

Mechanics of Structures (CE130N)

Lab 3

1 Objective

The objective of this lab is to expose you to solving differential equations which govern mechanical phenomenon, not through hand calculations, but through numerical methods. As you will see in your studies, as the mechanical system becomes more and more complex, hand solutions will become more and more difficult to obtain; in certain cases analytical closed form solutions may not even be accessible. By using numerical methods, even when these closed form solutions are not available one will still be able to obtain good approximations for the exact solution.

In this lab you will be using the software MATLAB and its built in functions to solve the mechanical problem of,

- Tension-compression bars,
- Bending of beams.

As mentioned repetitively in the lecture, both of these phenomenon are governed by a differential equation with one variable, x the position. Such problems can be solved conveniently in MATLAB. In this lab you will be asked to conduct the following things:

1. Understand the procedure of using MATLAB to solve the tension-compression bar problem.
2. Use MATLAB to implement the solution of bending of beams.

2 Tension-compression bar

2.1 Governing differential equation

The governing equations for a tension-compression bar are the following,

- Equilibrium

$$\frac{dR}{dx} + b(x) = 0 \quad (1)$$

The distributed load $b(x)$ can vary along the length of the bar.

- Kinematics

$$\varepsilon = \frac{du}{dx} \quad (2)$$

- Constitutive relation

$$\sigma = E(x)\varepsilon \quad (3)$$

The Young's modulus $E(x)$ can vary along the length of the bar.

- Resultant definition

$$R = A(x)\sigma \quad (4)$$

Here we have assumed a simple system, where the stress σ is constant across any cross-section. The cross-sectional area $A(x)$ can vary along the length of the bar.

In the lecture these four relations have been combined to obtain a single equation representing equilibrium in terms of the displacement,

$$\frac{d}{dx} \left[EA \frac{du}{dx} \right] + b = 0. \quad (5)$$

In order to utilize the solver in MATLAB, one must employ the governing equations in first-order form,

$$\boxed{\frac{dy}{dx} = \mathbf{f}(\mathbf{y}, x)}, \quad (6)$$

where \mathbf{y} is a vector of unknown variables, and \mathbf{f} is a vector of known functions depending on \mathbf{y} and the position x . For the tension-compression bar, we choose the variables u and R as the unknown variables (since the boundary conditions are enforced on u and R). Recall from the solution of systems of equations that one must have the same number of equations as variables. Thus we must obtain 2 differential equations involving u and R from the four governing equations above. These can be obtained by eliminating the variables ε and σ from Eqns. (1-4). This yields the two equations,

$$\begin{aligned} \frac{dR}{dx} + b &= 0, \\ R &= EA \frac{du}{dx}. \end{aligned}$$

The two equations can be rewritten as,

$$\frac{d}{dx} \begin{bmatrix} u \\ R \end{bmatrix} = \begin{bmatrix} R \\ \frac{E(x)A(x)}{-b(x)} \end{bmatrix}.$$

By defining,

$$\begin{aligned} \mathbf{y} &:= \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} u \\ R \end{bmatrix}, \\ \mathbf{f}(\mathbf{y}, x) &:= \begin{bmatrix} f_1(\mathbf{y}, x) \\ f_2(\mathbf{y}, x) \end{bmatrix} = \begin{bmatrix} y_2 \\ \frac{y_2}{\frac{E(x)A(x)}{-b(x)}} \end{bmatrix}, \end{aligned} \quad (7)$$

one obtains the desired first-order form,

$$\frac{d}{dx} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ \frac{y_2}{\frac{E(x)A(x)}{-b(x)}} \end{bmatrix}.$$

2.2 Boundary condition

To solve the differential equation, one must apply boundary conditions. For the second-order differential equation (5), one requires 2 boundary conditions.

