Mechanics of Structures (CE130N)
Lab 14

1 Objective

The objective of this lab is to understand how one can solve non-linear equations by using the Newton-Raphson method. Though many classes of problems in mechanics can be treated with linear equations, there are an even larger class of interesting problems which are governed by non-linear equations. The Newton-Raphson method is one method which one can apply to solve these equations. The Newton-Raphson method is motivated through 1 variables and extended to multi-variables. The extension is straightforward such that a well written MATLAB code should work for both cases. Problems in 1 to 3 variables are treated with examples from mechanics to illustrate the method.

2 Newton-Raphson method in 1-variable

Let $f(u)$ be a non-linear function in u such that one would like to find the solution to the equation,

$$f(u) = 0 \; .$$

The idea behind Newton-Raphson is to iteratively approach the exact solution. Given an initial guess of u^0, one would like to fix this guess by adding Δu such that $u^0 + \Delta u$ is the exact solution,

$$f(u^0 + \Delta u) = 0.$$

Conducting a Taylor series expansion around $u = u^0$ one has,

$$0 = f(u^0 + \Delta u) = f(u^0) + f'(u^0)\Delta u + (\text{higher order terms in } \Delta u) \; .$$

This expansion is approximated as,

$$0 = f(u^0 + \Delta u) \approx f(u^0) + f'(u^0)\Delta u \; ,$$

such that one solves,

$$0 = f(u^0) + f'(u^0)\Delta u \; ,$$

to find the update Δu by,

$$\Delta u \;\; = \;\; -f'(u^0)^{-1} f(u^0) \; .$$

This yields a new approximation u^1 for the solution,

$$\begin{aligned} u^1 &= u^0 + \Delta u \\ &= u^0 - f'(u^0)^{-1} f(u^0) \; . \end{aligned}$$

One can proceed to find better approximations by repeating the procedure above, replacing u^0 with u^1 and so on, and continue untill one has found a solution which is good enough, i.e., $|f(u^n)| < \text{tolerance}$. The steps can be summarized as follows,

1. Guess initial solution $\hat{u} = u^0$, and compute residual $f(\hat{u})$.
2. While $|f(\hat{u})| > \text{abstol}$,
   (a) Compute tangent stiffness: $f'(\hat{u})$.
   (b) Compute update: $\Delta u = -f'(\hat{u})^{-1} f(\hat{u})$.
   (c) Update solution: $\hat{u} \leftarrow \hat{u} + \Delta u$.
   (d) Compute new residual: $f(\hat{u})$.

There are two aspects of the Newton-Raphson method that one should always keep in mind,

- The Newton-Raphson method is *LOCALLY* convergent. This means that unless the initial guess u^0 is close enough to the exact solution, you may not converge to the solution.
- The Newton-Raphson method is locally *QUADRATICALLY* convergent. This means that when the solution starts converging, the error (residual) decreases quadratically with increasing iteration, i.e., you get twice as many digits of accuracy with each iteration.

3 Newton-Raphson method in N-variables

Let $f_i(u_1, \ldots, u_N)$ $(i = 1, \ldots, N)$ be N non-linear functions. Each function is a function in N variables, $u_1, \ldots, u_N$. One can use the compact notation,

$$\mathbf{f}(\mathbf{u}),$$

to denote this, where we have defined,

$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix}$$

$$\mathbf{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix} .$$

We would like to find the solution $\mathbf{u}$ to the equation,

$$\mathbf{f}(\mathbf{u}) = \mathbf{0} .$$

We can approach this in the same way as the 1-variable case, such that we would like to iteratively find the exact solution. Given an initial guess of $\mathbf{u}^0$, one would like to fix this guess by adding $\Delta\mathbf{u}$ so that $\mathbf{u}^0 + \Delta\mathbf{u}$ is the exact solution,

$$\mathbf{f}(\mathbf{u}^0 + \Delta\mathbf{u}) = \mathbf{0} ,$$

or equivalently,

$$f_i(u_1^0 + \Delta u_1, \ldots, u_N^0 + \Delta u_N) = 0 \quad (i = 1, \ldots, N) .$$

