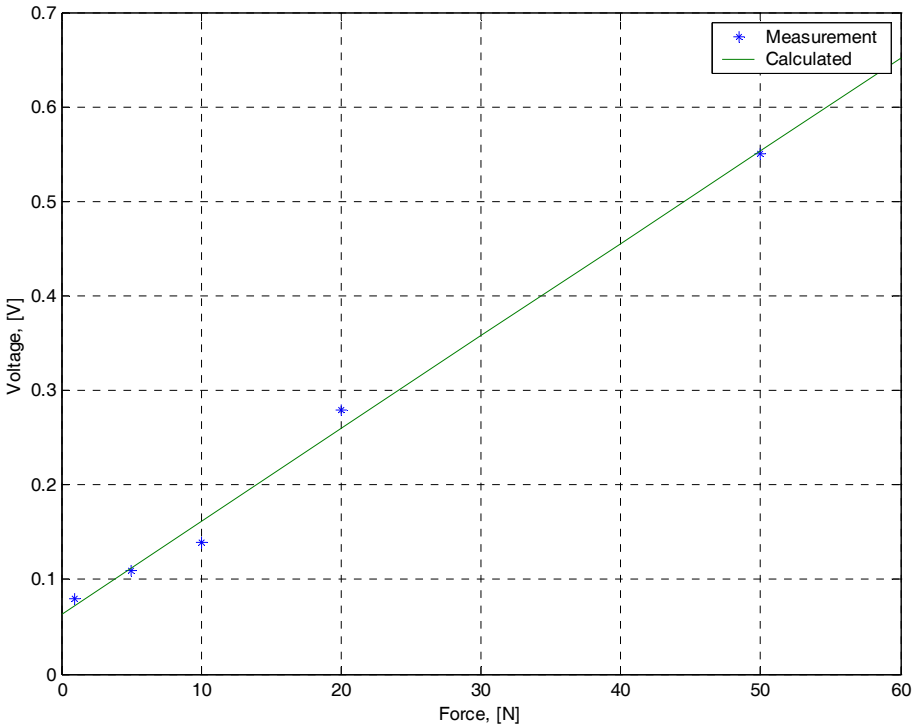


Figure Graph showing measured force and voltage together with calculated data according to optimum linear model.

65

a) Begin with opening the binary file `stiff.dat` for reading.

```
>> fid = fopen('stiff.dat','rb');
```

Read the contents of the file into the vector `data`.

```
>> data = fread(fid,'float32');
```

Close the input data file

```
>> fclose(fid);
```

b) Partition the data vector into three vectors: a frequency vector `f`, a force vector `F` and a displacement vector `x`.

```
>> f = data(1:3:end);
>> F = data(2:3:end);
>> x = data(3:3:end);
```

c) Calculate the dynamic stiffness vector `k`. And finally plot the stiffness in a lin-log diagram.

```
>> k = F./x;
>> semilogy(f,k)
>> xlabel('Frequency, [Hz]');
>> ylabel('Dynamic stiffness, [N/m]')
```

66

a)

```
function f = profile(x)

% Calculates road profile f(x) at position x.

% Ulf Carlsson, 2002-03-14.

global m k c V bumpsize;

% Function m-file that calculates the spatial derivative of
% the road profile. The bump is placed on interval [-0.5, 0.5].

if (x <= 0.5 & x >= -0.5)
    f = bumpsize*(cos(pi*x))^2;
else
    f = 0.001*(2*rand(1)-1);
end;

% End of function profile

function fprim = profileprim(x)

% Function m-file for road profile derivative.

% Ulf Carlsson, 2002-04-26.

global m k c V bumpsize;

% The bump is placed on interval [-0.5, 0.5].

fprim = 0;
if (x <= 0.5 & x >= -0.5)
   fprim = -bumpsize*pi*sin(2*pi*x);
end;

% End of function profileprim.
```

