Vibration Theory, Vol. 1B

linear vibration theory, MATLAB exercises

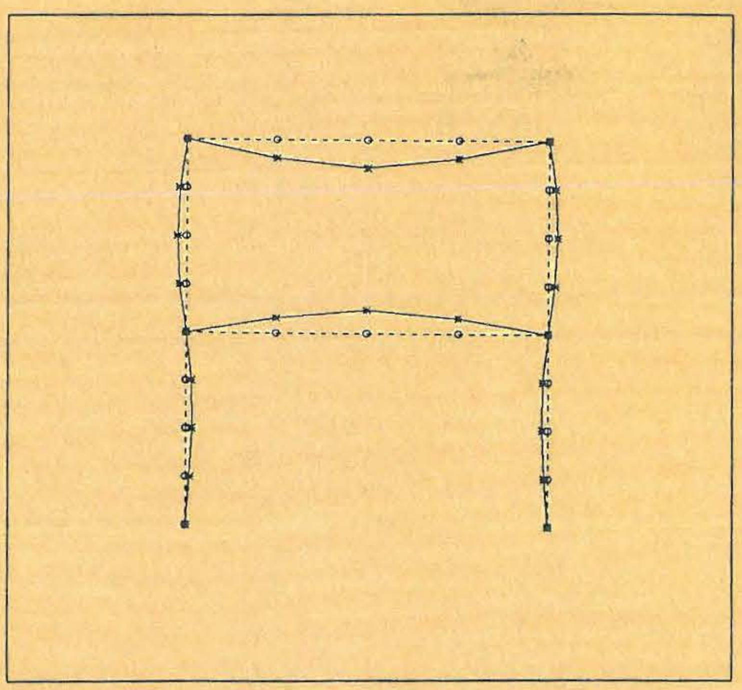
Asmussen, J. C.; Nielsen, Søren R. K.

Publication date:
1996

VIBRATION THEORY, VOL. 1B

Linear Vibration Theory
MATLAB Exercises

John C. Asmussen
Søren R.K. Nielsen

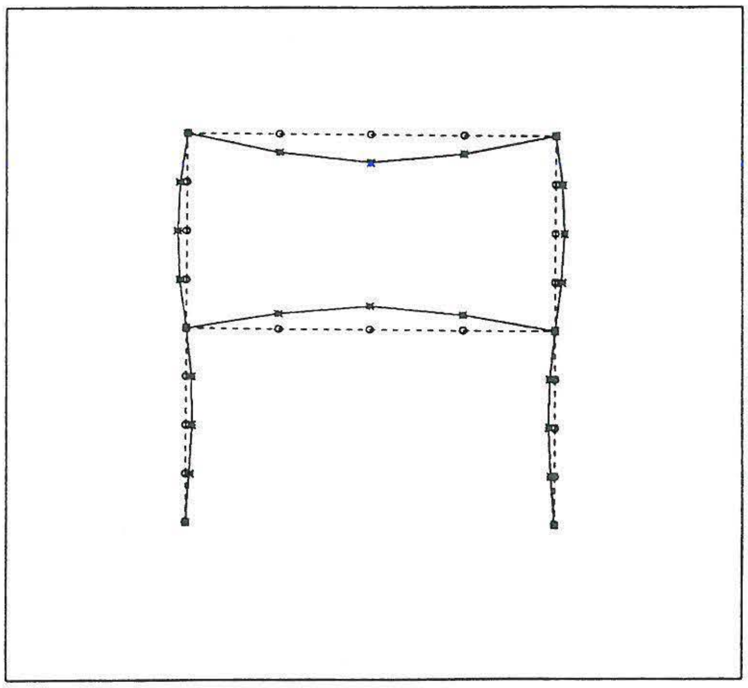


Aalborg tekniske Universitetsforlag
February, 1996

VIBRATION THEORY, VOL. 1B

Linear Vibration Theory
MATLAB Exercises

John C. Asmussen
Søren R.K. Nielsen



Aalborg tekniske Universitetsforlag
February, 1996

PREFACE

The present collection of MATLAB exercises has been published as a supplement to the text-book, *Svingningsteori, Bind 1* and the collection of exercises in *Vibration theory, Vol. 1A, Solved Problems.* Throughout the exercises references are made to these books.

The purpose of the MATLAB exercises is to give a better understanding of the physical problems in linear vibration theory and to suppress the mathematical analysis used to solve the problems. For this purpose the MATLAB environment is excellent.

It is possible to get help to every file by typing *help exerc??* or *help plot??*, where ?? denotes the number of the exercises. All m-files will be available on a discette.

University of Aalborg, February, 1996
Søren R.K. Nielsen
John C. Asmussen

Contents

Chapter 1

Lecture 1

Problem 1 and 4 of lecture 1 are solved theoretically. The solutions to these problems can be found in Nielsen [2]. The following MATLAB exercises replaces the corresponding problem 2 and 3.

1.1 MATLAB Exercise 1

The Fourier series of problem 1, lecture 1 is considered and the following truncated series is considered.

$$X_n(t) = \sum_{m=-n}^{n} A_m e^{i\omega_n t} \tag{1.1}$$

Plot the function $X_n(t)$ as a function of time for $t \in]-4\pi, 4\pi[$ and $n = 3, 10, 30$. Compare the different Fourier series with the sawtooth function graphically.

1.1.1 MATLAB Solution

The MATLAB functions below (exercl.m, plotl.m) shows how MATLAB exercise 1, lecture 1 could be solved. If the following four orders are given:

```
[x1,t]=exerc1(1000,3);
[x2,t]=exerc1(1000,10);
[x3,t]=exerc1(1000,30);
plot1(t,x1,x2,x3);
```

Then figure 1.1 will appear on the screen. An explanation of the input numbers can be found in the head of the MATLAB functions (exercl.m, plotl.m).

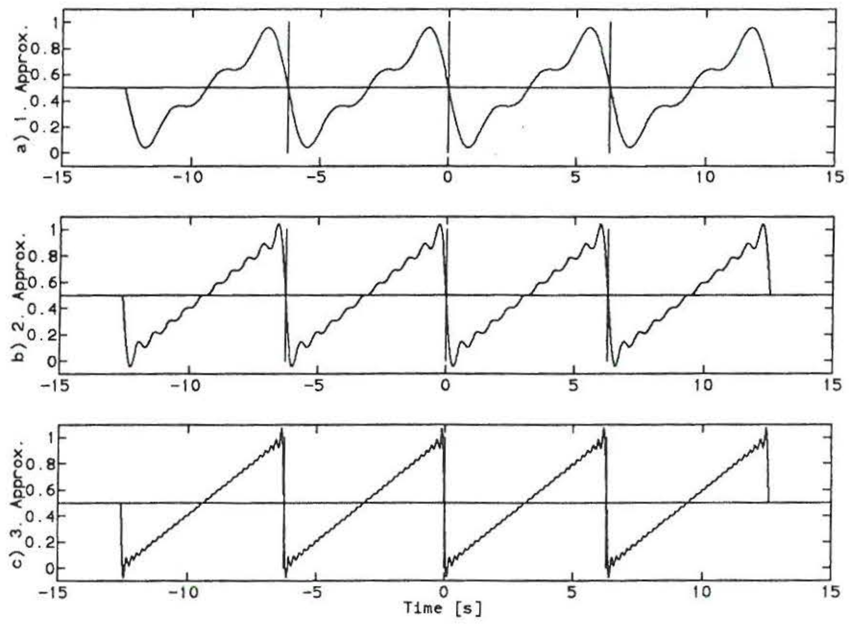


Figure 1.1: *Sawtooth approximations between* $]-4\pi, 4\pi[$*, as a function of the number of harmonics,* n. *a)* n=*3, b)* n=*10, c)* n=*30.*

1.1.2 MATLAB Files

```
*****************************************************************
  MATLAB      File for MATLAB Exercise 1, Lecture 1
  Function    EXERC1.M
  Purpose     To calculate the Fourier series from Exercise
              1, Lecture 1.
  Call        exerc1(n,no)
  n           Number of discrete time points between -4*pi and 4*pi.
  no          Number of summations in Fourier series
  Return      (Fx,t)
  Xno         Vector with sum of Fourier series
  t           Vector with time components
*****************************************************************
function [Xno,t]=exerc1(n,no);
dt=(4*pi+4*pi)/(n-1);
t=-4*pi:dt:4*pi;
for l=1:n,
Xno(l)=0;
for m=-no:no,
if (m==0) Am=0.5; else Am=i/(2*m*pi); end;
Xno(l)=Am*exp(i*m*t(l))+Xno(l);
end;
end;
*****************************************************************
```

```
*****************************************************************
  MATLAB     File for MATLAB Exercise 1, Lecture 1
  Function   PLOT1.M
  Purpose    To plot the sawtooth approximations from the
             Fourier series in Problem 1, Lecture 1
  Call       plot1(t,x1,x2,x3)
  t          Time vector
  x1         First sawtooth approximation
  x2         Second sawtooth approximation
  x3         Third sawtooth approximation
  Return     Graphical plot of approximations
*****************************************************************
function plot1(t,x1,x2,x3);
x1=real(x1); x2=real(x2); x3=real(x3);
tt(1)=-15; tt(2)=15; xx(1)=0.5; xx(2)=0.5;
tt1(1)=-2*pi; tt1(2)=-2*pi; xx1(1)=1; xx1(2)=0;
tt2(1)=0; tt2(2)=0; xx2(1)=1; xx2(2)=0;
tt3(1)=2*pi; tt3(2)=2*pi; xx3(1)=1; xx3(2)=0;
clg;
subplot(3,1,1),plot(t,x1,tt,xx,tt1,xx1,tt2,xx2,tt3,xx3);
ylabel('a) 1. Approx.');
axis([-15 15 -0.1 1.1]);
subplot(3,1,2),plot(t,x2,tt,xx,tt1,xx1,tt2,xx2,tt3,xx3);
ylabel('b) 2. Approx.');
axis([-15 15 -0.1 1.1]);
subplot(3,1,3),plot(t,x3,tt,xx,tt1,xx1,tt2,xx2,tt3,xx3);
ylabel('c) 3. Approx.');
xlabel('Time [s]');
axis([-15 15 -0.1 1.1]);
*****************************************************************
```

1.2 MATLAB Exercise 2

The system of problem 4, lecture 1 is considered. Let the initial conditions of the system be given as follows:

$$x_0 = \frac{a}{100} \quad , \qquad \dot{x}_0 = -2\frac{a}{100}\omega_0 \tag{1.2}$$

Plot the displacement response $x(t)$, and the velocity response, $\dot{x}(t)$, as a function of time in the interval $t \in [0, T]$. On the plot the displacement response is made non-dimensional with respect to x_0, and the velocity response is made non-dimensional with respect to $|\dot{x}_0|$. Use the following data:

$$l = 1m, \quad a = 0.3m, \quad S = 1kN, \quad m = 0.5kg,\ 2.0kg \tag{1.3}$$

Compare the responses with respect to the different eigenfrequencies graphically (the time periods should be equal). This should be done by comparing the two different time-displacement graphs and the two different time-velocity graphs.

1.2.1 MATLAB Solution

The MATLAB functions below (exerc2.m, plot2.m) shows how MATLAB exercise 2, lecture 1 could be solved. If the following three orders are given:

[x1,dx1,t]=exerc2(1,0.3,1,0.5,1000,0.01,0.003,-0.006);
[x2,dx2,t]=exerc2(1,0.3,1,2.0,1000,0.01,0.003,-0.006);
plot2(t,x1,x2,dx1,dx2);

Then figure 1.2 will appear on the screen. An explanation of the input numbers can be found in the head of the MATLAB functions (exerc2.m, plot2.m). Compare the two different time-displacement graphs and the two different time-velocity graphs.

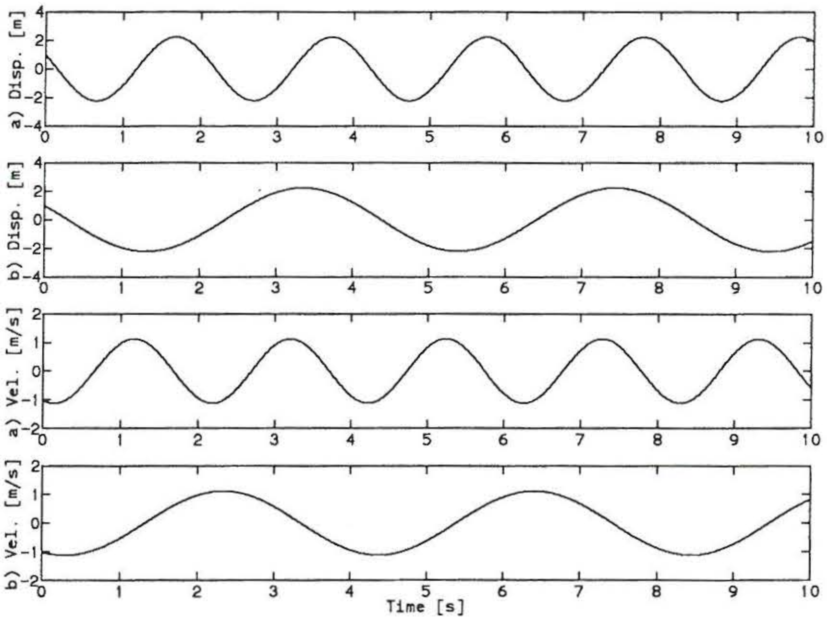


Figure 1.2: *Displacement and velocity responses for a)* m=*0.5kg and b)* m=*2.0kg.*

1.2.2 MATLAB Files

```
****************************************************************
 MATLAB     File for MATLAB Excercise 2, Lecture 1
 Function   EXERC2.M
 Purpose    To calculate the displacement response and the
            velocity response of the system described in
            Problem 4, Lecture 1
 Call       exerc2(l,a,S,m,n,dt,x0,dx0) (see fig. Problem 4)
 l          Length of stucture
 a          Distance to mass
 S          Tension Force
 m          Mass
 n          Number of points in discrete time
 dt         Time difference between points
 x0         Initial displacement
 dx0        Initial velocity/eigenfrequency
 Return     (x,dx,t)
 x          Displacement response vector
 dx         Velocity response vector
 t          Time vector
****************************************************************
function [x,dx,t]=exerc2(l,a,S,m,n,dt,x0,dx0)
om0=sqrt(l*S/(m*a*(l-a)));
dx0=dx0*om0;
for i=1:n,
t(i)=(i-1)*dt;
x(i)=x0*cos(om0*t(i))+dx0*sin(om0*t(i))/om0;
dx(i)=-x0*om0*sin(om0*t(i))+dx0*cos(om0*t(i));
```

```
end;
x=x/x0;
dx=dx/sqrt(dx0^2);
*****************************************************************

*****************************************************************
 MATLAB    File for MATLAB Exercise 2, Lecture 1
 Function  PLOT2.M
 Purpose   To plot the displacement response and the ve-
           locity response in Problem 4, Lecture 1
 Call      plot2(t,x1,x2,dx1,dx2)
 t         Time vector
 x1        Displacement (1. eigenfrequency)
 x2        Displacement (2. eigenfrequency)
 dx1       Velocity (1. eigenfrequency)
 dx2       Velocity (2. eigenfrequency)
 Return    Graphical plot
*****************************************************************
function plot2(t,x1,x2,dx1,dx2)
clg;
subplot(4,1,1),plot(t,x1);
ylabel('a) Disp. [m]');
subplot(4,1,2),plot(t,x2);
ylabel('b) Disp. [m]');
subplot(4,1,3),plot(t,dx1);
ylabel('a) Vel. [m/s]');
subplot(4,1,4),plot(t,dx2);
xlabel('Time [s]');
ylabel('b) Vel. [m/s]');
*****************************************************************
```

Chapter 2

Lecture 2

Problem 1 and 3 of lecture 2 are solved theoretically. The solutions of these problems can be found in Nielsen [2]. The following MATLAB exercise 3 replaces the corresponding problem 2, lecture 2.

2.1 MATLAB Exercise 3

The initial condition of the system in problem 3, lecture 2, is defined by the clock-wise rotation from the statical equilibrium state ϕ.

$$\phi_0 = \frac{\pi}{24} \quad , \quad \dot{\phi}_0 = \frac{\pi}{12}\omega_0 \tag{2.1}$$

Plot the vertical displacement $x(t)$ and the velocity $\dot{x}(t)$ of the beam at the position of the linear spring, k, as a function of the non-dimensional time $\frac{t}{T_0}$, where $T_0 = \frac{\omega_0}{2\pi}$ is the fundamental eigenperiod. Use the following data:

$$b = 1m, \quad a = \frac{2}{3}b, \quad k = 4\pi^2\frac{N}{m}, \quad m = 1kg, \quad c = 12\pi, 6\pi, 0.6\pi\frac{kg}{s} \tag{2.2}$$

For critical damping plot the vertical displacement $x(t)$ and the vertical velocity $\dot{x}(t)$ for a positive initial velocity, zero initial velocity and negative initial velocity.

2.1.1 MATLAB Solution

The MATLAB functions below (exerc3.m, plot3a.m) shows how MATLAB exercise 3, lecture 2 could be solved. An explanation of the input numbers can be found in the head of the MATLAB functions (exerc3.m, plot3a.m). If the following four orders are given:

```
[x1,dx1,t]=exerc3(1,2/3,4*pi^2,1,12*pi,1000,0.0025,pi/24,pi/12);
[x2,dx2,t]=exerc3(1,2/3,4*pi^2,1,6*pi,1000,0.0025,pi/24,pi/12);
[x3,dx3,t]=exerc3(1,2/3,4*pi^2,1,0.6*pi,1000,0.0025,pi/24,pi/12);
plot3a(t,x1,x2,x3,dx1,dx2,dx3);
```

Then figure 2.1 and figure 2.2 will appear on the screen. Notice that "RETURN" should be pressed to shift from figure 2.1 to figure 2.2. Compare the responses and the magnitude of the damping coefficient.

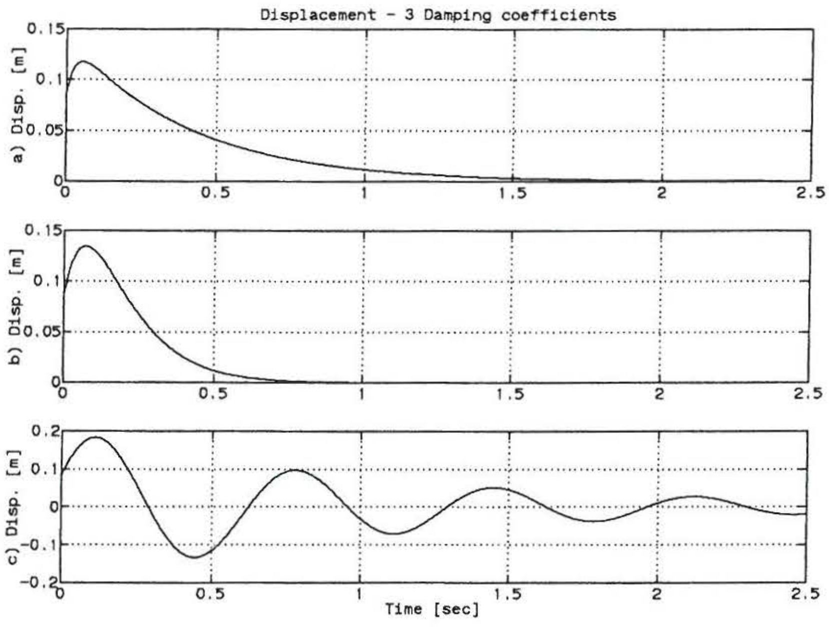


Figure 2.1: *Displacement responses from different damping coefficients. a) $c = 12\pi\frac{kg}{s}$, b) $c = 6\pi\frac{kg}{s}$, c) $c = 0.6\pi\frac{kg}{s}$.*

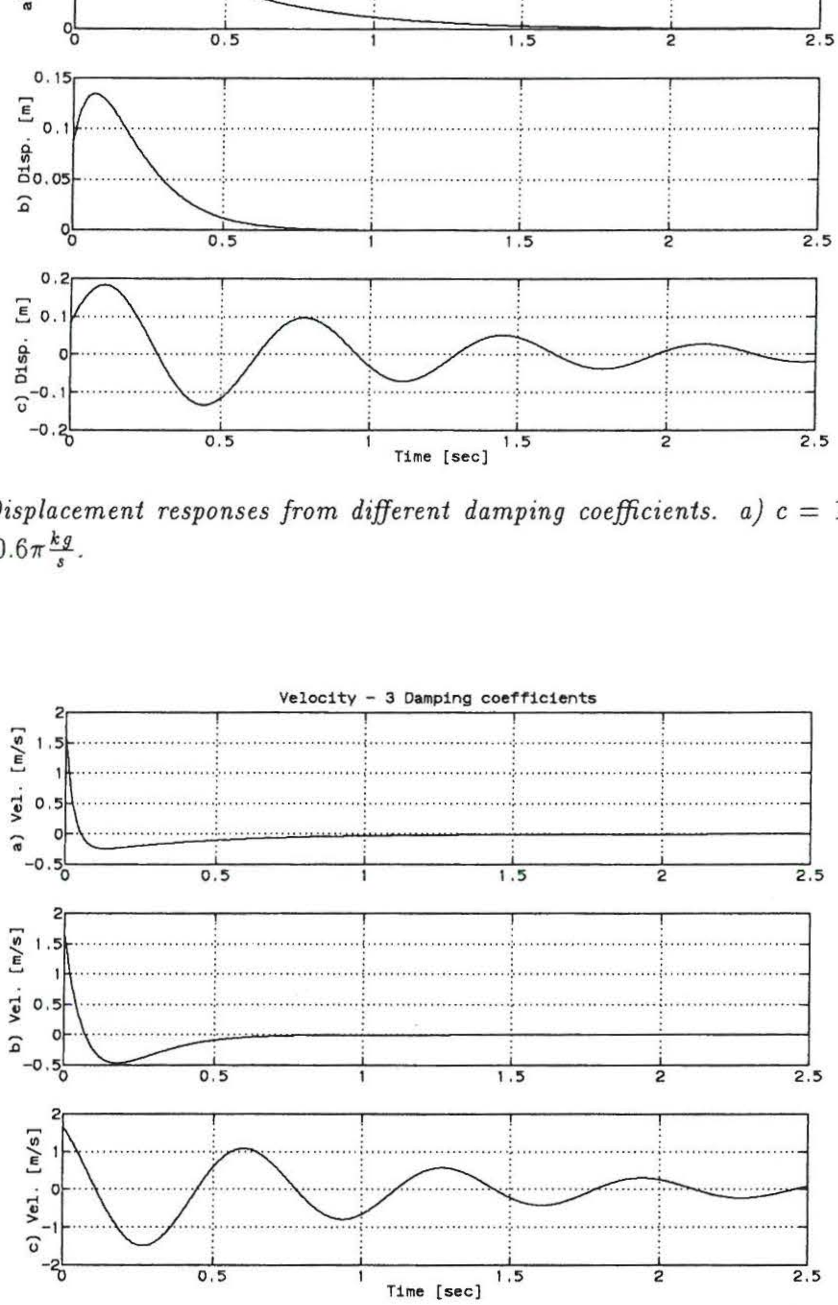


Figure 2.2: *Velocity responses from different damping coefficients. a) $c = 12\pi\frac{kg}{s}$, b) $c = 6\pi\frac{kg}{s}$, c) $c = 0.6\pi\frac{kg}{s}$.*

The displacement and velocity responses of the critical damped system with different initial velocities are plotted by the following four orders:

```
[x1,dx1,t]=exerc3(1,2/3,4*pi^2,1,6*pi,1000,0.0025,pi/24,pi/12);
[x2,dx2,t]=exerc3(1,2/3,4*pi^2,1,6*pi,1000,0.0025,pi/24,0);
[x3,dx3,t]=exerc3(1,2/3,4*pi^2,1,6*pi,1000,0.0025,pi/24,-pi/12);
plot3b(t,x1,x2,x3,dx1,dx2,dx3);
```

Figure 2.3 and figure 2.4 will appear on the screen. Notice that "RETURN" should be pressed to shift from figure 2.3 to figure 2.4. Compare the responses and the different initial velocities.