Conducting a Taylor series expansion around $\mathbf{u} = \mathbf{u}_0$, one has for equation i,

$$0 = f_i(u_1^0 + \Delta u_1, \ldots, u_N^0 + \Delta u_N) = f_i(\mathbf{u}^0) + \frac{\partial f_i}{\partial u_1}(\mathbf{u}^0)\Delta u_1 + \cdots + \frac{\partial f_i}{\partial u_N}(\mathbf{u}^0)\Delta u_N + (\text{higher order terms}) .$$

Using the notation introduced in the Principle of Stationary Potential Energy where,

$$\frac{\partial f_i}{\partial \mathbf{u}}(\mathbf{u}^0) = \begin{bmatrix} \frac{\partial f_i}{\partial u_1}(\mathbf{u}^0) \\ \vdots \\ \frac{\partial f_i}{\partial u_N}(\mathbf{u}^0) \end{bmatrix} ,$$

one can write compactly,

$$0 = f_i(\mathbf{u}^0 + \Delta\mathbf{u}) = f_i(\mathbf{u}^0) + \frac{\partial f_i}{\partial \mathbf{u}}(\mathbf{u}_0) \cdot \Delta\mathbf{u} + (\text{higher order terms}) .$$

Additionally,

$$\mathbf{0} = \mathbf{f}(\mathbf{u}^0 + \Delta\mathbf{u}) = \mathbf{f}(\mathbf{u}^0) + \begin{bmatrix} \frac{\partial f_1}{\partial \mathbf{u}}(\mathbf{u}^0) \cdot \Delta\mathbf{u} \\ \vdots \\ \frac{\partial f_N}{\partial \mathbf{u}}(\mathbf{u}^0) \cdot \Delta\mathbf{u} \end{bmatrix} + (\text{higher order terms}) .$$

Defining the matrix,

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}^0),$$

whose (i,j) entry is,

$$\frac{\partial f_i}{\partial u_j}(\mathbf{u}^0),$$

one finally has the compact expression,

$$\begin{aligned}\mathbf{0} = \mathbf{f}(\mathbf{u}^0 + \Delta\mathbf{u}) = \mathbf{f}(\mathbf{u}^0) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}^0) \cdot \Delta\mathbf{u} \\ + (\text{higher order terms}) \ .\end{aligned}$$

This expansion is approximated as,

$$\mathbf{0} = \mathbf{f}(\mathbf{u}^0 + \Delta\mathbf{u}) \approx \mathbf{f}(\mathbf{u}^0) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}^0) \cdot \Delta\mathbf{u} \ ,$$

such that one solves,

$$\mathbf{0} = \mathbf{f}(\mathbf{u}^0) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}^0) \cdot \Delta\mathbf{u} \ ,$$

to find the update $\Delta\mathbf{u}$ by,

$$\Delta\mathbf{u} \quad = \quad -\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}^0)^{-1}\mathbf{f}(\mathbf{u}^0) \ .$$

This yields a new approximation $\mathbf{u}^1$ for the solution,

$$\begin{aligned}\mathbf{u}^1 &= \mathbf{u}^0 + \Delta\mathbf{u} \\ &= \mathbf{u}^0 - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}^0)^{-1}\mathbf{f}(\mathbf{u}^0) \ .\end{aligned}$$

One can proceed to find better approximations by repeating the procedure above, replacing $\mathbf{u}^0$ with $\mathbf{u}^1$ and so on, and continue until one has found a solution which is good enough, i.e., $\|\mathbf{f}(\mathbf{u}^n)\| < \text{tolerance}$. The steps can be summarized as follows,

1. Guess initial solution $\hat{\mathbf{u}} = \mathbf{u}^0$, and compute residual $\mathbf{f}(\hat{\mathbf{u}})$.
2. While $\|\mathbf{f}(\hat{\mathbf{u}})\| > \text{abstol}$,
   (a) Compute tangent stiffness: $\dfrac{\partial \mathbf{f}}{\partial \mathbf{u}}(\hat{\mathbf{u}})$.
   (b) Compute update: $\Delta\mathbf{u} = -\dfrac{\partial \mathbf{f}}{\partial \mathbf{u}}(\hat{\mathbf{u}})^{-1}\mathbf{f}(\hat{\mathbf{u}})$.
   (c) Update solution: $\hat{\mathbf{u}} \leftarrow \hat{\mathbf{u}} + \Delta\mathbf{u}$.
   (d) Compute new residual: $\mathbf{f}(\hat{\mathbf{u}})$.