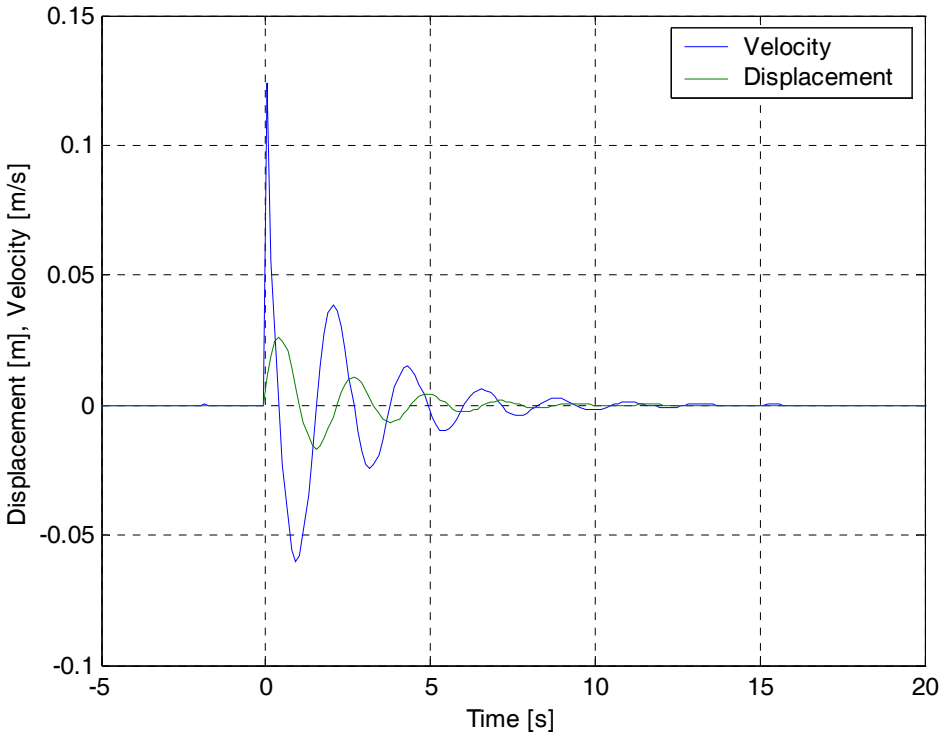
b)

```
function bump = funcbump(t,y)

% Function m-file for evaluating the right-hand side of
% governing equations for SDOF car running over bumpy road.
% The road profile is calculated in function profile.

% Ulf Carlsson, 2002-03-14.

global m k c V bumpsize;

% At time t the position is x = Vt.

A = [c/m k/m;-1 0];
bump = [k*profile(V*t)/m;0]+[c*V*profileprim(V*t)/m;0]-A*y;

% End of function funcbump
```

c and d)

```
% bumproad.m
% m-file for simulating the motion of a car running at speed V
% over a road bump with size bumpsize.

% Ulf Carlsson, 2002-03-14.

clear

% Define as global parameters that are used in function
% m-files.

global m k c V bumpsize;

% Initialize parameter values.

V = 50/3.6;
m = 250;
k = 2000;
c = 200;
bumpsize = 0.3;

ystart = [0;0];     % Initialiaze starting values.

% Call equation solver y'={b}+Ay. Right-hand side is
% calculated in function 'funcbump'. Solution is evaluated in
% 200 equidistant points at time interval [-5,20]. Solution is
% returned in displ. T is the time vector.

[T,displ] = ode45('funcbump',[linspace(-5,20,200)],ystart);

% Plot simulated displacement in a diagram.

plot(T,displ);
xlabel('Time [s]');
ylabel('Displacement [m], Velocity [m/s]');
grid
legend('Velocity','Displacement')

% End of m-file bumproad.m
```

d) The diagram below shows the vertical displacement and velocity as functions of time.



