UNIVERSITY OF CALIFORNIA AT BERKELEY
Department of Civil and Environmental Engineering
Structural Engineering, Mechanics and Materials

Mechanics of Structures (CE130N)
Lab 13

1 Objective

The objective of this lab is to see how the Principle of Virtual Work can be applied to solve mechanical problems governed by differential equations in a similar way to the application of the Principle of Stationary Potential Energy. The Principle of Virtual Work yields exactly the same results as the Principle of Stationary Potential Energy when one has a conservative system and the solution space $\mathcal{S}$ and test function space $\mathcal{V}$ are chosen to be the same. The Principle of Virtual Work has the added flexibility that one can treat non-conservative systems as well as the case $\mathcal{S} \neq \mathcal{V}$.

In combination with the Principle of Virtual Work, a selection of the solution space $\mathcal{S}$ and test function space $\mathcal{V}$ which consist of "hat functions" are made. This is essentially an introduction to the finite element method. The problem of torsion of bars is treated to illustrate the method.

2 Steps in applying the Principle of Virtual Work

The Principle of Virtual Work can be summarized by the following steps.

1. Form the expression for the virtual work.

$$\text{Internal Virtual Work} \quad = \quad \text{External Virtual Work.}$$

Depending on the problem one has,

$$\text{Internal Virtual Work} \quad = \quad \begin{cases} \displaystyle\int_0^L \delta u'(x) R(x)\, dx & \text{(Tension-compression bar)} \\ \displaystyle\int_0^L \delta\varphi'(z) T(z)\, dz & \text{(Torsion bar)} \end{cases} ,$$

and depending on the loading one has different forms for the External Virtual Work. For example, the torsion bar with a point torque $\bar{T}$ at the end $x = L$ and distributed torque $t(z)$ takes the form,

$$\text{External Virtual Work} \quad = \quad \int_0^L \delta\varphi(z) t(z)\, dz + \delta\varphi(L)\bar{T} \, .$$

2. Determine the functions $u(x)$ in the solution space $\mathcal{S}$ (form for the approximate solution),

$$\mathcal{S} := \left\{ u(x) | \sum_{i=1}^{N} u_i f_i(x) \right\} ,$$

where u_i are the coefficients to be determined and $f_i(x)$ are determined functions, and the functions $\delta u(x)$ in the test function space $\mathcal{V}$,

$$\mathcal{V} := \left\{ \delta u(x) | \sum_{j=1}^{N_\delta} \delta u_j g_j(x) \right\} ,$$

where δu_j are the arbitrary coefficients and $g_j(x)$ are determined functions.

The proper selection of $\mathcal{S}$ and $\mathcal{V}$ is crucial in obtaining a good approximation. They must satisfy the following conditions,

- $f_i(x)$ must satisfy the kinematic boundary conditions, $g_j(x)$ must be equal to zero on the kinematic boundary conditions.
- $f_i(x)$ $(i = 1, \ldots, N)$ must be linearly independent to each other, $g_j(x)$ $(j = 1, \ldots, N_\delta)$ must be linearly independent to each other.
- $f_i(x), g_j(x)$ must be nonzero for enough derivatives to make sense in the stiffness matrix.

3. Insert the from for the solution $u(x) \in \mathcal{S}$ and the test function $\delta u(x) \in \mathcal{V}$ into the expression for the virtual work. One must employ the constitutive relations here,

$$\begin{aligned} R(u(x)) = &\quad AEu'(x), \quad \text{(Tension compression bar)}, \\ T(\varphi(z)) = &\quad GJ\varphi'(z), \quad \text{(Torsion bar)} \, . \end{aligned}$$

This results in the expressions,

$$\begin{aligned} & \sum_{i=1}^{N_\delta} \delta u_i \left\{ \sum_{j=1}^{N} K_{ij} u_j - F_i \right\} = 0 \\ \Leftrightarrow \quad & \sum_{i=1}^{N} \delta u_i h_i = 0 \\ \Leftrightarrow \quad & \delta \mathbf{u}^T \mathbf{h} = \mathbf{0} \, . \end{aligned}$$

Here we have defined $\mathbf{h} := \mathbf{K}\mathbf{u} - \mathbf{F}$ and $\delta\mathbf{u}$ as the vector with entries δu_i. The entries of the matrix $\mathbf{K}$ are determined from the Internal Virtual Work and have the form,

$$K_{ij} \quad = \quad \begin{cases} \int_0^L g_i' AE f_j' dx & (\text{Tension} - \text{compression bar}) \\ \int_0^L g_i' GJ f_j' dx & (\text{Torsion bar}) \end{cases} .$$

and,

$$F_i \quad = \quad \begin{cases} \int_0^L g_i b dx + \text{term from point loads} & (\text{Tension-compression bar}) \\ \int_0^L g_i t dz + \text{term from point loads} & (\text{Torsion bar}) \end{cases} .$$

For the last equation, $\delta\mathbf{u}^T\mathbf{h} = 0$, to hold for arbitrary $\delta\mathbf{u}$, one must have,

$$\mathbf{h} = \mathbf{K}\mathbf{u} - \mathbf{F} = 0 \, ,$$

which determines the system of equations one must solve for $\mathbf{u}$, which are the undetermined coefficients.

