OPTIMIZATION (MT2528)

ASSIGNMENT 2

Name: Mora Akhil

Personal number: 9507286954

Task 1

Given a spring system as shown in the figure below.

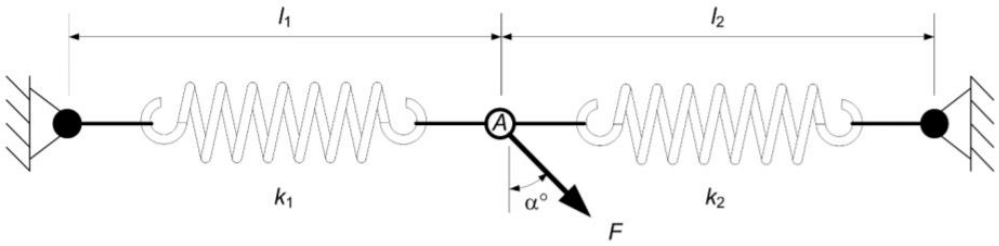


Figure 1: Spring system

Problem statement: To find the new equilibrium position of A.

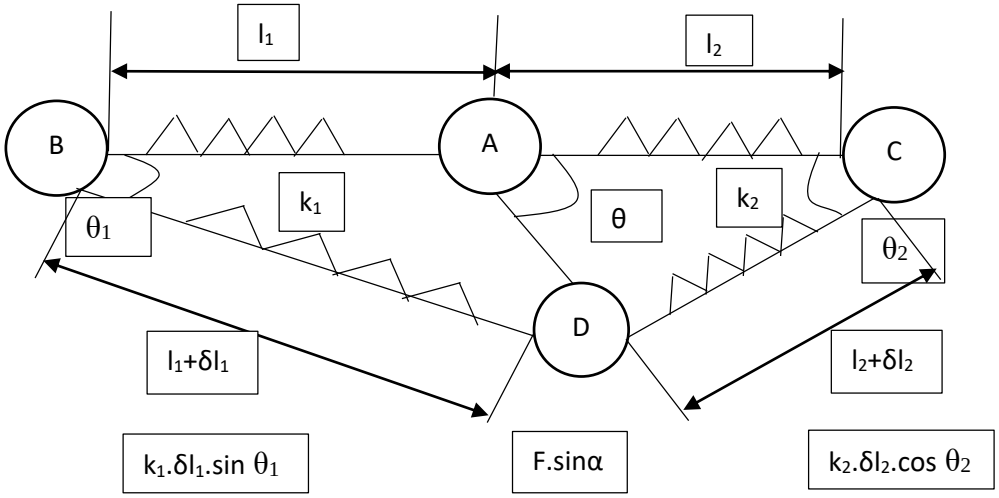
Table 1: Given parameters

Parameter	Value
l_1	100 mm
l_2	100 mm
k_1	100 N/m
k_2	800 N/m
F	$\sqrt{50}$
α	45°

Method: Random walk algorithm.

Solution procedure and approach:

Free body diagram:



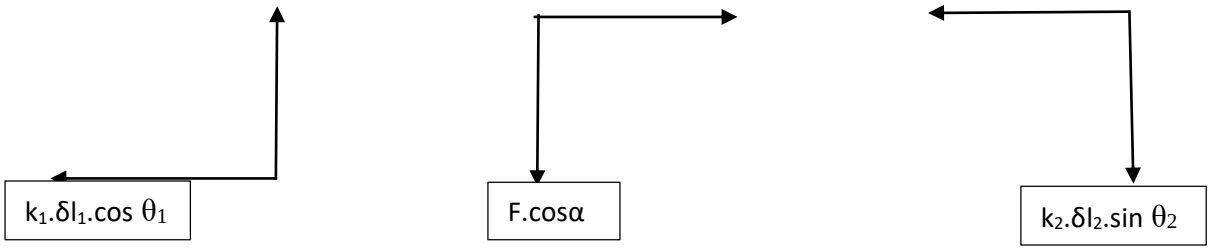


Figure 2: Free body diagram

For equilibrium: $Fsin\alpha - k_2.\delta l_2.cos\theta_2 - k_1.\delta l_1.cos\theta_1 = 0$ → Equation (1)

$$Fcos\alpha + k_2.\delta l_2.sin\theta_2 - k_1.\delta l_1.sin\theta_1 = 0$$ → Equation (2)

Fixing the co-ordinate system for the given spring system.

Let the initial position of A be, at the origin of the co-ordinate system, point B on the negative X-axis and point C on the positive X-axis.

- ⇨ Co-ordinates of A = (0,0)
- ⇨ Co-ordinates of B = ($-l_1$,0)
- ⇨ Co-ordinates of C = (l_1,0)

Let the co-ordinates of point D be (x,y).

Let AD=l and angle CAD=θ

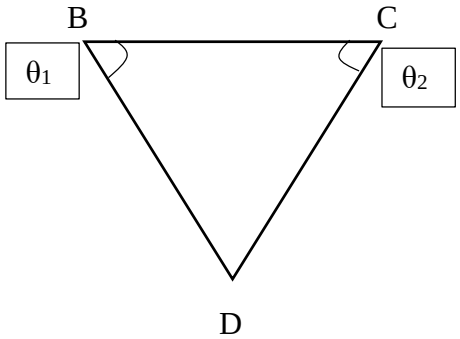
To evaluate the objective function, the lengths of BC, BD, CD and the angles between elongated and initial states of springs 1 and 2, i.e., θ_1 and θ_2 respectively have to be determined.

Length of BC = l_1+l_2

Length of BD = $\sqrt{(l_1 + x)^2 + y^2}$

Length of CD = $\sqrt{(l_2 - x)^2 + y^2}$

For determining the angles θ_1 and θ_2, consider triangle BCD.



BC = l_1+l_2; BD = $l_1+\delta l_1$; CD = $l_2+\delta l_2$

δl_1 = Elongation in spring 1 = BD-l_1

δl_2 = Elongation in spring 2 = CD-l_2

Applying cosine rules to angles θ_1 and θ_2, we get

$$\theta_1 = cos^{-1}\left(\frac{BC^2 + BD^2 - CD^2}{2 \times BC \times BD}\right) = cos^{-1}(\frac{(l_1 + l_2)^2 + (l_1 + \delta l_1)^2 - (l_2 + \delta l_2)^2}{2 \times (l_1 + l_2) \times (l_1 + \delta l_1)})$$

$$\theta_2 = cos^{-1}\left(\frac{BC^2 + CD^2 - BD^2}{2 \times BC \times CD}\right) = cos^{-1}(\frac{(l_1 + l_2)^2 + (l_2 + \delta l_2)^2 - (l_1 + \delta l_1)^2}{2 \times (l_1 + l_2) \times (l_2 + \delta l_2)})$$

Approach: Formulating the objective function. Adding equations (1) and (2),

Objective function:

$$F(sin\alpha + cos\alpha) + k_2.\delta l_2.(sin\theta_2 - cos\theta_2) - k_1.\delta l_1.(sin\theta_1 + cos\theta_1)$$

From the above equations, the objective function can be evaluated for different values of x and y.

