MA519: Numerical Linear Algebra

Lab 2: Cholesky Factorisation

We've recently learned what Cholesky Factorisation is. Now we'll implement it, and investigate some of its properties.

1 The finite difference method

First, lets create a one-dimensional discrete Laplacian T_N, (our $(-1, 2, -1)$ matrix that we've studied do much) and from that make the two-dimensional version using that

$$T_{NN} = I \otimes T_N + T_N \otimes I.$$

An easy way to do this in Matlab is to use the sparse function as shown below. Here ... means a single Matlab line is split in two. I'll explain the syntax of the `sparse` function in class. The `speye` function returns the identity matrix, in a special "sparse" format.

```
N = 8; % points on 1D mesh
TN = sparse(2:N, 1:N-1, -1, N,N) ...
  + sparse(1:N, 1:N, 2,N,N) ...
  + sparse(1:N-1, 2:N, -1, N,N);
TNN = kron(speye(N),TN) ...
  + kron(TN, speye(N));
```

Now visualise the sparsity pattern using `spy(TN)` and `spy(TNN)`, as shown in Figure 1

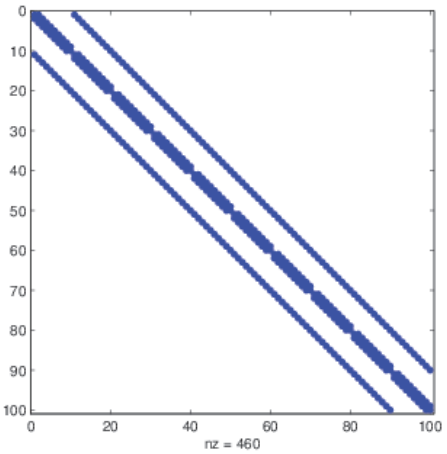

Figure 1: Sparsity pattern of T_{NN} with $N = 10$

To use this to solve a partial differential equation

$$-\Delta u = 1 \text{ on } \Omega = (0,1) \times (0,1)$$

with $u = 0$ on the boundary, try

```
m=N^2;
h =1/N;
b = h^2*ones(m,1);
x = A\b;
x = reshape(x,N,N)';
figure(2); mesh(x);
```

1.1 Cholesky (I)

First we'll investigate the efficiency of Cholesky factorisation, using the approach presented as Algorithm 2.11 in (Demmel, 1997)

```
for j=1:m
   L(j,j) = sqrt( A(j,j) ...
     - L(j,1:j-1)*L(j,1:j-1)');
   for i=j+1:m
      L(i,j) = (A(i,j) - ...
         L(i,1:j-1)*L(j,1:j-1)')/L(j,j);
   end
end
```

Rather than typing this in, you can download the code from the course website. The program is called

`TestCholesky1.m`

That program also verifies that the correct factorisation was indeed computed by reporting the norm $\|A - LL^T\|_2$.

This programme is very slow. What is the largest value of $m = N^2$ for which the factorisation will run in less than 10 seconds?

1.2 Sparse Cholesky

We learned in Lecture 21 that $A = T_{NN}$ is a banded $m \times m$ matrix, with $m = N^2$, with a band-width of N, and that if $A = LL^T$ then L is also a banded matrix, with band-width N. Modify the code above so that you only compute the necessary values of L.

1.3 Cholesky (II)

In Lecture 20 we also studied a version of the Cholesky algorithm as presented in Chap 4 of (Davis, 2006)

```
for j=1:m
   L(j, 1:j-1) = ...
     (L(1:j-1, 1:j-1) \ A(1:j-1,j))';
   L(j,j) = sqrt(A(j,j) ...
     - L(j,1:j-1)*L(j,1:j-1)');
end
```

Using this approach, what size matrix can you factorise in under 10 seconds?

Again, try to optimise this implementation so that it computes only the necessary factors.

Extra question: If you are comfortable with programming in Matlab, try writing a recursive Cholesky factorisation, based on the ideas outlined in Section 20.2 of

the notes. Note, however, this is feasible only for small matrices: large ones would require too many levels of recursion.

1.4 How sparse are your factors?

The `spy` function that displays the sparsity pattern of a matrix also tells you how many non-zero entries it has. This appears as `nz=x` below the figure. For example, if you factorise $A = T_{NN}$, with $N = 10$, and display the sparsity pattern of L, you should see that $nz = 1009$, as shown in Figure 2.

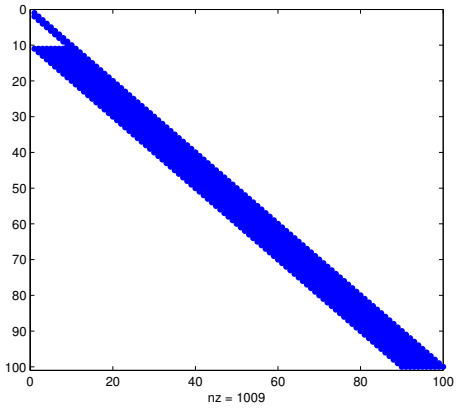

Figure 2: Sparsity pattern of L

It is possible to permute the rows and columns of A with a permutation matrix, P, so that the factorisation of $P^T AP$ has far fewer non-zeros than the factorisation of A.

In next week's class, we'll investigate how to do this. Right now, we'll use a built-in Matlab function that uses the *approximate minimum permutation*.

Set `p = amd(A);`. This returns a *permutation* vector, p. For example, if $A = T_{NN}$ with $N = 10$, you get $p = (26, 25, 50, 40, ...)$. Then set $B = A(p, p)$, as shown in Figure 3. This means that b_{ij} will be equal to $a_{p_i p_j}$. Equivalently, to form the permutation matrix explicitly, set `P=sparse(p, 1:m,1)`, and then $B = P^T AP$,

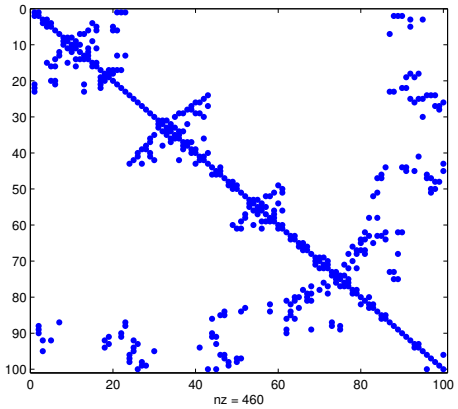

Figure 3: Sparsity pattern of $B = A(p, p)$

Now you should find that the Cholesky factor of B has only 648 non-zeros: a saving of 35% on the unpermuted matrix.

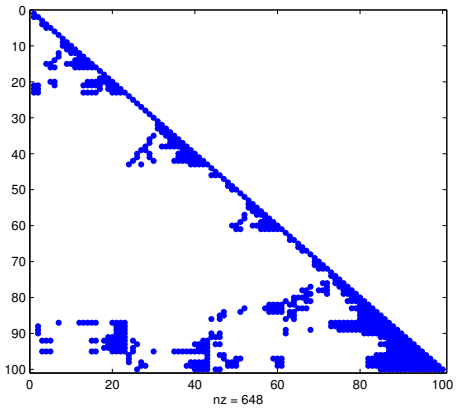

Figure 4: Sparsity pattern of the Cholesky factor of B