As one observes, the steps are completely identical to the 1 variable case.

4 Exercise

4.1 Download files

1. Download the file `newton_raphson.zip` into your `ce130n/programs` directory and unzip it.
2. Go to the `ce130n/programs/newton_raphson/exercise/` directory, and execute the file `init.m`. This will set the necessary paths to run the files.

YOU MUST RUN THE FILE init.m EVERYTIME YOU START UP MATLAB.

4.2 1-variable case

4.2.1 Complete Newton-Raphson function

Complete the function `newton_raphson.m` by filling in the appropriate lines.

4.2.2 1 variable example

Run the Newton-Raphson scheme to solve the non-linear equation,

$$f(u) = (u-1)^2 - \frac{1}{2} \; .$$

The derivative is computed as,

$$f'(u) = 2(u-1) \; .$$

To solve for the solution to $f(u) = 0$ using the function `newton_raphson.m`, one must define the function handles for the residual $f(u)$ and tangent stiffness $f'(u)$. The example can be run with the following MATLAB code,

```
>> param.N = 1;                    % -- Number of variables
>> f_resid = @(z) (z-1)^2-1/2;     % -- Define residual
>> f_stiff = @(z) 2*(z-1);         % -- Define stiffness
>> param.u0= 0;                    % -- Define starting value
>> param.history=1;                % -- Save iteration history
>> nrsol = newton_raphson(f_resid,f_stiff,param); % -- Compute N-R
>> plot_history(nrsol);            % -- Plot iteration vs. residual
>> plot1d_history(f_resid,nrsol,param); % -- Plot convergence of solution
```

You can also specify the range to plot the figures. Type

```
>> help plot_history
>> help plot1d_history
```

for more details.

Things to check:

- Make sure you understand how the Newton-Raphson proceeds.
- Does the solution change with the initial guess? If so, for which initial guess do you get which solution?
- Are there values which you do not get a solution? Why?
- How does the residual get smaller with each iteration near the last few iterations? Do you see quadratic convergence?
- Try some other functions for $f(u)$ and observe the behavior of obtaining a solution.

4.3 2 variable example

4.3.1 Simple example

Run the Newton-Raphson scheme to solve the non-linear equation,

$$\mathbf{f}(\mathbf{u}) = \begin{bmatrix} f_1(u_1,u_2) \\ f_2(u_1,u_2) \end{bmatrix} = \begin{bmatrix} u_1^2 + u_2^2 - 1 \\ u_1 + u_2 - 1 \end{bmatrix} .$$

The derivative is computed as,

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \end{bmatrix} = \begin{bmatrix} 2u_1 & 2u_2 \\ 1 & 1 \end{bmatrix}$$

To solve for the solution to $\mathbf{f}(\mathbf{u}) = \mathbf{0}$ using the function `newton_raphson.m`, one must define the function handles for the residual $\mathbf{f}(\mathbf{u})$ and tangent stiffness $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}$. The example can be run with the following MATLAB code,

```
>> param.N = 2;                                       % -- Number of variables
>> f_resid = @(z) [z(1)^2+z(2)^2-1;z(1)+z(2)-1;]; % -- Define residual
>> f_stiff = @(z) [2*z(1), 2*z(2); 1, 1;];        % -- Define stiffness
>> param.u0= [1;2;];                                  % -- Define starting value
>> param.history=1;                                   % -- Save iteration history
>> nrsol = newton_raphson(f_resid,f_stiff,param); % -- Compute N-R
>> plot_history(nrsol);                     % -- Plot iteration vs. residual
>> plot2d_history(f_resid,nrsol,param); % -- Plot convergence of solution
```

You can also specify the range to plot the figures. Type

```
>> help plot_history
>> help plot2d_history
```

for more details.

Things to check:

- Does the solution change with the initial guess? (HINT: Try `u0=[1/2;0;];`).
- Are there values which you do not get a solution? Why?
- How does the residual get smaller with each iteration near the last few iterations? Do you see quadratic convergence?
- Try some other functions for $\mathbf{f}(\mathbf{u})$ and observe the behavior of obtaining a solution.