Initially, a step length of 0.01mm and step direction of 1°is selected randomly and the objective function is evaluated. Further, step incremental of 0.0001mm, step direction of 1° is chosen and the new position is determined. It is evaluated using the following relation.

$$x^{t+1} = x^t + \alpha S$$

Where x^{t+1} is the next position, x^t is the current position, α is step length and S is unit vector in search direction.

The tolerance limit is set for 0.0001. If the objective function is less than the tolerance limit, the new position is determined.

The objective function is satisfied for the tolerance limit at,

New equilibrium position: x=13.4mm and y=4mm.

Steepest Decent Method

Method: Steepest Decent algorithm.

Solution procedure and approach:

The objective function formulated above is evaluated using steepest decent algorithm.

The gradients and hessian matrix of the objective function are calculated using the following formulae.

$$Gradient = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}$$

$$Hessian = \begin{bmatrix} \frac{\partial^2 f}{\partial x} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y} \end{bmatrix}$$

The gradient matrix is equated to zero and the hessian matrix is calculated. If the hessian matrix is less than zero, the values of x and y are noted and the objective function is evaluated. If the

objective function is satisfied, the values are considered to be in feasible region. All such values that satisfies the conditions are considered to be feasible set. The tolerance limit set is 0.0001.

The problem is started with the initial values of x and y as 0.0001 and 0.0001. The algorithm terminates when the solution satisfies tolerance limit. The optimal solution is found to be:

New equilibrium position: x = 15.6mm and y = 3.2mm

Task 2

Given a cantilever beam of I-section as shown in the figure below.

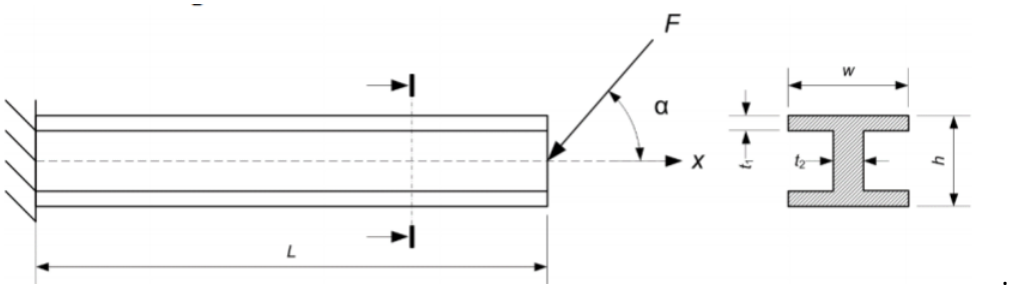


Figure 3: I-section steel beam

Table 2: Given parameters

Parameter	Value
E	210 GPa
G	70 GPa
R_{el}	190 MPa
F	75 kN
L	1.2 m
α	45°

Solution procedure and approach:

The given force is resolved into horizontal and vertical components.

Horizontal component = Fcosα

Vertical component = Fsinα

Fcosα induces compressive stress and Fsinα induces bending stress in the beam.

Compressive stress, $\boldsymbol{\sigma}_N = \frac{-Fcos\alpha}{A}$ (negative since compression)

Bending stress, $\boldsymbol{\sigma}_b = \frac{My}{I}$

Where M = Bending moment = FLsinα

y=Distance from base to centroid = $t_1+(h-2t_1)/2$

I=Moment of inertia of cross section about XX axis

Effective stress $\boldsymbol{\sigma} = \frac{-Fcos\alpha}{A} + \frac{My}{I}$

Deflection, $\delta = (\boldsymbol{\sigma} * l^3)/(3*E*I*A)$

The length of the beam is large compared to the other dimensions of the cross-section. Therefore, the deformation effect of shear stress is relatively small and hence it is neglected.

The problem is solved for different dimensions of t_1, t_2 to minimize the weight of the beam satisfying the following constraints using optimization toolbox in MATLAB.

Effective stress < Yield stress

Maximum deflection < 15mm

Minimizing the weight of the beam implies minimizing the dimensions of cross-section since the length and density of the beam are constant.

Objective function: Minimisation of area of the given beam.

$$\text{Area} = 2.w.t_1+(h-2.t_1).t_2$$

Constraints: Effective stress should be less than material yield stress and maximum deflection should be less than 15mm.

Effective stress = $\frac{-F\cos\alpha}{A} + \frac{My}{I} < 190e6$

Maximum deflection = $(\boldsymbol{\sigma} * l^3)/(3*E*I*A) < 15e\text{-}3$

The optimised width and height are determined for different values of t_1 and t_2.

The problem is solved using optimisation toolbox in MATLAB.

The following is the feasible set obtained, minimising the objective function while satisfying the constraints.

(i) For $t_1 = t_2 = 10$ mm, the optimized parameters are,
h = 175 mm, w = 131 mm, Cross-sectional area = 0.0018 m^2

(ii) For $t_1 = t_2 = 15$ mm, the optimized parameters are,
h = 175 mm, w = 18 mm, Cross-sectional area = 0.0027 m^2

(iii) For $t_1 = t_2 = 20$ mm, the optimized parameters are,
h = 175 mm, w = 100 mm, Cross-sectional area = 0.0031 m^2

(iv) For $t_1 = t_2 = 25$ mm, the optimized parameters are,
h = 133 mm, w = 45 mm, Cross-sectional area = 0.0044 m^2

(v) For $t_1 = t_2 = 30$ mm, the optimized parameters are,
h = 43.5 mm, w = 131 mm, Cross-sectional area = 0.0085 m^2

(vi) For $t_1 = t_2 = 35$ mm, the optimized parameters are,
h = 47 mm, w = 150 mm, Cross-sectional area = 0.0097 m^2

(vii) For $t_1 = t_2 = 40$ mm, the optimized parameters are,
h = 29.8 mm, w = 15 mm, Cross-sectional area = 0.01 m^2

(viii) For $t_1 = t_2 = 45$ mm, the optimized parameters are,
h = 21 mm, w = 150 mm, Cross-sectional area = 0.01 m^2

(ix) For $t_1 = t_2 = 50$ mm, the optimized parameters are,
h = 12 mm, w = 150 mm, Cross-sectional area = 0.0106 m^2

From the above feasible set, it is observed that, the objective function was minimised much in case (i), while satisfying the constraints.

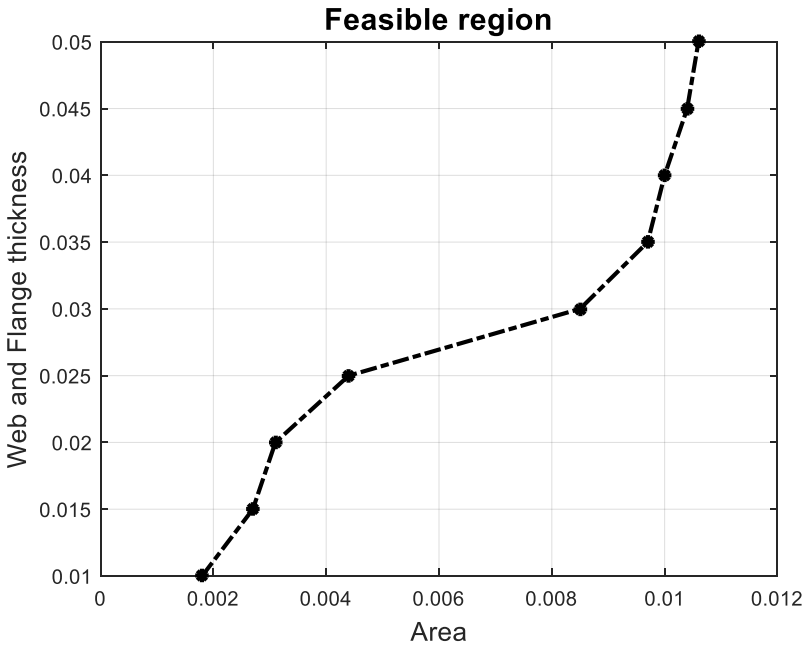


Figure 4: Feasible region

Optimal solution: t_1=10mm; t_2=10mm; h=175mm; w=131mm; Area = 0.0018 m^2

Task 3

An anchor system is with three beams and two connecting rods is given as shown in the figure below.

