## 2.6. Exercise

**P2.1** Find the roots of the following nonlinear vector equations using the Newton-Raphson method:

$$\mathbf{P}(\mathbf{u}) \equiv \begin{Bmatrix} u_1 + u_2 \\ u_1^2 + u_2^2 \end{Bmatrix} = \begin{Bmatrix} 3 \\ 9 \end{Bmatrix} \equiv \mathbf{f}\,.$$

Use the initial estimate $\mathbf{u}^0 = \{1,\ 5\}^T$ and convergence tolerance = $10^{-5}$. Discuss the convergence rate.

**Solution:**

The Jacobian matrix and the residual term are defined first as

$$\mathbf{K}_T = \frac{\partial \mathbf{P}}{\partial \mathbf{u}} = \begin{bmatrix} 1 & 1 \\ 2u_1 & 2u_2 \end{bmatrix}.$$

$$\mathbf{R} = \mathbf{f} - \mathbf{P}(\mathbf{u}) = \begin{Bmatrix} 3 - u_1 - u_2 \\ 9 - u_1^2 - u_2^2 \end{Bmatrix}.$$

Iteration 0;

$$\mathbf{u}^0 = \begin{Bmatrix} 1 \\ 5 \end{Bmatrix} \quad \mathbf{K}_T^0 = \begin{bmatrix} 1 & 1 \\ 2 & 10 \end{bmatrix} \quad \mathbf{R}^0 = \begin{Bmatrix} -3 \\ -17 \end{Bmatrix} \quad \mathrm{conv} = 3.25$$

Iteration 1;

$$\Delta\mathbf{u}^0 = \begin{Bmatrix} -1.625 \\ -1.375 \end{Bmatrix} \quad \mathbf{u}^1 = \begin{Bmatrix} -0.625 \\ 3.625 \end{Bmatrix} \quad \mathbf{K}_T^0 = \begin{bmatrix} 1 & 1 \\ -1.25 & 7.25 \end{bmatrix} \quad \mathbf{R}^0 = \begin{Bmatrix} 5.266 \\ -4.531 \end{Bmatrix}$$

Iteration 2;

$$\Delta\mathbf{u}^0 = \begin{Bmatrix} 0.533 \\ -0.533 \end{Bmatrix} \quad \mathbf{u}^1 = \begin{Bmatrix} -0.092 \\ 3.092 \end{Bmatrix} \quad \mathbf{K}_T^0 = \begin{bmatrix} 1 & 1 \\ -0.184 & 6.184 \end{bmatrix} \quad \mathbf{R}^0 = \begin{Bmatrix} 0.0 \\ -0.568 \end{Bmatrix}$$

Iteration 3;

$$\Delta\mathbf{u}^0 = \begin{Bmatrix} 0.089 \\ -0.089 \end{Bmatrix} \quad \mathbf{u}^1 = \begin{Bmatrix} -0.003 \\ 3.0003 \end{Bmatrix} \quad \mathbf{K}_T^0 = \begin{bmatrix} 1 & 1 \\ -0.0053 & 6.005 \end{bmatrix} \quad \mathbf{R}^0 = \begin{Bmatrix} 0.0 \\ -0.016 \end{Bmatrix}$$

Iteration 4;

$$\Delta\mathbf{u}^0 = \begin{Bmatrix} 0.003 \\ -0.003 \end{Bmatrix} \quad \mathbf{u}^1 = \begin{Bmatrix} -0.0 \\ 3.0 \end{Bmatrix} \quad \mathbf{K}_T^0 = \begin{bmatrix} 1 & 1 \\ 0.0 & 6.0 \end{bmatrix} \quad \mathbf{R}^0 = \begin{Bmatrix} 0.0 \\ -0.00001 \end{Bmatrix}$$

