Assignment 4 — 2.086 Fall 2015

Due: *20 October 2015 at 5 PM.*

Upload your solution folder to Stellar as a zip file named "`YOURNAME_ASSIGNMENT_4.zip`"; the folder must include the script for each question, in the proper format, as well as any other MATLAB functions that your scripts call. **Remember to verify your upload by downloading it, and running `grade_o_matic`, before the assignment is due**. Full instructions are given in the "Preparing Assignments for `Grade_o_matic`" document.

---

1. **Gaussian Elimination.** (40 points) Write a program that performs Gaussian elimination to solve a linear system $Au = f$, where A is a given $n \times n$ matrix, f is a given $n \times 1$ vector, and u is the $n \times 1$ solution vector. The two input parameters are (matrix) `A` and (vector) `f`. You may assume in this problem that A is symmetric positive-definite (SPD), so no row or column exchanges are needed for GE to succeed; you can also assume that $2 \leq n \leq 100$. Note n is not in the set of input parameters. The three outputs are (matrix) `U`, the upper triangular matrix into which A is transformed during GE; (vector) `fhat` $\leftrightarrow \hat{f}$ such that $Uu = \hat{f}$; and the solution (vector) `u`.

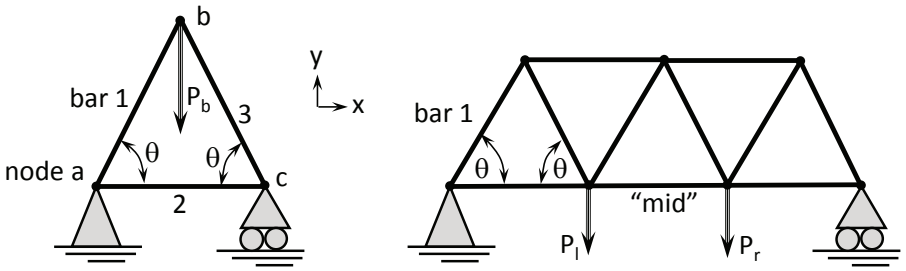
   Basic Gaussian elimination involves two distinct operations: upper-triangularization of A while also generating $\hat{f}$, followed by back-substitution. In the backsubstitution, remember to find $u(n)$ *first*, then $u(n-1)$, and so on. A pseudo-code for the upper-triangularization, over-writing A as we go, is

```
for iPivot = 1 : n-1                                  (which pivot)
    for jRow = iPivot+1 : n                           (which row below the pivot)
        pivot = A(iPivot,iPivot)
        scale = A(jRow,iPivot) / pivot
        A(jRow,:) = A(jRow,:) - A(iPivot,:) * scale
    end
end
```

   In testing your code, note that you can create an SDP matrix with $A = Q^{\mathrm{T}}Q$, where Q is *any* $n \times n$ full-rank matrix; `Q = randn(n,n)` will almost always do, at least for smaller n. You can verify your algorithm by checking that indeed $Au = f$, or by comparing with Matlab's backslash solution. Do not include backslash in your submitted code.

2. **Modeling and Solving for Truss Loads.** (35 points) Cranes, bridges, and many other structures employ trusses assembled from a large number of slender bars. In this problem, you will develop a complete linear model for a realistic truss, and then apply a strengthened version of your Gaussian elimination routine from Problem 1 to solve for the various loads.

   Referencing the left figure below, in which θ and P_b are fixed, static force balances generate the necessary equations while the unknowns are the ground forces, and the axial loads in each bar. Writing the horizontal and vertical force balances for the three nodes (left to right), the



equations unfold as follows:

$$\begin{aligned}
f_{a,x} + t_1 \cos\theta + t_2 &= 0 \\
f_{a,y} + t_1 \sin\theta &= 0 \\
-t_1 \cos\theta + t_3 \cos\theta &= 0 \\
-t_1 \sin\theta - t_3 \sin\theta - P_b &= 0 \\
-t_2 - t_3 \cos\theta &= 0 \text{ (since the horizontal ground force at } c \text{ is zero)} \\
t_3 \sin\theta + f_{c,y} &= 0.
\end{aligned}$$

Note we take the load in each bar to be *positive in tension*, and thus call the loads t_i's. The ground forces are taken positive when they push on the truss in the positive x and y directions as drawn. So $f_{a,x}$ is the force exerted by the ground onto node a, pushing the structure to the right. P_b is positive pointing down as drawn.

In the language of linear algebra, the equations above fit $Au = f$:

$$\begin{bmatrix}
1 & 0 & \cos\theta & 1 & 0 & 0 \\
0 & 1 & \sin\theta & 0 & 0 & 0 \\
0 & 0 & -\cos\theta & 0 & \cos\theta & 0 \\
0 & 0 & -\sin\theta & 0 & -\sin\theta & 0 \\
0 & 0 & 0 & -1 & -\cos\theta & 0 \\
0 & 0 & 0 & 0 & \sin\theta & 1
\end{bmatrix}
\begin{Bmatrix}
f_{a,x} \\ f_{a,y} \\ t_1 \\ t_2 \\ t_3 \\ f_{c,y}
\end{Bmatrix}
=
\begin{Bmatrix}
0 \\ 0 \\ 0 \\ P_b \\ 0 \\ 0
\end{Bmatrix}.$$

Your modeling task is to work out the similar equations for the larger truss in the right-side figure. You can apply exactly the same process as above, and adopt any labeling scheme you choose. There will be fourteen equations (horizontal and vertical force balances at each of seven nodes), and fourteen unknowns (two vertical ground forces, the left-side horizontal ground force, and eleven tensions). The right-hand side of your $Au = f$ should include the two vertical loads P_l and P_r, as drawn. In trouble-shooting, use the facts that the vertical ground forces must sum to $P_l + P_r$, and the horizontal ground force on the left side is zero. If you do all your labeling and the force balances moving left-to-right, you'll see a recurring structure near A's diagonal: indeed this system is banded diagonal, which allows $\mathcal{O}(n)$ computational cost, e.g., in a much longer truss.