Figure 5: Anchor design

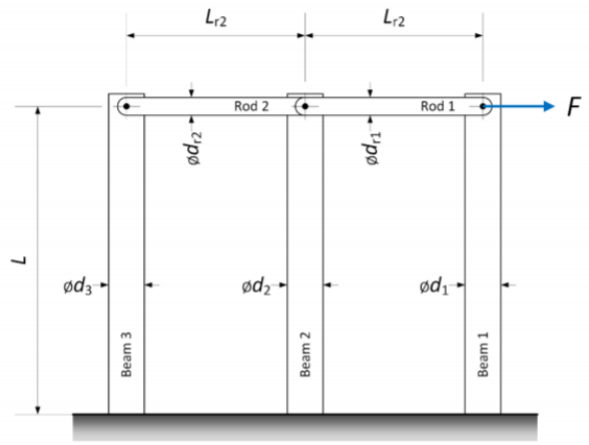


Figure 6: Schematic description

Table 3: Given parameters

Parameter	Value	Unit
L	1	m
L_{r1}	1	m
L_{r2}	1	m
E	70	GPa
ρ	2700	Kg/m^3

Solution procedure:

Let P_1 and P_2 be the tensile forces in rod 1 and rod 2 due to applied force 'F'.

Force at the joint of beam 1 and rod 1= ($F-T_1$)

Force at the joint of beam 2 and common pin joint of rods 1 and 2 = (T_1-T_2)

Force at the joint o beam 3 and rod 2=T_2

Deflection in rod 1, $\delta_{r1} = \frac{T_1 l_{r2}}{A_{r2} E}$

Deflection in rod 2, $\delta_{r2} = \frac{T_2 l_{r2}}{A_{r2} E}$

Deflection in beam 1, $\delta_1 = \frac{(F-T_1) l^3}{3EI}$

Deflection in beam 2, $\delta_2 = \frac{(T_1-T_2) l^3}{3EI}$

Deflection in beam 3, $\delta_3 = \frac{T_2 l^3}{3EI}$

$$\delta_1 = \delta_{r1} + \delta_2$$

$$\delta_2 = \delta_{r2} + \delta_3$$

Objective function: (i) Minimization of induced stress, $\frac{(F-T_1) l^3}{3EA}$

(ii) Minimization of weight, $d_1^2 + d_2^2 + dr_1^2 + dr_2^2$

Constraints: (i) $(F-T_1)/(\pi d_1^2)$, $(T_1-T_2)/(\pi d_2^2)$, $(T_2)/(\pi d_3^2) \leq 127e6$

(ii) $(F-T_1)$, (T_1-T_2), $T_2 \leq 400$

Approach:

The problem is solved using optimisation toolbox in MATLAB using *'fmincon'*.

The objective function and constraints are given in one function and the formulae of stresses are given in another function. Interior-point algorithm is chosen with the tolerance limit of 1e-9. The step size in the determination of optimal parameters is 5.8e-10. The algorithm terminated when the objective functions are minimised while satisfying the constraints.

Optimal solution:

Diameter of beam 1 = 10 mm

Diameter of beam 2 = 10 mm

Diameter of beam 3 = 10 mm

Diameter of rod 1 = 4.5 mm

Diameter of rod 2 = 5.5 mm

TASK 4:

The spring problem in task 1 is solved using genetic algorithm for the determination of equilibrium position.

Solution procedure and approach:

Continuous genetic algorithm is used for solving the problem. The function *'contgaf'* for continuous genetic algorithm is used. Initial population generated is decimal values. The following are the steps followed in solving the problem.

(i) Formulation of objective function
(ii) Number of variables is mentioned (2 variables)
(iii) The range is assumed, i.e., lower and upper bounds for both the variables, x and y (1mm – Lower bound; 100mm – Upper bound).
(iv) Initial population of 50 chromosomes is generated.
(v) Number of generations, 50 is selected.
(vi) A mutation rate of 0.2 is selected.
(vii) Proportion of the population mated is 0.6.

The problem is solved with all the above steps in a sequential order for the optimal solution.

Optimal solution:

New equilibrium position: x = 14.7mm; y = 1mm

REFERENCES

(1) Introduction to design optimisation, Wall J.
(2) Introduction to Optimum Design, Aurora, J.S.
(3) Numerical Methods using MATLAB by G.R.Lindfield and J.E.T.Penny
(4) https://se.mathworks.com
(5) OPTIMIZATION Toolbox in MATLAB.

APPENDIX:

Task 1:

```
function dis=dbtp(x1,x2,y1,y2)
dis=sqrt((x2-x1).^2+(y1-y2).^2);
end

%% Spring problem - Random walk
clc;
clear
close all;
F=sqrt(50);
alpha=(pi/4);
k1=100;
k2=800;
l1=100e-3;
l2=100e-3;
x1=0;   y1=0;
x2=-l1; y2=0;
x3=l2;  y3=0;
l(:,1)=0.01;
t(:,1)=1/(180/pi);
x4(:,1)=l(:,1).*cos(t(:,1));
y4(:,1)=l(:,1).*sin(t(:,1));
t(2:1000)=(2:1000)*(pi/180);
u=0.00001;
for j=2:1000
    tt=t(j-1);
for i=2:100
BC=dbtp(x2,x3,y2,y3);
BD=dbtp(x2,x4(:,i-1),y2,y4(:,i-1));
CD=dbtp(x3,x4(:,i-1),y3,y4(:,i-1));
dell1=BD-l1;
dell2=CD-l2;
tt1=acosd((BC^2+BD.^2-CD.^2)/(2*BC.*BD));
tt2=acosd((BC^2+CD.^2-BD.^2)/(2*BC*CD));
OF(j-1,i-1)=F*(sin(alpha)+cos(alpha))+k2*dell2*(sin(tt2)-cos(tt2))-
k1*dell1*(sin(tt1)+cos(tt1));
if(OF(j-1,i-1)<0)
    OF(j-1,i-1)=-OF(j-1,i-1);
end
if(OF(j-1,i-1)<0.001)
display(OF(j-1,i-1))
display(l(:,i-1))
display(tt)
break;
else
   l(:,i)=l(:,i-1) + (u*tt);
```

```
   x4(:,i)=l(:,i).*cos(tt);
   y4(:,i)=l(:,i).*sin(tt);
end
end
end
```