In order to apply boundary conditions in the solver in MATLAB, one must define a function which returns a residual of how much the boundary conditions are not satisfied; a residual of zero implies that the boundary conditions are satisfied exactly. The function has the form,

$$\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b))$$

where \mathbf{g} is vector of functions depending on the value of \mathbf{y} evaluated at the boundaries $x = a$ and $x = b$ (Here we assume the problem is defined on the interval $[a, b]$).

To clarify the form of the function, consider the boundary condition,

$$\begin{aligned} u(0) &= \bar{u}_0, \\ R(L) &= \bar{R}_L, \end{aligned}$$

where the displacement is known as \bar{u}_0 at the end point $x = 0$, and the force is known as \bar{R}_L at the end point $x = L$. The vector defining the residual of how much the boundary condition is not satisfied is,

$$\begin{bmatrix} u(0) - \bar{u}_0 \\ R(L) - \bar{R}_L \end{bmatrix}.$$

Using the correspondence between u, R and \mathbf{y} defined in Eqn. (7), one defines \mathbf{g} as,

$$\mathbf{g}(\mathbf{y}(0), \mathbf{y}(L)) := \begin{bmatrix} g_1(\mathbf{y}(0), \mathbf{y}(L)) \\ g_2(\mathbf{y}(0), \mathbf{y}(L)) \end{bmatrix} = \begin{bmatrix} y_1(0) - \bar{u}_0 \\ y_2(L) - \bar{R}_L \end{bmatrix}. \quad (8)$$

2.3 Numerical method

To solve the differential equation in first-order form (6), we will use the function `BVP4C` implemented in MATLAB. The following point is important to keep in mind when using numerical methods to solve differential equations.

A NUMERICAL METHOD ALWAYS GIVES APPROXIMATE SOLUTIONS.

It is only in special cases that one obtains the exact solution. There are cases when it seems like one has the exact solution, but this is only because the solution has been obtained to a high-degree of accuracy. Thus when invoking a numerical method, one must adjust the degree of accuracy to which one desires an approximate solution. The accuracy of the approximate solution $\mathbf{y}_{\text{approx}}$ can be measured by the relative accuracy, defined as,

$$\frac{\|\mathbf{y}_{\text{approx}} - \mathbf{y}_{\text{exact}}\|}{\|\mathbf{y}_{\text{exact}}\|},$$

where $\mathbf{y}_{\text{exact}}$ is the exact solution.

In the case of using `BVP4C` to solve the differential equation (6), there are three parameters one can adjust to determine the attained accuracy in the approximate solution. Let us assume we would like to solve the differential equation on the interval $[a, b]$. The three parameters are,

1. x_i ($i = 1, \dots, N$): The points at which you want to satisfy the differential equation, where N is the total number of nodes and $x_1 = a$ and $x_N = b$.
2. *RELTOL*: `BVP4C` will return an approximate solution $\mathbf{y}_{\text{approx}}$ which satisfies the relation,

$$\left\| \frac{d\mathbf{y}_{\text{approx}}(x_i)}{dt} - \mathbf{f}(\mathbf{y}_{\text{approx}}(x_i), x_i) \right\| \leq \|\mathbf{f}(\mathbf{y}_{\text{approx}}, x_i)\| \text{RELTOL}.$$

for $i = 1, \dots, N$.

3. *ABSTOL*: `BVP4C` will return an approximate solution $\mathbf{y}_{\text{approx}}$ which satisfies the relation,

$$\left\| \frac{d\mathbf{y}_{\text{approx}}(x_i)}{dt} - \mathbf{f}(\mathbf{y}_{\text{approx}}(x_i), x_i) \right\| \leq \text{ABSTOL},$$

for $i = 1, \dots, N$.

The number and location of the points x_i must be placed so that one will be able to sufficiently represent the behavior of the solution on $[a, b]$. A smaller value of *ABSTOL* and *RELTOL* will lead to a more accurate solution but will in general require more time to obtain the solution. Typically, *ABSTOL* and *RELTOL* are chosen between the values of 1×10^{-1} and 1×10^{-16} . (This lower bound in accuracy is due to finite precision of numbers representable on a computer.)