4.3.2 2-bar truss example

Consider the 2-bar truss in Figure 1. It is assumed that $L_x = L_y = 1$ and $AE = 1$. In the previous lectures you have learned how to compute the displacement under the given load, under the assumption of *SMALL DISPLACEMENTS*. As long as the material is elastic, under this assumption of *SMALL DISPLACEMENTS*, the truss is stable. This is not true in reality. Consider a rubber truss. You can imagine that if you apply enough load, the rubber truss will buckle and flip. This can be treated by including the effect of non-linear geometry.

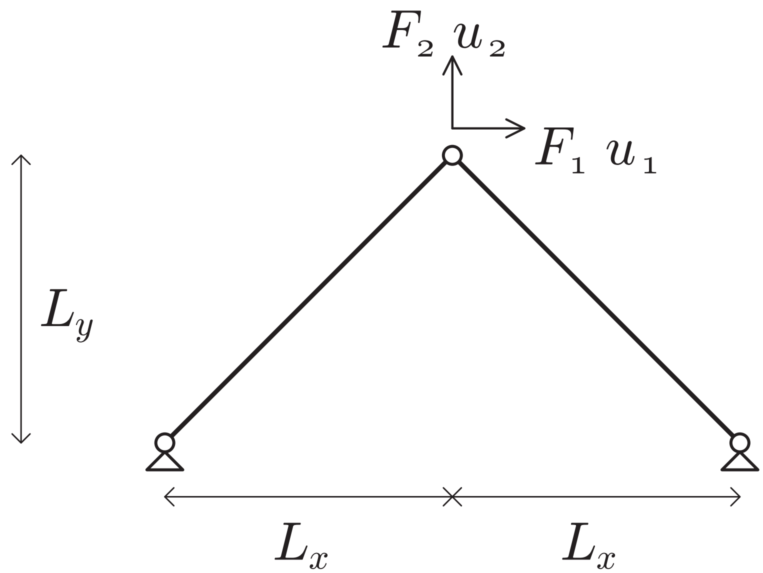

Figure 1: 2 bar truss configuration

Compare the governing equations for the two different cases. For the linear case,

$$
\begin{aligned}
\mathbf{F} &= N_1\mathbf{n}_1 + N_2\mathbf{n}_2 \ , \\
\mathbf{n}_1 &= \frac{1}{L_0}\begin{bmatrix} L_x \\ L_y \end{bmatrix}, \\
\mathbf{n}_2 &= \frac{1}{L_0}\begin{bmatrix} -L_x \\ L_y \end{bmatrix}, \\
L_0 &= \sqrt{(\ L_x)^2 + (L_y)^2}, \\
N_1(\mathbf{u}) &= AE\frac{\mathbf{n}_1^T\mathbf{u}}{L_0}, \\
N_2(\mathbf{u}) &= AE\frac{\mathbf{n}_2^T\mathbf{u}}{L_0} \ ,
\end{aligned}
$$

and thus one defines,

$$
\begin{aligned}
\mathbf{f}(\mathbf{u}) &:= \left[\mathbf{n}_1\mathbf{n}_1^T\frac{AE}{L_0} + \mathbf{n}_2\mathbf{n}_2^T\frac{AE}{L_0}\right]\mathbf{u} - \mathbf{F} = \mathbf{0} \ , \\
\mathbf{K} &:= \frac{\partial\mathbf{f}}{\partial\mathbf{u}} = \left[\mathbf{n}_1\mathbf{n}_1^T\frac{AE}{L_0} + \mathbf{n}_2\mathbf{n}_2^T\frac{AE}{L_0}\right] \ .
\end{aligned}
$$

