Institutionen för datavetenskap
Umeå universitet

12 november 2010
Numeriska metoder för civilingenjörer

Theme: Network models and linear systems Part II

Rules:

- Each student should hand in an individually completed report at latest **November 22 at 12.00 (noon)**.
- You may discuss the problem among fellow students. If you receive considerable help from someone, say so in your solutions.
- Do not copy solutions or code from others. Do not lend your solution or your code to other students.

Computer exercises

1. Write a Matlab function that computes the pressure in the nodes of a given arbitrary network by setting up and solving the equation system (10) in Part I of this theme. The function head should be

   ```
   function p = computepressure(B, e, s, k0)
   ```

   Input parameters are an incidence matrix B (not the truncated $\bar{B}$) for an arbitrary directed graph, a vector $e = (\epsilon_1, \ldots, \epsilon_m)$ with conductivity parameters for all edges in the graph (the length of e should be equal to the number of columns of B), a vector s (the length of s should be equal to the number of rows of B) containing the volume flows at the nodes of the graph, and an integer k_0 that specifies which node that should be "grounded"; that is, it should hold that $p_{k_0} = 0$.

   Hint: The Matlab function `diag`[1] is useful!

   To test your program, choose conductivity parameters as in problem 2 (b) in the preparatory exercises, solve the problem with you function, and verify that you get the same answer as when calculating by hand!

2. Make up your own network with at least 10 nodes. Construct the incidence matrix for the directed graph, and solve equation (10) in the theme introduction for your network using $\epsilon_i = 1$ for all edges i and $s_i = 1$ for all nodes i except the one that is grounded (which could be any node).

3. By comparing the structure of the graphs with the structure of the matrices you obtain from problems 1 and 2, infer a rule for how many nonzero elements there are at a particular row in matrix $\bar{B}E\bar{B}^T$.

4. Modify your code so that it solves equation (10) in the introduction to the theme with B instead of $\bar{B}$ and with s instead of $\bar{s}$, that is, without "grounding" a node. What happens if you try to solve the modified problem? Compute $BEB^T p$ for $p = (1, \ldots, 1)^T$, and explain what the problem is!

[1]To read the documentation, type `doc diag` or `help diag` at the Matlab prompt. Note that documentation for *any* Matlab command is available by typing `doc` or `help` before the command.

5. Now assume that we would like to study much larger graphs; for instance those that could represent the network of water pipes in a residential area with hundreds of thousands of houses. Download from the course home page the files `Bc0.mat` and `Bc1.mat`, which contains incidence matrices of dimensions 399×1146 and 1545×4536. The Matlab command `load Bc0` and `load Bc1` loads these matrices from current directory. Use your code to solve problems with these incidence matrices, using any values for the sources and any (positive) values for the conductivities. Time the solution of the linear system associated with the incidence matrix in `Bc0.mat` using the commands `tic` and `toc` (read the documentation of these commands!). Then compute approximately the time t_f required for one floating point operation using the estimate of how many flops Gaussian elimination take. Use this value of t_f to estimate how long time the solution of the linear system associated with the incidence matrix in `Bc1.mat` will take. Compare with the time it really takes and comment!

6. As we have seen, network problems gives rise to large, sparse linear systems. There are specialized algorithms for such systems that are much more efficient and much less memory demanding than Gaussian elimination of the kind that we study in this course. (Note that our strategy to work explicitly with incidence matrices is not advisable when the graphs are very large, since these matrices contain mostly zeros.) Such specialized algorithms are available in Matlab. The matrix should then be stored in a special *sparse format,* in which only the nonzero elements are stored. (The documentation for the command `sparse` contains more information about the sparse format.) If a matrix K is already available as an “ordinary” matrix, it can be converted to sparse format through the command `K=sparse(K)`. Matlab’s backslash operator, being “smart”, recognizes when the matrix is defined in sparse format and choses then the specialized algorithm. For both matrices in problem 5, compare the times required for solution of the linear system when `sparse(K)` is used instead of `K`, where `K` is the matrix $\bar{B}E\bar{B}^T$. Is there any benefit with the specialized algorithm?