3 Beam in bending

3.1 Governing differential equation

The governing equations for a beam in bending are the following,

- Equilibrium

$$\frac{dV}{dx} + q(x) = 0, \quad (9)$$

$$\frac{dM}{dx} + V = 0 \quad (10)$$

The distributed load $q(x)$ can vary along the length of the beam.

- Kinematics

$$\theta = \frac{dv}{dx}, \quad (11)$$

$$\kappa = \frac{d\theta}{dx}. \quad (12)$$

- Effective constitutive relation

$$M = E(x)I(x)\kappa \quad (13)$$

The Young's modulus $E(x)$ and second moment of inertia $I(x)$ can vary along the length of the beam.

In the lecture, these relations have been combined to obtain a single equation representing equilibrium in terms of the displacement,

$$\frac{d^2}{dx^2} \left[EI \frac{d^2 v}{dx^2} \right] = q. \quad (14)$$

In order to utilize the solver in MATLAB, one must employ the governing equations in first-order form,

$$\boxed{\frac{dy}{dx} = \mathbf{f}(\mathbf{y}, x)}, \quad (15)$$

where \mathbf{y} is a vector of unknown variables, and \mathbf{f} is a vector of known functions depending on \mathbf{y} and the position x . For the beam in bending, we choose the variables v , θ , M , and V as the unknown variables (since the boundary conditions are enforced on these quantities). Recall from the solution of systems of equations that one must have the same number of equations as variables. Thus we must obtain 4 differential equations from the governing equations above. These can be obtained by eliminating the variable κ from Eqns. (9-13). This yields the four equations,

$$\begin{aligned} \frac{dV}{dx} + q(x) &= 0, \\ \frac{dM}{dx} + V &= 0, \\ \theta &= \frac{dv}{dx}, \\ M &= E(x)I(x) \frac{d\theta}{dx}. \end{aligned}$$

The four equations can be rewritten as,

$$\frac{d}{dx} \begin{bmatrix} v \\ \theta \\ M \\ V \end{bmatrix} = \begin{bmatrix} \theta \\ M \\ \frac{E(x)I(x)}{-V} \\ -q(x) \end{bmatrix}.$$

By defining,

$$\begin{aligned} \mathbf{y} &:= \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} v \\ \theta \\ M \\ V \end{bmatrix}, \\ \mathbf{f}(\mathbf{y}, x) &:= \begin{bmatrix} f_1(\mathbf{y}, x) \\ f_2(\mathbf{y}, x) \\ f_3(\mathbf{y}, x) \\ f_4(\mathbf{y}, x) \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ \frac{y_2 y_3}{E(x)I(x)} \\ -y_4 \\ -q(x) \end{bmatrix}, \end{aligned} \quad (16)$$

one obtains the desired first-order form,

$$\frac{d}{dt} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} y_2 \\ y_3 \\ \frac{y_2 y_3}{E(x)I(x)} \\ -y_4 \\ -q(x) \end{bmatrix}.$$

3.2 Boundary condition

To solve the differential equation, one must apply boundary conditions. For the fourth-order differential equation (14), one requires 4 boundary conditions.

In order to apply boundary conditions in the solver in MATLAB, one must define a function which returns a residual of how much the boundary conditions are not satisfied; a residual of zero implies that the boundary conditions are satisfied exactly. The function has the form,

$$\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b))$$

where \mathbf{g} is vector of functions depending on the value of \mathbf{y} evaluated at the boundaries $x = a$ and $x = b$.