67

The following code contains solution to all parts of the exercise.

```
% triangle.m
% Code demonstrating how to use matrix
% operations to perform simple animated movies.

% Ulf Carlsson, 2002-08-14.

clear

% Part a).

% Create window with fixed corners for displaying
% the figures.

fig = figure;
axis([-10 10 -10 10]);

% Part b).

% Create two vectors one containing the x-coordinates
% the other one the y-coordinates. Note that the triangle
% is closed by finishing in the starting point.

x = [-5 5 0 -5];
y = [-5 -5 5 -5];
```

```
% Draw triangle in window using line 'command'.

line(x,y,'color','black');

% Part c).

% Translate the triangle in 2 units along the horizontal axis.
% Do this using matrix operations. First introduce the corner
% matrix triangle with x-coordinates in the first row and
% y-coordinates in the second row.

triang = [x;y];

% Then introduce a unit rigid body translation matrix filled
% with ones and having the same dimension as the corner
% matrix.

rigid = ones(size(triang));

% Then introduce a 2 by 2 diagonal point displacement matrix
% having the coordinate translations on the diagonal.

xdispl = 2;
ydispl = 0;
displace = [xdispl 0;0 ydispl];

% The triangle is translated by adding the matrix product
% between the displacement matrix and the rigid body
% translation matrix to the original triange matrix.

transltri = triang + displace*rigid;

% The triangle is translated by adding the matrix product
% between the translation matrix and the rigid body matrix
% translation to the original triange matrix.

transltri = triang + transl*rigid;

x = transltri(1,:);
y = transltri(2,:);
line(x,y,'color','blue');

% Part d).

% Zooming in on a figure can be accomplished by introducing a
% diagonal 2 by 2 zoom matrix with the zoom factor as diagonal
% elements.Isotropic zoom is obtained if the diagonal elements % are
equal.

xzoom = 1.2;    % Zoom factor 1.2 in x-direction
yzoom = 1.2;    % Zoom factor 1.2 in y-direction
zoom = diag([xzoom yzoom]);

% The zooming is performed by multiplying the corner matrix
% with the zoom matrix from the left.
```

```
zoomtria = zoom*triang;
x = zoomtria(1,:);
y = zoomtria(2,:);
line(x,y,'color','green');

% Part e).

% Rotating a figure an angle v in counter-clockwise direction
% can be performed by multiplying the corner matrix with a
% rotation matrix. Introduce the 2 by 2 rotation matrix
% rotate.

v = 15;         % Rotation angle [degrees]
fi = v*pi/180;  % Rotation angle [radians]
rotate = [cos(fi) sin(fi);-sin(fi) cos(fi)];

% Rotation in the opposite direction is obtained by
% transposing the rotation matrix.

% Rotate the triangle by multiplying the corner matrix with
% the rotation matrix.

rottri = rotate*triang;
x = rottri(1,:);
y = rottri(2,:);
line(x,y,'color','red');

% End of code
```

68

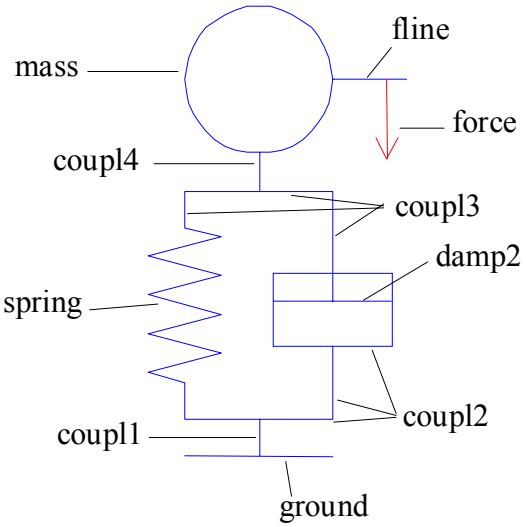
```
% triangmov.m

% Ulf Carlsson, 2002-03-12.

clear

% Prepare plot window with DoubleBuffer property set on.

fig = figure;
set(fig,'DoubleBuffer','on');

% Part a). Define triangle.

x = [-5 5 0 -5];
y = [-5 -5 5 -5];
triang = [x;y];

% Define rotation matrix B.

v = 1;
b = v/180*pi;
B = [cos(b) -sin(b);sin(b) cos(b)];

% Animate triangle rotation.
```

```
newtriang = triang;
N = 360/v;
for n = 1:N
    x = newtriang(1,:);
    y = newtriang(2,:);
    plot(x,y);                  %   Plot the triangle.
    axis([-10 10 -10 10]);      %   Use fixed axis.
    axis('square');             %   A square axis creates
                                %   an isotropic triangle.
    drawnow;                    %   Executes pending graphic
                                %   commands.
    newtriang = B*newtriang;    %   Rotate the triangle 1 degree.
end;
```

% End of triangmov.

69

The code below generates a plot representing the SDOF system according to the figure below.