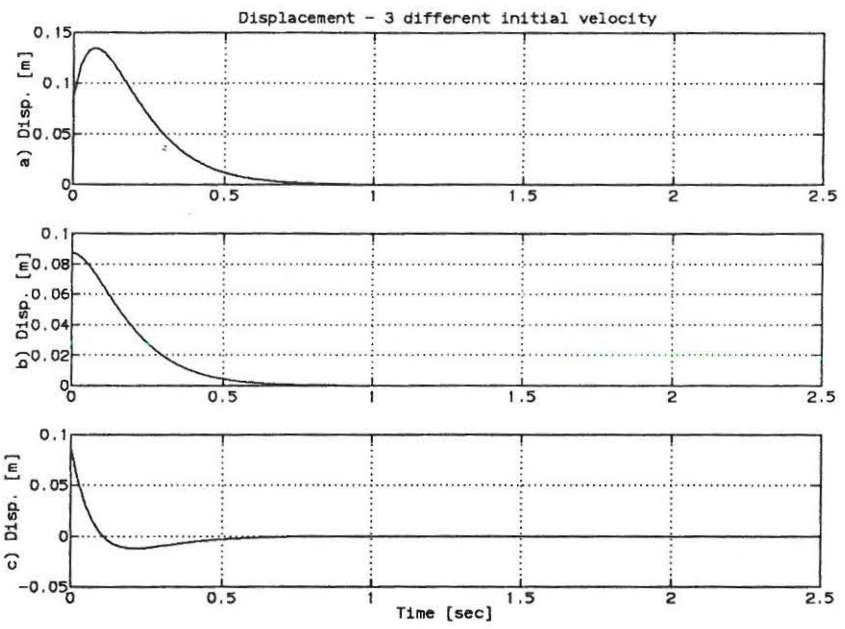


Figure 2.3: *Displacement responses for critical damping with three different initial velocities. a)* $\dot{\phi}_0 = \frac{\pi}{12}s^{-1}$, *b)* $\dot{\phi}_0 = 0s^{-1}$, *c)* $\dot{\phi}_0 = -\frac{\pi}{12}s^{-1}$. *The initial rotation is* $\frac{\pi}{24}$.

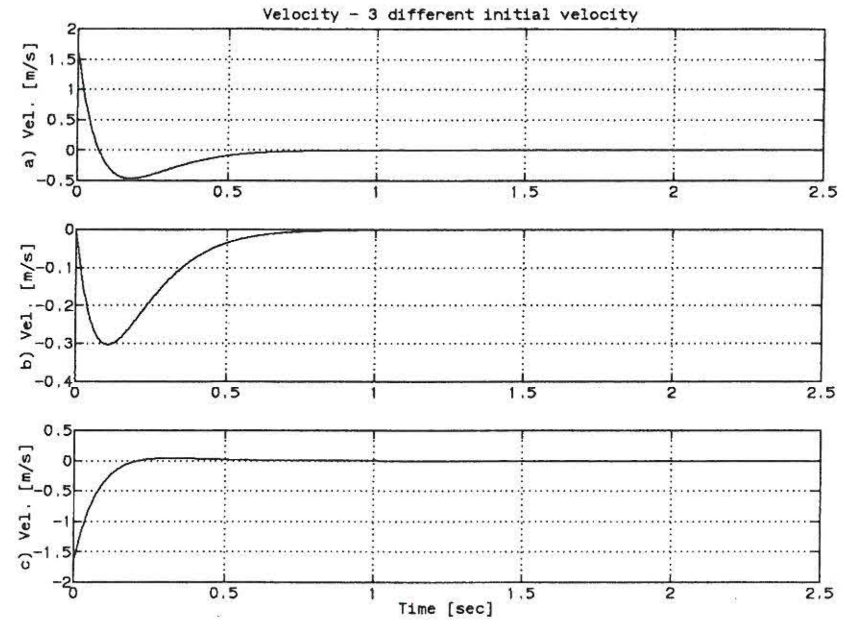


Figure 2.4: *Velocity responses for critical damping with three different initial velocities. a)* $\dot{\phi}_0 = \frac{\pi}{12}s^{-1}$, *b)* $\dot{\phi}_0 = 0s^{-1}$, *c)* $\dot{\phi}_0 = -\frac{\pi}{12}s^{-1}$. *The initial rotation is* $\frac{\pi}{24}$.

2.1.2 MATLAB Files

```
*****************************************************************
```

MATLAB	File for MATLAB Exercise 3, Lecture 2
Function	EXERC3.M
Purpose	To calculate the displacement and velocity response of the system in the spring point of the system in problem 3, lecture 2.
Call	exerc3(b,a,k,m,c,n,dt,x0,dx0) (see fig. Prob. 3)
b	Length of stucture
a	Distance to mass and damping
k	Stifness of linear spring
m	Mass
c	Damping constant
n	Number of points in discrete time
dt	Time difference between points
x0	Initial displacement
dx0	Initial velocity/eigenfrequency
Return	(x,dx,t)
x	Displacement response vector
dx	Velocity response vector
t	Time vector

```
*****************************************************************
function [x,dx,t]=exerc3(b,a,k,m,c,n,dt,x0,dx0)

om0=(b/a)*sqrt(k/m);
z=c/(2*sqrt(k*m));
x0=x0*a;
dx0=dx0*om0*a;

if (z>1),
 A =( dx0+(z+sqrt(z^2-1))*om0*x0)/(2*om0*sqrt(z^2-1));
 B =(-dx0-(z-sqrt(z^2-1))*om0*x0)/(2*om0*sqrt(z^2-1));
 C1=(-z+sqrt(z^2-1))*om0;
 C2=(-z-sqrt(z^2-1))*om0;
 end;

 if (z<1),
 C1=om0*sqrt(1-z^2);
 C2=(dx0+z*om0*x0)/C1;
 end;

 for i=1:n,
 t(i)=(i-1)*dt;

 if (z<1),
 x(i)=exp(-z*om0*t(i))*(x0*cos(C1*t(i))+C2*sin(C1*t(i)));
 dx(i)=-z*om0*x(i)+exp(-z*om0*t(i))*(-x0*C1*sin(C1*t(i))+C1*C2*cos(C1*t(i)));
 end;

 if (z==1),
 x(i)=exp(-om0*t(i))*(x0+(dx0+om0*x0)*t(i));
 dx(i)=-om0*x(i)+exp(-om0*t(i))*(dx0+om0*x0);
 end;

 if (z>1),
 x(i)=A*exp(C1*t(i))+B*exp(C2*t(i));
 dx(i)=C1*A*exp(C1*t(i))+C2*B*exp(C2*t(i));
 end;
 end;
 ****************************************************************


 ****************************************************************
   MATLAB     File for MATLAB Exercise 3, Lecture 2
   Function   PLOT3A.M
   Purpose    To plot the displacement response and the ve-
              locity response in Problem 3, Lecture 2
   Call       plot3a(t,x1,x2,x3,dx1,dx2,dx3)
   t          Time vector
   x1         Displacement (1. Damping coefficient)
   x2         Displacement (2. Damping coefficient)
   x3         Displacement (3. Damping coefficient)
   dx1        Velocity (1. Damping coefficient)
   dx2        Velocity (2. Damping coefficient)
   dx3        Velocity (3. Damping coefficient)
   Return     Graphical plots
 ****************************************************************
function plot3a(t,x1,x2,x3,dx1,dx2,dx3)
clg;
subplot(3,1,1),plot(t,x1);
ylabel('Disp. [m]');
title('Displacement - 3 Damping coefficients')
subplot(3,1,2),plot(t,x2);
ylabel('Disp. [m]');
subplot(3,1,3),plot(t,x3);
ylabel('Disp. [m]');
xlabel('Time [sec]')
pause
clg;
subplot(3,1,1),plot(t,dx1);
ylabel('Vel. [m/s]');
title('Velocity - 3 Damping coefficients')
subplot(3,1,2),plot(t,dx2);
ylabel('Vel. [m/s]');
subplot(3,1,3),plot(t,dx3);
```

```
ylabel('Vel. [m/s]');
xlabel('Time [sec]')
******************************************************************
```

```
******************************************************************
 MATLAB     File for MATLAB Excercise 3, Lecture 2
 Function   PLOT3B.M
 Purpose    To plot the displacement response and the ve-
            locity response in Problem 3, Lecture 2
 Call       plot3b(t,x1,x2,x3,dx1,dx2,dx3)
 t          Time vector
 x1         Displacement (1. Damping coefficient)
 x2         Displacement (2. Damping coefficient)
 x3         Displacement (3. Damping coefficient)
 dx1        Velocity (1. Damping coefficient)
 dx2        Velocity (2. Damping coefficient)
 dx3        Velocity (3. Damping coefficient)
 Return     Graphical plot
******************************************************************
function plot3b(t,x1,x2,x3,dx1,dx2,dx3)
clg;
subplot(3,1,1),plot(t,x1);
ylabel('a) Disp. [m]');
title('Displacement - 3 different initial velocity')
grid;
subplot(3,1,2),plot(t,x2);
ylabel('b) Disp. [m]');
grid;
subplot(3,1,3),plot(t,x3);
ylabel('c) Disp. [m]');
xlabel('Time [sec]');
grid;
pause
clg;
subplot(3,1,1),plot(t,dx1);
ylabel('a) Vel. [m/s]');
title('Velocity - 3 different initial velocity')
grid;
subplot(3,1,2),plot(t,dx2);
ylabel('b) Vel. [m/s]');
grid;
subplot(3,1,3),plot(t,dx3);
ylabel('c) Vel. [m/s]');
xlabel('Time [sec]');
grid;
******************************************************************
```

Chapter 3

Lecture 3

Problem 1,3 and 4 of lecture 3 are solved theoretically. The solutions to these problems can be found in Nielsen [2]. The following MATLAB exercise replaces the corresponding problem 2 of lecture 3.

3.1 MATLAB Exercise 4

Verify that the equation of motion of a linear viscous single degree-of-freedom system can be written on the following *state vector form*:

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ -\frac{k}{m}x(t) - \frac{c}{m}\dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \frac{3}{4} f_0 cos(\omega t) \tag{3.1}$$

Where the initial conditions are:

$$x(0) = x_0 \ , \quad \dot{x}(0) = \dot{x}_0 \tag{3.2}$$

Assume the initial values of the displacement and velocity of problem 1, lecture 3 are: $x_0 = \dot{x}_0 = 0$.

Verify upon insertion in the equation of motion and the initial conditions that the solution becomes:

$$x(t) = |X| \left[cos(\omega t - \Psi) - e^{-\zeta\omega_0 t} \left(cos(\Psi)cos(\omega_d t) + sin(\omega_d t)(\frac{\zeta cos(\Psi)}{\sqrt{1-\zeta^2}} + \frac{\omega}{\omega_d} sin(\Psi)) \right) \right] \tag{3.3}$$

Where $|X|$, Ψ are the amplitude and phase angle of the stationary harmonic motion, ω_0, ω_d are the undamped and damped circular eigenfrequency, ζ is the damping ratio determined in problem 1, lecture 3 and ω is the frequency of the force. The velocity response is:

$$\begin{aligned} \dot{x}(t) = |X|[\quad & -\omega sin(\omega t - \Psi) + \\ & \zeta\omega_0 e^{-\zeta\omega_0 t}(cos(\Psi)cos(\omega_d t) + sin(\omega_d t)(\tfrac{\zeta cos(\Psi)}{\sqrt{1-\zeta^2}} + \tfrac{\omega}{\omega_d} sin(\Psi))) - \\ & e^{-\zeta\omega_0 t}(-\omega_d cos(\Psi) sin(\omega_d t) + \omega_d cos(\omega_d t)(\tfrac{\zeta cos(\Psi)}{\sqrt{1-\zeta^2}} + \tfrac{\omega}{\omega_d} sin(\Psi))] \end{aligned} \tag{3.4}$$

Solve (3.1) numerically using a 4th order Runge-Kutta scheme and compare the solution with the analytical solution (3.3). Use the following data:

$$\omega = \omega_0 = 2\pi Hz \ , \quad k = 6\pi^2 \frac{N}{m} \ , \quad m = 1kg \ , \quad a = 1m \ , \quad c = 0.04\pi \frac{kg}{s} \ , \quad f_0 = \frac{0.32\pi^2 N}{3} \tag{3.5}$$

Plot the force versus the displacement and velocity response of the system for different frequencies of the force, ω.

3.1.1 MATLAB Solution

The MATLAB functions below (exerc4a.m, plot4a.m) shows how MATLAB exercise 4, lecture 3 could be solved. If the following three orders are given:

```
[t1,x1,dx1,tt,xt,dxt]= exerc4a(16*pi^2,0.04*pi,1,0.32*pi^2/3,2*pi,40,0.25);
[t2,x2,dx2,tt,xt,dxt]= exerc4a(16*pi^2,0.04*pi,1,0.32*pi^2/3,2*pi,100,0.1);
[t3,x3,dx3,tt,xt,dxt]= exerc4a(16*pi^2,0.04*pi,1,0.32*pi^2/3,2*pi,200,0.05);
plot4a(t1,t2,t2,x1,x2,x3,dx1,dx2,dx3,tt,xt,dxt);
```

Then figure 3.1 and figure 3.2 will appear on the screen. Notice that "RETURN" should be pressed to shift from figure 3.1 to figure 3.2. An explanation of the input numbers can be found in the top of the MATLAB functions (exerc4a.m, plot4a.m). Notice that the numerical approximation converges as the time intervals decreases.

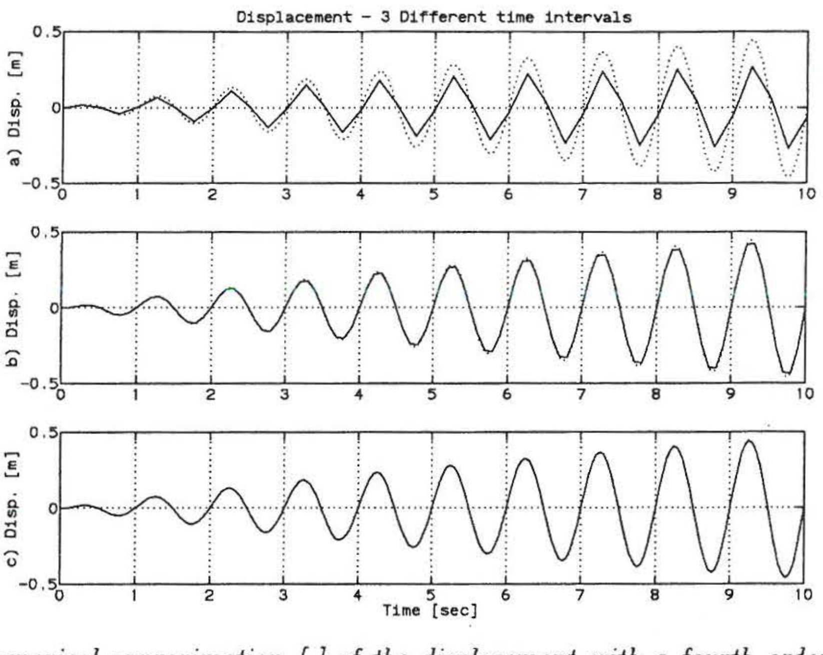


Figure 3.1: *Numerical approximation [-] of the displacement with a fourth order Runge-Kutta scheme and theoretical solution [..]. a)* $\Delta t=\frac{T_0}{4}$*, b)* $\Delta t=\frac{T_0}{10}$*, c)* $\Delta t=\frac{T_0}{20}$.

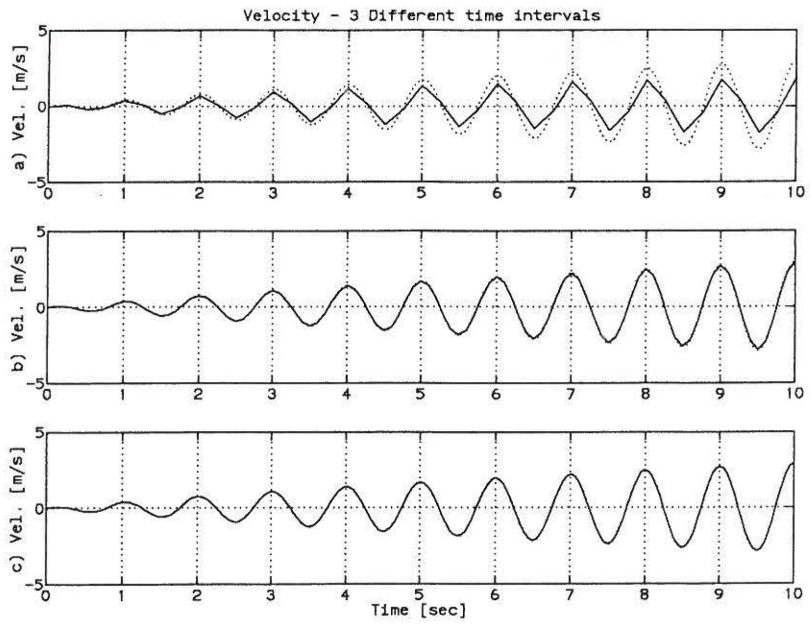


Figure 3.2: *Numerical approximation [-] of the velocity with a fourth order Runge-Kutta scheme and theoretical solution [..]. a)* $\Delta t = \frac{T_0}{4}$, *b)* $\Delta t = \frac{T_0}{10}$, *c)* $\Delta t = \frac{T_0}{20}$.

To plot the force versus the displacement and velocity response with different frequencies of the force, perform the following 2 examples.

Example 1: $\omega = 2.05\omega_0$. If the MATLAB files exerc4b.m and plot4b.m where implemented, the following two orders should be given:

```
[t,xt,dxt,Ft]=exerc4b(16*pi^2,0.04*pi,1,0.32*pi^2/3,2.05*pi,400,0.05);
plot4b(t,xt,dxt,Ft);
```

Then the following graph will appear on the screen:

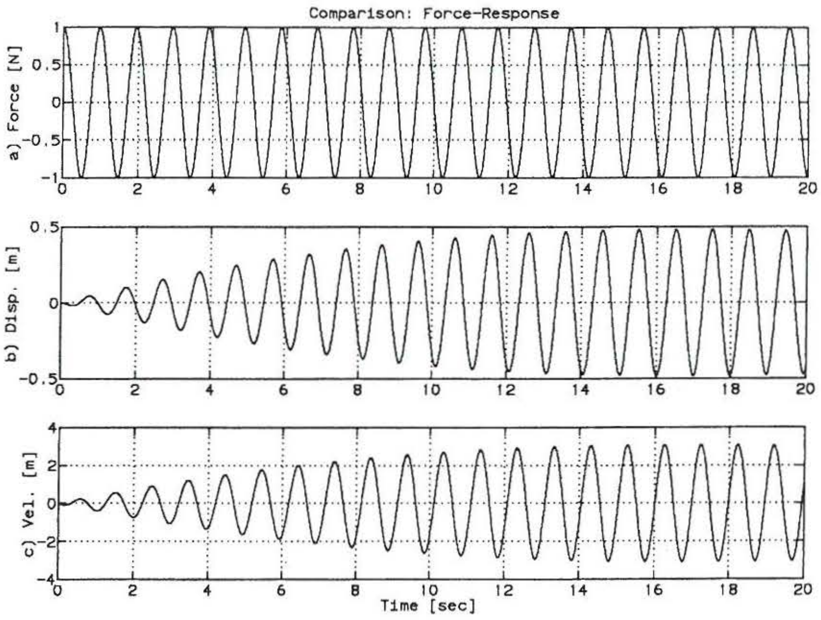


Figure 3.3: *Response of system subjected to a harmonic force where $\omega_0 < \omega$. a) Harmonic excitation process, b) displacement response c) velocity response.*

Example 2: $\omega = 2.05\pi$. If the MATLAB files exerc4c.m and plot4b.m where implemented, the following two orders should be given:

*[t,xt,dxt,Ft]=exerc4c(16*pi^2,0.04*pi,1,0.32*pi^2/3,1.95*pi,400,0.05);*
plot4b(t,xt,dxt,Ft);

Then the following graph will appear on the screen:

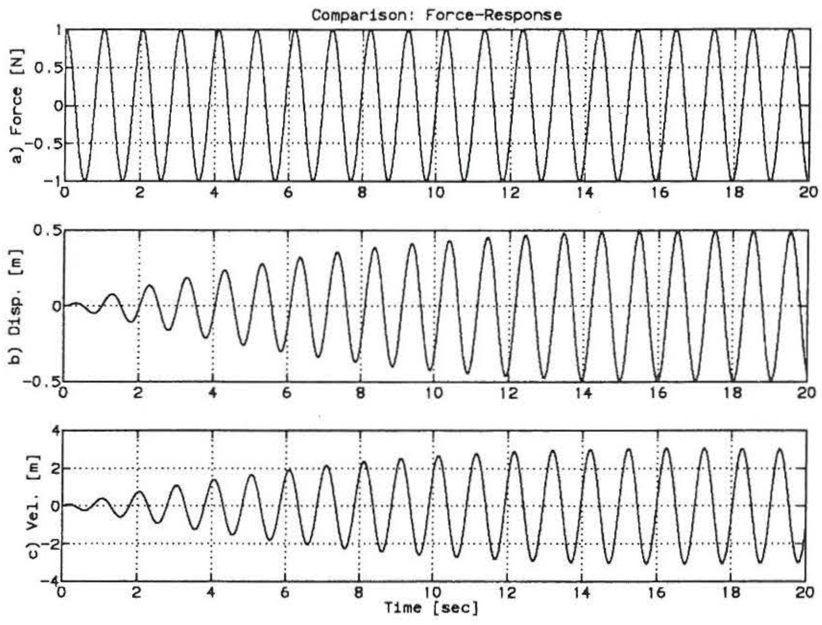


Figure 3.4: *Response of system subjected to a harmonic force where $\omega_0 > \omega$. a) Harmonic excitation process, b) displacement response c) velocity response.*