To clarify the form of the function, consider the boundary condition where the end at $x = 0$ is fixed and the end at $x = L$ has a point force \bar{V}_L ,

$$\begin{aligned} v(0) &= \bar{v}_0 = 0, \\ \theta(0) &= \bar{\theta}_0 = 0, \\ M(L) &= \bar{M}_L = 0, \\ V(L) &= \bar{V}_L. \end{aligned}$$

The vector defining the residual of how much the boundary condition is not satisfied is,

$$\begin{bmatrix} v(0) - \bar{v}_0 \\ \theta(0) - \bar{\theta}_0 \\ M(L) - \bar{M}_L \\ V(L) - \bar{V}_L \end{bmatrix}.$$

Using the correspondence between v, θ, M, V and \mathbf{y} defined in Eqn. (16), one defines \mathbf{g} as,

$$\mathbf{g}(\mathbf{y}(0), \mathbf{y}(L)) := \begin{bmatrix} g_1(\mathbf{y}(0), \mathbf{y}(L)) \\ g_2(\mathbf{y}(0), \mathbf{y}(L)) \\ g_3(\mathbf{y}(0), \mathbf{y}(L)) \\ g_4(\mathbf{y}(0), \mathbf{y}(L)) \end{bmatrix} = \begin{bmatrix} y_1(0) - \bar{v}_0 \\ y_2(0) - \bar{\theta}_0 \\ y_3(L) - \bar{M}_L \\ y_4(L) - \bar{V}_L \end{bmatrix}. \quad (17)$$

4 MATLAB tips

To use the MATLAB BVP4C solver, one must know how to use function handles. In basic programming, one learns how to pass “numbers” into a function. Analogously to this one can also pass a “function” into a function. This is done through the use of function handles.

The simple exercise below illustrates how they are used. We will combine the two functions:

- A function which takes a number as an argument and returns its 3rd power.
- A function which takes a function as an argument and displays the function value evaluated at 3.

Exercise

1. Write a function called `cubic` which takes a number `x` as an argument and returns its 3rd power,

```
function y = cubic(x)
y = x^3;
```

and save this as the file `cubic.m`.

2. Write a function which takes a function `func` as an argument and prints the value of `func` evaluated at 3,

```
function eval_print(func)
func(3)
```

and save this as the file `eval_print.m`.

3. Execute the function `eval_print` with `cubic` as the input argument in the command window,

```
>> eval_print(cubic)
```

You will see that this gives an error. The proper way of executing this is,

```
>> eval_print(@cubic)
```

The `@` tells MATLAB that the argument `cubic` is a function. Thus whenever MATLAB requires a function handle, one must be careful not to omit the `@`.

5 MATLAB solution method with BVP4C

To solve the given boundary value in MATLAB, one can use the built-in ODE solver `bvp4c`.

BVP4C Solve boundary value problems for ODEs by collocation.

`SOL = BVP4C(ODEFUN,BCFUN,SOLINIT,OPTIONS)` integrates a system of ordinary differential equations of the form $y' = f(x,y)$ on the interval $[a,b]$, subject to general two-point boundary conditions of the form $bc(y(a),y(b)) = 0$. `ODEFUN` and `BCFUN` are function handles. For a scalar X and a column vector Y , `ODEFUN(X,Y)` must return a column vector representing $f(x,y)$. For column vectors YA and YB , `BCFUN(YA,YB)` must return a column vector representing $bc(y(a),y(b))$.

In MATLAB, type `help bvp4c` to see more information on the function.

A boundary value problem (BVP) is a differential equation with values defined at the boundaries of the problem. To solve a BVP in MATLAB using the `BVP4C` solver, one must go through the following steps.

1. Convert the differential equation into first-order form (6) and choose the unknown variables in the vector y . Using this representation, construct the function `ODEFUN`.
2. Construct a function g which returns a residual of how much the boundary conditions are not satisfied as in Eqn.(8). Using this representation, construct the function `BCFUN`.
3. Use the function `BVPINIT` to construct an initial solution structure `SOLINIT`.
4. Use the function `BVPSET` to construct an options structure `OPTIONS`, which determine the degree of accuracy to which the solution is obtained.
5. Pass the arguments `ODEFUN` and `BCFUN` as function handles and the structures `SOLINIT` and `OPTIONS` into the function `BVP4C`.

In the following function and structure explanations, $nvar$ defined the number of unknown variables in the vector y defined in Eqn. (6).