% sdofanim.m

```
% Ulf Carlsson, 2002-04-18.

clear;

% Initialise parameters.

m = 1;
k = 1e4;
c = 1;
f = 15.8717;    % f(265) from exercise 48.
om = 2*pi*f;
F = 50;

% Generate displacement from results in exercise 48.

t = [0:0.001:1];
y0 = F/sqrt((k-om^2*m)^2+om^2*c^2);
```

```
phi = atan(om*c/(k-om^2*m));
y = y0*sin(om*t+phi);

% Create circle representing the mass.

s = [0:1:40]*2*pi/40;
xmass = cos(s);
ymass = sin(s)+5;

% Create ground at y = 0.

xground = [-1 1];
yground = [0 0];

% Create connection to ground.

xcoupl1 = [0 0];
ycoupl1 = [0 0.5];

% Create rigid (non-moving) part of damper and spring

xcoupl2 = [-1 -1 1 1];
ycoupl2 = [1 0.5 0.5 1.5];
xdamp1 = [0.2 0.2 1.8 1.8];
ydamp1 = [2.5 1.5 1.5 2.5];

% Create moving part of damper

xdamp2 = [0.2 1.8];
ydamp2 = [2 2];

% Create undeformed spring.

xs = [-0.5 0.5];
xspring = [0 xs xs xs xs 0]-1;
yspring = [0 1:2:15 16]/16;

% Create part of spring and damper connected to mass

ycoupl3 = [3 3.5 3.5 2];
ycoupl4 = [3.5 4];

% Create connecting line between force and mass

xfline = [1 2];
yfline = [5 5];

% Create force vector.

xforce = [1.75 1.75];
forcemagn = 1.5*sin(om*t);    % Magnitude force
xarrowhead = [1.6 1.75 1.9];
yarrowhead = [-0.3 0 -0.3];

% Create window for animation. Set DoubleBuffer = 'on' in order
% to prevent unsteady animations.
```

```
fh = figure;
set(fh,'DoubleBuffer','on');

% Animate.

for n = 1:length(t)
   yspringmov = 1+yspring*(2+y(n));
   plot(xground,yground,'b',xcoupl1,ycoupl1,'b',xcoupl2,...
       ycoupl2,'b',xdamp1,ydamp1,'b',xdamp2,ydamp2+y(n),...
       'b',xspring,yspringmov,'b',xcoupl2,ycoupl3+y(n),...
       'b',xcoupl1,ycoupl4+y(n),'b',xmass,ymass+y(n),...
       'b',xfline,yfline+y(n),'b',xforce,...
       [y(n) y(n)+forcemagn(n)]+5,'r',xarrowhead,...
       yarrowhead*sign(forcemagn(n))+5+forcemagn(n)+y(n)...
       ,'r');
   axis([-5 5 -2 8]);
   axis('square');
   axis off;
   drawnow;
end;
```

% End of code sdofanim.m

70

One possibility is to modify the solution to exercise 65. The vertical motion y now represents the cars vertical displacement. The horizontal motion is accounted for by adding a constant increment, representing constant speed, to all x-coordinates.

