

Mechanics of Structures (CE130N)

Lab 8

1 Objective

The objective of this lab is to understand that for a stable system, the state of equilibrium corresponds NOT only to a stationary point of the total potential energy but a minimum.

In this lab you will,

- Use the MATLAB function `fminunc`, which computes the minimum of a function, to compute the minimum of potential energies for the two variable problem for several cases.
- Use the MATLAB function `fminunc` to compute the equilibrium state of truss structures.

2 Principle of stationary potential energy

Define the following quantites,

Π_{total} : Total potential energy of the mechanical system,
 Π_{elastic} : Elastic energy in the mechanical system,
 Π_{load} : Energy due to the load,

and define,

$$\Pi_{\text{total}} := \Pi_{\text{elastic}} + \Pi_{\text{load}} .$$

Assume that all the energy quantites noted above depend on N variables or displacements, u_1, \dots, u_N , i.e.,

$$\Pi_{\text{total}}(u_1, \dots, u_N) .$$

By defining the vector \mathbf{u} ,

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

we can denote the dependence as,

$$\Pi_{\text{total}}(\mathbf{u}) .$$

The principle of stationary potential energy states,

The mechanical system is in equilibrium $\Leftrightarrow \Pi_{\text{total}}$ is stationary.

More concretely this implies the following,

The mechanical system is in equilibrium at $\hat{\mathbf{u}} \Leftrightarrow \frac{\partial \Pi_{\text{total}}}{\partial u_i}(\hat{\mathbf{u}}) = 0$ for $i = 1, \dots, N$

This principle allows us to look for the states of equilibrium of the mechanical system by looking for the stationary points of the potential energy. This principle also tells us if that there are no stationary points, then there are no states of equilibrium.

Additionally, if the system is stable one can state the following,

For **stable** systems, the potential energy,

$$\Pi_{\text{total}} = \Pi_{\text{elastic}} + \Pi_{\text{load}} ,$$

is not only stationary for equilibrium states, but will be a **minimum**.

3 Potential energy for linear mechanical systems

For the case of linear mechanical systems, Π_{total} can always be written as,

$$\Pi_{\text{total}}(\mathbf{u}) = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{u}^T \mathbf{F},$$

where \mathbf{K} is a N -by- N matrix and \mathbf{F} is a size N vector. We assume here that \mathbf{K} is a symmetric matrix.

By employing the principle of stationary potential energy,

$$\frac{\partial \Pi_{\text{total}}}{\partial u_i}(\mathbf{u}) = 0, \text{ for } i = 1, \dots, N,$$

one obtains,

$$\mathbf{0} = \frac{\partial \Pi_{\text{total}}}{\partial \mathbf{u}} = \mathbf{K} \mathbf{u} - \mathbf{F}.$$

Thus the stationary points (equilibrium points) $\hat{\mathbf{u}}$ satisfy the relation,

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

As we have seen in the last lab, the behavior of the solutions for linear mechanical systems can be understood by looking at the properties of the matrix \mathbf{K} . Here we will assume again that \mathbf{K} is symmetric.

- \mathbf{K} is symmetric positive definite (all eigenvalues are positive):
 - There is one stationary point which is also a minimum.
 - The system is stable.
- \mathbf{K} has both positive and negative eigenvalues :
 - There is one stationary point which is a saddle point (not a minimum).
 - The system is unstable.
- \mathbf{K} has a zero eigenvalue:
 - There may be no stationary points or there may be multiple.
 - The system is unstable.

Truss structures:

For truss structures, the total potential energy without the application of boundary conditions is,

$$\Pi_{\text{total}}(\mathbf{u}) = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{u}^T \mathbf{F},$$

where \mathbf{K} is the N -by- N stiffness matrix, \mathbf{u} is a size N vector of nodal displacements, and \mathbf{F} is a size N vector of applied nodal forces. We assume here that \mathbf{K} is a symmetric matrix. N is the total number of degrees-of-freedom. Recall the remark made that \mathbf{K} is a singular matrix (has an eigenvalue equal to zero) without the application of boundary conditions.