% Function for Steepest Descent method

```
function [f g H]=stpdcnt(X)
x=X(1);
y=X(2);
f=(100*((x + 1/10)^2 + y^2)^(1/2) - 10)*((5*((x + 1/10)^2 - (x - 1/10)^2 +
1/25))/(2*((x + 1/10)^2 + y^2)^(1/2)) - (1 - (25*((x + 1/10)^2 - (x -
1/10)^2 + 1/25)^2)/(4*((x + 1/10)^2 + y^2)))^(1/2)) - (800*((x - 1/10)^2 +
y^2)^(1/2) - 80)*((5*((x - 1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x - 1/10)^2
+ y^2)^(1/2)) - (1 - (25*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*((x -
1/10)^2 + y^2)))^(1/2)) + 10;
if nargout > 1
    g=[(100*((x + 1/10)^2 + y^2)^(1/2) - 10)*(1/((x + 1/10)^2 + y^2)^(1/2)
+ ((25*((4*(x + 1/10)^2)/5 - (4*(x - 1/10)^2)/5 + 4/125))/(4*((x + 1/10)^2
+ y^2)) - (25*(2*x + 1/5)*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*((x +
1/10)^2 + y^2)^2))/(2*(1 - (25*((x + 1/10)^2 - (x - 1/10)^2 +
1/25)^2)/(4*(x + 1/10)^2 + 4*y^2))^(1/2)) - (5*(2*x + 1/5)*((x + 1/10)^2 -
(x - 1/10)^2 + 1/25))/(4*((x + 1/10)^2 + y^2)^(3/2))) + (800*((x - 1/10)^2
+ y^2)^(1/2) - 80)*(1/((x - 1/10)^2 + y^2)^(1/2) + ((25*((4*(x - 1/10)^2)/5
- (4*(x + 1/10)^2)/5 + 4/125))/(4*((x - 1/10)^2 + y^2)) + (25*(2*x -
1/5)*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*((x - 1/10)^2 +
y^2)^2))/(2*(1 - (25*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x -
1/10)^2 + 4*y^2))^(1/2)) + (5*(2*x - 1/5)*((x - 1/10)^2 - (x + 1/10)^2 +
1/25))/(4*((x - 1/10)^2 + y^2)^(3/2))) - (400*(2*x - 1/5)*((5*((x - 1/10)^2
- (x + 1/10)^2 + 1/25))/(2*((x - 1/10)^2 + y^2)^(1/2)) - (1 - (25*((x -
1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*((x - 1/10)^2 + y^2)))^(1/2)))/((x -
1/10)^2 + y^2)^(1/2) + (50*(2*x + 1/5)*((5*((x + 1/10)^2 - (x - 1/10)^2 +
1/25))/(2*((x + 1/10)^2 + y^2)^(1/2)) - (1 - (25*((x + 1/10)^2 - (x -
1/10)^2 + 1/25)^2)/(4*((x + 1/10)^2 + y^2)))^(1/2)))/((x + 1/10)^2 +
y^2)^(1/2); (800*((x - 1/10)^2 + y^2)^(1/2) - 80)*((5*y*((x - 1/10)^2 - (x
+ 1/10)^2 + 1/25))/(2*((x - 1/10)^2 + y^2)^(3/2)) + (25*y*((x - 1/10)^2 -
(x + 1/10)^2 + 1/25)^2)/(4*(1 - (25*((x - 1/10)^2 - (x + 1/10)^2 +
1/25)^2)/(4*(x - 1/10)^2 + 4*y^2))^(1/2)*((x - 1/10)^2 + y^2)^2)) -
(100*((x + 1/10)^2 + y^2)^(1/2) - 10)*((5*y*((x + 1/10)^2 - (x - 1/10)^2 +
1/25))/(2*((x + 1/10)^2 + y^2)^(3/2)) + (25*y*((x + 1/10)^2 - (x - 1/10)^2
+ 1/25)^2)/(4*(1 - (25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(x +
1/10)^2 + 4*y^2))^(1/2)*((x + 1/10)^2 + y^2)^2)) - (800*y*((5*((x - 1/10)^2
- (x + 1/10)^2 + 1/25))/(2*((x - 1/10)^2 + y^2)^(1/2)) - (1 - (25*((x -
1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*((x - 1/10)^2 + y^2)))^(1/2)))/((x -
1/10)^2 + y^2)^(1/2) + (100*y*((5*((x + 1/10)^2 - (x - 1/10)^2 +
1/25))/(2*((x + 1/10)^2 + y^2)^(1/2)) - (1 - (25*((x + 1/10)^2 - (x -
1/10)^2 + 1/25)^2)/(4*((x + 1/10)^2 + y^2)))^(1/2)))/((x + 1/10)^2 +
y^2)^(1/2)];
    if nargout>2
          H=[(100*((5*((x + 1/10)^2 - (x - 1/10)^2 + 1/25))/(2*((x +
1/10)^2 + y^2)^(1/2)) - (1 - (25*((x + 1/10)^2 - (x - 1/10)^2 +
1/25)^2)/(4*((x + 1/10)^2 + y^2)))^(1/2)))/((x + 1/10)^2 + y^2)^(1/2) -
(800*((x - 1/10)^2 + y^2)^(1/2) - 80)*(((25*((4*(x - 1/10)^2)/5 - (4*(x +
1/10)^2)/5 + 4/125))/(4*((x - 1/10)^2 + y^2)) + (25*(2*x - 1/5)*((x -
1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*((x - 1/10)^2 + y^2)^2))^2/(4*(1 -
(25*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x - 1/10)^2 +
4*y^2))^(3/2)) - (5*((x - 1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x - 1/10)^2
+ y^2)^(3/2)) + (2/((x - 1/10)^2 + y^2) - (25*((x - 1/10)^2 - (x + 1/10)^2
+ 1/25)^2)/(2*((x - 1/10)^2 + y^2)^2) + (25*(2*x - 1/5)^2*((x - 1/10)^2 -
(x + 1/10)^2 + 1/25)^2)/(2*((x - 1/10)^2 + y^2)^3) + (25*(2*x - 1/5)*((4*(x
- 1/10)^2)/5 - (4*(x + 1/10)^2)/5 + 4/125))/(2*((x - 1/10)^2 +
```

```
y^2)^2))/(2*(1 - (25*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x -
1/10)^2 + 4*y^2))^(1/2)) + (2*x - 1/5)/((x - 1/10)^2 + y^2)^(3/2) +
(15*(2*x - 1/5)^2*((x - 1/10)^2 - (x + 1/10)^2 + 1/25))/(8*((x - 1/10)^2 +
y^2)^(5/2))) - (800*((5*((x - 1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x -
1/10)^2 + y^2)^(1/2)) - (1 - (25*((x - 1/10)^2 - (x + 1/10)^2 +
1/25)^2)/(4*((x - 1/10)^2 + y^2)))^(1/2)))/((x - 1/10)^2 + y^2)^(1/2) -
(100*((x + 1/10)^2 + y^2)^(1/2) - 10)*((5*((x + 1/10)^2 - (x - 1/10)^2 +
1/25))/(2*((x + 1/10)^2 + y^2)^(3/2)) - ((25*((4*(x + 1/10)^2)/5 - (4*(x -
1/10)^2)/5 + 4/125))/(4*((x + 1/10)^2 + y^2)) - (25*(2*x + 1/5)*((x +
1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*((x + 1/10)^2 + y^2)^2))^2/(4*(1 -
(25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(x + 1/10)^2 +
4*y^2))^(3/2)) + ((25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(2*((x +
1/10)^2 + y^2)^2) - 2/((x + 1/10)^2 + y^2) - (25*(2*x + 1/5)^2*((x +
1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(2*((x + 1/10)^2 + y^2)^3) + (25*(2*x +
1/5)*((4*(x + 1/10)^2)/5 - (4*(x - 1/10)^2)/5 + 4/125))/(2*((x + 1/10)^2 +
y^2)^2))/(2*(1 - (25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(x +
1/10)^2 + 4*y^2))^(1/2)) + (2*x + 1/5)/((x + 1/10)^2 + y^2)^(3/2) -
(15*(2*x + 1/5)^2*((x + 1/10)^2 - (x - 1/10)^2 + 1/25))/(8*((x + 1/10)^2 +
y^2)^(5/2))) + (800*(2*x - 1/5)*(1/((x - 1/10)^2 + y^2)^(1/2) + ((25*((4*(x
- 1/10)^2)/5 - (4*(x + 1/10)^2)/5 + 4/125))/(4*((x - 1/10)^2 + y^2)) +
(25*(2*x - 1/5)*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*((x - 1/10)^2 +
y^2)^2))/(2*(1 - (25*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x -
1/10)^2 + 4*y^2))^(1/2)) + (5*(2*x - 1/5)*((x - 1/10)^2 - (x + 1/10)^2 +
1/25))/(4*((x - 1/10)^2 + y^2)^(3/2))))/((x - 1/10)^2 + y^2)^(1/2) +
(100*(2*x + 1/5)*(1/((x + 1/10)^2 + y^2)^(1/2) + ((25*((4*(x + 1/10)^2)/5 -
(4*(x - 1/10)^2)/5 + 4/125))/(4*((x + 1/10)^2 + y^2)) - (25*(2*x + 1/5)*((x
+ 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*((x + 1/10)^2 + y^2)^2))/(2*(1 -
(25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(x + 1/10)^2 +
4*y^2))^(1/2)) - (5*(2*x + 1/5)*((x + 1/10)^2 - (x - 1/10)^2 +
1/25))/(4*((x + 1/10)^2 + y^2)^(3/2))))/((x + 1/10)^2 + y^2)^(1/2) +
(200*(2*x - 1/5)^2*((5*((x - 1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x -
1/10)^2 + y^2)^(1/2)) - (1 - (25*((x - 1/10)^2 - (x + 1/10)^2 +
1/25)^2)/(4*((x - 1/10)^2 + y^2)))^(1/2)))/((x - 1/10)^2 + y^2)^(3/2) -
(25*(2*x + 1/5)^2*((5*((x + 1/10)^2 - (x - 1/10)^2 + 1/25))/(2*((x +
1/10)^2 + y^2)^(1/2)) - (1 - (25*((x + 1/10)^2 - (x - 1/10)^2 +
1/25)^2)/(4*((x + 1/10)^2 + y^2)))^(1/2)))/((x + 1/10)^2 + y^2)^(3/2),
(400*(2*x - 1/5)*((5*y*((x - 1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x -
1/10)^2 + y^2)^(3/2)) + (25*y*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(1
- (25*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x - 1/10)^2 +
4*y^2))^(1/2)*((x - 1/10)^2 + y^2)^2)))/((x - 1/10)^2 + y^2)^(1/2) -
(800*((x - 1/10)^2 + y^2)^(1/2) - 80)*(y/((x - 1/10)^2 + y^2)^(3/2) +
((25*y*((4*(x - 1/10)^2)/5 - (4*(x + 1/10)^2)/5 + 4/125))/(2*((x - 1/10)^2
+ y^2)^2) + (25*y*(2*x - 1/5)*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/((x -
1/10)^2 + y^2)^3)/(2*(1 - (25*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x
- 1/10)^2 + 4*y^2))^(1/2)) + (15*y*(2*x - 1/5)*((x - 1/10)^2 - (x + 1/10)^2
+ 1/25))/(4*((x - 1/10)^2 + y^2)^(5/2)) + (25*y*((25*((4*(x - 1/10)^2)/5 -
(4*(x + 1/10)^2)/5 + 4/125))/(4*((x - 1/10)^2 + y^2)) + (25*(2*x - 1/5)*((x
- 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*((x - 1/10)^2 + y^2)^2))*((x -
1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(8*(1 - (25*((x - 1/10)^2 - (x + 1/10)^2
+ 1/25)^2)/(4*(x - 1/10)^2 + 4*y^2))^(3/2)*((x - 1/10)^2 + y^2)^2)) -
(100*((x + 1/10)^2 + y^2)^(1/2) - 10)*(y/((x + 1/10)^2 + y^2)^(3/2) +
((25*y*((4*(x + 1/10)^2)/5 - (4*(x - 1/10)^2)/5 + 4/125))/(2*((x + 1/10)^2
+ y^2)^2) - (25*y*(2*x + 1/5)*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/((x +
1/10)^2 + y^2)^3)/(2*(1 - (25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(x
+ 1/10)^2 + 4*y^2))^(1/2)) - (15*y*(2*x + 1/5)*((x + 1/10)^2 - (x - 1/10)^2
+ 1/25))/(4*((x + 1/10)^2 + y^2)^(5/2)) + (25*y*((25*((4*(x + 1/10)^2)/5 -
(4*(x - 1/10)^2)/5 + 4/125))/(4*((x + 1/10)^2 + y^2)) - (25*(2*x + 1/5)*((x
+ 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*((x + 1/10)^2 + y^2)^2))*((x +
1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(8*(1 - (25*((x + 1/10)^2 - (x - 1/10)^2
+ 1/25)^2)/(4*(x + 1/10)^2 + 4*y^2))^(3/2)*((x + 1/10)^2 + y^2)^2)) -
(50*(2*x + 1/5)*((5*y*((x + 1/10)^2 - (x - 1/10)^2 + 1/25))/(2*((x +
```

```
1/10)^2 + y^2)^(3/2)) + (25*y*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(1
- (25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(x + 1/10)^2 +
4*y^2))^(1/2)*((x + 1/10)^2 + y^2)^2)))/((x + 1/10)^2 + y^2)^(1/2) +
(800*y*(1/((x - 1/10)^2 + y^2)^(1/2) + ((25*((4*(x - 1/10)^2)/5 - (4*(x +
1/10)^2)/5 + 4/125))/(4*((x - 1/10)^2 + y^2)) + (25*(2*x - 1/5)*((x -