Below is the list of MATLAB programs that solve the nonlinear equations.

```
%
% P2.1 Newton-Raphson method
%
tol = 1.0e-5;  iter = 0;  c = 0;
u = [1; 5];    uold = u;
f = [3; 9];
P = [u(1)+u(2); u(1)^2+u(2)^2];
R = f - P;
conv= (R(1)^2+R(2)^2)/(1+f(1)^2+f(2)^2);
fprintf('\n iter       u1       u2         conv       c');
fprintf('\n %3d  %8.5f %8.5f %12.3e %7.5f',iter,u(1),u(2),conv,c);
while conv > tol && iter < 20
 Kt = [1 1; 2*u(1) 2*u(2)];
 delu = Kt\R;
 u = uold + delu;
 P = [u(1)+u(2); u(1)^2+u(2)^2];
 R = f - P;
 conv= (R(1)^2+R(2)^2)/(1+f(1)^2+f(2)^2);
 c = abs(3-u(2))/abs(3-uold(2))^2;
 iter = iter + 1;
 uold = u;
 fprintf('\n %3d  %8.5f %8.5f %12.3e %7.5f',iter,u(1),u(2),conv,c);
end
```

The following table shows the convergence history of the Newton-Raphson method. Note that the residual reduction is approximately quadratic, and the parameter *c* converges to a constant. Thus, the convergence rate is two.

| Iteration | $u_1$ | $u_2$ | conv | c |
|---|---|---|---|---|
| 0 | 1.0000 | 5.0000 | 3.275E+00 | 0.000 |
| 1 | −0.6250 | 3.6250 | 2.256E−01 | 0.156 |
| 2 | −0.0919 | 3.0919 | 3.550E−03 | 0.235 |
| 3 | −0.0027 | 3.0027 | 2.790E−06 | 0.314 |
| 4 | −0.0000 | 3.0000 | 2.171E−12 | 0.333 |
| 5 | −0.0000 | 3.0000 | 1.324E−24 | 0.333 |

**P2.2** Using the modified Newton-Raphson method, solve the nonlinear equations in P2.1. Compare the convergence rate with the Newton-Raphson method.

**Solution**:
In the modified Newton-Raphson method, a constant Jacobian matrix is used for all iterations. Below is the list of MATLAB programs for solving the problem.

```
%
% P2.2 Modified Newton-Raphson method
%
tol = 1.0e-5;  iter = 0;  c = 0;
u = [1; 5];    uold = u;
f = [3; 9];
P = [u(1)+u(2); u(1)^2+u(2)^2];
```

```
R = f - P;
conv= (R(1)^2+R(2)^2)/(1+f(1)^2+f(2)^2);
Kt = [1 1; 2*u(1) 2*u(2)];
fprintf('\n iter       u1       u2         conv       c');
fprintf('\n %3d  %8.5f %8.5f %12.3e %7.5f',iter,u(1),u(2),conv,c);
while conv > tol && iter < 20
delu = Kt\R;
 u = uold + delu;
 P = [u(1)+u(2); u(1)^2+u(2)^2];
 R = f - P;
 conv= (R(1)^2+R(2)^2)/(1+f(1)^2+f(2)^2);
 c = abs(3-u(2))/abs(3-uold(2))^2;
 iter = iter + 1;
 uold = u;
 fprintf('\n %3d  %8.5f %8.5f %12.3e %7.5f',iter,u(1),u(2),conv,c);
end
fprintf('\n');
```

The following table shows the convergence history of the modified Newton-Raphson method. Note that the rate of residual reduction is slower than that of the standard Newton-Raphson method, and the parameter *c* does not converges to a constant.

| Iteration | $u_1$ | $u_2$ | conv | c |
|---|---|---|---|---|
| 0 | 1.0000 | 5.0000 | 3.275E+00 | 0.000 |
| 1 | −0.6250 | 3.6250 | 2.256E−01 | 0.156 |
| 2 | −0.0586 | 3.0586 | 1.412E−03 | 0.150 |
| 3 | −0.0138 | 3.0138 | 7.592E−05 | 4.017 |
| 4 | −0.0034 | 3.0034 | 4.584E−06 | 17.879 |
| 5 | −0.0009 | 3.0009 | 2.840E−07 | 73.280 |

**P2.3** Using the Broyden method, solve the nonlinear equations in P2.1. Compare the convergence rate with the Newton-Raphson method.