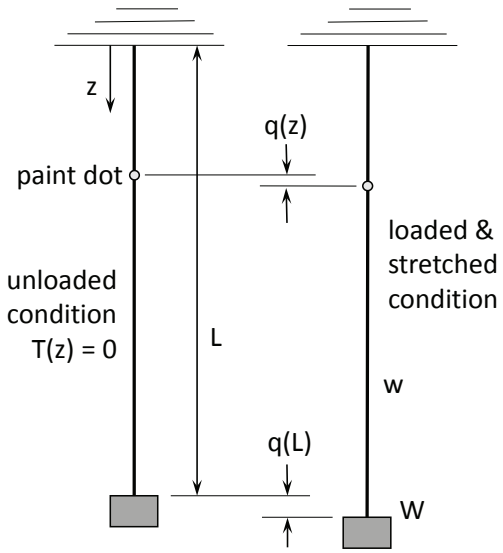
When you have assembled A and f, notice A is neither symmetric nor positive-definite; use `spy(A)` or `imagesc(A)` to visualize its structure. This fact in itself is not fatal for your Gaussian elimination solver, although a zero on the diagonal might well be (try it!). We need to strengthen the algorithm from Problem 1. In particular, we ask you to augment your code with a section that adaptively exchanges rows in U, to obtain the largest pivots in an absolute

value sense. This "partial pivoting" is not hard: consider `[junk,ind] = max(abs(U(i:n,i))` to identify the row that can provide the largest i'th pivot, and then how to swap this row with row i. Partial pivoting is a standard – and crucial – operation that smart GE solvers employ.

Once your truss model is correct and the augmented GE solver works, this problem is practically done. The two input parameters are (vector) `P`, containing the loads $[P_l, P_r]$ (in arbitrary force units); and the angle `theta` $\leftrightarrow \theta$, in degrees. Assume that $-100 \leq P_l \leq 100$, $-100 \leq P_r \leq 100$, and $5^\circ \leq \theta \leq 85^\circ$. There are two output variables: `t_1`, tension in the upper-leftmost member as drawn; and `t_mid`, tension in the middle member as drawn. We ask you to use your strong GE, not Matlab's backslash, to solve the linear system (the graders will be checking).

3. **Hanging Tow-Cable.** (25 points) The wreck of the *Titanic* was found in 1985, by a team that towed a heavy camera "sled" on a two-mile-long vertical cable from a ship. Manufacturing such a cable is an engineering accomplishment in its own right, and in this problem we'll consider how it stretches under immense tensile loads. The cable in the *Titanic* search had a diameter of 1.7cm, with an outer armor of plough steel strands, and an inner core of copper conductors and optical fibers. If a cable like this stretches too much, damage to the terminations and to the fibers can occur.

   Consider the cable hanging from a stationary ship, with a heavy sled at the bottom. Taking z as the coordinate along the cable, and q as the stretching displacement, the governing differential equation is $dq/dz = T(z)/EA$, where $T(z)$ is local tension, and EA is the cable stiffness, provided by Young's modulus E, and the cable's cross-sectional area A (note EA has units of force). You can think of $q(z)$ this way: Paint a dot at location z when the cable is vertical but unloaded and unstretched. As illustrated in the figure, $q(z)$ is the (small) distance that this dot *moves down as the cable is loaded.*



   Defining the unstretched cable length as L, the in-water cable weight per unit length as w, and sled in-water weight as W, we have tension $T(z) = (L - z)w + W$. Implementing a forward-differences discretization creates precisely the tridiagonal system described in **YPKP**

Section 25.3. Namely, setting the stepsize $\Delta z = L/n$, we can write (using the nomenclature of YPKP, i.e., the f's are forces) $k_i = EA/\Delta z, 1 \le i \le n$; $f_i = w\Delta z, 1 \le i \le n-1$; and $f_n = w\Delta z + W$.

a. (15 points) We ask you to write a program that sets up the tridiagonal linear system, and solves for the vector of displacements q_i of the cable. The *fixed* physical parameters are $L = 4000m$, $EA = 4.5 \times 10^7 N$, $w = 14N/m$, and $W = 9000N$; these should all be set inside your code. The single `grade_o_matic` input parameter is `n`; you may assume that $2 \le n \le 1000$. The two output variables are `firstDisp` $\leftrightarrow q(z = \Delta z)$, displacement of the first node below the surface; and `endDisp` $\leftrightarrow q(z = L)$, displacement of the lower end of the cable. Note we can integrate the governing equation to get an *exact*, closed-form solution $q(z = L) = L(Lw/2+W)/EA$, which may be useful in checking computations. The A-matrix in this problem is SPD, so you can run your basic GE algorithm on it with no row exchanges if you choose; you may also solve the system with backslash (full or sparse).

b. (10 points) To see a dramatic story unfold, compare the computational effort for four approaches, with several large n values (larger than 1000 if your computer can handle it):

- evil inverse, i.e., $u = A^{-1}f$ (how many elements in A^{-1} are nonzero?!)
- your basic Gaussian elimination scheme
- full backslash, i.e., $u = A\backslash f$, with A defined as a full matrix
- sparse backslash, i.e., $u = A\backslash f$, with A defined as a sparse matrix

Use `tic` and `toc` for timing. Type your observations and an explanation into a pdf file `A4Q3.pdf`.