ODEFUN

`[F] = ODEFUN(X,Y)`

- This function serves the purpose of $\mathbf{f}(\mathbf{y}, x)$ mentioned in Eqn.(6).

- *INPUT:*

X : Scalar value defining the position x

Y : Vector ($nvar \times 1$) representing \mathbf{y} evaluated at x .

- *OUTPUT:*

F : Vector ($nvar \times 1$) representing \mathbf{f} evaluated at x using \mathbf{y} , i.e., $\mathbf{f}(\mathbf{y}(x), x)$.

- *EXAMPLE:* For the tension-compression bar in 1D with $E = 1$, $A = 1$ and $b(x) = \sin\left(\frac{\pi}{2}x\right)$, the function is defined as,

```
function [fxy] = bar1d_ode(x,y)

% -- Define material property and geometry
E = 1;
A = 1;
L = 1;

% -- Define distributed load
b = sin(x*pi/(2*L));

% -- Define function
fxy = [y(2)/(E*A);
      -b;];
end
```

BCFUN

`[RES] = BCFUN(YA,YB)`

- To impose boundary conditions in MATLAB one must define a function g of the form,

$$\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)), \quad (18)$$

where $\mathbf{y}(a)$ denotes the value of the function \mathbf{y} at $x = a$, and $\mathbf{y}(b)$ denotes the value of the function \mathbf{y} at $x = b$. This function returns a vector of residuals which defines how much the boundary conditions are not satisfied; the function \mathbf{g} takes the value $\mathbf{g} = \mathbf{0}$ if the value of \mathbf{y} at $x = a, b$ exactly satisfy the boundary condition.

- *INPUT:*

YA :Vector ($nvar \times 1$) defining the value of \mathbf{y} at position $x = a$

YB :Vector ($nvar \times 1$) defining the value of \mathbf{y} at position $x = b$

- *OUTPUT:*

RES :Vector ($nvar \times 1$) representing \mathbf{g} .

- *EXAMPLE:* For the tension-compression bar in 1D with boundary conditions,

$$\begin{aligned} u(0) &= \bar{u}_0, \\ R(L) &= \bar{R}_L. \end{aligned}$$

One defines the function \mathbf{g} in the following way,

$$\mathbf{g}(\mathbf{y}(0), \mathbf{y}(L)) := \begin{bmatrix} y_1(0) - \bar{u}_0 \\ y_2(L) - \bar{R}_L \end{bmatrix}.$$

```
function [res] = bar1d_bc(ya,yb)

% -- Boundary Conditions (BC)
%   u: displacement
%   f: force

ua = 0;
fb = 1;

res= [ya(1)-ua;
      yb(2)-fb];

end
```

BVPSET

OPTIONS = BVPSET('RelTol', RELTOL, 'AbsTol', ABSTOL)

- To set the degree of desired accuracy in the approximate solution, one must construct an options structure OPTIONS to pass to BVP4C.

- *INPUT:*

RELTOL : See Section 2.3 for definition.

ABSTOL : See Section 2.3 for definition.

- *OUTPUT:*

OPTIONS : A MATLAB structure.

BVPINIT

SOLINIT = BVPINIT(X, YINIT)

- One must construct a structure SOLINIT, defining initial parameters of the solution, to pass to BVP4C.

- *INPUT:*

X : Vector defining the nodes at which one desires the solution.

YINIT : Vector ($nvar \times 1$) defining the initial solution.

- *OUTPUT:*

SOLINIT : A MATLAB structure.

6 Exercise

6.1 Function handles

Try the exercise in Section 4 regarding usage of function handles in MATLAB. Make sure you understand the importance of adding the @ mark.