**Solution:**

The Broyden method uses the exact Jacobian at the first iteration, and progressively update it at every iteration using the increments of solution and increments of residuals. Below is the list of MATLAB programs that solves the nonlinear equations.

```
%
% P2.3 Broyden method
%
tol = 1.0e-7;  iter = 0;  c = 0;
u = [1; 5];    uold = u;
f = [3; 9];
P = [u(1)+u(2); u(1)^2+u(2)^2];
R = P - f;     Rold = R;
conv= (R(1)^2+R(2)^2)/(1+f(1)^2+f(2)^2);
Kt = [1 1; 2*u(1) 2*u(2)];
fprintf('\n iter       u1       u2         conv       c');
fprintf('\n %3d  %8.5f %8.5f %12.3e %7.5f',iter,u(1),u(2),conv,c);
while conv > tol && iter < 20
 delu = -Kt\R;
 u = uold + delu;
 P = [u(1)+u(2); u(1)^2+u(2)^2];
```

```
 R = P - f;
 conv= (R(1)^2+R(2)^2)/(1+f(1)^2+f(2)^2);
 c = abs(3-u(2))/abs(3-uold(2))^2;
 delR = R - Rold;
 Kt = Kt + (delR-Kt*delu)*delu'/norm(delu)^2;
 uold = u;  Rold = R;
 iter = iter + 1;
 fprintf('\n %3d  %8.5f %8.5f %12.3e %7.5f',iter,u(1),u(2),conv,c);
end
fprintf('\n');
```

The following table shows the convergence history of the Broyden method. Note that the rate of residual reduction is slower than that of the standard Newton-Raphson method, and the parameter *c* does not converges to a constant. However, the method performs a little better than the modified Newton-Raphson method.

| Iteration | $u_1$ | $u_2$ | conv | c |
|---|---|---|---|---|
| 0 | 1.0000 | 5.0000 | 3.275E+00 | 0.000 |
| 1 | −0.6250 | 3.6250 | 2.256E−01 | 0.156 |
| 2 | −0.0758 | 3.0758 | 2.387E−03 | 0.194 |
| 3 | −0.0128 | 3.0128 | 6.531E−05 | 2.229 |
| 4 | −0.0003 | 3.0003 | 3.897E−08 | 1.917 |

**P2.4** Using the incremental force method, solve the equations in P2.1. Use five equal-interval load steps.

**Solution**:
Below is the list of MATLAB programs for solving the problem.

```
%
% P2.4 Incremental force method
%
tol = 1.0e-5;   u = [1; 5];
fprintf('\n  inc       u1       u2       f1       f2');
for i=1:5
 f = i*0.2*[3; 9];
 P = [u(1)+u(2); u(1)^2+u(2)^2];
 R = f - P;
 conv= (R(1)^2+R(2)^2)/(1+f(1)^2+f(2)^2);
 iter = 0;
 while norm(R) > tol && iter < 20
  Kt = [1 1; 2*u(1) 2*u(2)];
  delu = Kt\R;
  u = u + delu;
  P = [u(1)+u(2); u(1)^2+u(2)^2];
  R = f - P;
  conv= (R(1)^2+R(2)^2)/(1+f(1)^2+f(2)^2);
  iter = iter + 1;
 end
 fprintf('\n %3d  %8.5f %8.5f %8.5f %8.5f',i,u(1),u(2),f(1),f(2));
end
fprintf('\n');
```

The following table shows the convergence history of the incremental force method. Note that the force terms increase regularly as they are inputs, while the displacement terms are irregular.

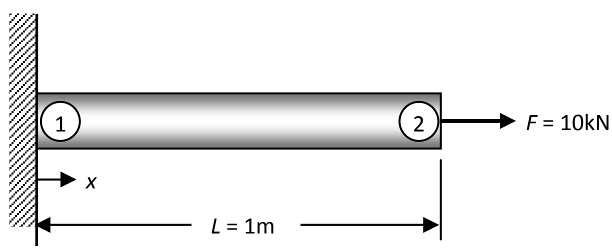
| Increment | $u_1$ | $u_2$ | $f_1$ | $f_2$ |
|---|---|---|---|---|
| 1 | −0.6000 | 1.2000 | 0.6000 | 1.8000 |
| 2 | −0.6000 | 1.8000 | 1.2000 | 3.6000 |
| 3 | −0.4748 | 2.2748 | 1.8000 | 5.4000 |
| 4 | −0.2697 | 2.6697 | 2.4000 | 7.2000 |
| 5 | −0.0003 | 3.0000 | 3.0000 | 9.0000 |