4. Solve the linear system of equations,

$$\mathbf{K}\mathbf{u} = \mathbf{F} \, .$$

to obtain the approximation,

$$u(x) \quad = \quad \sum_{i=1}^{N} u_i f_i(x) \, .$$

3 Exercise: Torsion bar solved with finite elements

3.1 Download files

1. Download the file `pvw.zip` into your `ce130n/programs` directory and unzip it.
2. Go to the `ce130n/programs/pvw/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.

3.2 Functions used for the approximation

In this exercise you will compute approximate solutions for a bar in torsion using the Principle of Virtual Work. For the solution space $\mathcal{S}$ and test function space $\mathcal{V}$, one will select hat functions, which are defined as,

$$h_m(a_1,a_2,a_3;x) \quad := \quad \begin{cases} 0 & (x < a_1) \\ \dfrac{x-a_1}{a_2-a_1} & (a_1 \le x \le a_2) \\ \dfrac{a_3-x}{a_3-a_2} & (a_2 \le x \le a_3) \\ 0 & (a_3 < x) \end{cases}$$

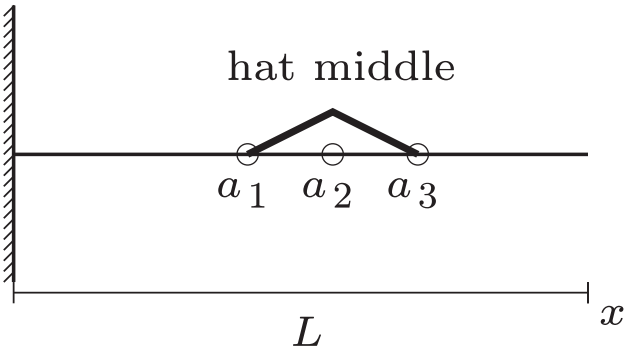

Figure 1: `hat_middle` function

One should have the following Figure 1 of the function in mind. The function has a value of zero outside the interval $[a_1, a_3]$, has a value of 1 at a_2, and behaves linearly. For a given a_1, a_2, a_3 one can plot this function in MATLAB by using the MATLAB function `hat_middle` with the lines of code,

```
>> a1     = -3;
>> a2     = -1;
>> a3     =  0.5;
>> dorder =  0;                          % -- Order of derivative
>> hm= @(z)hat_middle(z,dorder,a1,a2,a3);
>> fplot(hm,[-4,2]);
```

At the ends of the domain (bar) one has special types of hat functions which have only ”half” of the hat as shown

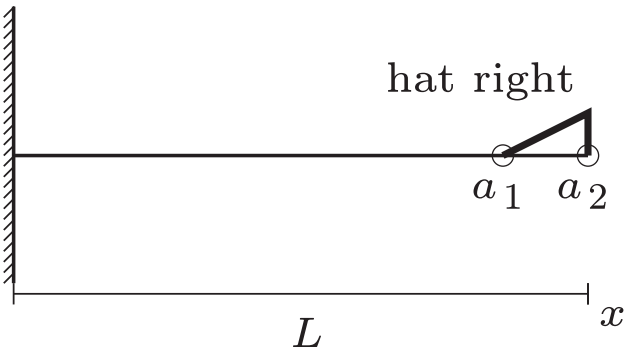

Figure 2: `hat_right` function

in Figure 2. These are defined as,

$$h_r(a_1, a_2; x) \quad := \quad \begin{cases} 0 & (x < a_1) \\ \dfrac{x - a_1}{a_2 - a_1} & (a_1 \leq x \leq a_2) \\ 0 & (a_3 < x) \end{cases}$$

The function has a value of zero outside the interval $[a_1, a_2]$, has a value of 1 at a_2, and behaves linearly. For a given a_1, a_2 one can plot this function in MATLAB by using the MATLAB function `hat_right` with the lines of code,

```
>> a1     = -3;
>> a2     = -1;
>> dorder =  0;                          % -- Order of derivative
>> hm= @(z)hat_right(z,dorder,a1,a2);
>> fplot(hm,[-4,2]);
```

There exists also the symmetric "left" version of this function which is called `hat_left.m`.

Things to check:

- Try plotting the three types of hat functions, `hat_middle.m`, `hat_right.m`, `hat_left.m`, and observe their behavior.
- What do the derivatives of the hat functions look like?

3.3 The solution space and test function space for a fixed-free torsion bar

The torsion bar with governing differential equation,

$$GJ\varphi''(x) + t \quad = \quad 0,$$

with fixed boundary condition at $x = 0$ is treated.