3.1.2 MATLAB Files

```
*****************************************************************
 MATLAB     File for MATLAB Exercise 4, Lecture 3
 Function   EXERC4A.M
 Purpose    To calculate the response of the system de-
            scribed in problem 1, lecture 3. A Runge-Kutta
            fourth order method is used.
 Call       exerc4a(k,c,m,f0,omf,n,dt) (fig.Prob. 1)
 k          Stifness of linear spring
 c          Damping constant
 m          Mass
 f0         Amplitude of force
 omf        Eigenfrequency of harmonic force
 n          Number of points in approximtation
 dt         Time distance between points
 Return     (x,dx,t)
 x1         Displacement response vector
 x2         Velocity response vector
 xt         Theoretical solution
 t          Corresponding time vector
*****************************************************************
function [t,x1,x2,tt,xt,dxt]=exerc4a(k,c,m,f0,omf,n,dt)

om0=0.5*sqrt(k/m);
z=c/sqrt(m*k);

constants used in differential equation
C1=om0^2;
C2=2*z*om0;
C3=(0.75*f0)/m;
for l=1:n+1, x1(l)=0; x2(l)=0; t(l)=0; end;

x1(1)=0; Initial conditions - zero displacement
x2(1)=0; Initial conditions - zero velocity
t(1)=0;

for l=1:n,
k1=0.5*dt*(-C1*x1(l)-C2*x2(l)+C3*cos(omf*t(l)));
K=0.5*dt*(x2(l)+0.5*k1);
k2=0.5*dt*(-C1*(x1(l)+K)-C2*(x2(l)+k1)+C3*cos(omf*(t(l)+0.5*dt)));
k3=0.5*dt*(-C1*(x1(l)+K)-C2*(x2(l)+k2)+C3*cos(omf*(t(l)+0.5*dt)));
LL=dt*(x2(l)+k3);
k4=0.5*dt*(-C1*(x1(l)+LL)-C2*(x2(l)+2*k3)+C3*cos(omf*(t(l)+dt)));
x1(l+1)=x1(l)+dt*(x2(l)+(1/3)*(k1+k2+k3));
x2(l+1)=x2(l)+(1/3)*(k1+2*k2+2*k3+k4);
t(l+1)=t(l)+dt;
end;
theoretical solution
if (om0==omf) psi=pi/2;
else psi=atan((2*z*om0*omf)/(om0^2-omf^2));
end;
X=(0.75*f0)/(m*sqrt((om0^2-omf^2)^2+4*(z^2)*(om0^2)*(omf^2)));
omd=om0*sqrt(1-z^2);

laengde=n*dt;
dt=laengde/2000;
n=2000;

for l=1:n+1,
tt(l)=(l-1)*dt;
Displacement response
xh1=cos(omf*tt(l)-psi);
xh2=cos(psi)*cos(omd*tt(l));
xh3=sin(omd*tt(l))*((z*cos(psi))/(sqrt(1-z^2))+(omf/omd)*sin(psi));
```

```
xt(l)=X*(xh1-exp(-z*om0*tt(l))*(xh2+xh3));
Velocity response
dxh1=-omf*sin(omf*tt(l)-psi);
dxh2=xh2;
dxh3=xh3;
dxh4=-omd*cos(psi)*sin(omd*tt(l));
dxh5=omd*cos(omd*tt(l))*((z*cos(psi))/(sqrt(1-z^2))+(omf/omd)*sin(psi));
dxt(l)=X*(dxh1+z*om0*exp(-z*om0*tt(l))*(dxh2+dxh3)-exp(-z*om0*tt(l))*(dxh4+dxh5));
end;
*****************************************************************
```

```
*****************************************************************
  MATLAB      File for MATLAB Exercise 4, Lecture 3
  Function    PLOT4A.M
  Purpose     To plot the displacement response and the ve-
              locity response in Problem 1, Lecture 4
  Call        plot4a(t1,t2,t3,x1,x2,x3,dx1,dx2,dx3,tt,xt,dxt)
  t1          Time vector
  t2          Time vector
  t3          Time vector
  x1          Displacement (1. Damping coefficient)
  x2          Displacement (2. Damping coefficient)
  x3          Displacement (3. Damping coefficient)
  dx1         Velocity (1. Damping coefficient)
  dx2         Velocity (2. Damping coefficient)
  dx3         Velocity (3. Damping coefficient)
  tt          Time vector
  xt          Theoretical Displacement
  dxt         Theoretical Velocity
  Return      Graphical plot
*****************************************************************
function plot4a(t1,t2,t3,x1,x2,x3,dx1,dx2,dx3,tt,xt,dxt)
clg;
subplot(3,1,1),plot(t1,x1,tt,xt,':');
ylabel('a) Disp. [m]');
title('Displacement - 3 Different time intervals')
grid;
axis([0 10 -0.5 0.5]);
subplot(3,1,2),plot(t2,x2,tt,xt,':');
ylabel('b) Disp. [m]');
grid;
axis([0 10 -0.5 0.5]);
subplot(3,1,3),plot(t3,x3,tt,xt,':');
ylabel('c) Disp. [m]');
grid;
axis([0 10 -0.5 0.5]);
pause
clg;
subplot(3,1,1),plot(t1,dx1,tt,dxt,':');
ylabel('a) Vel. [m/s]');
title('Velocity - 3 Different time intervals')
grid;
axis([0 10 -5 5]);
subplot(3,1,2),plot(t2,dx2,tt,dxt,':');
ylabel('b) Vel. [m/s]');
grid;
axis([0 10 -5 5]);
subplot(3,1,3),plot(t3,dx3,tt,dxt,':');
ylabel('c) Vel. [m/s]');
grid;
axis([0 10 -5 5]);

*****************************************************************
```

```
****************************************************************
 MATLAB     File for MATLAB Exercise 4, Lecture 3
 Function   EXERC4B.M
 Purpose    To calculate the response of the system de-
            scribed in problem 1, lecture 3. A Runge-Kutta
            fourth order method is used.
 Call       exerc4b(k,c,m,f0,omf,n,dt) (fig.Prob. 1)
 k          Stifness of linear spring
 c          Damping constant
 m          Mass
 f0         Amplitude of force
 omf        Eigenfrequency of harmonic force
 n          Number of points in approximtation
 dt         Time distance between points
 Return     (tt,xt,dxt,Ft)
 tt         Time vector
 xt         Displacement response vector
 dxt        Velocity response vector
 Ft         Force vector
****************************************************************
function [tt,xt,dxt,Ft]=exerc4b(k,c,m,f0,omf,n,dt)

om0=0.5*sqrt(k/m);
z=c/sqrt(m*k);

if (om0==omf) psi=pi/2;
else psi=atan((2*z*om0*omf)/(om0^2-omf^2));
end;
X=(0.75*f0)/(m*sqrt((om0^2-omf^2)^2+4*(z^2)*(om0^2)*(omf^2)));
omd=om0*sqrt(1-z^2);

laengde=n*dt;
dt=laengde/4000;
n=4000;

for l=1:n+1,
tt(l)=(l-1)*dt;
Displacement response
xh1=cos(omf*tt(l)-psi);
xh2=cos(psi)*cos(omd*tt(l));
xh3=sin(omd*tt(l))*((z*cos(psi))/(sqrt(1-z^2))+(omf/omd)*sin(psi));
xt(l)=X*(xh1-exp(-z*om0*tt(l))*(xh2+xh3));
Velocity response
dxh1=-omf*sin(omf*tt(l)-psi);
dxh2=xh2;
dxh3=xh3;
dxh4=-omd*cos(psi)*sin(omd*tt(l));
dxh5=omd*cos(omd*tt(l))*((z*cos(psi))/(sqrt(1-z^2))+(omf/omd)*sin(psi));
dxt(l)=X*(dxh1+z*om0*exp(-z*om0*tt(l))*(dxh2+dxh3)-exp(-z*om0*tt(l))*(dxh4+dxh5));
Ft(l)=cos(omf*tt(l));
end;
****************************************************************
```

```
*****************************************************************
 MATLAB     File for MATLAB Exercise 4, Lecture 3
 Function   PLOT4b.M
 Purpose    To plot the displacement response and the ve-
            locity response versus the force on the system
            in Problem 1, Lecture 4
 Call       plot4b(t,xt,dxt,Ft)
 t          Time vector
 xt         Displacement
 dxt        Velocity
 Ft         Force applied to the system
 Return     Graphical plot
*****************************************************************

function plot4b(t,xt,dxt,Ft)
clg;
subplot(3,1,1),plot(t,Ft);
ylabel('a) Force [N]');
title('Comparison: Force-Response');
grid;
subplot(3,1,2),plot(t,xt);
ylabel('b) Disp. [m]');
grid;
subplot(3,1,3),plot(t,dxt);
ylabel('c) Vel. [m]');
xlabel('Time [sec]')
grid;
*****************************************************************
```

```
*****************************************************************
 MATLAB     File for MATLAB Exercise 4, Lecture 3
 Function   EXERC4C.M
 Purpose    To calculate the response of the system de-
            scribed in problem 1, lecture 3. A Runge-Kutta
            fourth order method is used.
 Call       exerc4c(k,c,m,f0,omf,n,dt) (fig.Prob. 1)
 k          Stifness of linear spring
 c          Damping constant
 m          Mass
 f0         Amplitude of force
 omf        Eigenfrequency of harmonic force
 n          Number of points in approximtation
 dt         Time distance between points
 Return     (tt,xt,dxt,Ft)
 tt         Time vector
 xt         Displacement response vector
 dxt        Velocity response vector
 Ft         Force vector
*****************************************************************
function [tt,xt,dxt,Ft]=exerc4c(k,c,m,f0,omf,n,dt)
om0=0.5*sqrt(k/m);
z=c/sqrt(m*k);
if (om0==omf) psi=pi/2;
else psi=atan((2*z*om0*omf)/(om0^2-omf^2));
end;
X=(0.75*f0)/(m*sqrt((om0^2-omf^2)^2+4*(z^2)*(om0^2)*(omf^2)));
omd=om0*sqrt(1-z^2);
laengde=n*dt;
dt=laengde/4000;
n=4000;
for l=1:n+1,
tt(l)=(l-1)*dt;
```

```
xh1=cos(omf*tt(l)-psi);
xh2=cos(psi)*cos(omd*tt(l));
xh3=sin(omd*tt(l))*((z*cos(psi))/(sqrt(1-z^2))+(omf/omd)*sin(psi));
xt(l)=X*(xh1-exp(-z*om0*tt(l))*(xh2+xh3));
dxh1=-omf*sin(omf*tt(l)-psi);
dxh2=xh2;
dxh3=xh3;
dxh4=-omd*cos(psi)*sin(omd*tt(l));
dxh5=omd*cos(omd*tt(l))*((z*cos(psi))/(sqrt(1-z^2))+(omf/omd)*sin(psi));
dxt(l)=X*(dxh1+z*om0*exp(-z*om0*tt(l))*(dxh2+dxh3)-exp(-z*om0*tt(l))*(dxh4+dxh5));
Ft(l)=cos(omf*tt(l));
end;
```

Chapter 4

Lecture 4

Problem 1 and 3 of lecture 4 are solved theoretically. The solutions to these problems can be found in Nielsen [2]. The following MATLAB exercises replaces problem 2 of lecture 4.

4.1 MATLAB Exercise 5

Determine the eigenvalues λ and eigenvectors $\boldsymbol{\Psi}$ for the following linear eigenvalue problem:

$$(\mathbf{B} - \lambda\mathbf{A}) = \mathbf{0} \tag{4.1}$$

Where **A** and **B** are real and symmetric matrices.

Example 1 (Example 3.5 in Nielsen [1])

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \tag{4.2}$$

Compare results with (3.43), (3.61), and (3.62) in Nielsen [1]. Notice that both **A** and **B** are symmetric and positive definite matrices in this case.

Example 2 (Example 3.10 in [1])

$$\mathbf{A} = \left[\begin{array}{cc|cc} 0.3 & -0.2 & 1 & 0 \\ -0.2 & 0.5 & 0 & 2 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{array}\right], \quad \mathbf{B} = \left[\begin{array}{cc|cc} 200 & -100 & 0 & 0 \\ -100 & 200 & 0 & 0 \\ \hline 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -2 \end{array}\right] \tag{4.3}$$

Compare the results with (3.197), (3.102), (3.230), (3.239), (3.240) and (3.241) in Nielsen [1].

4.1.1 MATLAB Solution

The MATLAB function below (exerc5a.m) shows how MATLAB exercise 5, lecture 4 could be solved. If the following orders are given:

```
A(1,1:2)=[1 0];
A(2,1:2)=[0 2];
B(1,1:2)=[2 -1];
B(2,1:2)=[-1 2];
```

```
[V,D]=exerc5a(A,B);
```

Then the eigenvalues and corresponding eigenvectors of example 1 will appear on the screen. Notice that the eigenvectors only can be calculated except for an arbitrary constant.

Eigenvalues:

$$\begin{bmatrix} 0.6340 & 0 \\ 0 & 2.3660 \end{bmatrix} \tag{4.4}$$

Eigenvectors:

$$\begin{bmatrix} 0.7321 & -2.7321 \\ 1 & 1 \end{bmatrix} \tag{4.5}$$

To solve example 2 give the following orders:

```
A(1,1:4)=[0.3 -0.2 1 0];
A(2,1:4)=[-0.2 0.5 0 2];
A(3,1:4)=[1 0 0 0];
A(4,1:4)=[0 2 0 0];
B(1,1:4)=[200 -100 0 0];
B(2,1:4)=[-100 200 0 0];
B(3,1:4)=[0 0 -1 0];
B(4,1:4)=[0 0 0 -2];
[V,D]=exerc5b(A,B);
```

Notice again that the eigenvectors only can be calculated except an arbitrary constant.

Eigenvalues:

$$\begin{bmatrix} -0.0725+7.9620i & 0 & 0 & 0 \\ 0 & -0.0725-7.9620i & 0 & 0 \\ 0 & 0 & -0.2025+15.3804i & 0 \\ 0 & 0 & 0 & -0.2025-15.3804i \end{bmatrix} \tag{4.6}$$

Eigenvectors:

$$\begin{bmatrix} 0.7321+0.0050i & 0.7321-0.0050i & -2.7313+0.0364i & -2.7313-0.0364i \\ 1 & 1 & 1 & 1 \\ -0.0933+5.8283i & -0.0933-5.8283i & -0.0066-42.0159i & -0.0066+42.0159i \\ -0.0725+7.9620i & -0.0725-7.9620i & -0.2025+15.3804i & -0.2025-15.3804i \end{bmatrix} \tag{4.7}$$

The eigenvectors are the two first rows in each column. What are the relation between the two last rows in each column and the eigenvalues ? (The answer is given in Nielsen [1] p. 77, (3-209)).

4.1.2 MATLAB Files

```
*****************************************************************
  MATLAB      File for MATLAB Exercise 5, Lecture 4
  Function    EXERC5A.M
  Purpose     To calculate the eigenvalues and corresponding
              eigenvectors of the eigenvalue problem of a
              linear system.
  Call        exerc5a(A,B)
  A           Matrix (see problem text)
  B           Matrix (see problem text)
  Return      (x,dx,t)
  Lambda      Eigenvalues
  Phi         Eigenvectors
*****************************************************************
function [V,D] = exerc5a(A,B)
[V,D]=eig(B,A);
Eigenvalues=D
Eigenvectors=V
*****************************************************************
```

```
*****************************************************************
  MATLAB      File for MATLAB Exercise 5, Lecture 4
  Function    EXERC5B.M
  Purpose     To calculate the eigenvalues and corresponding
              eigenvectors of the eigenvalue problem of a
              linear system.
  Call        exerc5b(A,B)
  A           Matrix (see problem text)
  B           Matrix (see problem text)
  Return      (x,dx,t)
  Lambda      Eigenvalues
  Phi         Eigenvectors
*****************************************************************
function [V,D] = exerc5b(A,B);
[V,D]=eig(B,-A);
for i=1:4, V(:,i)=V(:,i)/V(2,i); end;
C1=V(:,1); C2=V(:,4); C3=D(1,1); C4=D(4,4);
V(:,1)=C2; V(:,4)=C1;
D(1,1)=C4; D(4,4)=C3;
Eigenvalues=D
Eigenvectors=V
*****************************************************************
```

4.2 MATLAB Exercise 6

Consider the linear second order single degree of freedom system from problem 1, lecture 3 in Nielsen [2]. Instead of a harmonic excitation force the system is excited by an impulse:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = f(t) \tag{4.8}$$

The initial conditions are:

$$x(0) = \dot{x}(0) = 0 \tag{4.9}$$

Because of the initial conditions the response of the linear oscillator is only determined by Duhamel's integral.

$$x(t) = \int_0^t h(t-\tau)f(\tau)d\tau \tag{4.10}$$

The impulse excitation is shown in figure 7.2.

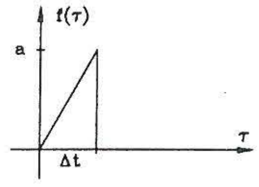


Figure 4.1: *Impulse excitation of a linear single degree of freedom system.*

The analytical solution to the vibration problem can be found in Appendix A. Verify upon insertion in the equation of motion (4.8) that the solutions for $t \leq \Delta t$ becomes:

$$x(t) = \frac{a}{\Delta t m \omega_0^2}\left(t - \frac{2\zeta}{\omega_0} + \frac{e^{-\zeta\omega_0 t}}{\omega_d\omega_0^2}(2\zeta\omega_0\omega_d cos(\omega_d t) + (\zeta^2\omega_0^2 - \omega_d^2)sin(\omega_d t))\right) \tag{4.11}$$

Solve (4.8) numerically using a 4th order Runge-Kutta scheme and compare the solutions with the analytical solution (4.11). Use the following data:

$$\omega_0 = 2\pi Hz\ , \quad k = 16\pi^2\frac{N}{m}\ , \quad m = 1kg\ , \quad c = 0.04\pi\frac{kg}{s}\ , \quad a = 1N\ , \quad \Delta t = 0.5s \tag{4.12}$$

Plot the displacement and the velocity response of the system for different approximations.

Solve (4.10) using Simpson integration. Compare the solution with the analytical solution (4.11). Use the data from (4.12). Plot the displacement of the system for different approximations. It is important to notice that the vibration problem can be solved using two different approaches: A numerical approximation of the differential equations or a numerical integration of the Duhamel integral.

4.2.1 MATLAB Solution

The MATLAB functions below (exerc6a.m, plot6a.m) shows how the numerical approximation of the differential equation in MATLAB exercise 5, lecture 4 could be performed. If the following four orders are given:

```
[t1,x1,dx1,tt,xt,dxt]=exerc6a(16*pi^2,0.04*pi,1,1,0.5,25,0.2);
[t2,x2,dx2,tt,xt,dxt]=exerc6a(16*pi^2,0.04*pi,1,1,0.5,50,0.1);
[t3,x3,dx3,tt,xt,dxt]=exerc6a(16*pi^2,0.04*pi,1,1,0.5,1000,0.005);
plot6a(t1,t2,t3,x1,x2,x3,dx1,dx2,dx3,tt,xt,dxt);
```

Then figure 4.2 and figure 4.3 will appear on the screen. Notice that "RETURN" should be pressed to shift from figure 4.2 to figure 4.3. An explanation of the input numbers can be found in the head of the MATLAB functions (exerc6a.m, plot6a.m). Notice how the numerical approximation converges to the theoretical solution as the time intervals decreases.

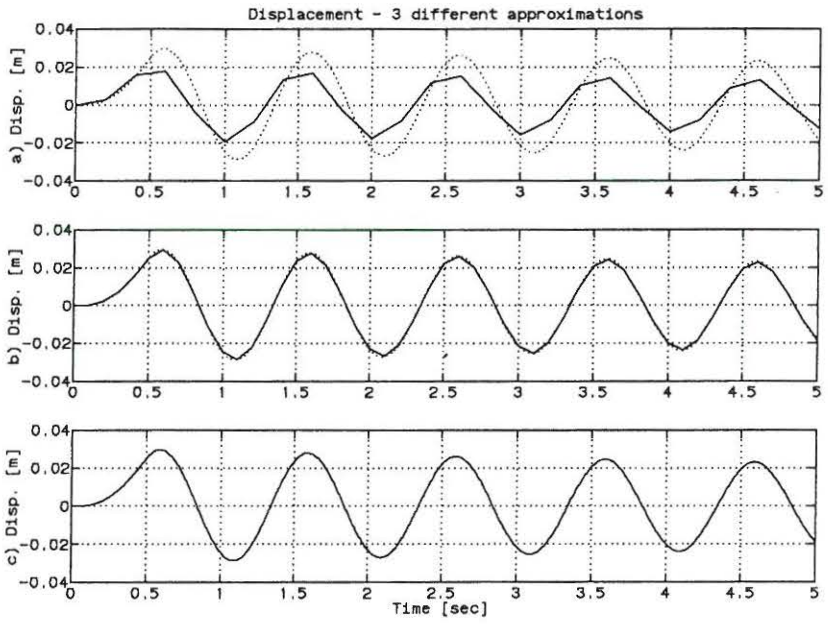


Figure 4.2: *Numerical approximation of the displacement with a fouth order Runge-Kutta scheme compared with theoretical solution. a)* $\Delta t = \frac{T_0}{5}$*, b)* $\Delta t = \frac{T_0}{10}$*, c)* $\Delta t = \frac{T_0}{200}$.

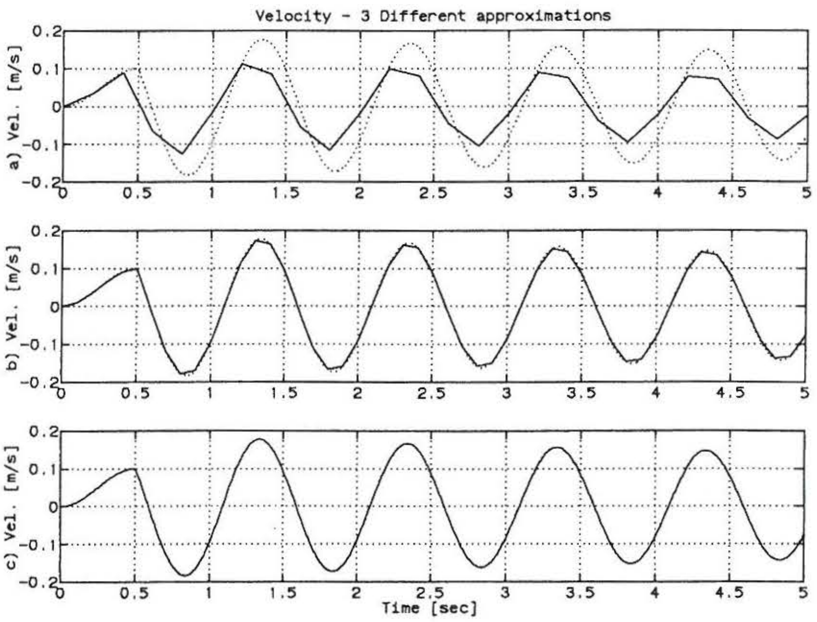


Figure 4.3: *Numerical approximation of the velocity with a fouth order Runge-Kutta scheme compared with theoretical solution. a)* $\Delta t = \frac{T_0}{5}$*, b)* $\Delta t = \frac{T_0}{10}$*, c)* $\Delta t = \frac{T_0}{200}$.