**P2.5** Consider a uniform bar with a constant Young's modulus $E = 100\text{MPa}$, cross-sectional area $A = 2\times10^{-4}\text{m}^2$, and a unit length $L = 1\text{m}$. The applied force $F = 10\text{kN}$ is large enough such that the relation between displacement and strain is nonlinear:

$$\varepsilon(u) = \frac{\mathrm{d}u}{\mathrm{d}x} + \frac{1}{2}\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)^2$$

Using a single two-node bar element, calculate the displacement at the tip and strain of the element. Assume stress-strain relation is linear and constant cross-sectional area and length. Use an increment force method with ten equal force increments.
**Hint:** The virtual strain can be obtained through variation of the strain as

$$\varepsilon(\overline{u}) = \frac{\mathrm{d}\overline{u}}{\mathrm{d}x} + \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}\overline{u}}{\mathrm{d}x}$$




**Solution:**
For a bar element, the displacement is interpolated by

$$u(x) = N_1(x)d_1 + N_2(x)d_2 = \frac{x}{L}d_2$$

Note that the boundary condition, d1 = 0, is applied. From the displacement-strain relation, the strain is interpolated by

$$\varepsilon = \frac{1}{2L^2}(d_2^2 + 2Ld_2)$$

In the weak form of the bar requires the virtual strain $\varepsilon(\overline{u})$, which can be written in terms of the virtual nodal displacement.

$$\varepsilon(\overline{u}) = \frac{1}{L^2}(d_2 + L)\overline{d}_2$$

The relation between stress and strain is linear:

$$\sigma = E\varepsilon = \frac{E}{2L^2}(d_2^2 + 2Ld_2)$$

The weak form of the nonlinear equation becomes

$$\overline{d}_2\left[\int_0^L \frac{1}{L^2}(d_2 + L)\frac{E}{2L^2}(d_2^2 + 2Ld_2)A\mathrm{d}x = F\right]$$

After integration, the nonlinear finite element equation can be obtained as

$$\frac{EA}{2L^3}(d_2^3 + 3Ld_2^2 + 2L^2d_2) = F$$

Below is the list of MATLAB programs that solves for the above nonlinear equation.

```
%
% P2.5 Nonlinear strain bar
%
fprintf('\n inc        F       u  strain');
tol = 1.e-5;  u = 0;
for i=1:10
 strain = 0.5*u^2 + u;
 f = 0.1*i;
 iter = 0;
 P = u^3+3*u^2+2*u;
 R = f - P;
 conv= R^2/(1+f^2);
 while conv > tol && iter < 20
  Kt = 3*u^2+6*u+2;
  delu = R/Kt;
  u = u + delu;
  strain = 0.5*u^2 + u;
  P = u^3+3*u^2+2*u;
  R = f - P;
  conv= R^2/(1+f^2);
  iter = iter + 1;
 end
 fprintf('\n %3d  %7.5f %7.5f %7.5f',i,f,u,strain);
end
```

The following table shows the force, displacement, and strain at each force increment. When displacement is small (e.g., the first increment), the different between displacement and strain is small, while the difference becomes large as displacement increases. This is because of the nonlinear displacement-strain relation.

| Increment | Force (kN) | Displacement | Strain |
|---|---|---|---|
| 1 | 1.0 | 0.0467 | 0.0478 |
| 2 | 2.0 | 0.0880 | 0.0919 |
| 3 | 3.0 | 0.1254 | 0.1333 |
| 4 | 4.0 | 0.1597 | 0.1725 |

| | | | |
|---|---|---|---|
| 5 | 5.0 | 0.1915 | 0.2098 |
| 6 | 6.0 | 0.2222 | 0.2469 |
| 7 | 7.0 | 0.2499 | 0.2812 |
| 8 | 8.0 | 0.2763 | 0.3144 |
| 9 | 9.0 | 0.3013 | 0.3467 |
| 10 | 10.0 | 0.3252 | 0.3781 |

**P2.6** Solve Problem P2.5 using the secant method. Do not use the incremental force method. Discuss about the convergence rate.