In order to apply the boundary conditions, one must identify the set of free degrees-of-freedom id_f and the set of known displacement degrees-of-freedom id_d . Using these two sets, one can reorder and partition the quantities \mathbf{K} , \mathbf{u} ,

and \mathbf{F} such that the total energy has the form,

$$\begin{aligned}
\Pi_{\text{total}}(\mathbf{u}) &= \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{u}^T \mathbf{F} \\
&= \frac{1}{2} \begin{bmatrix} \mathbf{u}_f \\ \mathbf{u}_d \end{bmatrix}^T \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fd} \\ \mathbf{K}_{df} & \mathbf{K}_{dd} \end{bmatrix} \begin{bmatrix} \mathbf{u}_f \\ \mathbf{u}_d \end{bmatrix} - \begin{bmatrix} \mathbf{u}_f \\ \mathbf{u}_d \end{bmatrix}^T \begin{bmatrix} \mathbf{F}_f \\ \mathbf{F}_d \end{bmatrix} \\
&= \frac{1}{2} \mathbf{u}_f^T \mathbf{K}_{ff} \mathbf{u}_f - \mathbf{u}_f^T (\mathbf{F}_f - \mathbf{K}_{fd} \mathbf{u}_d) + \left(\frac{1}{2} \mathbf{u}_d^T \mathbf{K}_{dd} \mathbf{u}_d - \mathbf{u}_d^T \mathbf{F}_d \right) \\
&= \frac{1}{2} \mathbf{u}_f^T \mathbf{K}_{ff} \mathbf{u}_f - \mathbf{u}_f^T \mathbf{r}_f + \Pi_{\text{constant}} .
\end{aligned}$$

Here we have defined,

$$\begin{aligned}
\mathbf{r}_f &:= \mathbf{F}_f - \mathbf{K}_{fd} \mathbf{u}_d, \\
\Pi_{\text{constant}} &:= \left(\frac{1}{2} \mathbf{u}_d^T \mathbf{K}_{dd} \mathbf{u}_d - \mathbf{u}_d^T \mathbf{F}_d \right) .
\end{aligned}$$

The definition of Π_{constant} comes from the fact that this contribution is a constant and independent of the displacement of the free degrees-of-freedom \mathbf{u}_f . Since \mathbf{u}_d is also known and constant, the dependence of Π_{total} on \mathbf{u} is actually,

$$\Pi_{\text{total}}(\mathbf{u}) \Rightarrow \Pi_{\text{total}}(\mathbf{u}_f) ,$$

and depends only \mathbf{u}_f . Thus the function that one must find the stationary points is,

$$\Pi_{\text{total}}(\mathbf{u}_f) = \frac{1}{2} \mathbf{u}_f^T \mathbf{K}_{ff} \mathbf{u}_f - \mathbf{u}_f^T \mathbf{r}_f ,$$

where we have omitted Π_{constant} since it is a constant. By looking for the stationary points one obtains the conditions,

$$\mathbf{0} = \frac{\partial \Pi_{\text{total}}}{\partial \mathbf{u}_f} = \mathbf{K}_{ff} \mathbf{u}_f - \mathbf{r}_f .$$

Thus the stationary points (equilibrium points) $\hat{\mathbf{u}}_f$ satisfy the relation,

$$\mathbf{K}_{ff} \hat{\mathbf{u}}_f = \mathbf{r}_f .$$

This equation looks identical to that which we have solved earlier. As long as the system is stable, i.e., \mathbf{K}_{ff} has all positive eigenvalues, $\Pi_{\text{total}}(\hat{\mathbf{u}}_f)$ is the minimum.

4 MATLAB function minimization function FMINUNC

To find the minimum of a scalar-valued function (Π_{total}) depending on a vector argument \mathbf{u} , one can employ MATLAB's function minimization function FMINUNC.

FMINUNC finds a local minimum of a function of several variables.

`[X,FVAL,EXITFLAG] = FMINUNC(FUN,X0,OPTIONS)` starts at X_0 and attempts to find a local minimizer X of the function FUN . FUN accepts input X and returns a scalar function value F evaluated at X . X_0 can be a scalar, vector or matrix.

The function is minimized with the default optimization parameters replaced by values in the structure `OPTIONS`, an argument created with the `OPTIMSET` function.

`FVAL` is the value of the objective function FUN at the solution X .

`EXITFLAG` describes the exit condition of FMINUNC.