71

The following code contains one possible solution to the problem.

```
function sdofgui(action)

% Graphic User Interface: sdofgui.m.
% Program for calculating and plotting the frequency response %
function (FRF)
% for a mechanical single degree of freedom system.
% Model parameters: m - mass [kg]
%                   k - spring constant [N/m]
%                   c - damping constant [Ns/m].
% The FRF is evaluated at Nf equidistant frequency values on
% the interval [low,high].

% Ulf Carlsson, 2002-03-19.

global massField stiffField dampField m k c
global lowField highField nlinField low high Nf
global absAxes phaseAxes

if nargin < 1,
   action = 'initialisation';
end;
if strcmp(action,'initialisation'),
```

```
% Initialise the interface window

   ValFig = figure('Name','SDOF System GUI','NumberTitle',...
     'off','position',[20 60 750 480]);

% End of part a).

% Part b). Initialise close button and text

   closeBtn = uicontrol('style','pushbutton','string',...
     'Close','callback','close;','position',...
     [87 30 40 20],'foregroundcolor',[1 0 0]);

% End of part b).

% Part c). Create headers for model parameters, frequency
% parameters, calculation and finally diagrams.

   textHeader1 = uicontrol('style','text','position',...
     [50 450 118 13],'string','MODEL PARAMETERS:', ...
     'BackgroundColor',[0.8 0.8 0.8]);
   textHeader2 = uicontrol('style','text','position',...
     [50 330 110 13],'string','FREQUENCY RANGE:',...
     'BackgroundColor',[0.8 0.8 0.8]);
   textHeader3 = uicontrol('style','text','position',...
     [66 200 77 13],'string','CALCULATION:',...
     'BackgroundColor',[0.8 0.8 0.8]);
   textHeader4 = uicontrol('style','text','position',...
     [390 450 209 13],'string',...
     'DIAGRAMS: Frequency Response Function',
     'BackgroundColor',[0.8 0.8 0.8]);

% End of part c).

% Part d). Initialise numeric values for model and frequency
% parameters and place lead texts at suitable positions.

   m = 1;           % Initialise the mass
   k = 1;           % Initialise the spring constant
   c = 1;           % Initialise the damping constant
   low = 0;  % Initialise the lower frequency limit
   high = 100;      % Initialise the upper frequency limit
   Nf = 101; % Initialise the number of frequency lines
   textMass = uicontrol('style','text','position',...
     [10 423 66 13],'string','Mass m, [kg]: ', ...
     'BackgroundColor',[0.8 0.8 0.8]);
   textStiff = uicontrol('style','text','position',...
     [10 395 119 13],'string','Spring constant k, [N/m]: ' ...
     ,'BackgroundColor',[0.8 0.8 0.8]);
   textDamp = uicontrol('style','text','position',...
     [10 368 136 13],'string','Damping constant c, [Ns/m]:'...
     ,'BackgroundColor',[0.8 0.8 0.8]);
   textLow = uicontrol('style','text','position',...
     [10 303 127 13],'string',...
     'Lower frequency limit, [Hz]: ','BackgroundColor'...
     ,[0.8 0.8 0.8]);
```

```
   textHigh = uicontrol('style','text','position',...
     [10 276 127 13],'string',...
     'Upper frequency limit, [Hz]: ','BackgroundColor',...
     [0.8 0.8 0.8]);
   textNlin = uicontrol('style','text','position',...
     [10 249 129 13],'string',...
     'Number of frequency lines : ','BackgroundColor',...
     [0.8 0.8 0.8]);

% End of part d).

% Part e). Initialise fields for editable numeric values.

% Initialise editable mass field
   massField = uicontrol('style','edit','position',...
     [90 420 60 20],'string',num2str(m),'callback',...
     'sdofgui(''newmass'');');
% Initialise editable spring field
   stiffField = uicontrol('style','edit','position',...
     [145 393 60 20],'string',num2str(k),'callback',...
     'sdofgui(''newstiff'');');
% Initialise editable damper field
   dampField = uicontrol('style','edit','position',...
     [160 366 60 20],'string',num2str(c),'callback',...
     'sdofgui(''newdamp'');');
% Initialise frequency parameter fields
   lowField = uicontrol('style','edit','position',...
     [150 300 60 20],'string',num2str(low),'callback',...
     'sdofgui(''newlow'');');
   highField = uicontrol('style','edit','position',...
     [150 273 60 20],'string',num2str(high),'callback',...
     'sdofgui(''newhigh'');');
   nlinField = uicontrol('style','edit','position',...
     [150 246 60 20],'string',num2str(Nf),'callback',...
     'sdofgui(''newNf'');');

% End of part e). Part of solution to e) is located below in
% elseif strcmp(…) command part.

% Part f). Initialise calculation button and text.

   calcBtn = uicontrol('style','pushbutton','string',...
     'FRF','callback','sdofgui(''FRFcalc'');',...
     'position',[83 170 40 20]);

% End of part f). Part of solution to f) is located below.

% Part g). Initialise diagrams for graphic presentation.

   absAxes = axes('position',[0.4 0.58 0.5 0.3],'NextPlot',...
     'replace');
   phaseAxes = axes('position',[0.4 0.18 0.5 0.3],...
     'NextPlot','replace');

% End of part g). Part of solution to g) is located below.
```

```
% Part e) continued.

% Execute whenever editable mass field is hitted
elseif strcmp(action,'newmass'),
% Convert "mass" string to numeric value
   m = str2num(get(massField,'string'));
% Execute whenever editable spring field is hitted
elseif strcmp(action,'newstiff'),
% Convert "stiff" string to numeric value
   k = str2num(get(stiffField,'string'));
% Execute whenever editable damper field is hitted
elseif strcmp(action,'newdamp'),
% Convert "damp" string to numeric value
   c = str2num(get(dampField,'string'));
elseif strcmp(action,'newlow'),
   low = str2num(get(lowField,'string'));
elseif strcmp(action,'newhigh'),
   high = str2num(get(highField,'string'));
elseif strcmp(action,'newNf'),
   Nf = str2num(get(nlinField,'string'));

% End of continued part e).

% Part f) continued.

elseif strcmp(action,'FRFcalc'),
% Define equidistant frequency vector.
   f = linspace(low,high,Nf);
   om = 2*pi*f;           % Calculate om = 2*pi*f
   om2 = om.^2;
   FRF = 1./(k+i*om*c-om2*m);   % Calculate FRF.

% End of continued part f).

% Part g) continued.

% Make absAxes current axis for next plot.
   subplot(absAxes);
   plot(f,abs(FRF));
   xlabel('Frequency, [Hz]')
   ylabel('abs(FRF) x/F, [m/N]')
   grid
   zoom on
% Make phaseAxes current axis for next plot.
   subplot(phaseAxes);
   plot(f,180/pi*angle(FRF));
   xlabel('Frequency, [Hz]')
   ylabel('phase(FRF) x/F, [degrees]')
   grid
   zoom on
% End of continued part g).

end;

% End of code
```