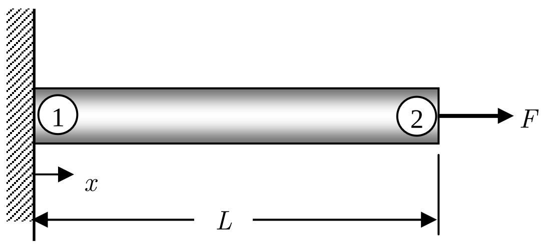
**Solution:**
For the derivation of nonlinear equation, refer to P2.5. Below is the list of MATLAB programs that solve for the nonlinear equation.

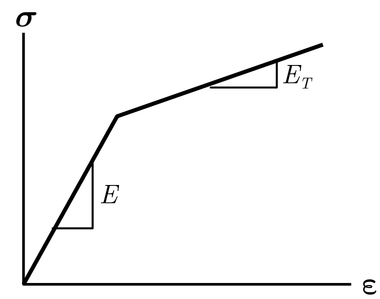
```
%
% P2.6 Nonlinear strain bar (secant method)
%
tol = 1.0e-5;      iter = 0;
u = 0.0;           uold = u;   f = 1;
P = u^3+3*u^2+2*u; Pold = P;
R = f - P;         conv= R^2/(1+f^2);
strain = 0.5*u^2 + u;
fprintf('\n iter       u  strain         conv');
fprintf('\n %3d  %7.5f %7.5f %12.3e',iter,u,strain,conv);
Ks = 3*u^2+6*u+2;
while conv > tol && iter < 20
 delu = R/Ks;
 u = uold + delu;
 strain = 0.5*u^2 + u;
 P = u^3+3*u^2+2*u;
 R = f - P;
 conv= R^2/(1+f^2);
 Ks = (P - Pold)/(u - uold);
 uold = u;
 Pold = P;
 iter = iter + 1;
 fprintf('\n %3d  %7.5f %7.5f %12.3e',iter,u,strain,conv);
end
```

The following table shows the convergence history. It converges in the fourth iteration. The convergence criterion reduces rapidly, but not as fast as the Newton-Raphson method.

| Iteration | Displacement | Strain | conv |
|---|---|---|---|
| 0 | 0.0000 | 0.0000 | 5.000E−01 |
| 1 | 0.5000 | 0.6250 | 3.838E−01 |
| 2 | 0.2667 | 0.3022 | 2.746E−02 |
| 3 | 0.3160 | 0.3659 | 6.859E−04 |
| 4 | 0.3252 | 0.3781 | 2.241E−06 |

**P2.7** Consider a uniform bar with cross-sectional area $A = 1\times10^{-4}\mathrm{m}^2$ and a unit length $L = 1\mathrm{m}$. The bar shows elasto-plastic material behavior as depicted in the figure. The plastic deformation starts at yields stress $\sigma_Y = 400\mathrm{MPa}$. In the elastic region, the Young's modulus $E = 200\mathrm{GPa}$, while in the plastic region, the tangent stiffness is $E_T = 20\mathrm{GPa}$. When a force $F = 50\mathrm{kN}$ is applied at the end, calculate tip displacement and stress of the element using one bar element. Use 10 equal-interval force increments. Plot the force-displacement curve. Assume displacement-strain relation is linear.




**Solution:**
The discrete weak form of the bar element can be written as

$$\bar{\mathbf{d}}^{\mathrm{T}}\int_0^L \mathbf{B}^{\mathrm{T}}\sigma A\,\mathrm{d}x = \bar{\mathbf{d}}^{\mathrm{T}}\mathbf{F}$$

where $\mathbf{d} = [d_1,\ d_2]^{\mathrm{T}}$ is the vector of nodal displacements, $\bar{\mathbf{d}}$ is the vector of virtual nodal displacements, $\mathbf{B}^{\mathrm{T}} = [-1,\ 1]/L$ is the displacement-strain matrix, and $\mathbf{F} = [F_1,\ F_2]^{\mathrm{T}}$ is the vector of applied forces. In order to simplify the following steps, the essential boundary condition can be applied in advance; i.e., $d_1 = \bar{d}_1 = 0$ . For simplicity of notation, $d = d_2$ and $F = F_2$ will be used in the following derivations. Then, the above discrete weak form becomes a scalar equation. The residual now becomes

$$\begin{aligned} R &= F - \int_0^L \frac{\sigma A}{L}\mathrm{d}x \\ \Rightarrow R &= F - \sigma(d)A \end{aligned}$$