Possible values of `EXITFLAG`

and the corresponding exit conditions are

- 1 Magnitude of gradient smaller than the specified tolerance.
- 2 Change in X smaller than the specified tolerance.
- 0 Maximum number of function evaluations or iterations reached.
- 1 Algorithm terminated by the output function.
- 2 Line search cannot find an acceptable point along the current search direction (only occurs in the medium-scale method).

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

To use this function, one must specify the scalar-valued function FUN to be minimized.

5 Exercise

5.1 Download files

1. Download the file `potentialenergy.zip` into your `cel30n/programs/` directory and unzip it.
2. Go to the `cel30n/programs/potentialenergy/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.

5.2 Computing the potential energy

Write a function which computes the total potential energy for a linear mechanical system,

$$\Pi_{\text{total}}(\mathbf{u}) = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{u}^T \mathbf{F},$$

given the stiffness matrix \mathbf{K} , forcing vector \mathbf{F} , and displacement vector \mathbf{u} . Complete the function, `truss_complete/core/potentialenergy.m`, by adding the line of code which computes the potential energy for a linear mechanical system.

5.3 Potential energy for a 2 variable case

Use the function `plotenergy2v.m` which constructs a surface and contour plot of the total potential energy for a 2 variable linear mechanical system in combination with the functions,

- `cel30n/programs/potentialenergy/core/potentialenergy.m`
- `fminunc.m`

to compute and visualize the process of energy minimization.

First define the stiffness matrix \mathbf{K} and forcing vector \mathbf{F} and plot the potential energy.

```
>> close all;
>> clear all;
>> K = DEFINE THE 2-BY-2 MATRIX;
>> F = DEFINE THE LOAD VECTOR;
>> param.surfaceplot_num = 1; % -- Plot the surface plot in Figure number 1
>> param.contourplot_num = 2; % -- Plot the contour plot in Figure number 2
>> param.contourplot_pause = 0.1;% -- Pause in seconds between contour plot
>> param.u1_range = [-3,3];
>> param.u2_range = [-3,3];
>> plotenergy2v(K,F,param); %-- Plot energy contour
```

Next compute the minimum of the potential energy by employing the MATLAB function `fminunc`. One must first set default parameters for the function which define the behavior of the function. Recall how we set optional parameters to solve the boundary value problem in Lab 3. The selection of the following parameters will define the accuracy and quality of the results obtained.

```
>> options = optimset; % -- Sets default options for the minimization
>> options.MaxIter = 50; % -- Maximum number of iterations
>> options.MaxFunEvals = 50; % -- Maximum number of function evaluations
>> options.TolX = 1e-3; % -- Tolerance in the input argument X
>> options.TolFun = 1e-3; % -- Tolerance in the output function value
>> options.FinDiffType = 'central';
% -- Use central difference for
% increased accuracy
>> options.Display = 'iter'; % -- Show solution process on screen
% iteratively
>> options.PlotFcns = @optimplotfval;
% -- Show visually the decrease in the
% function value
```

Finally call the MATLAB function which minimizes the potential energy,

```
>> u0 = [0;0;];
>> [uval,fval,exitflag] = fminunc(@(u)potentialenergy(K,F,u,param),u0,options);
```

In the output arguments,

- `uval` : the value of \mathbf{u} at the computed minimum
- `fval` : the function value of the potential energy computed at the obtained \mathbf{u} ,
- `exitflag` : gives information on whether the solution is valid or not.

In the input arguments,

<code>u0</code>	: the initial approximation to the value of \mathbf{u}
<code>options</code>	: a structure defining the options.
<code>@(u)potentialenergy(K,F,u,param)</code>	: the function handle to the potential energy function with variable arguments.

Potential energy for a 2 spring system

The expression for the matrix \mathbf{K} of the system constructed from 2 springs shown in Figure 1 is,

$$\mathbf{K} = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix},$$

where k_1 and k_2 are the stiffnesses of the springs. Define the displacement and load at the two nodes as u_1, F_1, u_2, F_2 .