The MATLAB functions below (exerc6b.m, plot6b.m) shows how the numerical integration of the Duhamel integral in MATLAB exercise 5, lecture 4 could be performed using Simpson integration. If the following four orders are given:

```
[t1,x1,tt,xt]=exerc6b(16*pi^2,0.04*pi,1,1,0.5,25,0.2);

[t2,x2,tt,xt]=exerc6b(16*pi^2,0.04*pi,1,1,0.5,50,0.1);

[t3,x3,tt,xt]=exerc6b(16*pi^2,0.04*pi,1,1,0.5,100,0.05);

plot6b(t1,t2,t3,x1,x2,x3,tt,xt);
```

Then figure 4.4 will appear on the screen. An explanation of the input numbers can be found in the head of the MATLAB functions (exerc6b.m, plot6b.m). Notice how the numerical approximation converges to the theoretical solution as the time intervals decreases.

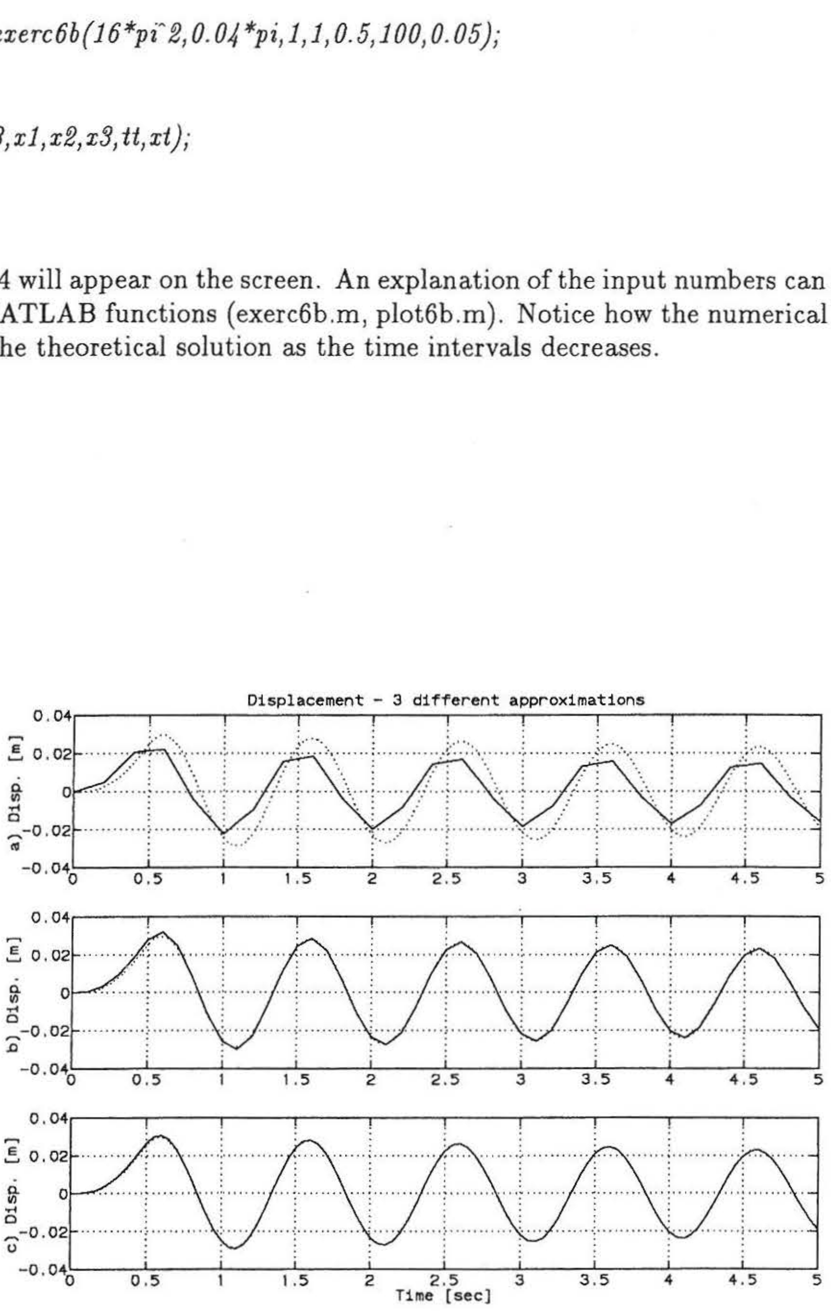


Figure 4.4: *Numerical approximation of the displacement calculated using Simpson integration compared with theoretical solution. a) $\Delta t = \frac{T_0}{5}$, b) $\Delta t = \frac{T_0}{10}$, c) $\Delta t = \frac{T_0}{200}$.*

4.2.2 MATLAB Files

```
*****************************************************************

  MATLAB     File for MATLAB Exercise 5, Lecture 4
  Function   EXERC6A.M
  Purpose    To calculate the response of the system de-
             scribed in MATLAB exercise 5, lecture 4. A
             Runge-Kutta fourth order method is used.
  Call       exerc6a(k,c,m,a,dta,n,dt)
  k          Stifness of linear spring
  c          Damping constant
  m          Mass
  a          Maximum value of force
  dta        Time length of impulse
  n          Number of points in approximtation
  dt         Time distance between points
  Return     (x,dx,t)
  t          Time vector - numerical solution
  x1         Displacement response vector - numerical
  x2         Velocity response vector - numerical
  tt         Time vector - theoretical solution
  xt         Displacement response vector - theoretical
  dxt        Velocity response vector - numerical solution
*****************************************************************
function [t,x1,x2,tt,xt,dxt]=exerc6a(k,c,m,a,dta,n,dt)
om0=0.5*sqrt(k/m);
z=c/sqrt(m*k);
omd=om0*sqrt(1-z^2);
C1=om0^2;
C2=2*z*om0;
for l=1:n+1, x1(l)=0; x2(l)=0; t(l)=0; tt(l)=0; end;
x1(1)=0; x2(1)=0; t(1)=0; for l=1:n,
if (l*dt¡=0.5), C3=a/(m*0.5); else C3=0; end;
k1=0.5*dt*(-C1*x1(l)-C2*x2(l)+C3*t(l));
K=0.5*dt*(x2(l)+0.5*k1);
k2=0.5*dt*(-C1*(x1(l)+K)-C2*(x2(l)+k1)+C3*(t(l)+0.5*dt));
k3=0.5*dt*(-C1*(x1(l)+K)-C2*(x2(l)+k2)+C3*(t(l)+0.5*dt));
LL=dt*(x2(l)+k3);
k4=0.5*dt*(-C1*(x1(l)+LL)-C2*(x2(l)+2*k3)+C3*(t(l)+dt));
x1(l+1)=x1(l)+dt*(x2(l)+(1/3)*(k1+k2+k3));
x2(l+1)=x2(l)+(1/3)*(k1+2*k2+2*k3+k4);
t(l+1)=t(l)+dt;
end;
c1=a/(m*om0^2*0.5); c4=z^2*om0^2;
c5=2*z*om0*omd; c6=omd^2;
c7=2*z/om0; c8=1/(omd*om0^2);
C1=0.5*c5*z*om0; C2=z*om0^2*omd;
C3=omd^3; C4=z^2*om0^2*omd;
C5=2*z*om0*omd^2; C6=z^3*om0^3;
time1=dta/500;
for l=1:501,
tt(l)=(l-1)*time1;
c3=exp(-z*om0*tt(l));
ct=omd*tt(l);
xt(l)=c1*(tt(l)-c7+c3*c8*((c4-c6)*sin(ct)+c5*cos(ct)));
dxt(l)=c1*(1+c3*c8*((-C1-C3)*cos(ct)+(C2-C5-C6)*sin(ct)));
end;
time2=(n*dt-dta)/2000;
x0=xt(501); dx0=dxt(501);
c1=(dx0+z*om0*x0)/omd;
for l=1:2001,
tt(501+l)=dta+l*time2;
tt1(l)=l*time2;
c3=exp(-z*om0*tt1(l));
```

```
ct=omd*tt1(l);
xt(501+l)=c3*(x0*cos(ct)+c1*sin(ct));
dxt(501+l)=-z*om0*xt(501+l)+c3*omd*(-x0*sin(ct)+c1*cos(ct));
end;
******************************************************************
```

```
******************************************************************
  MATLAB     File for MATLAB Exercise 5, Lecture 4
  Function   PLOT6a.M
  Purpose    To plot the displacement response and the ve-
             locity response in MATLAB Exercise 5, lecture 4
  Call       plot6a(t1,t2,t3,x1,x2,x3,dx1,dx2,dx3,tt,xt,dxt)
  t1         Time vector (1. Aproximation)
  t2         Time vector (2. Aproximation)
  t3         Time vector (3. Aproximation)
  x1         Displacement (1. Aproximation)
  x2         Displacement (2. Aproximation)
  x3         Displacement (3. Aproximation)
  dx1        Velocity (1. Aproximation)
  dx2        Velocity (2. Aproximation)
  dx3        Velocity (3. Aproximation)
  tt         Time vector
  xt         Theoretical Displacement
  dxt        Theoretical Velocity
  Return     Graphical plot
******************************************************************
function plot6a(t1,t2,t3,x1,x2,x3,dx1,dx2,dx3,tt,xt,dxt)
clg;
subplot(3,1,1),plot(t1,x1,tt,xt,':');
ylabel('a) Disp. [m]');
axis([0 5 -0.04 0.04]);
title('Displacement - 3 different approximations')
grid;
subplot(3,1,2),plot(t2,x2,tt,xt,':');
ylabel('b) Disp. [m]');
axis([0 5 -0.04 0.04]);
grid;
subplot(3,1,3),plot(t3,x3,tt,xt,':');
ylabel('c) Disp. [m]');
xlabel('Time [sec]');
axis([0 5 -0.04 0.04]);
grid;
pause
clg;
subplot(3,1,1),plot(t1,dx1,tt,dxt,':');
ylabel('a) Vel. [m/s]');
title('Velocity - 3 Different approximations')
axis([0 5 -0.2 0.2]);
grid;
subplot(3,1,2),plot(t2,dx2,tt,dxt,':');
ylabel('b) Vel. [m/s]');
axis([0 5 -0.2 0.2]);
grid;
subplot(3,1,3),plot(t3,dx3,tt,dxt,':');
ylabel('c) Vel. [m/s]');
xlabel('Time [sec]');
axis([0 5 -0.2 0.2]);
grid;
******************************************************************
```

```
*****************************************************************
 MATLAB    file for MATLAB Exercise 5, Lecture 4
 Function  EXERC6B.M
 Purpose   To calculate the response of the system de-
           scribed in MATLAB exercise 5, lecture 4.
           Simpsons integration method is used for Duha-
           mels integral
 Call      exerc6b(k,c,m,a,dta,n,dt)
 k         Stifness of linear spring
 c         Damping constant
 m         Mass
 a         Maximum value of force
 dta       Time length of impulse
 n         Number of points in approximation
 dt        Time distance between points
 Return    (t,x1,tt,xt)
 t         Time vector - numerical solution
 x1        Displacement response vector - numerical
 tt        Time vector - theoretical solution
 xt        Displacement response vector - theoretical
*****************************************************************
function [t,x1,tt,xt]=exerc6b(k,c,m,a,dta,n,dt)

om0=0.5*sqrt(k/m);
z=c/sqrt(m*k);
omd=om0*sqrt(1-z^2);

C1=1/(m*omd);
for i=1:n+1, x1(i)=0; t(i)=0; end;
for i=1:n,
ta=i*dt;
for j=1:i,
if (j*dt¿dta), a1=0; else a1=a; end;
t1=(j-1)*dt; t2=t1+0.5*dt; t3=t1+dt;
g1=C1*exp(-z*om0*(ta-t1))*sin(omd*(ta-t1))*a1*t1/dta;
g2=C1*exp(-z*om0*(ta-t2))*sin(omd*(ta-t2))*a1*t2/dta;
g3=C1*exp(-z*om0*(ta-t3))*sin(omd*(ta-t3))*a1*t3/dta;
simpsum=(g1+4*g2+g3);
x1(i+1)=x1(i+1)+simpsum;
end;
x1(i+1)=ta*x1(i+1)/(3*(2*i-1));
t(i+1)=i*dt;
end;

c1=a/(m*om0^2*0.5); c4=z^2*om0^2;
c5=2*z*om0*omd; c6=omd^2;
c7=2*z/om0; c8=1/(omd*om0^2);
C1=0.5*c5*z*om0; C2=z*om0^2*omd;
C3=omd^3; C4=z^2*om0^2*omd;
C5=2*z*om0*omd^2; C6=z^3*om0^3;

time1=dta/500;

for l=1:501,
tt(l)=(l-1)*time1;
c3=exp(-z*om0*tt(l));
ct=omd*tt(l);
xt(l)=c1*(tt(l)-c7+c3*c8*((c4-c6)*sin(ct)+c5*cos(ct)));
end;
dx0=c1*(1+c3*c8*((-C1-C3)*cos(ct)+(C2-C5-C6)*sin(ct)));
time2=(n*dt-dta)/2000;

x0=xt(501);
c1=(dx0+z*om0*x0)/omd;
```

```
for l=1:2001,
tt(501+l)=dta+l*time2;
tt1(l)=l*time2;
c3=exp(-z*om0*tt1(l));
ct=omd*tt1(l);
xt(501+l)=c3*(x0*cos(ct)+c1*sin(ct));
end;
*****************************************************************
```

```
*****************************************************************
```

MATLAB	file for MATLAB Exercise 5, Lecture 4
Function	PLOT6B.M
Purpose	To plot the displacement response MATLAB Exercise 5, lecture 4
Call	plot6b(t1,t2,t3,x1,x2,x3,tt,xt,dxt)
t1	Time vector (1. Aproximation)
t2	Time vector (2. Aproximation)
t3	Time vector (3. Aproximation)
x1	Displacement (1. Aproximation)
x2	Displacement (2. Aproximation)
x3	Displacement (3. Aproximation)
tt	Time vector
xt	Theoretical Displacement
dxt	Theoretical Velocity
Return	Graphical plot

```
*****************************************************************
function plot6b(t1,t2,t3,x1,x2,x3,tt,xt)
clg;
subplot(3,1,1),plot(t1,x1,tt,xt,':');
ylabel('a) Disp. [m]');
axis([0 5 -0.04 0.04]);
title('Displacement - 3 different approximations')
grid;
subplot(3,1,2),plot(t2,x2,tt,xt,':');
ylabel('b) Disp. [m]');
axis([0 5 -0.04 0.04]);
grid;
subplot(3,1,3),plot(t3,x3,tt,xt,':');
ylabel('c) Disp. [m]');
xlabel('Time [sec]');
axis([0 5 -0.04 0.04]);
grid;
*****************************************************************
```

Chapter 5

Lecture 5

Problem 2 of lecture 5 is solved theoretically. The solutions to these problems can be found in Nielsen [2]. The following MATLAB exercise replaces problem 1 of lecture 5.

5.1 MATLAB Exercise 7

The system of problem 2, lecture 5 is considered. The equation of motion and the initial conditions are:

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{K}\mathbf{x} = 0 , \quad t > 0 \tag{5.1}$$

$$\mathbf{x}_0 = \begin{bmatrix} 1 \\ \frac{1}{3} \end{bmatrix} , \quad \dot{\mathbf{x}}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.2}$$

Verify that (5.1) can be written in the following *state vector form*:

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} , \quad \mathbf{z}(0) = \mathbf{z}_0 \tag{5.3}$$

$$\mathbf{z}(t) = \begin{bmatrix} \mathbf{x}(t) \\ \dot{\mathbf{x}}(t) \end{bmatrix} , \quad \mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & \mathbf{0} \end{bmatrix} , \quad \mathbf{z}_0 = \begin{bmatrix} \mathbf{x}_0 \\ \dot{\mathbf{x}}_0 \end{bmatrix} \tag{5.4}$$

Solve (5.4) numerically using a 4th order Runge-Kutta scheme and compare the solution with the analytical solution (see Nielsen [2], p. 36, (12)). Use the following data:

$$k_1 = 8\pi^2 \frac{N}{m} , \quad k_2 = 16\pi^2 \frac{N}{m} , \quad m1 = 1kg , \quad m2 = 2kg \tag{5.5}$$

The eigenfrequencies can be found in Nielsen [2], p. 35 (3).

5.1.1 MATLAB Solution

The MATLAB functions below (exerc7.m, plot7.m) shows how MATLAB exercise 7, lecture 5 could be solved. If the following three orders are given:

```
[x1,xt,t1,tt]=exerc7(8*pi^2,16*pi^2,1,2,[1 1/3], [0 0],25,0.2);
[x2,xt,t2,tt]=exerc7(8*pi^2,16*pi^2,1,2,[1 1/3], [0 0],50,0.1);
plot7(t1,t2,tt,x1,x2,xt);
```

Then figure 5.1 will appear on the screen. An explanation of the input numbers can be found in the head of the MATLAB functions (exerc7.m, plot7.m)

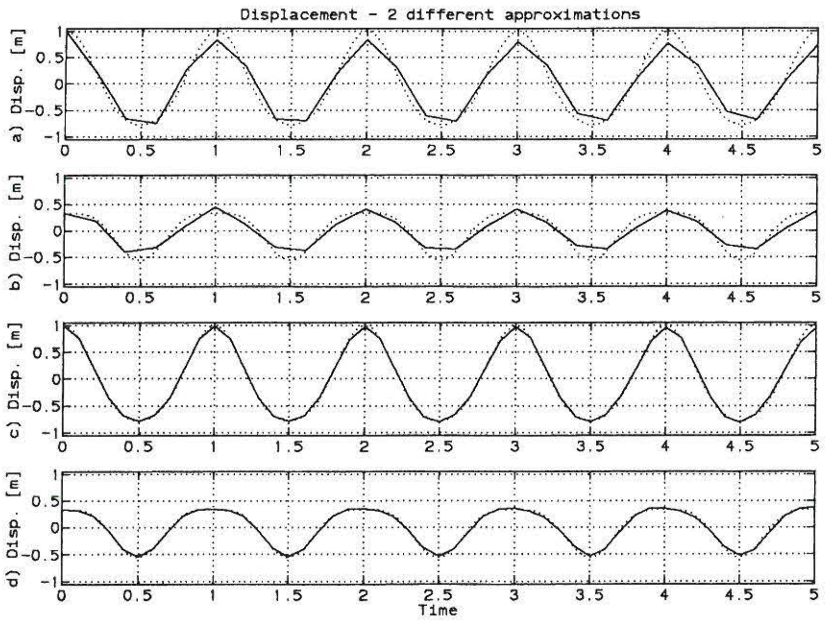


Figure 5.1: *Numerical approximation [—] with a fourth order Runge-Kutta scheme compared with the theoretically solution [· · ·]. a) $\Delta t = \frac{T_0}{5}$ 2. storey b) $\Delta t = \frac{T_0}{5}$ 1. storey c) $\Delta t = \frac{T_0}{10}$ 2. storey d) $\Delta t = \frac{T_0}{10}$1.storey.*

5.1.2 MATLAB Files

```
*********************************************************************
  MATLAB     File for MATLAB Exercise 7, Lecture 5
  Function   EXERC7.M
  Purpose    To calculate the response of the 2DOF system
             described in problem 2, lecture 5. A Runge-
             Kutta fourth order method is used.
  Call       exerc7(k1,k2,m1,m2,x0,dx0,n,dt) (fig.prob. 2)
  k1         Stifness of upper storey
  k2         Stifness of lower storey
  m1         Mass of upper storey
  m2         Mass of lower storey
  x0         Initial conditions vector, displacement
  dx0        Initial conditions vector, velocity
  n          Number of points in approximtation
  dt         Time distance between points
  Return     (xn,xt,tn,tt)
  xn         Displacement response matrix (approximation)
  xt         Displacement response matrix (Theoretically)
  tn         Time vector corresponding to xn
  tt         Time vector corresponding to xa
*********************************************************************
function [xn,xt,tn,tt]=exerc7(k1,k2,m1,m2,x0,dx0,n,dt)
om0(1)=sqrt(0.5*k1/m1); om0(2)=sqrt(2.0*k2/m2);
M(1,1)=m1; M(1,2)=0; M(2,1)=0; M(2,2)=m2;
K(1,1)=k1; K(1,2)=-k1; K(2,1)=-k1; K(2,2)=k1+k2;
Phi1(1)=2; Phi1(2)=1; Phi2(1)=-1; Phi2(2)=1;
P=[Phi1' Phi2'];
MI=inv(M);
PI=inv(P);
tn=0:dt:n*dt; xn=zeros(2,n+1); dxn=zeros(2,n+1);
xn=zeros(2,n+1); dxn=zeros(2,n+1);
xn(1,1)=x0(1); xn(2,1)=x0(2);
c=-MI*K;
```

```
for i=1:n,
k1=0.5*dt*c*xn(1:2,i);
K=0.5*dt*(dxn(1:2,i)+0.5*k1);
k2=0.5*dt*c*(xn(1:2,i)+K);
k3=0.5*dt*c*(xn(1:2,i)+K);
LL=dt*(dxn(1:2,i)+k3);
k4=0.5*dt*c*(xn(1:2,i)+LL);
xn(1:2,i+1)=xn(1:2,i)+dt*(dxn(1:2,i)+(k1+k2+k3)/3);
dxn(1:2,i+1)=dxn(1:2,i)+(1/3)*(k1+2*k2+2*k3+k4);
end;
a=PI*x0';
bb=PI*dx0'; b(1)=bb(1)/om0(1);, b(2)=bb(2)/om0(2);
dtt=n*dt/1001;
tt=0:dtt:1000*dtt; xt=zeros(2,1001);
for i=1:1001,
xt(1:2,i)=a(1)*Phi1'*cos(om0(1)*tt(i))+a(2)*Phi2'*cos(om0(2)*tt(i));
xt(1:2,i)=xt(1:2,i)+b(1)*Phi1'*sin(om0(1)*tt(i))+b(2)*Phi2'*sin(om0(2)*tt(i));
end;
*******************************************************************
```

```
*******************************************************************
  MATLAB file for MATLAB Exercise 7, Lecture 5
  Function PLOT7.M
  Purpose To plot the displacement response for a 2DOF
  system in MATLAB Exercise 7, lecture 5
  Call plot7(t1,t2,tt,x1,x2,xt)
  t1 Time vector (1. Aproximation)
  t2 Time vector (2. Aproximation)
  tt Time vector theoretically
  x1 Displacement matrix (1. Aproximation)
  x2 Displacement matrix (2. Aproximation)
  xt Displacement matrix theoretically
  Return Graphical plot
*******************************************************************
function plot7(t1,t2,tt,x1,x2,xt)
clg;
x11=x1(1,1:size(t1')); x12=x1(2,1:size(t1'));
xt1=xt(1,1:size(tt')); xt2=xt(2,1:size(tt'));
x21=x2(1,1:size(t2')); x22=x2(2,1:size(t2'));
subplot(4,1,1),plot(t1,x11,tt,xt1,':');
ylabel('a) Disp. [m]');
title('Displacement - 2 different approximations')
grid;
axis([0 5 -1.05 1.05]);
subplot(4,1,2),plot(t1,x12,tt,xt2,':');
ylabel('b) Disp. [m]');
grid;
axis([0 5 -1.05 1.05]);
subplot(4,1,3),plot(t2,x21,tt,xt1,':');
ylabel('c) Disp. [m]');
grid;
axis([0 5 -1.05 1.05]);
subplot(4,1,4),plot(t2,x22,tt,xt2,':');
ylabel('d) Disp. [m]');
grid; xlabel('Time')
axis([0 5 -1.05 1.05]);
*******************************************************************
```

Chapter 6

Lecture 6

Problem 1 and 3 of lecture 6 are solved theoretically. The solutions to these problems can be found in Nielsen [2]. The following MATLAB exercise replaces problem 2 of lecture 6.