Note that the residual is nothing but the equilibrium between external and internal forces: $P(d) = F$. The nonlinearity comes from stress calculation. Initially, the stress increases linearly with strain until it reaches the yield stress. The value of displacement at yield is

$$u_Y = \varepsilon_Y L = \frac{\sigma_Y L}{E} = 0.002$$

Thus, the internal force term is determined based on the magnitude of displacement as

$$\begin{cases} P(d) = \dfrac{EA}{L}d & \text{if}\quad d < d_Y \\ P(d) = \sigma_Y A + \dfrac{E_T A}{L}(d - d_Y) & \text{otherwise} \end{cases}$$

The Jacobian relation and stress calculation also have two branches as

$$\begin{cases} K_T(d) = \dfrac{EA}{L} & \text{if} \quad d < d_Y \\ K_T(d) = \dfrac{E_T A}{L} & \text{otherwise} \end{cases}$$

$$\begin{cases} \sigma(d) = \dfrac{E}{L} d & \text{if} \quad d < d_Y \\ \sigma(d) = \sigma_Y + \dfrac{E_T}{L}(d - d_Y) & \text{otherwise} \end{cases}$$

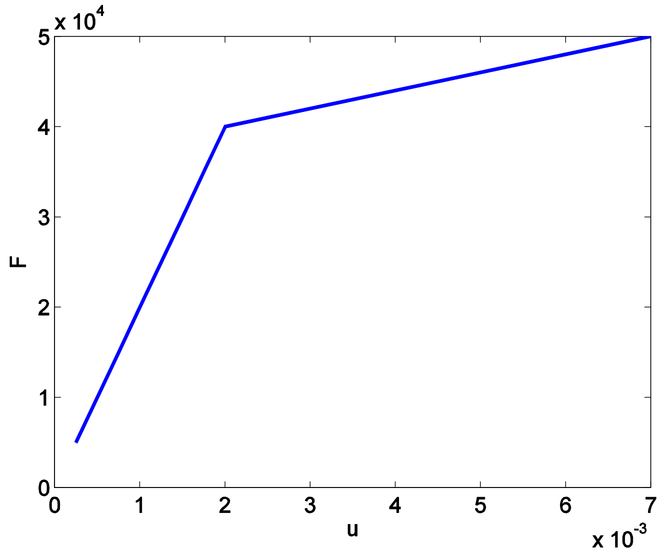
Note that actually elasto-plastic material show much more complex behavior that the one explained above. However, when the load is monotonically increasing, the above formulas work. Below is the list of MATLAB program that solves the elasto-plastic bar with ten increments.

```
%
% P2.7 Elasto-plastic bar
%
fprintf('\n inc         F       u     stress');
tol = 1.e-5;  u = 0;  uY = 0.002;
for i=1:10
 f = i*5000;
 iter = 0;
 if u < uY P = 2E7*u;
 else      P = 2E7*uY + 2E6*(u-uY); end
 R = f - P;
 conv= R^2/(1+f^2);
 while conv > tol && iter < 20
  if u < uY Kt = 2E7;
  else      Kt = 2E6; end
  delu = R/Kt;
  u = u + delu;
  if u < uY P = 2E7*u;
  else      P = 2E7*uY + 2E6*(u-uY); end
  R = f - P;
  conv= R^2/(1+f^2);
  iter = iter + 1;
 end
 if u < uY stress = 2E11*u;
 else      stress = 4E8 + 2E10*(u-uY); end
 fprintf('\n %3d  %8.1f %7.5f %10.3e',i,f,u,stress);
end
```

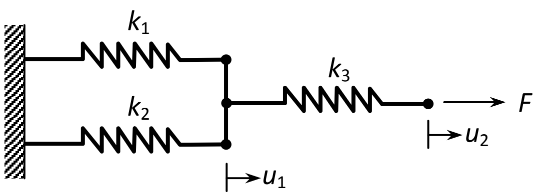
The following table and figure show the force-displacement relation at each increment. It is clear that plastic deformation starts at $F$ = 40kN at which stress becomes 400MPa. Since the stress-strain relation is linear in each increment, it only requires one iteration per increment.

