Theme: Network models and linear systems

Introduction

Various kinds of network problems arise again and again in many applications and yield linear systems of equations with particular properties. Figure 1 shows an example of a small network. Networks are modeled mathematically using graph theory. A graph consists of a collection of vertices (or nodes; noder in Swedish) and edges (kanter or bågar in Swedish); each edge connects two vertices. The network in Figure 1 is a directed graph that could represent an electrical network connecting components like resistors, capacitors, inductors, and electric sources. The network could alternatively illustrate transportation routes for goods between cities. More examples of systems that can be modeled using graphs are truss networks, computer networks, and the World Wide Web.

Here, we will take a closer look at networks of water pipes. Assume that each of the five numbered edges in Figure 1 represents a pipe through which water may flow. Each vertex represents either a source (the water company) of water or a water sink (residential houses, say). Assuming that we know the amount of water per time unit that is supplied or consumed at each vertex, we like to determine the water pressures that are needed at the various vertices. This is a relevant question when dimensioning pumps and pipes. We will describe a general methodology to solve this problem under simplified assumptions.

The arrows in Figure 1 indicate sign conventions for positive flux (flöde in Swedish) through the pipes. For example, water flows from vertex 1 to vertex 3 at a rate of \( u_1 \) m\(^3\)/s (\( u_1 < 0 \) indicates net flow from vertex 3 to vertex 1). Moreover, we know the rates \( s_i \) (in m\(^3\)/s) at which water is supplied or consumed at vertex \( i \); \( s_i > 0 \) means that water is added and \( s_i < 0 \) that water is removed from the network.

We assume that there are no leaks in the network. Thus, the flux of water out of vertex \( i \), \( -s_i \), must be exactly equal the sum of the fluxes that are coming into vertex \( i \) through the edges that are attached to vertex \( i \). The flow of water in the network (Figure 1) will then satisfy the equations

\[
-u_1 - u_2 = -s_1, \quad (\text{balance at node 1}),
\]

\[
u_2 - u_3 - u_5 = -s_2, \quad (\text{balance at node 2}),
\]

\[
u_1 + u_3 - u_4 = -s_3, \quad (\text{balance at node 3}),
\]

\[
u_4 + u_5 = -s_4, \quad (\text{balance at node 4}).
\]

Note that the plus or minus signs of equations (1) depend on the sign conventions; we have (arbitrary) assigned directions as in Figure 1 and defined positive sources as adding to the network.

Since there are no leaks in the system, and since water is hard to compress, it holds that everything that goes in must come out:

\[
s_1 + s_2 + s_3 + s_4 = 0. \quad (2)
\]

Equation (1) can be written in the matrix form

\[
Bu = -s, \quad (3)
\]

1See for instance Wikipedia’s article http://en.wikipedia.org/wiki/Graph_(mathematics)
2http://en.wikipedia.org/wiki/Directed_graph
Figure 1: An example of a network in terms of a directed graph with 4 vertices and 5 edges.

where

\[
B = \begin{pmatrix}
-1 & -1 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & -1 \\
1 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 1 & 1 \\
\end{pmatrix}, \quad u = \begin{pmatrix}
u_1 \\
u_2 \\
u_3 \\
u_4 \\
u_5 \\
\end{pmatrix}, \quad \text{and} \quad s = \begin{pmatrix}s_1 \\
s_2 \\
s_3 \\
s_4 \\
\end{pmatrix}.
\]

(4)

Matrix $B$ is called the incidence matrix\(^3\) of the directed graph and gives a complete description of how the network is connected—its “topology”. The incidence matrix reveals nothing, however, about the coordinates of the vertices or the edges (information that is not needed for the present problem). An incidence matrix for a directed graph has as many rows as vertices in the graph and as many columns as edges in the graph. Every column represents an edge and has $+1$ and $−1$ at the rows that correspond to the vertices in which there are inflow and outflow, respectively. All other entries in $B$ are zero. Note that the incidence matrix typically has more columns than rows, since a graph typically have more edges than vertices.