6.1 MATLAB Exercise 8

The system in problem 1, lecture 6 is considered again. The harmonic load subjected to mass, m_1, is replaced by the following load function (still subjected to mass, m_1).

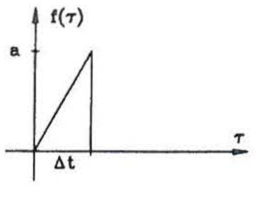


Figure 6.1: *Impulse excitation of linear 2DOF system in problem 1 lecture 6.*

The impulse excitation force subjected to mass m_1 is described by:

$$f(\tau) = \begin{cases} a\frac{\tau}{\Delta t}, & 0 < \tau \leq \Delta t \\ 0, & \tau > \Delta t \end{cases} \tag{6.1}$$

The system starts at rest to time $t = 0$ corresponding to the initial values $\mathbf{x}(0) = \mathbf{0}$, $\dot{\mathbf{x}}(0) = \mathbf{0}$. The indicated load function has previously been used in MATLAB exercise 6.

1: Determine the undamped circular eigenfrequencies and the undamped eigenmodes of the system.
2: Determine the motion of the system using expansion in undamped modal coordinates.
3: Compare the theoretical solution with a numerical approximation using a fourth order Runge-Kutta scheme.

6.1.1 MATLAB Solution

The governing differential equation of the system is:

$$\mathbf{M}\ddot{\mathbf{x}}(t) \;+\; \mathbf{K}\mathbf{x}(t) \;=\; \mathbf{f}(t) \tag{6.2}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \;,\; \mathbf{f}(t) = \begin{bmatrix} f(t) \\ 0 \end{bmatrix} \;,\; \mathbf{M} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \;,\; \mathbf{K} = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \tag{6.3}$$

The frequency condition is used to calculate the undamped circurlar eigenfrequencies and undamped eigenmodes.

$$\left(\mathbf{K} - \omega^2\mathbf{M}\right)\boldsymbol{\Phi} \;=\; \mathbf{0} \tag{6.4}$$

Undamped circular eigenfrequencies

$$\begin{aligned}
\det\left(\mathbf{K} - \omega^2\mathbf{M}\right) \;&=\; 0 \Rightarrow \\
\det\left(\begin{bmatrix} k_1 + k_2 - \omega^2 m_1 & -k_2 \\ -k_2 & k_2 + \omega^2 m_2 \end{bmatrix}\right) \;&=\; 0 \Rightarrow \\
(k_1 + k_2 - \omega^2 m_1)(k_2 - \omega^2 m_2) - (-k_2)(-k_2) \;&=\; 0 \Rightarrow \\
m_1 m_2 \omega^4 \;+\; (-k_1 m_2 - k_2 m_1 - k_2 m_2)\omega^2 \;+\; k_1 k_2 \;&=\; 0 \Rightarrow \\
\left.\begin{matrix} \omega_1^2 \\ \omega_2^2 \end{matrix}\right\} \;&=\; \frac{k_1 m_2 + k_2(m_1 + m_2) \pm \sqrt{k_1^2 m_2^2 + k_2^2(m_2 + m_1)^2 + 2k_1 k_2 m_2(m_2 - m_1)}}{2 m_1 m_2}
\end{aligned} \tag{6.5}$$

Undamped eigenmodes

If $\Phi_2^{(j)} = 1$ it then follows from the second equation of (6.4) that:

$$-\,k_2\Phi_1^{(j)} + (k_2 - \omega_j^2 m_2)\Phi_2^{(j)} = 0 \quad\Rightarrow\quad \Phi_1^{(j)} = 1 - \omega_j^2 \frac{m_2}{k_2} \tag{6.6}$$

Where the circular eigenfrequencies are given by (6.5).

Motion of system

The motion of the system can be describe by expansion of undamped modal coordinates:

$$\mathbf{x}(t) \;=\; q_1(t)\boldsymbol{\Phi}^{(1)} \;+\; q_2(t)\boldsymbol{\Phi}^{(2)} \tag{6.7}$$

The undamped eigenmodes of the system are known from (6.6). The modal coordinates $q_i(t)$ are found from (see Nielsen [1], p. 70, (3-166)):

$$q_i(t) \;=\; \int_0^t h_i(t - \tau)F_i(\tau)d\tau \tag{6.8}$$

Where:

$$h_i(t) \;=\; \begin{cases} 0 & t \leq 0 \\ \frac{1}{M_i\omega_i}sin(\omega_i t) & t > 0 \end{cases} \tag{6.9}$$

$$M_i = \mathbf{\Phi}^{(i)T}\mathbf{M}\mathbf{\Phi}^{(i)} = (\mathbf{\Phi}^{(i)})^2 m_1 + m_2 = \left(1-\omega_i^2\frac{m_2}{k_2})^2\right) m_1 + m_2 \tag{6.10}$$

$$F_i(t) = \mathbf{\Phi}^{(i)T}\mathbf{f}(t) = \Phi_1^{(i)}f(t) \tag{6.11}$$

$$\begin{aligned} q_i(t) &= \frac{\Phi_1^{(i)}}{M_i\omega_i}\int_0^{min(t,\Delta t)}\sin(\omega_i(t-\tau))f(\tau)d\tau = \frac{a\Phi_1^{(i)}}{\Delta t M_i\omega_i}\int_0^{min(t,\Delta t)}\sin(\omega_i(t-\tau))\tau d\tau \\ &= \frac{a\Phi_1^{(i)}}{\Delta t M_i\omega_i^2}\left[\tau\cos(\omega_i(t-\tau)) + \frac{1}{\omega_i}\sin(\omega_i(t-\tau))\right]_0^{min(t,\Delta t)} \\ &= \frac{a\Phi_1^{(i)}}{\Delta t M_i\omega_i^2}(\cos(\omega_i(t-min(t,\Delta t)))min(t,\Delta t)) + \frac{1}{\omega_i}\sin(\omega_i(t-min(t,\Delta t)) - \sin(\omega_i t)) \end{aligned} \tag{6.12}$$

Numerical approximation
The MATLAB functions below (exerc8.m, plot8.m) shows how the numerical approximation using a 4th order Runge-Kutta scheme could be solved. If the the following 4 orders are given:

```
[x1,xt,t1,tt]=exerc8(4000*pi^2,400*pi^2,1000,100,1 ,0.5,50,0.1);
[x2,xt,t2,tt]=exerc8(4000*pi^2,400*pi^2,1000,100,1 ,0.5,100,0.05);
[x3,xt,t3,tt]=exerc8(4000*pi^2,400*pi^2,1000,100,1 ,0.5,500,0.01);
plot8(t1,t2,t3,x1,x2,x3,tt,xt);
```

Then figure 6.2 and figure 6.3 will appear on the screen. Notice that "RETURN" should be pressed to shift from figure 6.2 to figure 6.3. An explanation of the input numbers can be found in the head of the MATLAB files.

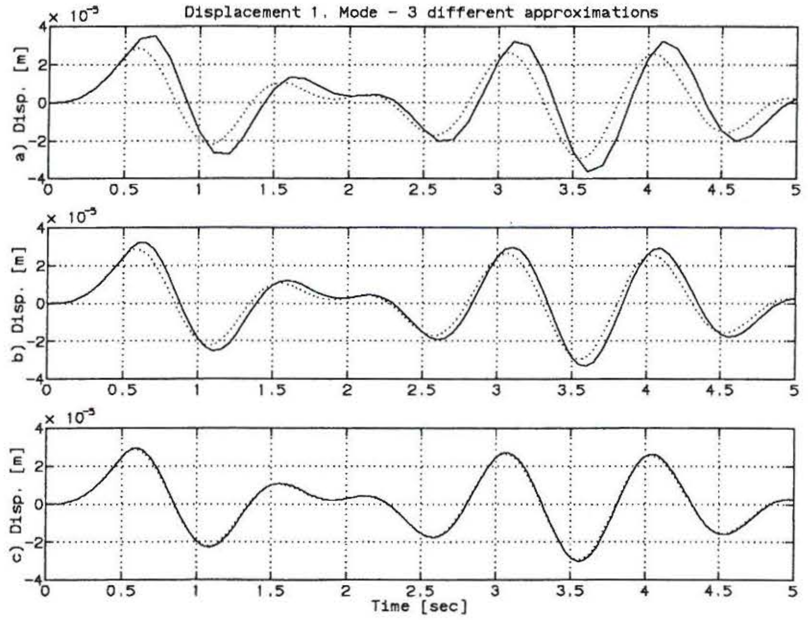


Figure 6.2: *Theoretical displacement response [· · ·] and three different numerical approximations [—] of first degree of freedom. a)* $\Delta t = \frac{T_1}{25}$ $(\frac{T_2}{14})$, *b)* $\Delta t = \frac{T_1}{50}$ $(\frac{T_2}{28})$, *c)* $\Delta t = \frac{T_1}{250}$ $(\frac{T_2}{140})$.

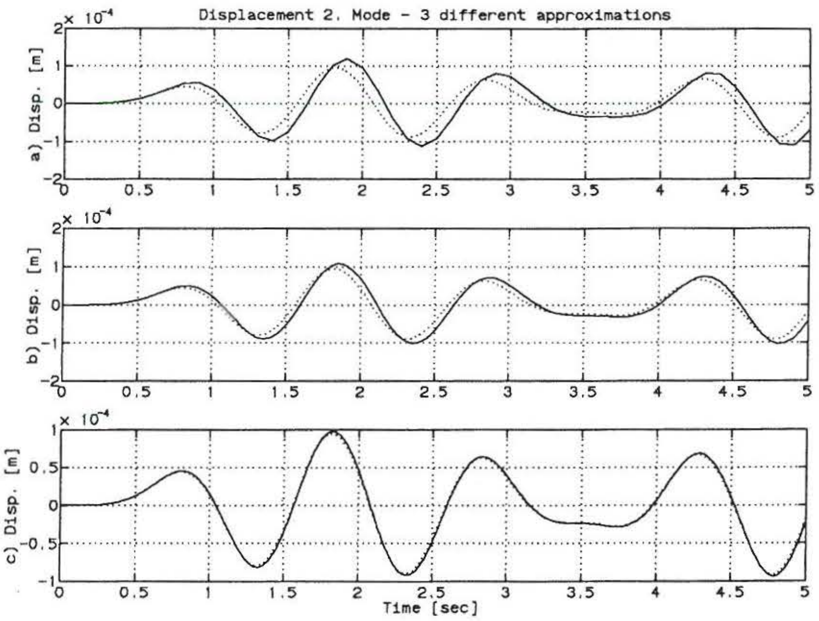


Figure 6.3: *Theoretical displacement response [· · ·] and three different numerical approximations [–] of second degree of freedom. a)* $\Delta t = \frac{T_1}{25}$ ($\frac{T_2}{14}$), *b)* $\Delta t = \frac{T_1}{50}$ ($\frac{T_2}{28}$), *c)* $\Delta t = \frac{T_1}{250}$ ($\frac{T_2}{140}$).

6.1.2 MATLAB Files

****

MATLAB	File for MATLAB Exercise 8, Lecture 6
Function	EXERC8.M
Purpose	To calculate the response of the 2DOF system described in problem 1, lecture 6. A Runge-Kutta fourth order method is used.
Call	exerc8(k1,k2,m1,m2,a,dta,n,dt)
k1	Stifness of first linear spring
k2	Stifness of second linear spring
m1	Mass 1
m2	Mass 2
a	Maximal impulse force
dta	Time lengt of impulse force
n	Number of points in approximtation
dt	Time distance between points
Return	(xn,xt,tn,tt)
xn	Displacement response matrix (approximation)
xt	Displacement response matrix (Theoretically)
tn	Time vector corresponding to xn
tt	Time vector corresponding to xa

****

```
function [xn,xt,tn,tt]=exerc8(k1,k2,m1,m2,a,dta,n,dt)
M(1,1)=m1; M(1,2)=0; M(2,1)=0; M(2,2)=m2;
K(1,1)=k1+k2; K(1,2)=-k2; K(2,1)=-k2; K(2,2)=k2;
[V,D]=eig(K,M) Phi2=V(:,1)/V(2,1); Phi1=V(:,2)/V(2,2);
w2=sqrt(D(1,1)); w1=sqrt(D(2,2));
MI=inv(M);
tn=0:dt:n*dt; xn=zeros(2,n+1); dxn=zeros(2,n+1);
c=-MI*K;
for i=1:n,
if (tn(i)¡=dta) f=[a/dta 0]'; else f=[0 0]'; end;
```

```
k1=0.5*dt*(c*xn(1:2,i)+MI*f*tn(i));
K=0.5*dt*(dxn(1:2,i)+0.5*k1);
k2=0.5*dt*(c*(xn(1:2,i)+K)+MI*f*(tn(i)+dt/2));
k3=0.5*dt*(c*(xn(1:2,i)+K)+MI*f*(tn(i)+dt/2));
L=dt*(dxn(1:2,i)+k3);
k4=0.5*dt*(c*(xn(1:2,i)+L)+MI*f*tn(i+1));
xn(1:2,i+1)=xn(1:2,i)+dt*(dxn(1:2,i)+(k1+k2+k3)/3);
dxn(1:2,i+1)=dxn(1:2,i)+(1/3)*(k1+2*k2+2*k3+k4);
end;
length=n*dt; dt=length/2000;
tt=0:dt:length;
xt=zeros(2,2000);
c1=a*Phi1(1)/(dta*w1^2*Phi1'*M*Phi1);
c2=a*Phi2(1)/(dta*w2^2*Phi2'*M*Phi2);
for i=1:2001,
t1=min(tt(i),dta);
xt(:,i)=Phi1*c1*(t1*cos(w1*(tt(i)-t1))+(1/w1)*(sin(w1*(tt(i)-t1))-sin(w1*tt(i))));
xt(:,i)=xt(:,i)+Phi2*c2*(t1*cos(w2*(tt(i)-t1))+(1/w2)*(sin(w2*(tt(i)-t1))-sin(w2*tt(i))));
end;
******************************************************************
```

```
******************************************************************
 MATLAB     File for MATLAB Exercise 8, Lecture 6
 Function   PLOT8.M
 Purpose    To plot the displacement response for a 2DOF
            system in MATLAB Exercise 8, lecture 6. Theo-
            retical response versus numerical approx.
 Call       plot8(t1,t2,t3,x1,x2,x3,tt,xt)
 t1         Time vector (1. Approximation)
 t2         Time vector (2. Approximation)
 t3         Time vector (3. Approximation)
 x1         Displacement matrix (1. Approximation)
 x2         Displacement matrix (2. Approximation)
 x3         Displacement matrix (3. Approximation)
 tt         Time Vector (Theoretical)
 xt         Displacement matrix theoretically
 Return     Graphical plots
******************************************************************
function plot8(t1,t2,t3,x1,x2,x3,tt,xt)
clg;
subplot(3,1,1),plot(t1,x1(1,:),tt,xt(1,:),':');
ylabel('a) Disp. [m]');
title('Displacement 1. Mode - 3 different approximations')
grid;
subplot(3,1,2),plot(t2,x2(1,:),tt,xt(1,:),':');
ylabel('b) Disp. [m]');
grid;
subplot(3,1,3),plot(t3,x3(1,:),tt,xt(1,:),':');
ylabel('c) Disp. [m]');
grid;
xlabel('Time [sec]')
pause; clg;
subplot(3,1,1),plot(t1,x1(2,:),tt,xt(2,:),':');
ylabel('a) Disp. [m]');
title('Displacement 2. Mode - 3 different approximations')
grid;
subplot(3,1,2),plot(t2,x2(2,:),tt,xt(2,:),':');
ylabel('b) Disp. [m]');
grid;
subplot(3,1,3),plot(t3,x3(2,:),tt,xt(2,:),':');
ylabel('c) Disp. [m]');
grid;
xlabel('Time [sec]')
******************************************************************
```

Chapter 7

Lecture 7

Problem 1 and 2 of lecture 7 are solved theoretically. The solutions to these problems can be found in Nielsen [2]. Both problems are considered as essential. It is recommended the groups to solve one of the problems in common. The other problem should be solved individually. The individually solved problem is replaced by the following MATLAB exercise. MATLAB exercise 9 is an extension of MATLAB exercise 8.

7.1 MATLAB Exercise 9

The system in problem 1 of lecture 6 is considered again. This system is not undamped as in problem 1 of lecture 6. The system and the excitation is shown in figure 7.1 and 7.2.

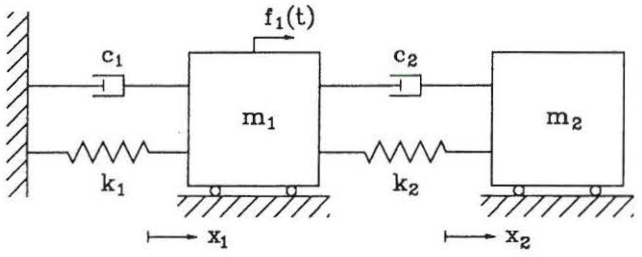


Figure 7.1: *Linear viscous damped two degree of freedom system.*

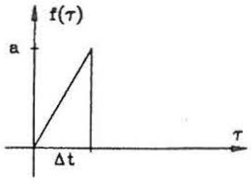


Figure 7.2: *Excitation subjected to mass m_1.*

The equations of motion becomes:

$$\begin{aligned} &\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{C}\dot{\mathbf{x}}(t) + \mathbf{K}\mathbf{x}(t) = \mathbf{f}(t)\,, \qquad t>0 \\ &\mathbf{x}(0) = \mathbf{0}\,, \quad \dot{\mathbf{x}}(0) = 0 \end{aligned} \tag{7.1}$$

Where:

$$\mathbf{M} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \mathbf{C} = \begin{bmatrix} c_1+c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix} \mathbf{K} = \begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \mathbf{f}(t) = \begin{bmatrix} f(t) \\ 0 \end{bmatrix} \tag{7.2}$$

The data for the system are:

$$\frac{k_1}{m_1} = 4\pi^2\frac{N}{mkg} m_1 = 1000kg \; \frac{k_2}{m_2} = 4\pi^2\frac{N}{mkg} \; m_2 = 100kg \; c_1 = 0.04\pi\frac{kg}{s} \; c_2 = 0.08\pi\frac{kg}{s} \tag{7.3}$$

1: Show, that a theoretically solution by expansion in undamped modal coordinates is impossible.
2: Determine the damped mode shapes $\Phi^{(j)}$ and the characteristic values λ_j of the system. (Rewrite to a linear eigenvalue problem of order 4.)
3: Determine the response of the system by expansion in damped modal coordinates and compare the solution with a numerical calculation using a 4*th* order Runge-Kutta scheme.

7.1.1 MATLAB Solution

Ad 1) To solve the linear differential equation in (7.1) by expansion in undamped modal coordinates the following decoubling condition should prevail.

$$\mathbf{\Phi}^{(i)}\mathbf{C}\mathbf{\Phi}^{(j)} = \begin{cases} 0 & i \neq j \\ 2\zeta_i\omega_i M_i & i=j \end{cases} \tag{7.4}$$

Define the following damping coefficients:

$$\frac{\mathbf{\Phi}^{(i)}\mathbf{C}\mathbf{\Phi}^{(j)}}{2\sqrt{\omega_i\omega_j M_i M_j}} = \begin{cases} \zeta_{ij} & j \neq i \\ \zeta_i & i=j \end{cases} \tag{7.5}$$

The undamped mode shapes and the eigenfrequencies are determined in problem 1 of lecture 6.

$$\omega^2 = \frac{k_1 m_1 + k_2(m_1+m_2) \pm \sqrt{(k_1^2 m_2^2 + k_2^2(m_1+m_2)^2 + 2k_1k_2m_2(m_2-m_1))}}{2m_1m_2} \tag{7.6}$$

$$\mathbf{\Phi}^{(i)} = \begin{bmatrix} 1-\omega_i^2\frac{m_2}{k_2} \\ 1 \end{bmatrix} \tag{7.7}$$

The damping ratios can be calculated theoretically. Alternatively it is choosen to apply a numerical calculation using a MATLAB function (exerc9a.m). If the following order are given and the MATLAB function has been implemented the decoubling matrix appears on the screen. An explanation to the input numbers can be found in the head of the MATLAB file.

*exerc9a(4000*pi^2,400*pi^2,1000,100,0.04*pi,0.08*pi);*

$$\begin{bmatrix} \zeta_1 & \zeta_{12} \\ \zeta_{21} & \zeta_2 \end{bmatrix} = 10^{-3}\cdot\begin{bmatrix} 0.0770 & 0.0938 \\ 0.0938 & 0.1403 \end{bmatrix} \tag{7.8}$$

As seen $\zeta_{12} = \zeta_{21} \neq 0$, so modal decoupling does not take place. Notice that $\zeta_{12} = 0.0938 < \sqrt{\zeta_1\zeta_2} = \sqrt{(0.0770\cdot 0.1403\cdot 10^{-6})}$. This condition must identically be fullfilled for a symmetric and dissipative damping matrix, see Nielsen [1], p. 72, (3-180). If the theoretically solution by expansion in undamped modal coordinates should be exact, the decoubling condition should be fulfilled. This is not the case for the system. A solution by expansion in undamped modal coordinates can only be approximate.

Ad 2) To calculate the damped mode shapes $\mathbf{\Phi}^{(j)}$ and the characteristic values λ_j the linear differential equations (7.1) is rewritten to state variables. This reduces the eigenvalue problem to a linear eigenvalue problem of dimension 4.

$$\mathbf{A}\dot{\mathbf{z}} + \mathbf{B}\mathbf{z} = \mathbf{0} \tag{7.9}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{C} & \mathbf{M} \\ \mathbf{M} & \mathbf{0} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & -\mathbf{M} \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix} \tag{7.10}$$

The linear eigenvalue problem reduces to:

$$(\lambda\mathbf{A} + \mathbf{B})\mathbf{\Psi} = \mathbf{0} \tag{7.11}$$

If (7.11) is solved the solutions $\mathbf{\Psi}$ whould have the form:

$$\mathbf{\Psi} = \begin{bmatrix} \mathbf{\Phi} \\ \lambda\mathbf{\Phi} \end{bmatrix} \tag{7.12}$$

If the following order are given and the MATLAB function (exerc9b.m) has been implemented the eigenvalues λ and the eigenvectors $\mathbf{\Psi}$ appears on the screen. An explanation to the input numbers can be found in the head of the MATLAB file. The eigenvectors are normalized to have $\Phi_2^{(j)} = 1$.