| Increment | Force (kN) | Displacement | Stress (MPa) |
|---|---|---|---|
| 1 | 5.0 | 0.00025 | 50 |
| 2 | 10.0 | 0.00050 | 100 |
| 3 | 15.0 | 0.00075 | 150 |
| 4 | 20.0 | 0.00100 | 200 |
| 5 | 25.0 | 0.00125 | 250 |
| 6 | 30.0 | 0.00150 | 300 |

| | | | |
|---|---|---|---|
| 7 | 35.0 | 0.00175 | 350 |
| 8 | 40.0 | 0.00200 | 400 |
| 9 | 45.0 | 0.00450 | 450 |
| 10 | 50.0 | 0.00700 | 500 |




**P2.8** Consider three nonlinear springs in the figure. The stiffness of each spring is given by $k_1 = 500 + 50u$, $k_2 = 200 + 100u$, and $k_3 = 500 + 100u$ where $u$ is the elongation of the spring. Solve the displacements at Nodes 1 and 2 using the Newton-Raphson method when $F = 100$.




**Solution:**

The finite element matrix equation for the three nonlinear springs can be written as

$$\begin{bmatrix} k_1 + k_2 + k_3 & -k_3 \\ -k_3 & k_3 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ F \end{Bmatrix}$$

By substituting the stiffness for all spring elements, the following nonlinear equations can be obtained:

$$\begin{cases} 500u_1^2 + 200u_1u_2 - 100u_2^2 + 1200u_1 - 500u_2 = 0 \\ 100u_1^2 - 200u_1u_2 + 100u_2^2 - 500u_1 + 500u_2 = 100 \end{cases}$$

which is in the form of $\mathbf{P}(\mathbf{u}) = \mathbf{F}$. The Jacobian matrix can be obtained by differentiating the above relation

$$\mathbf{K}_T = \begin{bmatrix} 100u_1 + 200u_2 + 1200 & 200u_1 - 200u_2 - 500 \\ 200u_1 - 200u_2 - 500 & -200u_1 + 200u_2 + 500 \end{bmatrix}$$

Below is the list of MATLAB programs that solves for the equilibrium of the three nonlinear springs

```
%
% P2.8 Three nonlinear springs
%
tol = 1.0e-5;  iter = 0;
u = [0; 0];    uold = u;
f = [0; 100];
P = [50*u(1)^2+200*u(1)*u(2)-100*u(2)^2+1200*u(1)-500*u(2)
     100*u(1)^2-200*u(1)*u(2)+100*u(2)^2-500*u(1)+500*u(2)];
R = f - P;
conv= (R(1)^2+R(2)^2)/(1+f(1)^2+f(2)^2);
fprintf('\n iter      u1      u2         conv');
fprintf('\n %3d  %7.5f %7.5f %12.3e',iter,u(1),u(2),conv);
while conv > tol && iter < 20
 Kt = [100*u(1)+200*u(2)+1200   200*u(1)-200*u(2)-500
       200*u(1)-200*u(2)-500   -200*u(1)+200*u(1)+500];
 delu = Kt\R;
 u = uold + delu;
 P = [50*u(1)^2+200*u(1)*u(2)-100*u(2)^2+1200*u(1)-500*u(2)
      100*u(1)^2-200*u(1)*u(2)+100*u(2)^2-500*u(1)+500*u(2)];
 R = f - P;
 conv= (R(1)^2+R(2)^2)/(1+f(1)^2+f(2)^2);
 uold = u;
 iter = iter + 1;
 fprintf('\n %3d  %7.5f %7.5f %12.3e',iter,u(1),u(2),conv);
end
```

The following table shows the convergence history of the Newton-Raphson method. Note that the iteration converges at the third iteration.