1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*((x - 1/10)^2 + y^2)^2))/(2*(1 -
(25*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x - 1/10)^2 +
4*y^2))^(1/2)) + (5*(2*x - 1/5)*((x - 1/10)^2 - (x + 1/10)^2 +
1/25))/(4*((x - 1/10)^2 + y^2)^(3/2))))/((x - 1/10)^2 + y^2)^(1/2) +
(100*y*(1/((x + 1/10)^2 + y^2)^(1/2) + ((25*((4*(x + 1/10)^2)/5 - (4*(x -
1/10)^2)/5 + 4/125))/(4*((x + 1/10)^2 + y^2)) - (25*(2*x + 1/5)*((x +
1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*((x + 1/10)^2 + y^2)^2))/(2*(1 -
(25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(x + 1/10)^2 +
4*y^2))^(1/2)) - (5*(2*x + 1/5)*((x + 1/10)^2 - (x - 1/10)^2 +
1/25))/(4*((x + 1/10)^2 + y^2)^(3/2))))/((x + 1/10)^2 + y^2)^(1/2) +
(400*y*(2*x - 1/5)*((5*((x - 1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x -
1/10)^2 + y^2)^(1/2)) - (1 - (25*((x - 1/10)^2 - (x + 1/10)^2 +
1/25)^2)/(4*((x - 1/10)^2 + y^2)))^(1/2)))/((x - 1/10)^2 + y^2)^(3/2) -
(50*y*(2*x + 1/5)*((5*((x + 1/10)^2 - (x - 1/10)^2 + 1/25))/(2*((x +
1/10)^2 + y^2)^(1/2)) - (1 - (25*((x + 1/10)^2 - (x - 1/10)^2 +
1/25)^2)/(4*((x + 1/10)^2 + y^2)))^(1/2)))/((x + 1/10)^2 + y^2)^(3/2);
(400*(2*x - 1/5)*((5*y*((x - 1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x -
1/10)^2 + y^2)^(3/2)) + (25*y*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(1
- (25*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x - 1/10)^2 +
4*y^2))^(1/2)*((x - 1/10)^2 + y^2)^2)))/((x - 1/10)^2 + y^2)^(1/2) -
(800*((x - 1/10)^2 + y^2)^(1/2) - 80)*(y/((x - 1/10)^2 + y^2)^(3/2) +
(15*y*(2*x - 1/5)*((x - 1/10)^2 - (x + 1/10)^2 + 1/25))/(4*((x - 1/10)^2 +
y^2)^(5/2)) + (25*y*((4*(x - 1/10)^2)/5 - (4*(x + 1/10)^2)/5 +
4/125))/(4*(1 - (25*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x - 1/10)^2
+ 4*y^2))^(1/2)*((x - 1/10)^2 + y^2)^2) + (25*y*((25*((4*(x - 1/10)^2)/5 -
(4*(x + 1/10)^2)/5 + 4/125))/(4*((x - 1/10)^2 + y^2)) + (25*(2*x - 1/5)*((x
- 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*((x - 1/10)^2 + y^2)^2))*((x -
1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(8*(1 - (25*((x - 1/10)^2 - (x + 1/10)^2
+ 1/25)^2)/(4*(x - 1/10)^2 + 4*y^2))^(3/2)*((x - 1/10)^2 + y^2)^2) +
(25*y*(2*x - 1/5)*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(2*(1 - (25*((x -
1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x - 1/10)^2 + 4*y^2))^(1/2)*((x -
1/10)^2 + y^2)^3)) - (100*((x + 1/10)^2 + y^2)^(1/2) - 10)*(y/((x + 1/10)^2
+ y^2)^(3/2) - (15*y*(2*x + 1/5)*((x + 1/10)^2 - (x - 1/10)^2 +
1/25))/(4*((x + 1/10)^2 + y^2)^(5/2)) + (25*y*((4*(x + 1/10)^2)/5 - (4*(x -
1/10)^2)/5 + 4/125))/(4*(1 - (25*((x + 1/10)^2 - (x - 1/10)^2 +
1/25)^2)/(4*(x + 1/10)^2 + 4*y^2))^(1/2)*((x + 1/10)^2 + y^2)^2) +
(25*y*((25*((4*(x + 1/10)^2)/5 - (4*(x - 1/10)^2)/5 + 4/125))/(4*((x +
1/10)^2 + y^2)) - (25*(2*x + 1/5)*((x + 1/10)^2 - (x - 1/10)^2 +
1/25)^2)/(4*((x + 1/10)^2 + y^2)^2))*((x + 1/10)^2 - (x - 1/10)^2 +
1/25)^2)/(8*(1 - (25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(x +
1/10)^2 + 4*y^2))^(3/2)*((x + 1/10)^2 + y^2)^2) - (25*y*(2*x + 1/5)*((x +
1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(2*(1 - (25*((x + 1/10)^2 - (x - 1/10)^2
+ 1/25)^2)/(4*(x + 1/10)^2 + 4*y^2))^(1/2)*((x + 1/10)^2 + y^2)^3)) -
(50*(2*x + 1/5)*((5*y*((x + 1/10)^2 - (x - 1/10)^2 + 1/25))/(2*((x +
1/10)^2 + y^2)^(3/2)) + (25*y*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(1
- (25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(x + 1/10)^2 +
4*y^2))^(1/2)*((x + 1/10)^2 + y^2)^2)))/((x + 1/10)^2 + y^2)^(1/2) +
(800*y*(1/((x - 1/10)^2 + y^2)^(1/2) + ((25*((4*(x - 1/10)^2)/5 - (4*(x +
1/10)^2)/5 + 4/125))/(4*((x - 1/10)^2 + y^2)) + (25*(2*x - 1/5)*((x -
1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*((x - 1/10)^2 + y^2)^2))/(2*(1 -
(25*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x - 1/10)^2 +
4*y^2))^(1/2)) + (5*(2*x - 1/5)*((x - 1/10)^2 - (x + 1/10)^2 +
1/25))/(4*((x - 1/10)^2 + y^2)^(3/2))))/((x - 1/10)^2 + y^2)^(1/2) +
(100*y*(1/((x + 1/10)^2 + y^2)^(1/2) + ((25*((4*(x + 1/10)^2)/5 - (4*(x -
1/10)^2)/5 + 4/125))/(4*((x + 1/10)^2 + y^2)) - (25*(2*x + 1/5)*((x +
1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*((x + 1/10)^2 + y^2)^2))/(2*(1 -
```

```
(25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(x + 1/10)^2 +
4*y^2))^(1/2)) - (5*(2*x + 1/5)*((x + 1/10)^2 - (x - 1/10)^2 +
1/25))/(4*((x + 1/10)^2 + y^2)^(3/2))))/((x + 1/10)^2 + y^2)^(1/2) +
(400*y*(2*x - 1/5)*((5*((x - 1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x -
1/10)^2 + y^2)^(1/2)) - (1 - (25*((x - 1/10)^2 - (x + 1/10)^2 +
1/25)^2)/(4*((x - 1/10)^2 + y^2)))^(1/2)))/((x - 1/10)^2 + y^2)^(3/2) -
(50*y*(2*x + 1/5)*((5*((x + 1/10)^2 - (x - 1/10)^2 + 1/25))/(2*((x +
1/10)^2 + y^2)^(1/2)) - (1 - (25*((x + 1/10)^2 - (x - 1/10)^2 +
1/25)^2)/(4*((x + 1/10)^2 + y^2)))^(1/2)))/((x + 1/10)^2 +
y^2)^(3/2),(100*((x + 1/10)^2 + y^2)^(1/2) - 10)*((15*y^2*((x + 1/10)^2 -
(x - 1/10)^2 + 1/25))/(2*((x + 1/10)^2 + y^2)^(5/2)) - (25*((x + 1/10)^2 -
(x - 1/10)^2 + 1/25)^2)/(4*(1 - (25*((x + 1/10)^2 - (x - 1/10)^2 +
1/25)^2)/(4*(x + 1/10)^2 + 4*y^2))^(1/2)*((x + 1/10)^2 + y^2)^2) - (5*((x +
1/10)^2 - (x - 1/10)^2 + 1/25))/(2*((x + 1/10)^2 + y^2)^(3/2)) +
(25*y^2*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/((1 - (25*((x + 1/10)^2 -
(x - 1/10)^2 + 1/25)^2)/(4*(x + 1/10)^2 + 4*y^2))^(1/2)*((x + 1/10)^2 +
y^2)^3) + (625*y^2*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^4)/(16*(1 - (25*((x
+ 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*(x + 1/10)^2 + 4*y^2))^(3/2)*((x +
1/10)^2 + y^2)^4)) - (800*((x - 1/10)^2 + y^2)^(1/2) - 80)*((15*y^2*((x -
1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x - 1/10)^2 + y^2)^(5/2)) - (25*((x -
1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(1 - (25*((x - 1/10)^2 - (x + 1/10)^2
+ 1/25)^2)/(4*(x - 1/10)^2 + 4*y^2))^(1/2)*((x - 1/10)^2 + y^2)^2) - (5*((x
- 1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x - 1/10)^2 + y^2)^(3/2)) +
(25*y^2*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/((1 - (25*((x - 1/10)^2 -
(x + 1/10)^2 + 1/25)^2)/(4*(x - 1/10)^2 + 4*y^2))^(1/2)*((x - 1/10)^2 +
y^2)^3) + (625*y^2*((x - 1/10)^2 - (x + 1/10)^2 + 1/25)^4)/(16*(1 - (25*((x
- 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(x - 1/10)^2 + 4*y^2))^(3/2)*((x -
1/10)^2 + y^2)^4)) - (800*((5*((x - 1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x
- 1/10)^2 + y^2)^(1/2)) - (1 - (25*((x - 1/10)^2 - (x + 1/10)^2 +
1/25)^2)/(4*((x - 1/10)^2 + y^2)))^(1/2)))/((x - 1/10)^2 + y^2)^(1/2) +
(100*((5*((x + 1/10)^2 - (x - 1/10)^2 + 1/25))/(2*((x + 1/10)^2 +
y^2)^(1/2)) - (1 - (25*((x + 1/10)^2 - (x - 1/10)^2 + 1/25)^2)/(4*((x +
1/10)^2 + y^2)))^(1/2)))/((x + 1/10)^2 + y^2)^(1/2) + (1600*y*((5*y*((x -
1/10)^2 - (x + 1/10)^2 + 1/25))/(2*((x - 1/10)^2 + y^2)^(3/2)) + (25*y*((x
- 1/10)^2 - (x + 1/10)^2 + 1/25)^2)/(4*(1 - (25*((x - 1/10)^2 - (x +
1/10)^2 + 1/25)^2)/(4*(x - 1/10)^2 + 4*y^2))^(1/2)*((x - 1/10)^2 +
y^2)^2)))/((x - 1/10)^2 + y^2)^(1/2) - (200*y*((5*y*((x + 1/10)^2 - (x -
1/10)^2 + 1/25))/(2*((x + 1/10)^2 + y^2)^(3/2)) + (25*y*((x + 1/10)^2 - (x
- 1/10)^2 + 1/25)^2)/(4*(1 - (25*((x + 1/10)^2 - (x - 1/10)^2 +
1/25)^2)/(4*(x + 1/10)^2 + 4*y^2))^(1/2)*((x + 1/10)^2 + y^2)^2)))/((x +
1/10)^2 + y^2)^(1/2) + (800*y^2*((5*((x - 1/10)^2 - (x + 1/10)^2 +
1/25))/(2*((x - 1/10)^2 + y^2)^(1/2)) - (1 - (25*((x - 1/10)^2 - (x +
1/10)^2 + 1/25)^2)/(4*((x - 1/10)^2 + y^2)))^(1/2)))/((x - 1/10)^2 +
y^2)^(3/2) - (100*y^2*((5*((x + 1/10)^2 - (x - 1/10)^2 + 1/25))/(2*((x +
1/10)^2 + y^2)^(1/2)) - (1 - (25*((x + 1/10)^2 - (x - 1/10)^2 +
1/25)^2)/(4*((x + 1/10)^2 + y^2)))^(1/2)))/((x + 1/10)^2 + y^2)^(3/2)];
    end
end


%% Spring problem - Steepest descent
clc;
clear all;
close all;
%% Given paramters
F=sqrt(50);
alpha=(45*pi/180);
k1=100;
k2=800;
l1=100e-3;
l2=100e-3;
```

```
%% Co-ordinates of points
syms x y
x1=0;   y1=0;           % A
x2=-l1; y2=0;           % B Fixed end of spring 1
x3=l2;  y3=0;           % C Fixed end of spring 2
x4=x;y4=y*;             % D - New position of A
%% Distances
BC=dbtp(x2,x3,y2,y3);
BD=dbtp(x2, ,y2,y4);
CD=dbtp(x3,x4,y3,y4);
%% Cosine and sine angles
costheeta1=(BC^2+BD^2-CD^2)/(2*BC*BD);
costheeta2=(BC^2+CD^2-BD^2)/(2*BC*CD);
sintheeta1=sqrt(1-costheeta1^2);
sintheeta2=sqrt(1-costheeta2^2);
%% Deflections
dell1=BD-l1;  % Deflection in spring 1
dell2=CD-l2;  % Deflection in spring 2
%% Objective function
OF=F*(sin(alpha)+cos(alpha))+k2*dell2*(sintheeta2-costheeta2)-
k1*dell1*(sintheeta1-costheeta1);
%% Differentials
dfx=diff(OF,x);
dfy=diff(OF,y);
%% Hessian matrix
h11=diff(dfx,x);
h12=diff(dfx,y);
h21=diff(dfy,x);
h22=diff(dfy,y);
%%
options=optimset('GradObj','on','Hessian','on');
[x,fval]=fminunc(@stpdcnt,[10e-3; 10e-3],options);
```

