**Peeter Joot**
**peeter.joot@gmail.com**

# ECE1254H Modeling of Multiphysics Systems. Lecture 12: Struts and Joints, Node branch formulation. Taught by Prof. Piero Triverio

## 1.1 Disclaimer

Peeter's lecture notes from class. These may be incoherent and rough.

## 1.2 Struts and Joints, Node branch formulation

Let's consider the simple strut system of fig. 1.1 again.

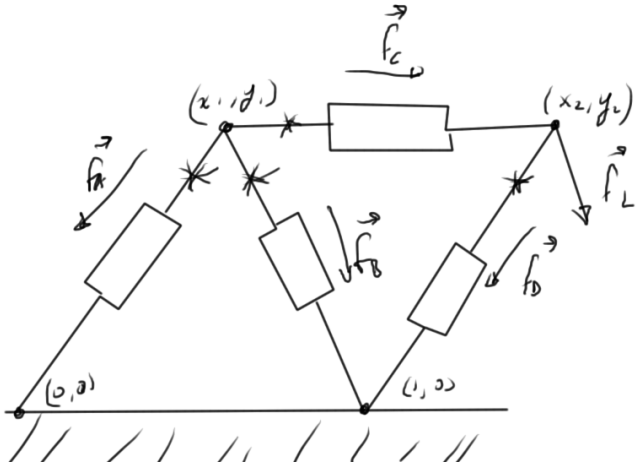

**Figure 1.1:** Simple strut system

Our unknowns are

1. Forces

   At each of the points we have a force with two components

$$\mathbf{f}_A = \left(f_{A,x}, f_{A,y}\right) \tag{1.1}$$

We construct a total force vector

$$\mathbf{f} = \begin{bmatrix} f_{A,x} \\ f_{A,y} \\ f_{B,x} \\ f_{B,y} \\ \vdots \end{bmatrix} \tag{1.2}$$

2. Positions of the joints

$$\mathbf{r} = \begin{bmatrix} x_1 \\ y_1 \\ y_1 \\ y_2 \\ \vdots \end{bmatrix} \tag{1.3}$$

Our given variables are

1. The load force $\mathbf{f}_L$.
2. The joint positions at rest.
3. parameter of struts.

*Conservation laws* The conservation laws are

$$\mathbf{f}_A + \mathbf{f}_B + \mathbf{f}_C = 0 \tag{1.4a}$$

$$-\mathbf{f}_C + \mathbf{f}_D + \mathbf{f}_L = 0 \tag{1.4b}$$

which we can put in matrix form

$$\begin{matrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{matrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{A,x} \\ f_{A,y} \\ f_{B,x} \\ f_{B,y} \\ f_{C,x} \\ f_{C,y} \\ f_{D,x} \\ f_{D,y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -f_{L,x} \\ -f_{L,y} \end{bmatrix} \tag{1.5}$$

Here the block matrix is called the incidence matrix $\mathbf{A}$, and we write

$$\boxed{A\mathbf{f} = \mathbf{f}_L.} \tag{1.6}$$

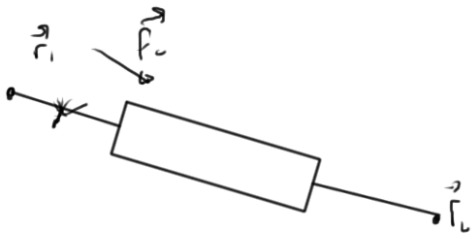

**Figure 1.2:** Strut spanning nodes

*Constitutive laws* Given a pair of nodes as in fig. 1.2.
each component has an equation relating the reaction forces of that strut based on the positions

$$f_{c,x} = S_x\left(x_1 - x_2, y_1 - y_2, p_c\right) \tag{1.7a}$$

$$f_{c,y} = S_y\left(x_1 - x_2, y_1 - y_2, p_c\right), \tag{1.7b}$$

where $p_c$ represent the parameters of the system. We write

$$\mathbf{f} = \begin{bmatrix} f_{A,x} \\ f_{A,y} \\ f_{B,x} \\ f_{B,y} \\ \vdots \end{bmatrix} = \begin{bmatrix} S_x\left(x_1 - x_2, y_1 - y_2, p_c\right) \\ S_y\left(x_1 - x_2, y_1 - y_2, p_c\right) \\ \vdots \end{bmatrix}, \tag{1.8}$$