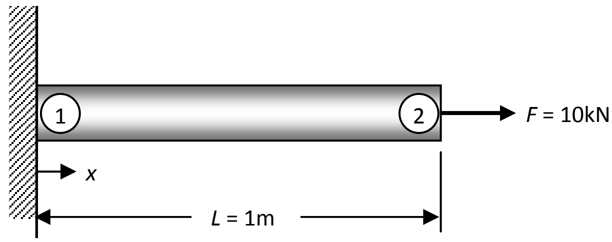
| Iteration | $u_1$ | $u_2$ | conv |
|---|---|---|---|
| 0 | 0.0000 | 0.0000 | 9.999E−01 |
| 1 | 0.1429 | 0.3429 | 1.688E−03 |
| 2 | 0.1380 | 0.3296 | 2.722E−05 |
| 3 | 0.1388 | 0.3316 | 5.390E−07 |

**P2.9** Consider a uniform bar in the figure. The stress-strain relation and displacement-strain relation are linear. However, the Young's modulus of the material varies according to the strain.

$$\sigma = E(u)\varepsilon(u), \quad \varepsilon(u) = \frac{\mathrm{d}u}{\mathrm{d}x}, \quad E(u) = E_0\left(1 - \frac{\mathrm{d}u}{\mathrm{d}x}\right)$$

When one element is used to model the bar, formulate the nonlinear equation with the tip displacement being unknown. Solve the tip displacement using the incremental force method with ten equal-interval increments. Use $E_0 = 1.0\text{GPa}$, $A = 10^{-4}\text{m}^2$, and $F =$

$25\text{kN}$. Plot the force-displacement curve. Test what happens when $F = 30\text{kN}$, and explain why.




**Solution:**

Since Node 1 is fixed for the element, it can be deleted. Thus, the weak form of the nonlinear bar becomes a single DOF equation. By denoting $d = d_2$ and $F = F_2$, the nonlinear equation becomes $P(d) = F$, where the internal force term is defined by

$$P(d) = \sigma(d)A = EA\varepsilon(d) = \frac{E_0 A}{L}\left(\varepsilon(d) - \varepsilon(d)^2\right)$$

Note that since the length of the element is a unit and since a linear bar element is used, displacement is identical to strain. Due to strain-dependent material properties, the Newton-Raphson method is used to find the displacement $d$. The Jacobian becomes

$$K_T = \frac{\mathrm{d}P}{\mathrm{d}d} = \frac{E_0 A}{L}(1 - 2d)$$

The first derivative on the right-hand side can be calculated by differentiating the stress-strain relation, and the second derivative from displacement-strain relation. Using these relations, we have

$$K_T = \frac{\mathrm{d}P}{\mathrm{d}d} = \frac{E_0 A}{L}(2d + 1)$$

Below is the list of MATLAB programs that solves for the nonlinear modulus bar.

```
%
% P2.9 Nonlinear modulus bar
%
fprintf('\n inc iter       F       u       stress');
tol = 1.e-5;  u = 0;
for i=1:10
 f = i*2500;
 iter = 0;
 stress = 1E9*(1-u)*u;
 P = stress*1E-4;
 R = f - P;
 conv= R^2/(1+f^2);
 while conv > tol && iter < 20
  Kt = 1E5*(1-2*u);
  delu = R/Kt;
  u = u + delu;
  stress = 1E9*(1-u)*u;
  P = stress*1E-4;
  R = f - P;
```

```
  conv= R^2/(1+f^2);
  iter = iter + 1;
 end
 fprintf('\n %3d %3d  %7.1f %7.5f %12.3e',i,iter,f,u,stress);
end
```

The following table and figure show the force-displacement relation during the force increments. Each increment converges in the second iteration. Note that the slope of the force-displacement curve becomes zero as the force approaches 25kN. In fact, the force will decrease if displacement further increases. However, this cannot be solved using Newton-Raphson method as the tangent stiffness becomes singular.

| Increment | Force (kN) | Displacement | Stress (MPa) |
|---|---|---|---|
| 1 | 2.5 | 0.0257 | 25.0 |
| 2 | 5.0 | 0.0528 | 50.0 |
| 3 | 7.5 | 0.0817 | 75.0 |
| 4 | 10.0 | 0.1127 | 100.0 |
| 5 | 12.5 | 0.1464 | 125.0 |
| 6 | 15.0 | 0.1838 | 150.0 |
| 7 | 17.5 | 0.2261 | 175.0 |
| 8 | 20.0 | 0.2764 | 200.0 |
| 9 | 22.5 | 0.3416 | 224.9 |
| 10 | 25.0 | 0.4802 | 249.6 |