Executing the code with parameter values according to the figure below gives the graphic user interface shown in the figure below.

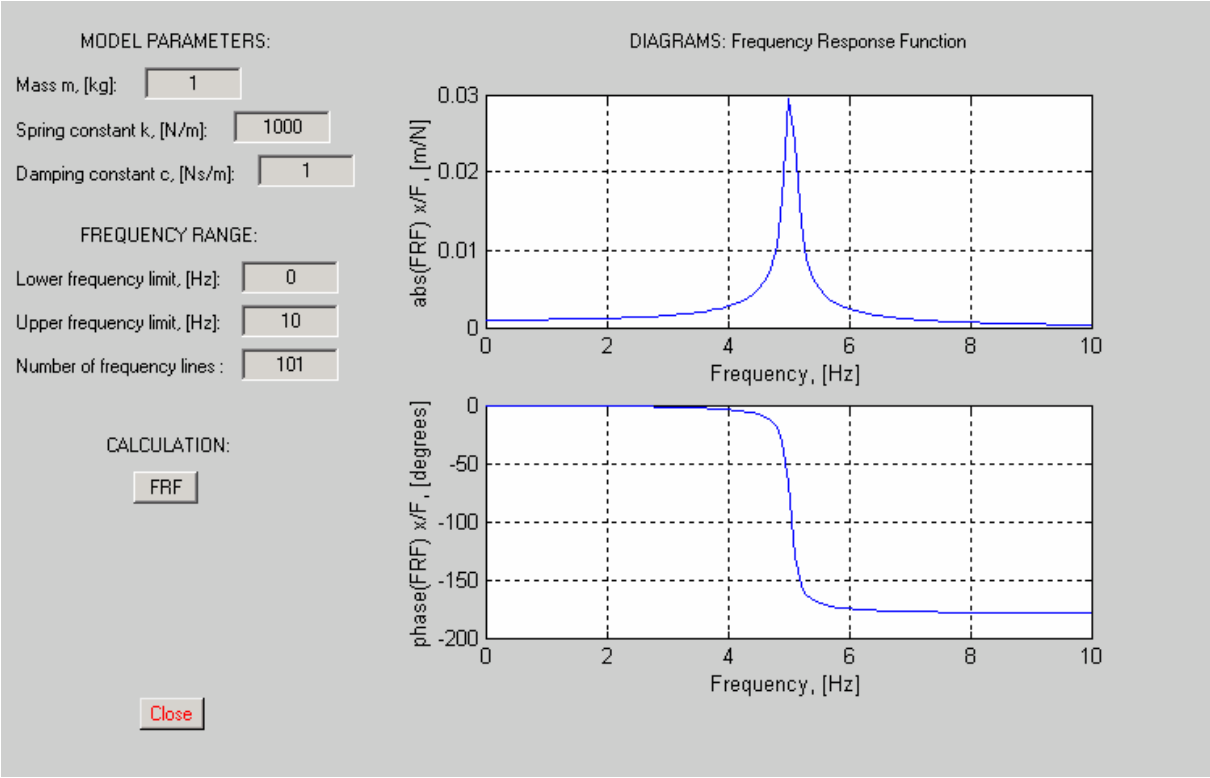


Figure Graphic user interface for calculation of mechanical single degree of freedom system frequency response functions.

72

Bold faced lines are additional lines for `Sound` button.

...

```
fileinpBtn = ...
uicontrol('Style','pushbutton','string','File','position',...
[5,395,40,20],'callback','[tdat,pdat,fsampl] = firstplot;');

% Create 'Sound' button.

fileinpBtn =
uicontrol('Style','pushbutton','string','Sound','position',...
[75,395,45,20],'callback','soundsc(pdat,fsampl);');

% Create exit pushbutton with red text.
```

...