or

$$\boxed{\mathbf{f} = S(\mathbf{r})} \tag{1.9}$$

*Putting the pieces together* The node branch formulation is

$$\begin{aligned} A\mathbf{f} - \mathbf{f}_L &= 0 \\ \mathbf{f} - S(\mathbf{r}) &= 0 \end{aligned} \tag{1.10}$$

We'll want to approximate this system using the Jacobian methods discussed, and can expect the cost of that Jacobian calculation to potentially be expensive. To move to the nodal formulation we eliminate forces (the equivalent of currents in this system)

$$\boxed{AS(\mathbf{r}) - \mathbf{f}_L = 0} \tag{1.11}$$

We cannot use this nodal formulation when we have struts that are so stiff that the positions of some of the nodes are fixed, but can work around that as before by introducing an additional unknown for each component of such a strut.

## 1.3 Damped Newton's method

We want to be able to deal with the oscillation that we can have in examples like that of fig. 1.3.

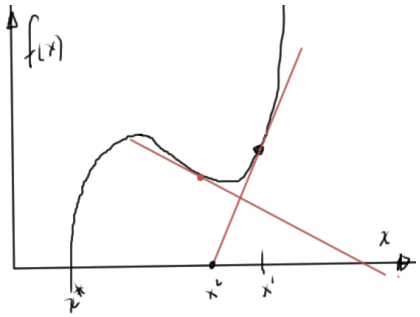

**Figure 1.3:** Oscillatory Newton's iteration

Large steps can be dangerous. We want to modify Newton's method as follows
Our algorithm is
Guess $\mathbf{x}^0$, $k = 0$.
**repeat**
  Compute $F$ and $J_F$ at $\mathbf{x}^k$
  Solve linear system $J_F(\mathbf{x}^k)\Delta\mathbf{x}^k = -F(\mathbf{x}^k)$
  $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k\Delta\mathbf{x}^k$
  $k = k + 1$
**until** converged
with $\alpha^k = 1$ we have standard Newton's method. We want to pick $\alpha^k$ so that we minimize

## 1.4 Continuation parameters

Newton's method converges given a close initial guess. We can generate a sequence of problems where the previous problem generates a good initial guess for the next problem.

An example is a heat conducting bar, with a final heat distribution. We can start the numeric iteration with $T = 0$, and gradually increase the temperatures until we achieve the final desired heat distribution.

Suppose that we want to solve

$$F(\mathbf{x}) = 0. \tag{1.12}$$

We modify this problem by introducing

$$\tilde{F}(\mathbf{x}(\lambda), \lambda) = 0, \tag{1.13}$$

where

- $\tilde{F}(\mathbf{x}(0), 0) = 0$ is easy to solve

- $\tilde{F}(\mathbf{x}(1), 1) = 0$ is equivalent to $F(\mathbf{x}) = 0$.
- (more on slides)

The source load stepping algorithm is

- Solve $\tilde{F}(\mathbf{x}(0), 0) = 0$, with $\mathbf{x}(\lambda_{\text{prev}} = \mathbf{x}(0)$
- (more on slides)

This can still have problems, for example, when the parameterization is multivalued as in fig. 1.4.

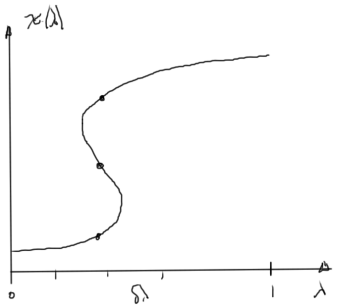

**Figure 1.4:** Multivalued parameterization

We can attempt to adjust $\lambda$ so that we move along the parameterization curve.