Denote the pressure (in Pa) at vertex $i$ by $p_i$. The flux of water through a particular edge depends on the difference of pressure between the two vertices that the edge connects. We assume a simple linear relationship so that the flux is proportional to the pressure difference:

\[
\begin{align*}
u_1 &= \epsilon_1(p_1 - p_3) \quad (\text{edge 1}), \\
u_2 &= \epsilon_2(p_1 - p_2) \quad (\text{edge 2}), \\
u_3 &= \epsilon_3(p_2 - p_3) \quad (\text{edge 3}), \\
u_4 &= \epsilon_4(p_3 - p_4) \quad (\text{edge 4}), \\
u_5 &= \epsilon_5(p_2 - p_4) \quad (\text{edge 5}),
\end{align*}
\]

(5)

where the coefficients $\epsilon_i \geq 0$ denotes the “conductivity” of that pipe; a property that can be determined by experiments, for instance. When writing equation (5) (which corresponds to Ohm’s law for electrical circuits) in matrix form, we see that the transpose of the incidence matrix also appears in the pressure–flux relation, that is

\[
u = −EB^T p, \tag{6}
\]

where

\[
E = \begin{pmatrix}
\epsilon_1 & 0 & 0 & 0 & 0 \\
0 & \epsilon_2 & 0 & 0 & 0 \\
0 & 0 & \epsilon_3 & 0 & 0 \\
0 & 0 & 0 & \epsilon_4 & 0 \\
0 & 0 & 0 & 0 & \epsilon_5 \\
\end{pmatrix} \quad \text{and} \quad p = \begin{pmatrix}p_1 \\
p_2 \\
p_3 \\
p_4 \\
\end{pmatrix}.
\]

(7)

\(^3\)http://en.wikipedia.org/wiki/Incidence_matrix
Recall, that the fluxes of water through the edges only depend on pressure differences. Thus, adding any constant to the pressure at all edges does not change the resulting flow. Hence, we can choose to use the pressure at, for example, vertex 4 as our reference, or equivalently (analogous to the concept of zero-potential “ground” in electrical circuits), we assume that the pressure is held at the constant value \( p_4 = 0 \). The unknown pressures \( p_1, p_2, \) and \( p_3 \) will then denote the pressure relative to the pressure at vertex 4. Also, we let \( s_1, s_2, \) and \( s_3 \) be given arbitrarily and define \( s_4 = -s_1 - s_2 - s_3 \) so that equation (2) is automatically satisfied. These simplifications imply that we may delete the fourth row in matrix \( B \) (since we already know what happens at vertex 4!) and remove \( s_4 \) and \( p_4 \) from the problem (since these are known). Hence, we have one less unknown (in \( u = -EB^T p \)) that yields one less equation (in the system \( Bu = -s \)). The reduced system is given by:

\[
\begin{align*}
\bar{B}u &= -\bar{s}, \\
u &= -\bar{E}B^T \rho,
\end{align*}
\]

where

\[
\bar{B} = \begin{pmatrix} -1 & -1 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & -1 \\
1 & 0 & 1 & -1 & 0 \end{pmatrix}, \quad \bar{s} = \begin{pmatrix} s_1 \\
2 s_2 \\
s_3 \end{pmatrix}, \quad \text{and} \quad \rho = \begin{pmatrix} p_1 \\
p_2 \\
p_3 \end{pmatrix}.
\]

Multiplying both sides of equation (8b) from the left with \( -\bar{B} \) and adding the resulting equation to equation (8a) yields the following equation in the reduced unknowns \( \bar{\rho} \):

\[
\bar{B}E\bar{B}^T \bar{\rho} = \bar{s}.
\]

**Theoretical exercises**

1. Compute, by hand, the \( LU \)-factors of matrix

\[
A = \begin{pmatrix}
1 & 1.5 & 1 & -1 \\
2 & 1 & 3 & -2 \\
0 & 2 & 1 & 1 \\
0 & 4 & 2 & 0
\end{pmatrix}.
\]

2. Consider equation system (10) in the particular case of the network of Figure 1.

   (a) Explicitly compute the left-hand-side matrix \( \bar{B}E\bar{B}^T \) for arbitrary values of the conductivity parameters \( \epsilon_1, \ldots, \epsilon_5 \).

   (b) Compute, by hand, the \( LU \) factorization of the above matrix for the particular conductivities \( \epsilon_1 = 1, \epsilon_2 = 2, \epsilon_3 = 2, \epsilon_4 = 3, \epsilon_5 = 4 \).

   (c) Use the \( LU \)-factorization you computed above to solve the system by hand for the right-hand side vector \( \bar{s} = (0, 4, 3)^T \).

**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). The function head should be

\[
\text{function } p = \text{computepressure}(B, e, s, k0)
\]

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` is useful!

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

2. Make up your own network with at least eight nodes. Construct the incidence matrix for the directed graph, and solve equation (10) 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) 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 $BE\bar{B}^T p$ for $p = (1, \ldots, 1)^T$, and explain what the problem is!

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 $399 \times 1146$ and $1545 \times 4536$, respectively. The Matlab commands `load Bc0` and `load Bc1` load 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?

---

4To 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.