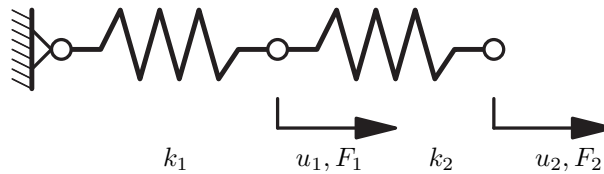


Figure 1: 2 variable system

For each of the following cases, compute the minimum of the potential energy and answer the questions,

- Does the potential energy have a minimum? Explain why or why not.
- Does the system have an equilibrium point? If it does, is the system stable at this equilibrium point?
- Do you obtain the same minimum for any initial approximation u_0 ? If there is a minimum, what is the value of the minimum and what is the displacement \mathbf{u} at the minimum. How accurate are your answers compared to the exact solution?
- Can you obtain a minimum by changing the initial approximation u_0 ?
- If the potential energy has a minimum, how do the approximations approach the minimum using the function `fminunc`?

1. $k_1 = 0.2, k_2 = 1, F_1 = 0, F_2 = 1, u_0 = [-4, 2]$:

2. $k_1 = -0.5, k_2 = 1, F_1 = 1, F_2 = 1, u_0 = [2, -4]$:

3. $k_1 = 0, k_2 = 1, F_1 = 1, F_2 = 1, u_0 = [0, -4]$:

4. $k_1 = 0, k_2 = 1, F_1 = 1, F_2 = -1, u_0 = [0, -4]$:

5.4 Potential energy for truss structures

Compute the minimum of the potential energy for truss structures using the function `fminunc`.

One must first compute the matrix \mathbf{K}_{ff} and vector \mathbf{r}_f . To obtain these one must run the following lines of MATLAB code.

```
>> clear all;
>> close all;
>> mesh_data_truss_example; % -- load data
>> plotmesh(mesh);          % -- plot mesh
>> formmatrices;            % -- form matrices and vectors
```

Then one must set the parameters which define the performance of the algorithm.

```
>> options = optimset; % -- Sets default options for the minimization
>> options.MaxIter      = 50; % -- Maximum number of iterations
>> options.MaxFunEvals  = 50; % -- Maximum number of function evaluations
>> options.TolX         = 1e-3; % -- Tolerance in the input argument X
>> options.TolFun       = 1e-3; % -- Tolerance in the output function value
>> options.FinDiffType = 'central';
                        % -- Use central difference for
                        %    increased accuracy
>> options.Display      = 'iter'; % -- Show solution process on screen
                        %    iteratively
>> options.PlotFcns     = @optimplotfval;
                        % -- Show visually the decrease in the
                        %    function value
```

Finally call the MATLAB function which minimizes the potential energy,

```
>> uf0 = [0;0;];
>> [ufval,fval,exitflag] =
    fminunc(@(uf)potentialenergy(Kff,Rf,uf),uf0,options);
```

In the output arguments,

<code>ufval</code>	: the value of \mathbf{u}_f at the computed minimum
<code>fval</code>	: the function of the potential energy computed at the obtained \mathbf{u}_f ,
<code>exitflag</code>	: gives information on whether the solution is valid or not.

In the input arguments,

<code>uf0</code>	: the initial approximation to the value of \mathbf{u}_f
<code>options</code>	: a structure defining the options.
<code>@(uf)potentialenergy(Kff,Rf,uf)</code>	: the function handle to the potential energy function with variable arguments.

One can display the obtained results by the following procedure,

```
>> mesh= mesh_setuf(mesh,ufval); % -- Set ufval into mesh
>> plotdefo(mesh,lel); % -- plot deformed configuration for check
```

For each of the following cases, compute the minimum of the potential energy and answer the questions,

- Does the potential energy have a minimum? Explain why or why not.
- Does the system have an equilibrium point? If it does, is the system stable at this equilibrium point?
- Do you obtain the same minimum for any initial approximation \mathbf{u}_f0 ? If there is a minimum, what is the value of the minimum and what is the displacement \mathbf{u}_f at the minimum. How accurate are your answers compared to the exact solution?
- Can you obtain a minimum by changing the initial approximation \mathbf{u}_f0 ?

```
1. mesh_data_truss_example.m  
   uf0=[0;0;];
```

```
2. mesh_data_truss_ex1.m  
   uf0=[0;1;0;0.5;];
```

```
3. mesh_data_truss_ex2.m
   uf0=[0;1;0;0.5;];
```

```
4. mesh_data_howe.m
   uf0=all zeros;
```