*[V,D]=exerc9b(4000*pi^2,400*pi^2,1000,100,0.04*pi,0.08*pi);*

Eigenvalues

$$\begin{bmatrix} -0.0004 + 5.3678i & 0 & 0 & 0 \\ 0 & -0.0004 - 5.3678i & 0 & 0 \\ 0 & 0 & -0.0001 + 7.3547i & 0 \\ 0 & 0 & 0 & -0.0001 - 7.3547i \end{bmatrix} \tag{7.13}$$

Eigenvectors

$$\begin{bmatrix} 0.2702 + 0.0001i & 0.2702 - 0.0001i & -0.3702 + 0.0003i & -0.3702 - 0.0003i \\ 1.0 & 1.0 & 1.0 & 1.0 \\ -0.0008 + 1.4501i & -0.0008 - 1.4501i & -0.0015 - 2.7224i & -0.0015 + 2.7224i \\ -0.0004 + 5.3678i & -0.0004 - 5.3678i & -0.0010 + 7.3547i & -0.0010 - 7.3547i \end{bmatrix} \tag{7.14}$$

Notice how both the eigenvalues and the eigenvectors turns out in complex conjugate pairs.

Ad 3) Especially the displacement response can be found by:

$$\mathbf{x}(t) = 2\sum_{j=1}^{2} Re(\mathbf{\Phi}^{(j)} q_j(t)) \tag{7.15}$$

Where $\mathbf{\Phi}^{(j)}$ are the damped mode shapes and damped modal coordinates are:

$$\begin{aligned}
q_j(t) &= e^{\lambda_j t}\int_0^t e^{-\lambda_j\tau}\tfrac{1}{m_j}\mathbf{\Phi}^{(j)T}\mathbf{f}(\tau)d\tau \\
&= e^{\lambda_j t}\int_0^t e^{-\lambda_j\tau}\tfrac{1}{m_j}\Phi_1^{(j)}f(\tau)d\tau \\
&= \tfrac{e^{\lambda_j t}}{m_j}\Phi_1^{(j)}\int_0^{min(t,\Delta t)} e^{-\lambda_j\tau}\tfrac{a\tau}{\Delta t}d\tau \\
&= \tfrac{ae^{\lambda_j t}}{\Delta t m_j}\Phi_1^{(j)}\left(\left[\tfrac{e^{-\lambda_j\tau}}{-\lambda_j}\tau\right]_0^{min(t,\Delta t)} + \tfrac{1}{\lambda_j}\int_0^{min(t,\Delta t)} e^{-\lambda_j\tau}d\tau\right) \\
&= \tfrac{ae^{\lambda_j t}}{\Delta t m_j}\Phi_1^{(j)}\left(\left[\tfrac{e^{-\lambda_j\tau}}{-\lambda_j}\tau\right]_0^{min(t,\Delta t)} - \tfrac{1}{\lambda_j^2}\left[e^{-\lambda_j\tau}\right]_0^{min(t,\Delta t)}\right) \\
&= -\tfrac{ae^{\lambda_j t}}{\Delta t m_j\lambda_j}\Phi_1^{(j)}\left(e^{-\lambda_j min(t,\Delta t)}min(t,\Delta t) + \tfrac{1}{\lambda_j}(e^{-\lambda_j min(t,\Delta t)} - 1)\right)
\end{aligned} \tag{7.16}$$

Where m_j is the damped modal mass given by (see Nielsen [1], p. 77, (3-212)).

$$\begin{aligned}
m_j &= \mathbf{\Phi}^{(j)T}\mathbf{C}\mathbf{\Phi}^{(j)} + 2\lambda_j\mathbf{\Phi}^{(j)T}\mathbf{M}\mathbf{\Phi}^{(j)} \\
&= (c_1+c_2)(\Phi_1^{(j)})^2 - 2c_1\Phi_1^{(j)} + c_2 + 2\lambda_j((\Phi_1^{(j)})^2 m_1 + m_2)
\end{aligned} \tag{7.17}$$

The damped mode shapes and the eigenvalues of the system are calculated in the solution to question 2. This means that if the last statement of (7.16) is used in (7.15) an analytical solution to the displacement response of the system is obtained. To compare this theoretical solution with a numerical approximation using a *4th* order Runge-Kutta scheme the MATLAB function (exerc9c.m) is used. An explanation of the input numbers can be found in the head of the MATLAB file. If the following order are given figure 7.3 and figure 7.4 will appear on the screen.

```
m1=1000; m2=100; c1=0.04*pi; c2=0.08*pi; k1=4000*pi^2; k2=400*pi^2;
[t1,x1,tt,xt]=exerc9c(V,D,m1,m2,c1,c2,k1,k2,100,0.5,25,0.02);
[t2,x2,tt,xt]=exerc9c(V,D,m1,m2,c1,c2,k1,k2,100,0.5,50,0.01);
[t3,x3,tt,xt]=exerc9c(V,D,m1,m2,c1,c2,k1,k2,100,0.5,1000,0.0005);
plot9(t1,t2,t3,x1,x2,x3,tt,xt);
```

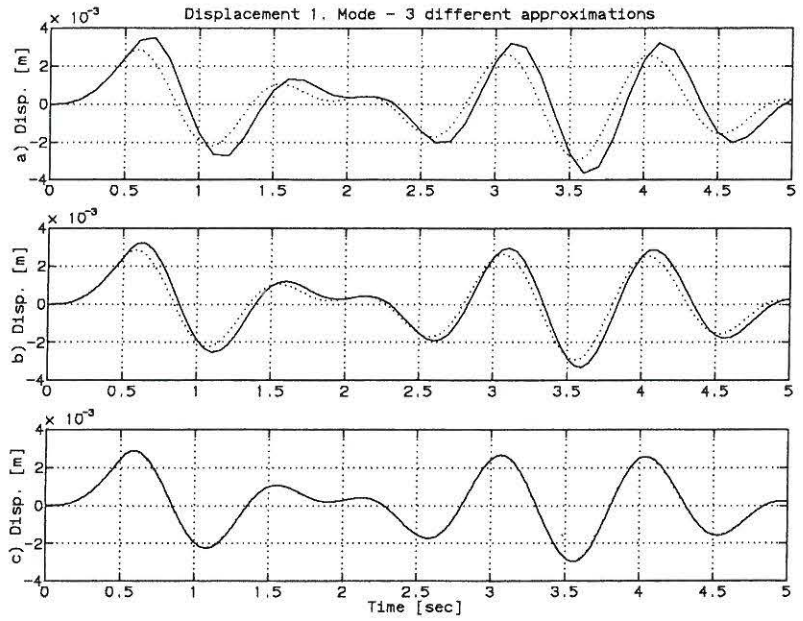


Figure 7.3: *Theoretical displacement response* [· · ·] *and 3 different numerical approximations [—] of first degree of freedom. a)* $\Delta t = \frac{T_1}{12} = \frac{T_2}{10}$ *b)* $\Delta t = \frac{T_1}{24} = \frac{T_2}{20}$, *c)* $\Delta t = \frac{T_1}{240} = \frac{T_2}{200}$.

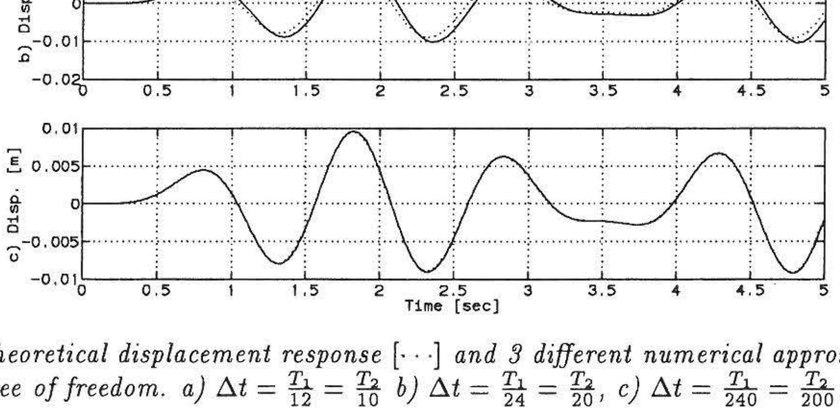


Figure 7.4: *Theoretical displacement response* [· · ·] *and 3 different numerical approximations [—] of second degree of freedom. a)* $\Delta t = \frac{T_1}{12} = \frac{T_2}{10}$ *b)* $\Delta t = \frac{T_1}{24} = \frac{T_2}{20}$, *c)* $\Delta t = \frac{T_1}{240} = \frac{T_2}{200}$.

As seen the numerical solution based on the Runge-Kutta scheme requires very small time steps before converging is achieved. A typical time step of $\Delta t = \frac{T_2}{200}$ should be compared with a time step of $\Delta t = \frac{T_0}{20}$ of a SDOF system. As n (number of DOF) is further increased the time step is even further reduced. Hence the Runge-Kutta scheme becomes increasingly ineffective at large values of n, and should be replaced by other integration schemes.

7.1.2 MATLAB Files

```
*****************************************************************
  MATLAB     File for MATLAB Exercise 9, lecture 7
  Function   EXERC9A.M
  Purpose    To calculate the decoubling matrix
  Call       exerc9a(k1,k2,m1,m2,c1,c2)
  k1         Stifness of linear spring 1
  k2         stifness of linear spring 2
  m1         Mass of mass 1
  m2         Mass of mass 2
  c1         Damping constant 1
  c2         Damping constant 2
  Return     Damping ratio matrix
*****************************************************************
function exerc9a(k1,k2,m1,m2,c1,c2)
konst=sqrt(k1^2*m2^2+k2^2*(m1+m2)^2+2*k1*k2*m2*(m2-m1));
w1=(k1*m2+k2*(m1+m2)-konst)/(2*m1*m2);
w2=(k1*m2+k2*(m1+m2)+konst)/(2*m1*m2);
w1=sqrt(w1); w2=sqrt(w2);
M(1,1)=m1; M(1,2)=0; M(2,1)=0; M(2,2)=m2;
Phi1=[1-w1^2*m2/k2 1]'; Phi2=[1-w2^2*m2/k2 1]';
C(1,1)=c1+c2; C(1,2)=-c2; C(2,1)=-c2; C(2,2)=c2;
decou(1,1)=Phi1'*C*Phi1;
decou(1,2)=Phi1'*C*Phi2;
decou(2,1)=Phi2'*C*Phi1;
decou(2,2)=Phi2'*C*Phi2;
mm1=Phi1'*M*Phi1; mm2=Phi2'*M*Phi2;
z11=decou(1,1)/(2*w1*mm1);
z12=decou(1,2)/(2*sqrt(w1*w2*mm1*mm2));
z21=decou(2,1)/(2*sqrt(w1*w2*mm1*mm2));
z22=decou(2,2)/(2*w2*mm2);
damp(:,1)=[z11 z21]'; damp(:,2)=[z12 z22]';
dampingratios=damp
*****************************************************************
```

```
*****************************************************************
  MATLAB     File for MATLAB Exercise 9, lecture 7.
  Function   EXERC9B.M
  Purpose    To calculate the eigenvalues and eigenvectors
             of linear state variable eigenvalue problem.
  Call       exerc9b(k1,k2,m1,m2,c1,c2)
  k1         Stifness of linear spring 1
  k2         stifness of linear spring 2
  m1         Mass of mass 1
  m2         Mass of mass 2
  c1         Damping constant 1
  c2         Damping constant 2
  Return     Eigenvalues and eigenvectors
  V          Normalized eigenvectors
  D          Eigenvalues
*****************************************************************
function [V,D]=exerc9b(k1,k2,m1,m2,c1,c2)
M(1,1)=m1; M(1,2)=0; M(2,1)=0; M(2,2)=m2;
C(1,1)=c1+c2; C(1,2)=-c2; C(2,1)=-c2; C(2,2)=c2;
K(1,1)=k1+k2; K(1,2)=-k2; K(2,1)=-k2; K(2,2)=k2;
O=zeros(2,2);
```

```
A1=[C M]'; A2=[M O]'; A=[A1 A2];
B1=[K O]'; B2=[O -M]'; B=[B1 B2];
[V,D]=eig(B,-A);
V(:,1)=V(:,1)/V(2,1); V(:,2)=V(:,2)/V(2,2);
V(:,3)=V(:,3)/V(2,3); V(:,4)=V(:,4)/V(2,4);
eigenvalues=D
Eigenvectors=V
******************************************************************
```

```
******************************************************************
 MATLAB     File for MATLAB Exercise 9, lecture 7.
 Function   EXERC9C.M
 Purpose    To calculate the displacement response of the
            system. Both the theoretical solution and a
            numerical solution using a 4th order Runge-
            Kutta scheme is used.
 Call       exerc9c(V,D,m1,m2,c1,c2,k1,k2,a,dta,n,dt)
 V          Normalized complex mode shape vectors
 D          Complex eigenvalues
 m1         Mass of mass 1
 m2         Mass of mass 2
 c1         Damping constant 1
 c2         Damping constant 2
 a          Maximal excitation
 dta        Duration of excitation
 n          Number of points in approximation
 dt         Time distance between approximations
 Return
 tn         Time vector for numerical approximation
 xn         Numerical approx. matrix for displacement
 tt         Time vector for theoretical solution
 xt         Displacement matrix for theoretical solution
******************************************************************
function [tn,xn,tt,xt]=exerc9c(V,D,m1,m2,c1,c2,k1,k2,a,dta,n,dt)
Phi1=V(1:2,2); Phi2=V(1:2,1);
L1=D(2,2); L2=D(1,1);
M(1,1)=m1; M(1,2)=0; M(2,1)=0; M(2,2)=m2;
C(1,1)=c1+c2; C(1,2)=-c2; C(2,1)=-c2; C(2,2)=c2;
K(1,1)=k1+k2; K(1,2)=-k2; K(2,1)=-k2; K(2,2)=k2;
MI=inv(M); F=[a/dta 0]';
tn=0:dt:n*dt; xn=zeros(2,n+1); dxn=zeros(2,n+1);
for i=1:n,
if (tn(i)¡=dta), F1=F; else F1=[0 0]'; end;
k1=0.5*dt*(-MI*C*dxn(:,i)-MI*K*xn(:,i)+MI*F1*tn(i));
KK=0.5*dt*(dxn(:,i)+0.5*k1);
k2=0.5*dt*(-MI*C*(dxn(:,i)+k1)-MI*K*(xn(:,i)+KK)+MI*F1*(tn(i)+0.5*dt));
k3=0.5*dt*(-MI*C*(dxn(:,i)+k2)-MI*K*(xn(:,i)+KK)+MI*F1*(tn(i)+0.5*dt));
LL=dt*(dxn(:,i)+k3);
k4=0.5*dt*(-MI*C*(dxn(:,i)+2*k3)-MI*K*(xn(:,i)+LL)+MI*F1*(tn(i)+dt));
xn(:,i+1)=xn(:,i)+dt*(dxn(:,i)+(1/3)*(k1+k2+k3));
dxn(:,i+1)=dxn(:,i)+(1/3)*(k1+2*k2+2*k3+k4);
end;
md1=(c1+c2)*Phi1(1)^2-2*c2*Phi1(1)+c2+2*L1*(Phi1(1)^2*m1+m2);
md2=(c1+c2)*Phi2(1)^2-2*c2*Phi2(1)+c2+2*L2*(Phi2(1)^2*m1+m2);
C1=a*Phi1(1)/(dta*md1*L1); C2=a*Phi2(1)/(dta*md2*L2);
dtt=n*dt/1000; tt=0:dtt:n*dt; xt=zeros(2,1001);
for i=1:1001,
t=min(tt(i),dta);
q1=-C1*exp(L1*tt(i))*(exp(-L1*t)*t+(1/L1)*(exp(-L1*t)-1));
q2=-C2*exp(L2*tt(i))*(exp(-L2*t)*t+(1/L2)*(exp(-L2*t)-1));
xt(:,i)=2*(real(Phi1*q1)+real(Phi2*q2));
end;
******************************************************************
```

```
*******************************************************************
 MATLAB     File for MATLAB Exercise 9, Lecture 7
 Function   PLOT9.M
 Purpose    To plot the displacement response for a 2DOF
            system in MATLAB Exercise 9, lecture 7. Theo-
            retical response versus numerical approx.
 Call       plot9(t1,t2,t3,x1,x2,x3,tt,xt)
 t1         Time vector (1. Approximation)
 t2         Time vector (2. Approximation)
 t3         Time vector (3. Approximation)
 x1         Displacement matrix (1. Approximation)
 x2         Displacement matrix (2. Approximation)
 x3         Displacement matrix (3. Approximation)
 tt         Time Vector (Theoretical)
 xt         Displacement matrix theoretically
 Return     Graphical plots
*******************************************************************
function plot9(t1,t2,t3,x1,x2,x3,tt,xt)
clg;
subplot(3,1,1),plot(t1,x1(1,:),tt,xt(1,:),':');
ylabel('a) Disp. [m]');
title('Displacement 1. Mode - 3 different approximations')
grid;
subplot(3,1,2),plot(t2,x2(1,:),tt,xt(1,:),':');
ylabel('b) Disp. [m]');
grid;
subplot(3,1,3),plot(t3,x3(1,:),tt,xt(1,:),':');
ylabel('c) Disp. [m]');
grid;
xlabel('Time [sec]')
pause; clg;
subplot(3,1,1),plot(t1,x1(2,:),tt,xt(2,:),':');
ylabel('a) Disp. [m]');
title('Displacement 1. Mode - 3 different approximations')
grid;
subplot(3,1,2),plot(t2,x2(2,:),tt,xt(2,:),':');
ylabel('b) Disp. [m]');
grid;
subplot(3,1,3),plot(t3,x3(2,:),tt,xt(2,:),':');
ylabel('c) Disp. [m]');
grid;
xlabel('Time [sec]')
*******************************************************************
```

Chapter 8

Lecture 8

Problem 3 of lecture 8 is solved theoretically. Problem 1 and 2 of lecture 8 are solved individually. The solutions to these problems can be found in Nielsen [1]. The following MATLAB exercises replaces the problems, which are solved individually. MATLAB exercise 10 is an extension of MATLAB exercise 8 and 9.

8.1 MATLAB Exercise 10

The systems in MATLAB exercise 8 and 9 are considered again.

1) Compare the theoretical solution of the system in MATLAB exercise 8 with a solution, where only 1 *undamped* modal coordinat is used (system reduction).

2) Compare the theoretical solution of the system in MATLAB exercise 9 with a solution, where only 1 *damped* modal coordinat is used (system reduction).

8.1.1 MATLAB Solution

If the second mode can be considered as not being dynamically excited, then only the quasi-static response of this mode is taken into account.

Ad 1) The response of the undamped two degree of freedom system in MATLAB exercise 8 using system reduction is:

$$\mathbf{x}(t) \;=\; q_1(t)\boldsymbol{\Phi}^{(1)} + \left(\mathbf{K}^{-1} - \frac{1}{\omega_1^2 M_1}\boldsymbol{\Phi}^{(1)}\boldsymbol{\Phi}^{(1)T}\right)\mathbf{f}(t) \tag{8.1}$$

If the quasi-static response is ignored the response becomes:

$$\mathbf{x}(t) \;=\; q_1(t)\boldsymbol{\Phi}^{(1)} \tag{8.2}$$

The loading $\mathbf{f}(t)$ is caused by a triangular impulse of duration $\Delta t = \frac{1}{2}T_1$ on the first mass. Hence for $t > \Delta t$ $\mathbf{f}(t) \equiv 0$ and (8.1) and (8.2) coincide. The discontinuity of (8.1) at the tune $t = \Delta t$ is caused by the discontinuity of the quasi-static part of the response due to the jump of the impulse. None of the approximations are good since the 2nd mode is excited significantly. The theoretically solution by expansion in undamped modal coordinates is known from MATLAB exercise 8. This solution and the approximation given by (8.1) are implemented in the MATLAB function below (exerc10a.m). If the following order are given.

*exerc10a(4000*pi^2,400*pi^2,1000,100,1,0.5,100,0.05);*

Then figure 8.1 will appear on the screen.

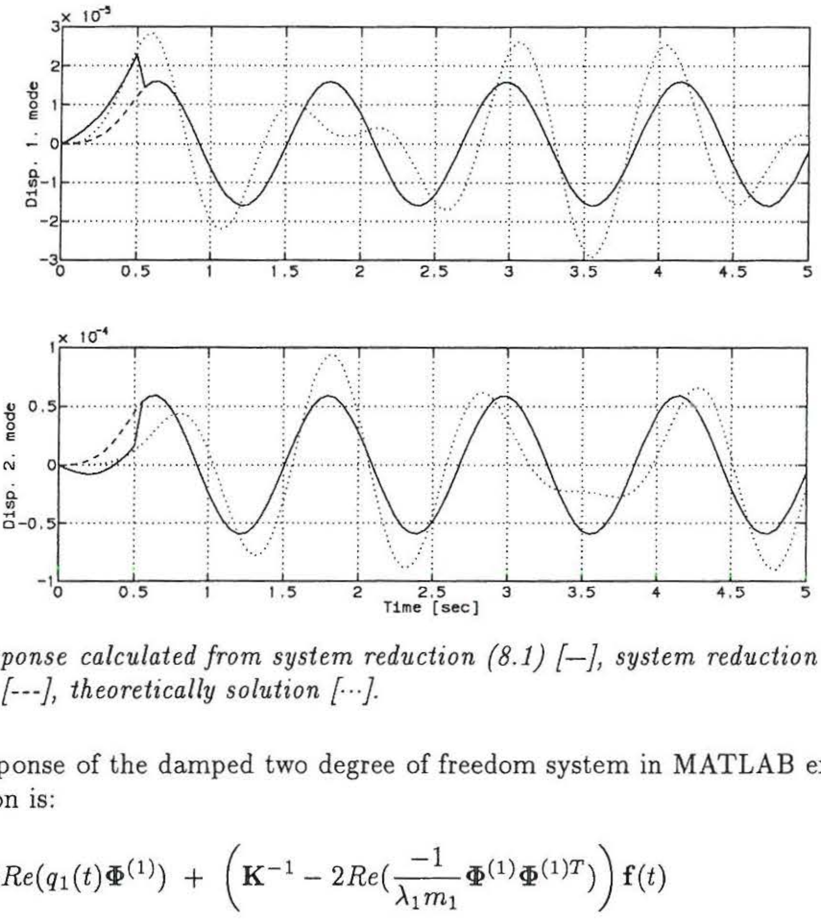


Figure 8.1: *Response calculated from system reduction (8.1) [—], system reduction without quasi-static response [---], theoretically solution [···].*

Ad 2) The response of the damped two degree of freedom system in MATLAB exercise 9 using system reduction is:

$$\mathbf{x}(t) \;=\; 2Re(q_1(t)\mathbf{\Phi}^{(1)}) \;+\; \left(\mathbf{K}^{-1} - 2Re(\frac{-1}{\lambda_1 m_1}\mathbf{\Phi}^{(1)}\mathbf{\Phi}^{(1)T})\right)\mathbf{f}(t) \qquad (8.3)$$

If the quasi-static response is ignored the response becomes:

$$\mathbf{x}(t) \;=\; 2Re(q_1(t)\mathbf{\Phi}^{(1)}) \qquad (8.4)$$

The theoretically solution by expansion in undamped modal coordinates is known from MATLAB exercise 9. This solution and the approximation given by (8.2) are implemented in the MATLAB function below (exerc10b.m). If the following order are given.

*exerc10b(1000,100,0.04*pi,0.08*pi,4000*pi^2,400*pi^2, 1,0.5,100,0.05);*

Then figure 8.2 will appear on the screen.