Task 2:

```
function A=area(x)
h=x(1);
w=x(2);
A=2*30e-3*x(2)+(x(1)-2*30e-3)*30e-3;
end


function [c1,c2,i]=constraints(x)
h=x(1);
w=x(2);
A1=10e-3*x(2);
A2=(x(1)-2*10e-3)*10e-3;
A3=10e-3*x(2);
A=A1+A2+A3;
y1=10e-3+(x(1)-2*10e-3)+10e-3/2;
y2=10e-3+(x(1)-2*10e-3);
y3=10e-3/2;
ybar=(A1*y1+A2*y2+A3*y3);
k1=abs(ybar-y1);
k2=abs(ybar-y2);
k3=abs(ybar-y3);
I=x(2)*(10e-3^3)/12+A1*k1^2+10e-3*((x(1)-2*10e-3)^3)/12+A2*k2^2+x(2)*(10e-
3^2)/12+A3*k3^2;
c1=((-75e3*cosd(45)/A)+(75e3*sind(45)*1200*(10e-3+(x(1)-2*10e-3)/2))/(I))-
190e6;
c2=((c1*1200^3)/(3*210e3*I*A))-15e-3;
```

```
i=0;
end

hmin=10e-3;
hmax=175e-3;
wmin=10e-3;
wmax=150e-3;
x0=[(hmin+hmax)/2 (wmin+wmax)/2];
lowbou=[hmin wmin];
uppbou=[hmax wmax];
options=optimoptions(@fmincon,'Algorithm','interior-
point','display','iter','TolX',1E-9,'TolFun',1E-9,'DiffMinChange',1E-6);
[x,fval,exitflag,output]=fmincon('area',x0',[],[],[],[],lowbou,uppbou,'cons
traints',options)
```