We will choose the solution function and test function to be a linear combination of the hat functions mentioned in the previous section. To define the hat functions, one must first define location of intervals and nodes (elements) on which the hat functions are to be defined, i.e., the location of the a_1, a_2, a_3 of the hat functions. Let us define a grid on the domain $[0, L]$ and break this into nel equidistant intervals of size $d := L/nel$ which results in $nel + 1$ nodes. Label the node locations as,

$$x_k \quad := \quad d(k-1)\,,$$

such that $x_1 = 0$ and $x_{nel+1} = L$. This defines the $N = N_\delta = nel$ functions used for the solution function $u(x)$ and test function $\delta(x)$. These two functions are defined as,

$$\begin{aligned}
\varphi(x) &= \sum_{i=1}^{N} u_i f_i(x) = \sum_{i=1}^{N-1} u_i h_m(x_i, x_{i+1}, x_{i+2}; x) + u_N h_r(x_{nel}, x_{nel+1}; x), \\
\delta\varphi(x) &= \sum_{j=1}^{N} \delta u_j g_j(x) = \sum_{j=1}^{N-1} \delta u_j h_m(x_j, x_{j+1}, x_{j+2}; x) + \delta u_N h_r(x_{nel}, x_{nel+1}; x).
\end{aligned}$$

Observe how all functions $f_i(x)$ satisfy the kinematic boundary conditions, i.e., zero displacement at $x = 0$, and how all functions $g_j(x)$ are equal to zero at the kinematic boundary condition. As a result $\varphi(0) = 0$ and $\delta\varphi(0) = 0$.

These functions are defined in the `pvw` structure in the file `torsion_ex2.m`. (The case for polynomials is given in `torsion_ex1.m`.) The functions and their derivatives can be visualized by the MATLAB code,

```
>> torsion_ex2;
>> plotproblem(pvw);                % -- Plot problem configuration
>> plotpvwf(pvw,'sf');              % -- Plot solution functions
>> plotpvwf(pvw,'tf');              % -- Plot test functions
```

Things to do:

- Plot the solution functions and test functions for various nel and observe their behavior.
- How can you modify the code so that you can treat the torsion bar under fixed-fixed boundary conditions?
  HINT: One must make sure that the function $\varphi(x)$ satisfies the kinematic boundary condition that $\varphi(0) = \varphi(L) = 0$ and that the function $\delta\varphi(x)$ satisfies $\delta\varphi(0) = \delta\varphi(L) = 0$. Save this configuration in a file `torsion_ex2b.m`.

3.4 Solving for the approximate solution

To solve for the approximate solution to the rotations $\varphi(x)$, one must determine the coefficients u_i $(i = 1, \ldots, N)$ from the linear system of equations,

$$\mathbf{K}\mathbf{u} = \mathbf{F}.$$

The form of the selection for the solution function and test function affect the structure of the stiffness matrix $\mathbf{K}$ and forcing vector $\mathbf{F}$.

One can form $\mathbf{K}$ and $\mathbf{F}$, solve for the coefficients $\mathbf{u}$, and plot the solution by the MATLAB code,

```
>> torsion_ex2;
>> K = computeK(pvw);
>> F = computeF(pvw);
>> u = K\F;
>> plotsol(pvw,u);                    % -- Plot solution
>> dorder = 0;
>> evaldisp(pvw,u,[0.5,1],dorder)     % -- Obtain rotation at x=0.5,1
>> dorder = 1;
>> evaldisp(pvw,u,[0.5,1],dorder)     % -- Obtain derivative of
>>                                    %    rotation at x=0.5,1
```

Things to do:

- How do the nonzero entries appear in the stiffness matrix $\mathbf{K}$? (Do you see any structure?)
- For the given case with two point torques at $x = 1/2L$ and $x = L$, how many elements (intervals) are required to obtain an exact solution? Explain why?
- Change the load to a distributed torque of $t(x) = 1$. How many elements (intervals) do you need to get a good approximation to the tip rotation at $x = L$? How good is the approximate solution at the nodes x_k for varying nel ?
  HINT: The finite element solution with hat functions in 1D has the nice property of nodal exactness.
- Using the file `torsion_ex2b.m`, compute for the rotations under a distributed torque $t = \sin(\pi x/L)$. What is the slope of convergence of the L^2 error for the rotation and its derivative as the number of elements increases from $[1, 25]$. Numerically one can compute the L^2 error with the following MATLAB code.

```
>> x      = linspace(0,1,1001);
>> ue     = (1/pi)^2*sin(pi*x);          % -- Exact solution for rotation
>> ee     = (1/pi)^1*cos(pi*x);          % -- Exact solution for derivative
>> ua     = evaldisp(pvw,u,x,0);         % -- Obtain approximation for rotation
>> ea     = evaldisp(pvw,u,x,1);         % -- Obtain approximation for derivative
>> l2e_u  = norm(ua-ue)/norm(ue);        % -- L2 error for rotation
>> l2e_e  = norm(ea-ee)/norm(ee);        % -- L2 error for derivative
```