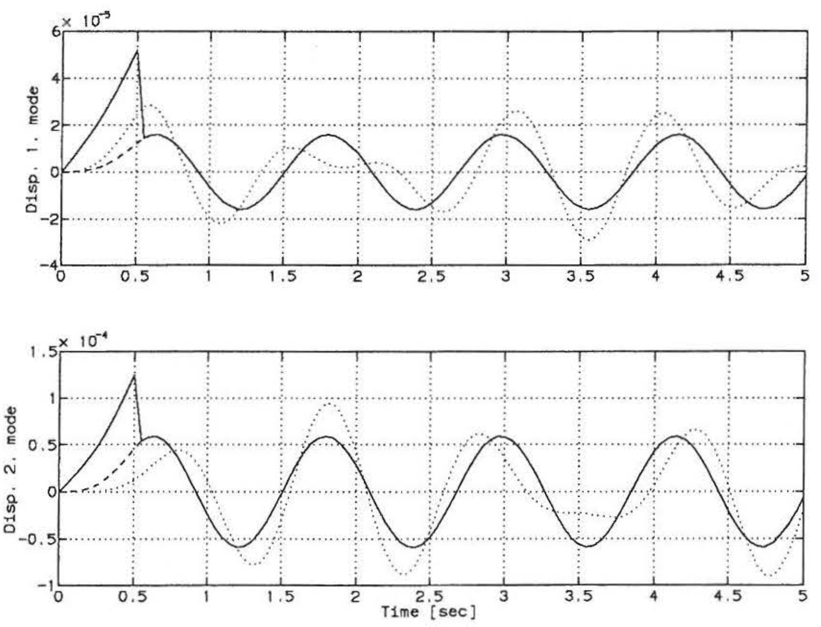


Figure 8.2: *Response calculated from system reduction (8.3) [—], system reduction without quasi-static correction (8.4) [---], theoretically solution [···].*

8.1.2 MATLAB Files

****

MATLAB	File for MATLAB Exercise 10, Lecture 8
Function	EXERC10a.M
Purpose	To calculate the response of the 2DOF system described in problem 1, lecture 6. The response is calculated using expansion in undamped modal coordinats and a approximation using system reduction.
Call	exerc10a(k1,k2,m1,m2,a,dta,n,dt)
k1	Stifness of first linear spring
k2	Stifness of second linear spring
m1	Mass 1
m2	Mass 2
a	Maximal impulse force
dta	Time lengt of impulse force
n	Number of points in response calculation
dt	Time distance between points
Return	Graphical plot - Comparison

****

```
function exerc10a(k1,k2,m1,m2,a,dta,n,dt)
M(1,1)=m1; M(1,2)=0; M(2,1)=0; M(2,2)=m2;
K(1,1)=k1+k2; K(1,2)=-k2; K(2,1)=-k2; K(2,2)=k2;
[V,D]=eig(K,M); Phi2=V(:,1)/V(2,1); Phi1=V(:,2)/V(2,2);
w2=sqrt(D(1,1)); w1=sqrt(D(2,2));
MI=inv(M); KI=inv(K);

t=0:dt:n*dt;
xt=zeros(2,length(t));
c1=a*Phi1(1)/(dta*w1^2*Phi1'*M*Phi1;
c2=a*Phi2(1)/(dta*w2^2*Phi2'*M*Phi2;
for i=1:length(t),
t1=min(t(i),dta);
xt(:,i)=Phi1*c1*(t1*cos(w1*(t(i)-t1))+(1/w1)*(sin(w1*(t(i)-t1))-sin(w1*t(i))));
xt(:,i)=xt(:,i)+Phi2*c2*(t1*cos(w2*(t(i)-t1))+(1/w2)*(sin(w2*(t(i)-t1))-sin(w2*t(i))));
end;

x=zeros(2,length(t));
c3=Phi1*Phi1'/(w1^2*(Phi1'*M*Phi1));
f=[a/dta 0]';
for i=1:length(t),
t1=min(t(i),dta);
if t(i)¡=dta, f1=f; else f1=[0 0]'; end;
x(:,i)=Phi1*c1*(t1*cos(w1*(t(i)-t1))+(1/w1)*(sin(w1*(t(i)-t1))-sin(w1*t(i))));
x1(:,i)=x(:,i);
x(:,i)=x(:,i)+(KI-c3)*f1*t(i);
end;

clg;
subplot(211),plot(t,xt(1,:),':',t,x(1,:),t,x1(1,:),'-');
ylabel('Disp. 1. mode');
grid;
subplot(212),plot(t,xt(2,:),':',t,x(2,:),t,x1(2,:),'-');
ylabel('Disp. 2. mode');
xlabel('Time [sec]');
grid;
*****************************************************************


*****************************************************************
  MATLAB     File for MATLAB Exercise 10, lecture 8
  Function   EXERC10B.M
  Purpose    To calculate the displacement response of the
             system. Both the theoretical solution and a
             approximate solution using system reduction
             is made
  Call       exerc10b(m1,m2,c1,c2,k1,k2,a,dta,n,dt)
  m1         Mass of mass 1
  m2         Mass of mass 2
  c1         Damping constant 1
  c2         Damping constant 2
  k1         Stifness constant 1
  k2         Stifness constant 2
  a          Maximal excitation
  dta        Duration of excitation
  n          Number of points in approximation
  dt         Time distance between approximations
  Return     Graphical plot - Comparison
*****************************************************************
function exerc10b(m1,m2,c1,c2,k1,k2,a,dta,n,dt)
M(1,1)=m1; M(1,2)=0; M(2,1)=0; M(2,2)=m2;
C(1,1)=c1+c2; C(1,2)=-c2; C(2,1)=-c2; C(2,2)=c2;
K(1,1)=k1+k2; K(1,2)=-k2; K(2,1)=-k2; K(2,2)=k2;
O=zeros(2,2);
A=[C M; M O]; B=[K O; O -M];
[V,D]=eig(B,-A);
V(:,1)=V(:,1)/V(2,1); V(:,2)=V(:,2)/V(2,2);
V(:,3)=V(:,3)/V(2,3); V(:,4)=V(:,4)/V(4,2);
Phi1=V(1:2,2); Phi2=V(1:2,1);
L1=D(2,2); L2=D(1,1);
MI=inv(M); KI=inv(K); F=[a/dta 0]';
md1=(c1+c2)*Phi1(1)^2-2*c1*Phi1(1)+c2+2*L1*(Phi1(1)^2*m1+m2);
md2=(c1+c2)*Phi2(1)^2-2*c1*Phi2(1)+c2+2*L2*(Phi2(1)^2*m1+m2);
C1=a*Phi1(1)/(dta*md1*L1); C2=a*Phi2(1)/(dta*md2*L2);
t=0:dt:n*dt; xt=zeros(2,length(t));
for i=1:length(t),
t1=min(t(i),dta);
q1=-C1*exp(L1*t(i))*(exp(-L1*t1)*t1+(1/L1)*(exp(-L1*t1)-1));
q2=-C2*exp(L2*t(i))*(exp(-L2*t1)*t1+(1/L2)*(exp(-L2*t1)-1));
xt(:,i)=2*(real(Phi1*q1)+real(Phi2*q2));
end;
for i=1:length(t),
t1=min(t(i),dta);
if (t(i)¡=dta), f=F; else f=[0 0]'; end;
q1=-C1*exp(L1*t(i))*(exp(-L1*t1)*t1+(1/L1)*(exp(-L1*t1)-1));
x(:,i)=2*real(q1*Phi1);
x1(:,i)=x(:,i);
x(:,i)=x(:,i)+(KI-2*real((1/(md1*L1))*(Phi1*Phi1')))*f*t(i);
end;
clg;
subplot(2,1,1),plot(t,x(1,:),t,xt(1,:),':',t,x1(1,:),'–');
ylabel('Disp. 1. mode');
grid;
subplot(2,1,2),plot(t,x(2,:),t,xt(2,:),':',t,x1(2,:),'–');
ylabel('Disp. 2. mode');
xlabel('Time [sec]');
grid;
******************************************************************
```

Chapter 9

Lecture 9

Problem 1 and 2 of lecture 10 is solved theoretically. The solution to these problems can be found in Nielsen [2]. The following MATLAB exercise replaces the problems which are solved individually.

9.1 MATLAB Exercise 11

Consider a continous Bernoilli-Euler beam. The beam is simply supported at both end sections A and B and loaded as shown in figure 9.1.

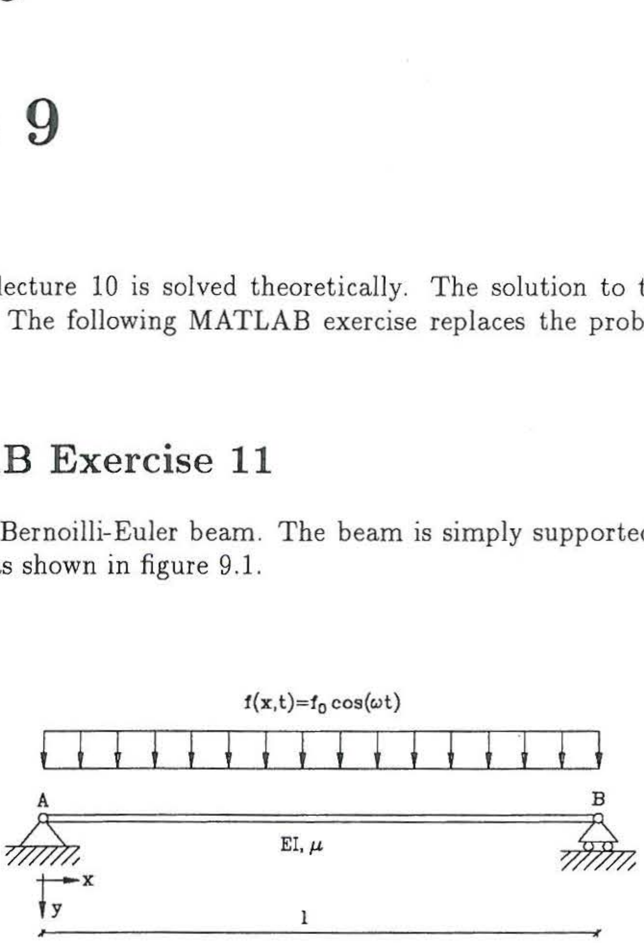


Figure 9.1: *Harmonic loaded continous Bernoulli-Euler beam.*

The beam is excited by the dynamic force per unit length $f(x,t) = f_0\cos(\omega t)$. The beam is undamped and the following data are valid:

$$l = 1m\ , \quad EI = 1Nm^2\ , \quad \mu = \pi^2\frac{kg}{m} \tag{9.1}$$

The initial conditions at time $t = 0$ are:

$$y(x,0) = 0\ , \qquad \dot{y}(x,0) = 0 \tag{9.2}$$

Question: Find the stationary motion of the beam. Animate the motion and try to change the frequency of the harmonic load with values in the vicinity of the 3 lowest natural frequencies of the beam.

9.1.1 MATLAB Solution

The force-balance differential equation of the system is, see Nielsen [1], p. 100, (4-9).

$$\frac{\partial^4 y(x,t)}{\partial x^4} + \mu\frac{\partial^2 y(x,t)}{\partial t^2} = f_0 \cos(\omega t) \tag{9.3}$$

Because the beam is simply supported as shown in figure 9.1 the boundary condition needed to solve (9.3) are:

$$y(0,t) \equiv 0\ , \quad y(l,t) \equiv 0\ , \quad \frac{\partial^2 y(0,t)}{\partial x^2} \equiv 0\ , \quad \frac{\partial^2 y(l,t)}{\partial x^2} \equiv 0 \tag{9.4}$$

In the present case a solution to the stationary displacement field can be obtained on closed form by direct integration of the partial differential equation for the beam element. The alternative is to use expansion in undamped modal coordiantes (series solution). The calculations can be found in Nielsen [2], p. 212 ff. The calcutions are omitted here. The solution is:

$$y(x,t) = U(x)\cdot\cos(\omega t) \tag{9.5}$$

Where:

$$\begin{aligned} U(x) = {} & \tfrac{f_0}{2\omega^2\mu}\left(-2 + \left(\tfrac{1-\cos(\lambda)}{\sin(\lambda)}\right)\sin\left(\lambda\tfrac{x}{l}\right) + \cos\left(\lambda\tfrac{x}{l}\right) + \right. \\ & \left. \left(\tfrac{1-\cosh(\lambda)}{\sinh(\lambda)}\right)\sinh\left(\lambda\tfrac{x}{l}\right) + \cosh\left(\lambda\tfrac{x}{l}\right)\right) \end{aligned} \tag{9.6}$$

$$\lambda^4 = \frac{\mu\omega^2 l^4}{EI} \tag{9.7}$$

The natural frequencies of the beam is given by, Nielsen [1], p. 111, (4-45):

$$\omega_i = \frac{i^2\pi^2}{l^2}\sqrt{\frac{EI}{\mu}}\ , \qquad i = 1,2,3,\ldots \tag{9.8}$$

Inserting the data from (9.1) in (9.8) yields:

$$\omega_i = i^2\cdot\pi\ , \qquad i = 1,2,3,\ldots \tag{9.9}$$

If the following order are given and the MATLAB function (exercl1.m) are implemented, the animation of the stationary motion of the beam will appear on the screen. Because of the limiting speed of the hardware the motion of the beam is animated in slow motion.

exercl1(?,1,10,pi^2,1,1);

9.1.2 MATLAB Files

****

MATLAB	File for MATLAB Exercise 11, Lecture 9
Function	EXERC11.M
Purpose	To animate the response of a simply supported Bernoulli-Euler beam load by a harmonic force. The beam is undamped and the solution is obtained in close form.
Call	exerc11(w,f0,t,mu,laen,EI)
w	Frequency of the harmonic force
f0	Amplitude of harmonic force
n	Number of load cycles to animate
mu	Mass per unit length of beam
laen	Length of beam
EI	Material constants of beam
Return	Graphical plots - Animation

```
****************************************************************
function [y]=exercl1(w,f0,n,mu,laen,EI);
f=w/(2*pi); dt=1/(20*f);
y(41,20)=zeros;
dl=laen/40; L=sqrt(sqrt(mu*w^2*laen^4/EI));
C1=(1-cos(L))/sin(L); C2=(1-cosh(L))/sinh(L); C3=f0/(2*w^2*mu);
C5=dl*L/laen;
for i=1:20,
t=(i-1)*dt;
C4=C3*cos(w*t);
for j=1:41,
l=(j-1)*C5;
y(j,i)=C4*(-2+C1*sin(l)+cos(l)+C2*sinh(l)+cosh(l));
end;
end;
ymax=max(max(y)); ymin=min(min(y));
xmax=laen+0.1*laen;
xmin=-0.1*laen;
ymax=2*ymax; ymin=2*ymin;
for i=1:41, x(i)=(i-1)*dl; end;
for j=1:n,
for i=1:20,
plot(x,y(:,i));
axis([xmin xmax ymin ymax]);
title('Animation of stationary response');
xlabel('Length of beam'); ylabel('Response of beam');
pause(0.5);
end;
end;
****************************************************************
```

Chapter 10

Lecture 11

Problem 1 June 16, 1989 are solved theoretically. The solutions to the problem can be found in Nielsen [2]. The following MATLAB exercise should also be calculated.

10.1 MATLAB Exercise 12

Consider a continuous 2-storey frame structure. The structure is shown in figure 10.1

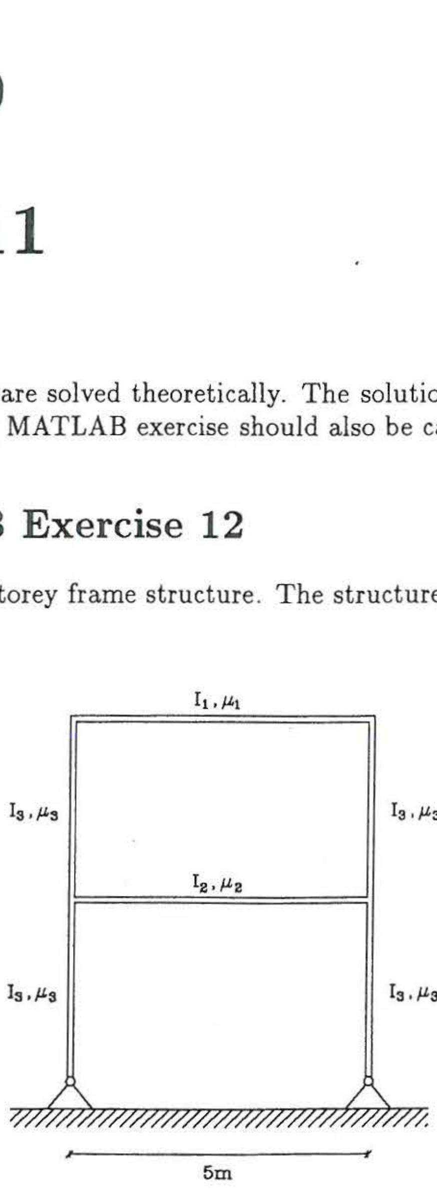


Figure 10.1: *2-storey frame structure.*

The following data is valid for the stifness and mass of the frame structure.

$$\begin{array}{lll} E_1 = 2.0 \cdot 10^{11} \frac{N}{m^2} & E_2 = E_1 & E_3 = E_1 \\ I_1 = 1.945 \cdot 10^{-5} m^4 & I_2 = 3.89 \cdot 10^{-5} m^4 & I_3 = 2.63 \cdot 10^{-5} m^4 \\ \mu_1 = 1000 \frac{kg}{m} & \mu_2 = 2000 \frac{kg}{m} & \mu_3 = 46 \frac{kg}{m} \end{array} \tag{10.1}$$

Question: Determine the undamped eigenfrequencies and eigen modes for the following number of system nodes, 6,12,24.

The theoretical solution for the 4 lowest eigenfrequencies are:

$$f_1 = 1.05Hz, \quad f_2 = 4.31Hz, \quad f_3 = 6.19Hz \quad f_4 = 7.17Hz \tag{10.2}$$

10.1.1 MATLAB Solution

The following orders calculates and plots the eigenfrequencies and the mode shapes of the structure when 6,12 and 24 degrees of freedom are used.
6 Node approximation.
exerc12a
F=[1.0281 4.2125 8.2186 11.3539]
plot12a(X,T,Shape)

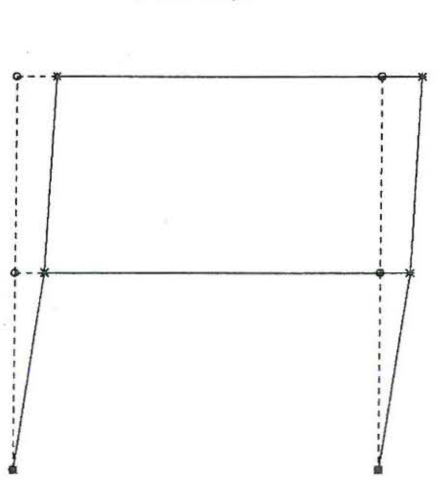


Figure 10.2: *6 Node approximation, 1. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

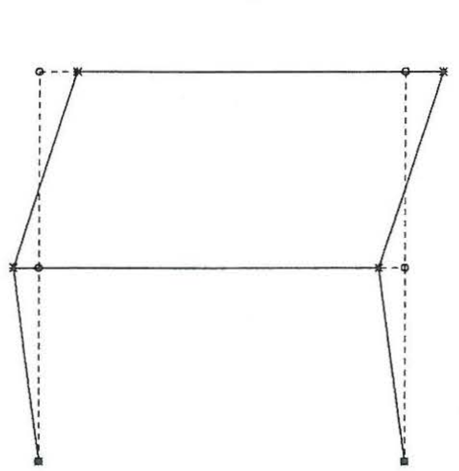


Figure 10.3: *6 Node approximation, 2. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

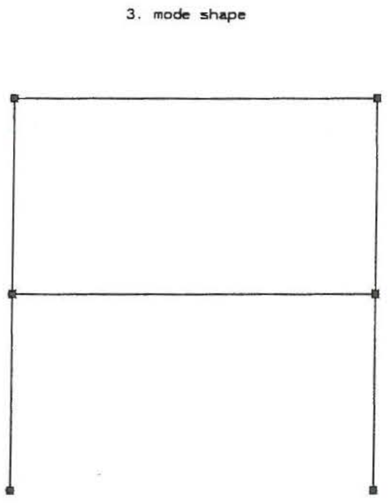


Figure 10.4: *6 Node approximation, 3. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

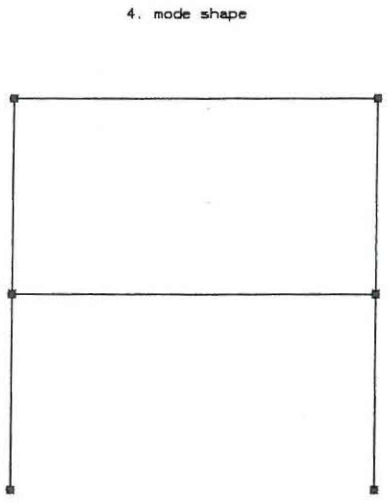


Figure 10.5: *6 Node approximation, 4. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

12 Node approximation
exerc12b
F=[1.0281 4.2120 6.1251 7.1140]
plot12b(X,T,Shape)

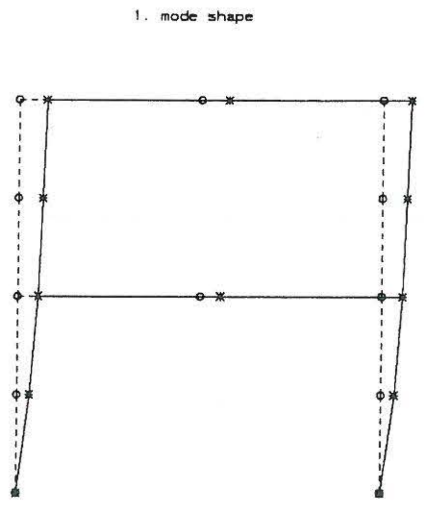


Figure 10.6: *12 Node approximation, 1. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

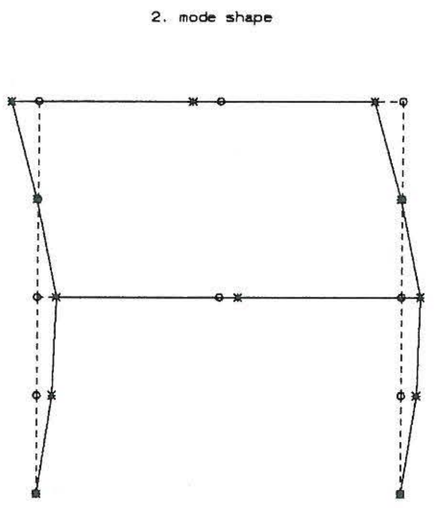


Figure 10.7: *12 Node approximation, 2. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

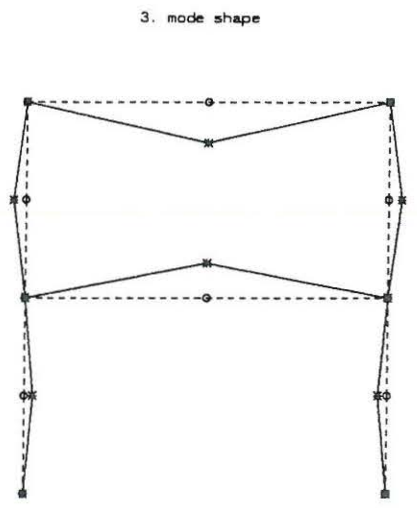


Figure 10.8: *12 Node approximation, 3. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