Task 3:

```
function [a,b]=const(x,var,minmat,maxmat)
F=var(4);
l=var(1);
Maxstr=var(6);
GroForce=var(5);
P2=F/(3+((12*x(3).^4)/(16*x(5).^2))+((9*(x(2).^4)*(x(3).^4))/(16*16*(x(4).^
2)*(x(5).^2))));
P1=((3*P2*(x(1).^4))/(16*(x(4).^2)))+(2*P2);
sig1=(((F-P1)*l*x(1)/2)/(pi*(x(1).^4)/64))-Maxstr;
sig2=(((P1-P2)*l*x(2)/2)/(pi*(x(2).^4)/64))-Maxstr;
sig3=(((P2)*l*x(3)/2)/(pi*(x(3).^4)/64))-Maxstr;
sig4=(P1/(pi*(x(4).^2)/4))-Maxstr;
sig5=(P2/(pi*(x(5).^2)/4))-Maxstr;
GrFor1=(F-P1)-GroForce;
GrFor2=(P1-P2)-GroForce;
GrFor3=P2-GroForce;
a=[sig1;sig2;sig3;sig4;sig5;GrFor1;GrFor2;GrFor3];
b=[];


function [del1,del2]=OFanch(x,parameters,minmat,maxmat)
l=parameters(1);
E=parameters(2);
den=parameters(3);
F=parameters(4);
P2=F/(3+((12*x(3).^4)/(16*x(5).^2))+((9*(x(2).^4)*(x(3).^4))/(16*16*(x(4).^
2)*(x(5).^2))));
P1=((3*P2*(x(1).^4))/(16*(x(4).^2)))+(2*P2);
del1=((F-P1)*(l.^3))/(3*E*(pi*(x(1).^4)/64));
del2=den*(((pi*l)/4)*((x(1).^2)+(x(2).^2)+(x(3).^2)+(x(4).^2)+(x(5).^2)));

clc;
close all;
clear all;
l=1;                % Lengths of beams
E=70e9;             % Young's modulus
den=2700;           % Density
F=1000;             % Applied force
GroForce=400;       % Maximum ground force
Maxstr=127e6;       % Maximum stress
var=[l E den F GroForce Maxstr];
d1low=1e-3;
d1up=10e-3;
d2low=1e-3;
```

```
d2up=10e-3;
d3low=1e-3;
d3up=10e-3;
dr1low=1e-3;
dr1up=10e-3;
dr2low=1e-3;
dr2up=10e-3;
x0=[(d1low+d1up)/2 (d2low+d2up)/2 (d3low+d3up)/2 (dr1low+dr1up)/2
(dr2low+dr2up)/2];
minmat=[d1low d2low d3low dr1low dr2low];
maxmat=[d1up d2up d3up dr1up dr2up];
options=optimoptions(@fmincon,'Algorithm','interior-
point','display','iter','TolX',1E-9,'Tolfun',1E-9,'DiffMinChange',1E-6);
[x, fval, exitflag,
output]=fmincon('OFanch',x0,[],[],[],[],minmat,maxmat,'const',options,var)
```