6.2 Download files

1. Download the file `bar_beam1d.zip` into your `cel30n/lab/` directory and unzip it.

6.3 Tension-compression bar

1. Go to the directory `bar_beam1d/exercise`.
2. Execute the file,

```
>> bar1d;
```

This should give you two plots. One showing the displacement u and force R , the other showing the relative error of the obtained approximate solution. The obtained solution is to a problem with,

$$\begin{aligned}A &= 1, \\E &= 1, \\L &= 1, \\b(x) &= \sin\left(\frac{\pi}{2L}x\right),\end{aligned}$$

and boundary condition,

$$\begin{aligned}u(0) &= 0, \\R(L) &= 0.\end{aligned}$$

The exact solution to this problem can be obtained through solving the differential equation by hand as,

$$\begin{aligned}u(x) &= \left(\frac{2}{\pi}\right)^2 \sin\left(\frac{\pi}{2L}x\right), \\R(x) &= \left(\frac{2}{\pi}\right) \cos\left(\frac{\pi}{2L}x\right).\end{aligned}$$

Try changing the two parameters *ABSTOL* and *RELTOL*. How does the accuracy of the approximate solution change? How does the time required to obtain the solution change?

3. Change the file so that you obtain the solution for the problem,

$$\begin{aligned}A &= 1, \\E &= 1, \\L &= 1, \\b(x) &= 0,\end{aligned}$$

and boundary condition,

$$\begin{aligned}u(0) &= 0, \\R(L) &= 1.\end{aligned}$$

What should the shape of the displacement look like?

4. Change the file so that you obtain the solution for the problem,

$$\begin{aligned} A &= 1, \\ E &= 1, \\ L &= 1, \\ b(x) &= \sin\left(\frac{2\pi}{L}x\right), \end{aligned}$$

and boundary condition,

$$\begin{aligned} u(0) &= 0, \\ u(L) &= 0. \end{aligned}$$

Do the obtained results match what those of HW #1 Problem 2.21 ?

5. **Bonus question:** Assume a tapered bar with an end load. The beam has a linearly varying cross sectional area of $(2 - \alpha)A_0$ at the end $x = 0$ and αA_0 at the end $x = L$,

$$A = A_0 \left[-\frac{2(1 - \alpha)}{L}x + (2 - \alpha) \right].$$

Assume $0 \leq \alpha \leq 1$. One observes that the volume of the bar is a constant value of A_0L for any α . If a designer would like to minimize the displacement at $x = L$ using a constant volume of material, what should the value of α be?

HINT: Solve the problem,

$$\begin{aligned} A &= A_0 \left[-\frac{2(1 - \alpha)}{L}x + (2 - \alpha) \right], \\ A_0 &= 1, \\ E &= 1, \\ L &= 1, \\ b(x) &= 0, \end{aligned}$$

and boundary condition,

$$\begin{aligned} u(0) &= 0, \\ R(L) &= 1. \end{aligned}$$

6.4 Beam in bending

Implement the differential equations for the beam and solve several problems.

1. Using the file `beam1d_incomplete.m` and by completing the two functions `beam1d_ode` and `beam1d_bc` solve the solution of the beam problem with parameters,

$$\begin{aligned} E &= 1, \\ I &= 1, \\ L &= 1, \\ q(x) &= q_0 = 1, \end{aligned}$$

and boundary condition,

$$\begin{aligned}u(0) &= 0, \\M(0) &= 0, \\u(L) &= 0, \\M(L) &= 0.\end{aligned}$$

This is a beam with both ends pinned and a uniform distributed load of 1. Is the displacement at the center of the beam what you expect from the classical solution? This solution is,

$$u(L/2) = \frac{5}{384} \frac{q_0 L^4}{EI}.$$

2. Change the file so that you obtain the solution for the problem,

$$\begin{aligned}E &= 1, \\I &= 1, \\L &= 1, \\q(x) &= 0,\end{aligned}$$

and boundary condition,

$$\begin{aligned}u(0) &= 0, \\\theta(0) &= 0, \\M(L) &= 0, \\V(L) &= 1.\end{aligned}$$

This is a cantilever beam with a load applied at the end. What should the shape of the displacement look like? Does the tip displacement match the results you have from your lecture notes?