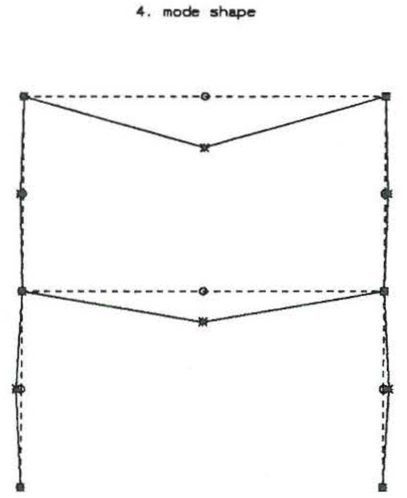


Figure 10.9: *12 Node approximation, 4. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

24 Node approximation

exerc12c

F=[1.0281 4.2116 6.0751 7.0395]

plot12c(X,T,Shape)

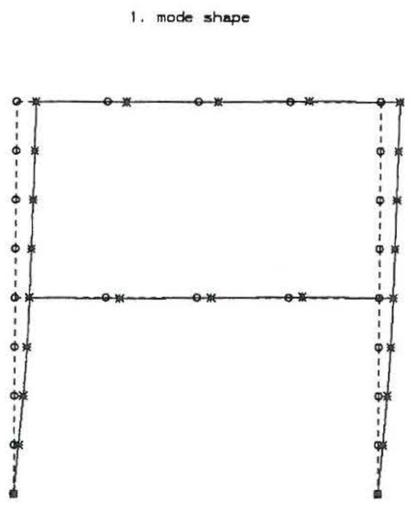


Figure 10.10: *24 Node approximation, 1. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

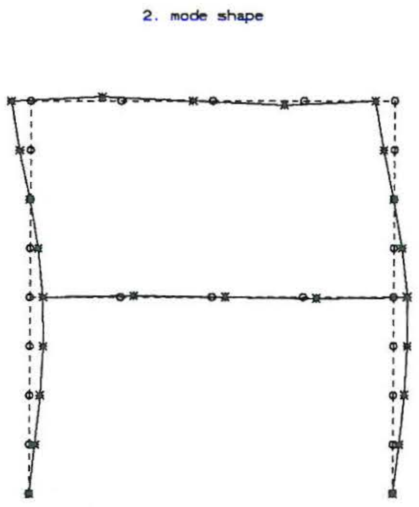


Figure 10.11: *24 Node approximation, 2. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

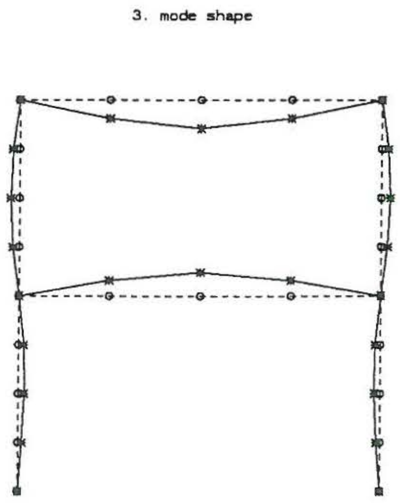


Figure 10.12: *24 Node approximation, 3. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

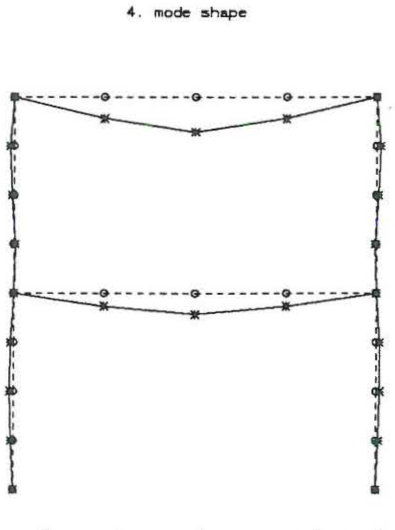


Figure 10.13: *24 Node approximation, 4. mode. o node points, * mode shape points, [- - -] undeformed structure, [—] deformed structure.*

The theoretical and the approximative eigenfrequencies of the frame structure are shown in tabel 10.1.

	Theory	6 Nodes	12 Nodes	24 Nodes
f_1	1.05	1.03	1.03	1.03
f_2	4.31	4.21	4.21	4.21
f_3	6.19	8.22	6.13	6.08
f_4	7.17	11.35	7.11	7.04

Table 10.1: *Eigenfrequencies*

10.1.2 MATLAB Files

All MATLAB files omitted.

Chapter 11

Lecture 12

Problem 2 June 26 1987 and problem 3 September 28 1989 are solved theoretically. The solutions to these problems can be found in Nielsen [2]. Furthermore MATLAB exercise 12 should be calculated, which is an extension of MATLAB exercise 11.

11.1 MATLAB Exercise 13

Consider the continuous 2-storey frame structure presented in MATLAB exercise 11. The structure is loaded as shown in figure 11.1.

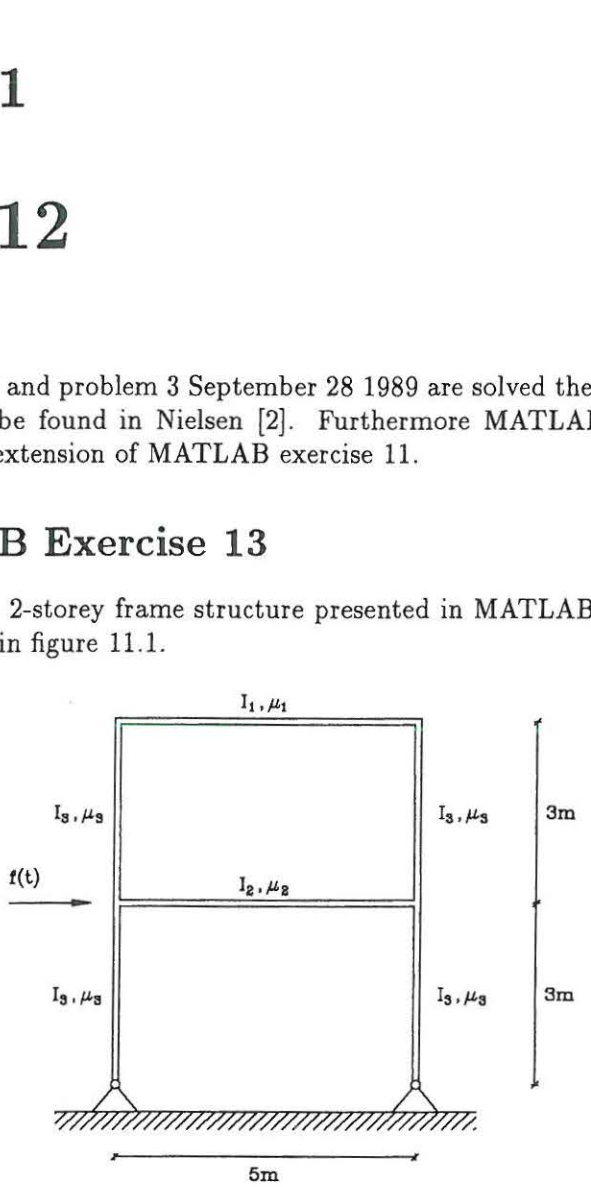


Figure 11.1: *Dynamically loaded 2-storey frame structure.*

The following data is valid for the stiffness and mass of the frame structure.

$$\begin{aligned} &E_1 = 2.0 \cdot 10^{11} \tfrac{N}{m^2} && E_2 = E_1 && E_3 = E_1 \\ &I_1 = 1.945 \cdot 10^{-5} m^4 && I_2 = 3.89 \cdot 10^{-5} m^4 && I_3 = 2.63 \cdot 10^{-5} m^4 \\ &\mu_1 = 1000 \tfrac{kg}{m} && \mu_2 = 2000 \tfrac{kg}{m} && \mu_3 = 46 \tfrac{kg}{m} \end{aligned} \tag{11.1}$$

The initial condition of the frame structure are:

$$\mathbf{x}(0) = \mathbf{0}\ , \quad \dot{\mathbf{x}}(0) = \mathbf{0} \tag{11.2}$$

In figure 11.2 the impulse load on the frame structure is shown.

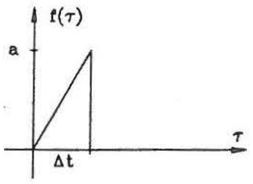


Figure 11.2: *Impulse load on 2-storey frame structure*

$$f(\tau) = \begin{cases} a\frac{\tau}{\Delta t}, & 0 < \tau \leq \Delta t \\ 0, & \tau > \Delta t \end{cases} \tag{11.3}$$

Where a is a positive constant.
Question: Determine the time-varying horisontal displacement of the stories and the moment in the upper column at the node, where the load is applied. Use 4 modal coordinates in the approximation. The mode shapes, eigenfrequencies, mass matrix and stifness matrix are known from MATLAB excercise 11 using the case with 24 systems nodes. In the approximation the quasi-static term of higher order modes are ignored. The damping matrix of the structure can be considered to decouple in a expansion in terms of undamped modal coordinates. The damping ratios of the first 4 modes and the constants describing the impulse load are:

$$\zeta_1 = 0.01\,, \quad \zeta_2 = 0.005\,, \quad \zeta_3 = 0.007\,, \quad \zeta_4 = 0.006 \tag{11.4}$$

$$a = 100\,, \qquad \Delta t = 0.2s. \tag{11.5}$$

11.1.1 MATLAB Solution

To calculate the time-varying moment at the excitation point the beam element connecting the node at the excitation point and the node right above is considered. Using standard FE-theory the end section shear forces P_1, P_3 and the end-section moments P_2, P_4 can be calculated from the conjugated nodal displacement u_1, u_3 and nodal rotations u_2, u_4 as follows

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = \frac{EI}{l^3} \begin{bmatrix} 12 & 6l & -12 & 6l \\ 6l & 4l^2 & -6l & 2l^2 \\ -12 & -6l^2 & 12 & -6l \\ 6l & 2l^2 & -6l & 4l^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{11.6}$$

The signs of the quantities has been depicted on figure 1.3.

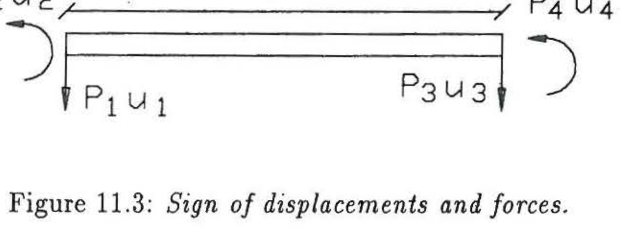


Figure 11.3: *Sign of displacements and forces.*

The displacement at any node point, j, of the discretized structure can be calculated from the following truncated modal expansion, see Nielsen [1], p. 86, (3-249):

$$x_j(t) \simeq \sum_{i=1}^{4} q_i(t)\Phi_j^{(i)} \tag{11.7}$$

Where j denotes the number of degree of freedom of the excitation point. The differential placement of the modal coordinates are given by the following second order differential equation.

$$\ddot{q}_i + 2\zeta_i\omega_i\dot{q}_i + \omega_i^2 q_i = \frac{1}{M_i}F_i \tag{11.8}$$

With the initial conditions:

$$q_i(0) = 0 , \quad \dot{q}_i(0) = 0 \tag{11.9}$$

The modal masses and the modal excitation are calculated from:

$$M_i = \mathbf{\Phi}^{(i)T}\mathbf{M}\mathbf{\Phi}^{(i)} \tag{11.10}$$

$$F_i = \mathbf{\Phi}^{(i)T}\mathbf{f} = \Phi_j^{(i)}f(t) \tag{11.11}$$

The solution to an equivalent differential equation to (11.8) is known from MATLAB exercise 4 and appendix A.

$$q_i(t) = \frac{\Phi_j^{(i)}a}{\Delta t\omega_i^2}\left[\frac{min(t,\Delta t)}{M_i} + h(t - min(t,\Delta t)) - h(t) + 2\zeta_i\omega_i(H(t - min(t,\Delta t)) - H(t))\right] \tag{11.12}$$

Where:

$$h(t) = \frac{1}{M_i\omega_{d,i}}e^{-\zeta_i\omega_i t}sin(\omega_{d,i}t) \tag{11.13}$$

$$H(t) = \frac{1}{M_i\omega_{d,i}\omega_i^2}[\omega_{d,i} - e^{-\zeta_i\omega_i t}(\zeta_i\omega_i\sin(\omega_{d,i}t) + \omega_{d,i}\cos(\omega_{d,i}t))] \tag{11.14}$$

The following steps are performed to obtain the solution to the question.

1) Calculate the global mass matrix **M**, the global stiffness matrix **K**, the eigenfrequencies ω_i and the mode shapes $\Phi^{(i)}$.

2) Calculate the modal mass M_i and the modal excitation F_i from (11.10) and (11.11).

3) Solve (11.8) for the four lowest modes to determine the modal coordinates.

4) Use (11.7) to calculate the displacement response of the excitation point.

5) Use (11.6) to calculate the time-varying moment of the excitation point.

Ad 1) By giving the following orders, equivalent to MATLAB excercise 11, the mass matrix, stifness matrix. eigenfrequencies and mode shapes will be calculated and available in the memory afterwards.

exerc12c

The variables are denoted M,K,F and Shape.

Ad 2) The modal masses and the modal excitations are calculated in the MATLAB file *exerc13a.m*. Give the following order.

[Mi Fi Shapes]=exerc13a(M,Shape)

Ad 3,4,5 Each of the three steps are performed in the MATLAB file exerc13b.m. Give the following order, and the graphs will be shown on the screen. Press enter to continue. The inputs are described in the head of the MATLAB file.

exerc13b(Mi,Fi,Shapes,F,100,0.2,2,0.01);

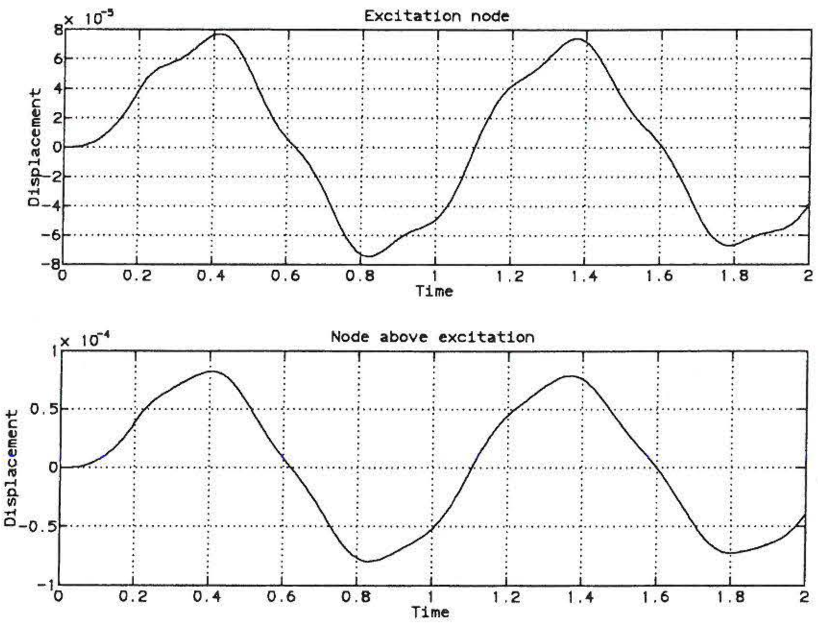


Figure 11.4: *Horizontal displacement response of the storeys. The upper graph shows the upper storey and the lower graph shows the lower storey.*

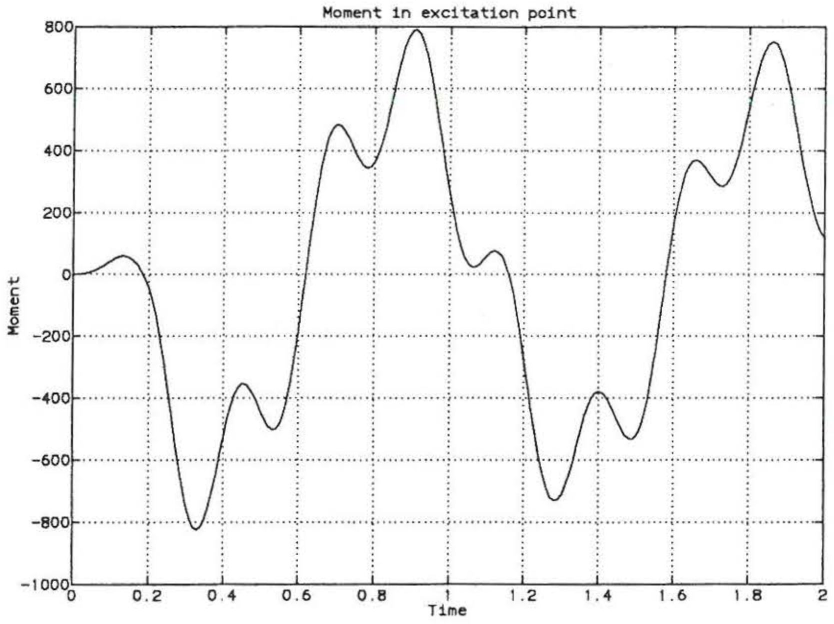


Figure 11.5: *Time-varying moment at the excitation points.*

11.1.2 MATLAB Files

All MATLAB files omitted.

Appendix A

Lecture 4

Consider the linear single degree of freedom system in problem 1, lecture 3 Nielsen [2]. Instead of a harmonic excitation force the system is excited by an impulse force. The impulse force is shown in figure A.1.

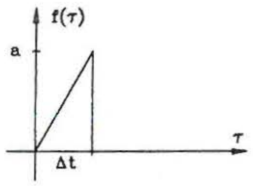


Figure A.1: *Impulse excitation of the system in problem 1, lecture 3 Nielsen [2].*

Determine the solution, $x(t)$, to the linear vibration problem of the system:

$$\begin{array}{l} \ddot{x}(t) \;+\; 2\zeta\omega_0\dot{x}(t) \;+\; \omega_0^2 x(t) \;=\; f(t)\;, \quad t > 0 \\ x(t=0) \;=\; 0\;, \quad \dot{x}(t=0) \;=\; 0 \end{array} \tag{A.1}$$

Where ζ is the damping ratio and ω_0 is the natural eigenfrequency. The impulse force on the system (see figure A.1) is described by:

$$f(\tau) \;=\; \begin{cases} a\frac{\tau}{\Delta t}, & 0 < \tau \leq \Delta t \\ 0, & \tau > \Delta t \end{cases} \tag{A.2}$$

The maximum impulse excitation is a and Δt is the duration of the impulse force.

Analytical Solution

Because the initial conditions of the system are zero (see (A.1)) the displacement response is given by Duhamel's integral (see Nielsen [1], p. 29, (2.131)):

$$x(t) \;=\; \int_0^t h(t-\tau)f(\tau)d\tau \;=\; \int_0^{min(t,\Delta t)} h(t-\tau)a\frac{\tau}{\Delta t}d\tau \tag{A.3}$$

The impulse response function, $h(t)$ is given by (see Nielsen [1], p. 27 (2.119)):

$$h(t) = \begin{cases} \frac{1}{m\omega_d} e^{-\zeta\omega_0 t} sin(\omega_d t) & t \geq 0 \\ 0 & t < 0 \end{cases} \tag{A.4}$$

Where m is the mass of the oscillator and $\omega_d = \omega_0\sqrt{1-\zeta^2}$ is the damped natural eigenfrequency. The integral in (A.3) is evaluated by partial integration.

$$\begin{aligned} x(t) &= -[H(t-\tau)f(\tau)]_0^t + \int_0^t H(t-\tau)f'(\tau)d\tau \\ &= -H(0)f(t) + H(t)f(0) + \int_0^{min(t,\Delta t)} H(t-\tau)\tfrac{a}{\Delta t}d\tau \\ &= \tfrac{a}{\Delta t}\int_0^{min(t,\Delta t)} H(t-\tau)d\tau \end{aligned} \tag{A.5}$$

Where $H(t)$ is the integral of $h(t)$ defined as:

$$H(t) = \int_0^t h(\tau)d\tau \quad \Rightarrow \quad H(0) = 0 \tag{A.6}$$

In the last statement of (A.5) it has been used that $H(0) = f(0) = 0$, $f'(\tau) = \frac{a}{\Delta t}$, $\tau \leq \Delta t$ and $f'(\tau) = 0$, $\tau > \Delta t$.

In order to evaluate the integral (A.6) the following initial value problem for $h(t)$ is used (see Nielsen [1], p. 27, (2.119)).

$$\begin{aligned} &\ddot{h}(\tau) + 2\zeta\omega_0\dot{h}(\tau) + \omega_0^2 h(\tau) = 0 , \quad \tau > 0 \\ &h(0) = 0 , \quad \dot{h}(0) = \tfrac{1}{m} \end{aligned} \tag{A.7}$$

Upon integrating (A.7) an expression for $H(t)$ as a function of $h(t)$ is obtained.

$$[\dot{h}(\tau)]_0^t + 2\zeta\omega_0[h(\tau)]_0^t + \omega_0^2[H(\tau)]_0^t = 0 \tag{A.8}$$

Inserting the initial conditions (A.7) provides:

$$H(t) = \frac{1}{\omega_0^2}\left(\frac{1}{m} - \dot{h}(t) - 2\zeta\omega_0 h(t)\right) \tag{A.9}$$

$$H(t) = \frac{1}{m\omega_d\omega_0^2}\left[\omega_d - e^{-\zeta\omega_0 t}\left(\zeta\omega_0 sin(\omega_d t) + \omega_d cos(\omega_d t)\right)\right] \tag{A.10}$$

The result of (A.10) is inserted in the last statement of (A.5).

$$\begin{aligned} x(t) &= \tfrac{a}{\Delta t\omega_0^2}\int_0^{min(t,\Delta t)}\left(\tfrac{1}{m} - \dot{h}(t-\tau) - 2\zeta\omega_0 h(t-\tau)\right)d\tau \\ &= \tfrac{a}{\Delta t\omega_0^2}\left[\tfrac{\tau}{m} + h(t-\tau) + 2\zeta\omega_0 H(t-\tau)\right]_0^{min(t,\Delta t)} \\ &= \tfrac{a}{\Delta t\omega_0^2}\left[\tfrac{min(t,\Delta t)}{m} + h(t-min(t,\Delta t)) - h(t) + (2\zeta\omega_0 H(t-min(t,\Delta t)) - H(t))\right] \end{aligned} \tag{A.11}$$

Where $h(t)$ is given by (A.4) and $H(t)$ is given by (A.10).

Bibliography

[1] Søren R.K. Nielsen *Linear svingningsteori*
Aalborg tekniske Universitetsforlag, December 1993, 2. udgave

[2] Søren R.K. Nielsen *Linear Vibration Theory, Solved Problems*
Aalborg tekniske Universitetsforlag, November 1993, 1. udgave

ISSN 1395-8232 U9601
Aalborg University
Dept. of Building Technology and Structural Engineering
Sohngaardsholmsvej 57
DK-9000 Aalborg, Denmark

Printed at Aalborg University