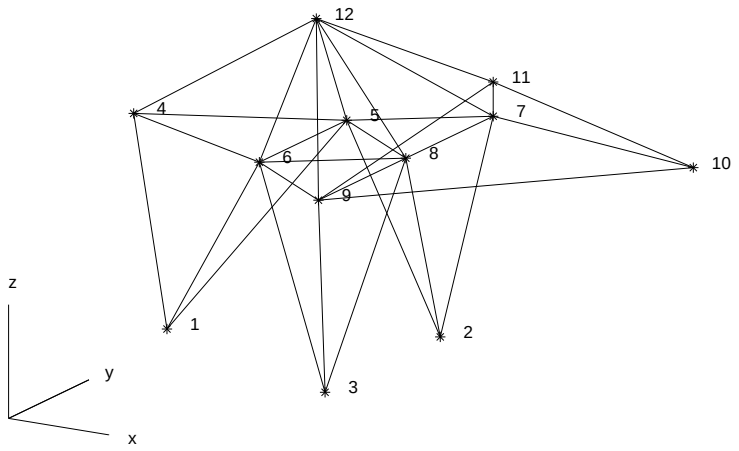
CS412 **Assignment 5 (as of 07nov00)** Fall 2000

due date: 09 November 2000

STAPLE your homework. MARK your homework clearly with your NAME. In addition, write the first letter of your LAST NAME boldly into the upper left corner of the first page of your homework . For this to be of real help, the vertical edge of the upper left corner should be longer than the horizontal edge.

1. (40 points) This problem illustrates the fact that, with `Matlab`'s backslash operator available, the only hard aspect of solving linear systems is usually the correct assembly of the linear system.



The three-dimensional structure shown above is to be analyzed to determine, for each member m, the static force f_m on it that results when a certain load is applied to joint 10. A negative force corresponds to compression, a positive force corresponds to expansion. The coordinates of the twelve joints, the connectivity data, and other information, are in the m-file `home5.m` available from the class page, along with some explanatory text. That's the m-file I made up for generating the above picture.

First some background. The equations for the forces are obtained as follows. At joint i, the balance of forces is expressed by the equation

$$\sum_m \mathbf{c}_{i,j} f_m = \mathbf{load}_i, \tag{1}$$

where the sum is over only those members m which connect to joint i and where j is the joint connected to i by m; further,

$$\mathbf{c}_{i,j} := (\mathbf{i} - \mathbf{j})/|\mathbf{i} - \mathbf{j}|,$$

with $\mathbf{i} = (x(i), y(i), z(i))$ the 3-vector pointing to joint i, and $|\mathbf{i} - \mathbf{j}|$ the distance between joint i and joint j; and, finally, $\mathbf{load}_i$ the load-vector applied to joint i. In our example, only joint 10 will carry a load.

Note that (1) constitutes three scalar equations; e.g., the first of these equations is

$$\sum_m \frac{x(i) - x(j)}{|\mathbf{i} - \mathbf{j}|} f_m = 0$$

(in case $i \neq 10$), with $|\mathbf{i} - \mathbf{j}|$ the Euclidean distance between the two joints i and j. Note also that $\mathbf{c}_{j,i} = -\mathbf{c}_{i,j}$.

Altogether, we obtain 36 equations, i.e., 3 equations for each of the 12 joints, from which the 30 unknowns, f_m, $m = 1, \ldots, 30$, are to be determined. Of these 36 equations, we eliminate 6 by the requirement that the structure should be fixed to the ground. This we can do in several ways. Here is one: for joint 1, we remove all three equations. This is the same as saying that we are applying just the right load at joint 1 to ensure that it won't move, whatever the other forces may be. We also remove the y and the z equation from joint 2 and the z equation from joint 3. This says that we are applying to joints 2 and 3 just the right load in the z-direction to keep them on the ground, and also apply just the right load in the y-direction to joint 2 to prevent the structure from rotating around joint 1.

Write (and hand in) a well-commented script that gets the data by reading in the arrays `m` and `j` from (a suitably modified file) `home5.m`, then generates from it the matrix for this system, as well as the right sides for the four loads to be considered, then uses the backslash to solve, in one blow, for the forces in the four cases when the load vector at joint 10 is

a) $(0, 0, -10)$
b) $(0, 3, -8)$
c) $(2, 0, -8)$
d) $(0, 0.5, -12)$

then, finally, answers the following question: Which of the given loads exceeds the manufacturer's spec that no extension force should exceed 35 units and no compressive force should exceed 20 units?

You will have to decide how to number the equations, and how to construct the matrix (e.g., by rows, or by columns). (E.g., I first generated the 36×30 matrix column by column, then converted it to the final 30×30 matrix by using `matlab`'s command `A(b,:) = [];` which removes the rows `b(1), b(2), ...` from `A`.)

As a check, for the first load, the force on members 15 and 16 is the same, and is close to 3 units. As a further check, since all these load vectors have a strong negative z-component, the member (10,11), i.e., the member connecting joint 10 to joint 11, is certain to be extended (as are the members (11,12), (4,12) and (1,4), to judge from the picture).

2. (30 points) A naive approach to least-squares approximation to given data (x_i, y_i), $i = 1{:}n$, by polynomials of degree $< k$ is to solve the corresponding normal equations `V'*V*? = V'*y` with `V` the corresponding Vandermonde matrix `[x.^(k-1), x.^(k-2), ..., x.^0]`. The solution, `c`, provides the coefficients of the least-squares polynomial fit. These can be used, e.g., in statements like `vals = polyval(c,z)`, to evaluate the least-squares fit at the points in `z`.

Write and run a script that compares the above naive approach to three other approaches for the following particular situation. Compute the least-squares approximation by cubic polynomials (i.e., $k = 4$) to the data `x = linspace(999,1001,10)'; x-1000; y = (ans.^2-1).*ans;`

(a) by the above approach;
(b) by using `matlab`'s command `polyfit`;
(c) by applying the above approach to the modified data `x-mean(x), y` (which means that `vals = polyval(c,z-mean(x))` would evaluate the resulting fit from the computed coefficient vector `c`);
(d) by using `matlab`'s command `polyfit`, but applied to the modified data `x-mean(x),y`.

Since the data being fitted come from a cubic polynomial, we would expect a perfect fit. Judge and compare all four approaches by (i) how well the values of the fitted polynomial at the data abscissae `x` reproduce the given ordinates `y`; and (ii) the accuracy of the coefficients obtained.

State your conclusion concerning these four approaches.

3. (2*5 points) For each of the following, state whether it is true or false, and give a brief reason (or evidence) for your answer.

1. The linear system that arises in cubic spline interpolation is diagonally dominant, hence can be solved safely by Gauss elimination without pivoting.
2. If `A` is a nonsquare matrix and `A'*A` is invertible, then, up to roundoff, `A\B` gives the same answer as `inv(A'*A)*B`.
3. The norm of a matrix *always* equals the biggest norm of one of its columns.
4. Forward- and backward-substitution takes as many flops as does the initial factorizing of the coefficient matrix of the linear system being solved.
5. Computing a QR factorization is twice as much work as computing a PLU factorization.