

CSUMS Seminar 5: Assignment on Fixed-Point Methods and Newton's Method

Number 1 below is due on Wednesday, October 22, 2008. Number 2 is due by noon on Tuesday, October 28, 2008. Please keep in mind that you will receive another small assignment on 10/22/08 that will be due on 10/28/08.

- Recall the nonlinear system from your last homework assignment.

$$\begin{aligned}x_1^2 - 10x_1 + x_2^2 + 6 &= 0 \\x_1x_2^2 - x_1 - 10x_2 + 8 &= 0.\end{aligned}$$

The associated fixed-point problem that you used for fixed-point iteration and the Gauss-Seidel method is

$$\begin{aligned}x_1 &= g_1(x_1, x_2) = \frac{x_1^2 + x_2^2 + 6}{10} \\x_2 &= g_2(x_1, x_2) = \frac{x_1x_2^2 + x_1 + 8}{10}\end{aligned}$$

- Use the Contraction Mapping Theorem to show that $\mathbf{G} = (g_1, g_2)^T : D \subset \mathcal{R}^2 \rightarrow \mathcal{R}^2$ has a unique fixed point in

$$D = \{(x_1, x_2)^T \mid 0 \leq x_1, x_2 \leq 1.5\}$$

and that $\mathbf{x}^{(k)} = \mathbf{G}(\mathbf{x}^{(k-1)})$ converges to a unique fixed point if $\mathbf{x}^{(0)} \in D$.

- Use the Contraction-Mapping Theorem to show that \mathbf{G} is not guaranteed a fixed point in

$$D = \{(x_1, x_2)^T \mid 2 \leq x_1, x_2 \leq 4\}$$

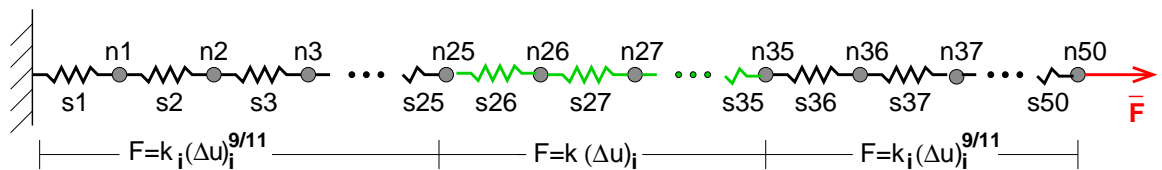
nor that the sequence generated by $\mathbf{x}^{(k)} = \mathbf{G}(\mathbf{x}^{(k-1)})$ is guaranteed to converge if a fixed-point does exist.

Note that the Contraction Mapping Theorem does not indicate a fixed point definitely does not exist if the appropriate assumptions are not satisfied. It only specifies the conditions under which a fixed point is guaranteed to exist. The same hold true for the portion of the theorem that guarantees convergence.

For help on this problem you can follow Example 2 in Section 10.1 in Burden and Faires.

- For the following problem you will program Newton's Method for nonlinear systems with a finite-difference approximation of the Jacobian matrix.

Consider the collection of 50 nonlinear springs in series illustrated below. Each spring is connected by what is termed a *node*, which for the purposes of this exercise is assumed to be massless.



The force in springs 1-25 and 36-50, F_i , where the i refers to the i^{th} spring, is given by

$$F_i = k_i(\Delta u_i)^{9/11}, \quad (1)$$

where $k_i = 1 + \frac{x}{20}$ is a spatially variable spring constant of the i^{th} spring and $\Delta u_i = u_i - u_{i-1}$ is the change in length from the resting length of this spring. The variable u_i represents the *displacement*, i.e. the distance that something moves from its original position, of the i^{th} node. Springs 26-35 are linear, and the force generated by stretching or compressing them is given by

$$F_i = k(\Delta u_i). \quad (2)$$

The spring constant $k = 2$ is the same for all of these springs.

The goal of your program will be to determine the displacement of each node in the system given a force \bar{F} applied to the right-most node (see illustration above). To do this, we must ensure that the forces balance at each node. For a node i the force balance equation is

$$F_{i+1} - F_i = 0,$$

which indicates that the force from the spring on one side of the node must balance the force of the spring on the other side of the node. Depending on the location of the node, the force in the spring will be given by (1) or (2). The force balance equation for node 50 incorporates the applied force such that the force balance equation for this node is

$$\bar{F} - F_{50} = 0.$$

To determine the displacements one needs to solve the nonlinear system of 50 equations and 50 unknowns numerically. The Matlab function $\mathbf{F}(\mathbf{u})$ which represents the system described above can be downloaded along with this assignment as the file `FSpring.m`.

Your assignment is to write a program that uses Newton's Method to solve for the displacements. Use finite differences with $\epsilon = 0.01$ to approximate your Jacobian matrix. For example, the entry $\frac{\partial f_3}{\partial u_{20}}$ should be approximated by

$$\frac{\partial f_3}{\partial u_{20}} \approx \frac{f_3(u_1^{(k)}, u_2^{(k)}, \dots, u_{20}^{(k)} + \epsilon, u_{21}^{(k)}, \dots, u_{50}^{(k)}) - f_3(u_1^{(k)}, u_2^{(k)}, \dots, u_{20}^{(k)}, u_{21}^{(k)}, \dots, u_{50}^{(k)})}{\epsilon},$$

where $\mathbf{u}^{(k)}$ refers to the approximate displacement at the k^{th} iteration of Newton's Method. It may be useful to write the code for the approximation of the Jacobian matrix as a separate Matlab function.

As discussed in class, for each nonlinear iteration, you need to solve a linear system. Use fixed-point iteration to solve these linear systems.

Last, but not least, be sure to include appropriate stopping criteria in your code: (i) stop when a tolerance of $\|\mathbf{u}^{(k)} - \mathbf{u}^{(k-1)}\|_{\infty} < 10^{-5}$ and (ii) stop when a maximum number of (either nonlinear or linear) iterations reaches a maximum value.

- (a) When coding large programs it is often beneficial to test whether your code works correctly for a simple case. To test your code, initially let $\bar{F} = 0$. If there is no force applied to the last node, all displacements should be zero.
- (b) Solve for the displacements when $\bar{F} = 1.0$; you are pulling your system of springs to the right.
- (c) Solve for the displacements when $\bar{F} = -0.2$; you are pushing your system of springs to the left.
- (d) Try playing around with the finite difference parameter ϵ . Does it have a lot of effect on your final solution and how quickly you converge to it?