These are implemented in functions, `l2bar_resid.m` and `l2bar_stiff.m`. For the non-linear case,

$$
\begin{aligned}
\mathbf{F} &= N_1\mathbf{n}_1 + N_2\mathbf{n}_2, \\
\mathbf{n}_1(\mathbf{u}) &= \frac{1}{L_1}\begin{bmatrix} L_x + u_1 \\ L_y + u_2 \end{bmatrix}, \\
\mathbf{n}_2(\mathbf{u}) &= \frac{1}{L_2}\begin{bmatrix} -L_x + u_1 \\ L_y + u_2 \end{bmatrix}, \\
L_1 &= \sqrt{(L_x + u_1)^2 + (L_y + u_2)^2}, \\
L_2 &= \sqrt{(-L_x + u_1)^2 + (L_y + u_2)^2}, \\
N_1(\mathbf{u}) &= AE\frac{L_1 - L_0}{L_0}, \\
N_2(\mathbf{u}) &= AE\frac{L_2 - L_0}{L_0} \ ,
\end{aligned}
$$

and thus one defines,

$$
\begin{aligned}
\mathbf{f}(\mathbf{u}) &:= N_1\mathbf{n}_1 + N_2\mathbf{n}_2 - \mathbf{F} \ , \\
\mathbf{K}(\mathbf{u}) &:= \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \left[\mathbf{n}_1\mathbf{n}_1^T\frac{AE}{L_0} + \mathbf{n}_2\mathbf{n}_2^T\frac{AE}{L_0}\right] \\
&\quad + \frac{N_1}{L_1}\left\{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \mathbf{n}_1\mathbf{n}_1^T\right\} + \frac{N_2}{L_2}\left\{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \mathbf{n}_2\mathbf{n}_2^T\right\} \ .
\end{aligned}
$$

These are implemented in functions, `nl2bar_resid.m` and `nl2bar_stiff.m`. Observe how for the non-linear case, the axial force N and vectors $\mathbf{n}$ depend on the displacement $\mathbf{u}$ in a non-linear fashion.

To solve for the solution to $\mathbf{f}(\mathbf{u}) = \mathbf{0}0$ using the function `newton_raphson.m`, one must define the function handles for the residual $\mathbf{f}(\mathbf{u})$ and tangent stiffness $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}$. The example can be run with the following MATLAB code,

```
>> param.N = 2;                                  % -- Number of variables
>> param.u0= [1;2;];                             % -- Define starting value
>> param.history=1;                              % -- Save iteration history
>> F       = [0;-0.1;];                          % -- Define loading
>> f_resid = @(z)nl2bar_resid(z,F);              % -- Nonlinear residual
>> f_stiff = @(z)nl2bar_stiff(z);                % -- Nonlinear stiffness
>> nrsol = newton_raphson(f_resid,f_stiff,param); % -- Compute N-R
>> plot_history(nrsol);                   % -- Plot iteration vs. residual
>> plot2d_history(f_resid,nrsol,param); % -- Plot convergence of solution
```

One can also run the case for varying load and plot the results,

```
>> param.N = 2;                                  % -- Number of variables
>> Fs = [ zeros(1,40);                           % -- Loading in minus y
>>        linspace( 0.0,-0.185*2,40);];          %    direction
>> plot2bartruss(Fs);
```

One plot gives you a pictorial view of the structure under loading and the other gives you the load (F_2) vs. deflection u_2 curve. The loading is assumed $F_1 = 0$.

Things to check:

- When you run the function `plot2bartruss.m` what happens after the load reaches close to a value of -0.2? Try to explain what is occuring.
- Look at the expressions for residual $\mathbf{f}(\mathbf{u})$ and tangent stiffness $\mathbf{K}(\mathbf{u})$ for the linear and non-linear case and distinguish the differences.
- Complete `func/l2bar_resid.m` and `func/l2bar_stiff.m` and modify the code `func/plot2bartruss.m` so that it runs the case with linear trusses. How does the load-displacement curve differ from the non-linear case?

4.4 3 variable example

4.4.1 Simple example

Run the Newton-Raphson scheme to solve the non-linear equation,

$$\mathbf{f}(\mathbf{u}) = \begin{bmatrix} f_1(u_1, u_2, u_3) \\ f_2(u_1, u_2, u_3) \\ f_3(u_1, u_2, u_3) \end{bmatrix} = \begin{bmatrix} u_1^2 + u_2^2 + u_3^2 - 1 \\ u_1^2 - 0.5 \\ u_3 - 0.25 \end{bmatrix} .$$