Task 4:

```
function [x,f]=contgaf(func,nv,range,pop,gens,mu,matenum)
pops=[]; fitv=[]; nc=pop;
chrom=rand(nc,nv);
a=range(1);
b=range(2);
pops=(b-a)*chrom+a;
for MainIter=1:gens
    for i=1:nc
        fitv(i)=feval(func,pops(i,:));
    end
    [sfit,indexf]=sort(fitv);
    nb=round(matenum*nc);
    if nb/2~=round(nb/2)
        nb=round(matenum*nc)+1;
    end
    fitbest=sfit(1:nb);
    prob=@(n) (nb-n+1)/sum(1:nb);
    rankv=prob([1:nb]);
    for i=1:nb
        cumprob(i)=sum(rankv(1:i));
    end
    mp=round(nb/2);
    randpm=rand(1,mp);
    randpd=rand(1,mp);
    mm=[];
    for j=1:mp
        if randpm(j)<cumprob(1)
            mm=[mm i];
        else
            for i=1:nb-1
                if (randpm(j)>cumprob(i)) && (randpm(j)<cumprob(i+1))
                    mm=[mm i+1];
                end
            end
        end
    end
    md=[];
    md=setdiff([1:nb],mm);
    xp=ceil(rand*nv);
    addpops=[];
    for i=1:mp
```

```
        pd=pops(indexf(md(i)),:);
        pm=pops(indexf(mm(i)),:);
        beta=rand;
        popm(xp)=pm(xp)-beta*(pm(xp)-pd(xp));
        popd(xp)=pd(xp)+beta*(pm(xp)-pd(xp));
        if xp==nv
            ch1=[pm(1:nv-1),pd(nv)];
            ch2=[pd(1:nv-1),pm(nv)];
        else
            ch1=[pd(1:xp),pm(xp+1:nv)];
            ch2=[pm(1:xp),pd(xp+1:nv)];
        end
        ch1(xp)=popm(xp);
        ch2(xp)=popd(xp);
        addpops=[addpops;ch1;ch2];
    end
    newpops=[];
    newpops=[pops(indexf(1:nc-nb),:); addpops];
    Nmut=ceil(mu*nv*(nc-1));
    for k=1:Nmut
        mui=ceil(rand*nc);
        muj=ceil(rand*nv);
        if mui~=indexf(1)
            newpops(mui,muj)=(b-a)*rand+a;
        end
    end
    pops=newpops;
end
f=sfit(1);
x=pops(indexf(1),:);

clc;
clear all;
close all;
ff=@(x) (100*((x(1) + 1/10).^2 + x(2).^2).^(1/2) - 10)*((5*((x(1) +
1/10).^2 - (x(1) - 1/10).^2 + 1/25))/(2*((x(1) + 1/10).^2 + x(2).^2)^(1/2))
- (1 - (25*((x(1) + 1/10).^2 - (x(1) - 1/10).^2 + 1/25)^2)/(4*((x(1) +
1/10).^2 + x(2).^2))).^(1/2)) - (800*((x(1) - 1/10).^2 + x(2).^2).^(1/2) -
80)*((5*((x(1) - 1/10).^2 - (x(1) + 1/10).^2 + 1/25))/(2*((x(1) - 1/10).^2
+ x(2).^2).^(1/2)) - (1 - (25*((x(1) - 1/10).^2 - (x(1) + 1/10).^2 +
1/25).^2)/(4*((x(1) - 1/10).^2 + x(2)^2))).^(1/2)) + 10;
[x,f]=contgaf(ff,2,[1e-3 100e-3],50,50,0.2,0.6)
```