

MATLAB Lecture 8. Special Matrices in MATLAB

Almost all matrices coming from practical applications have some kind of special structure. Finite difference methods lead to sparse matrices. Adjacency matrices to define graphs are also sparse. Temperature distributions give rise to multi-dimensional matrices. See [1] and [2] for more examples.

8.1 Sparse Matrices

The matrices that arise in applications like finite element modelling or communication networks contain only few nonzero elements. We call such matrices *sparse*. Instead of storing *all* elements of the matrices, we better leave out the zeroes and only store the nonzero elements. Besides memory efficiency, also the calculations go much faster.

With the command **sparse** we can generate sparse matrices. The structure of a matrix can be visualized with **spy**. For example,

```
>> a = sparse(4,6,3,10,10)    % one nonzero element
>> spy(a)                     % visualize structure
```

creates a 10×10 matrix a with $a(4,6) = 3$. The plot created by **spy** is not so exciting. With MATLAB's special matrices gallery, it is trivial to plot a Sierpinski gasket. We take the Pascal matrix p , defined by

$$p(i,j) = \binom{i+j-2}{j-1} = \frac{(i+j-2)!}{(i-1)!(j-1)!}.$$

With the command **pascal** we can generate Pascal matrices for any dimension. The matrix is not sparse, but we can create one, by replacing the even elements by zero, and the odd ones by one.

```
>> p32 = pascal(32);          % Pascal matrix of dimension 32
>> s32 = rem(p32,2);          % p32 mod 2
>> spy(s32)                   % displays Sierpinski gasket
```

We can convert a matrix from the dense to the sparse representation in two ways:

```
>> s8 = rem(pascal(8),2)      % 8x8 sparse matrix
>> sp8a = sparse(s8)          % convert to sparse matrix
>> [i,j,s] = find(s8);        % returns indices : s8(i,j) == s
>> [i,j,s]                    % displays indices and values
>> sp8b = sparse(i,j,s)       % second way to convert
```

We will use the second construction above to create the tridiagonal matrix

$$\begin{bmatrix} 2 & 1 & & & & \\ 1 & 2 & 1 & & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 2 & 1 \\ & & & & 1 & 2 \end{bmatrix}$$

This is a matrix with 2 on the diagonal, ones on the subdiagonals, and zeroes everywhere else. Let us create such a matrix of dimension 10:

```
>> i = [1:10 2:10 1:9]       % row indices
>> j = [1:10 1:9 2:10]       % column indices
>> s = [2*ones(1,10) ones(1,9) ones(1,9)] % nonzero elements
>> b = sparse(i,j,s)          % create the matrix
>> spy(b)                     % display the structure
```

We can apply the regular matrix operations to sparse matrices, for instance `eig` to compute the eigenvalues. But there is also the sparse version `eigs` which computes only a few eigenvalues iteratively. The reason for these special functions is that in practice, sparse matrices can be huge and one is seldomly interested in knowing all eigenvalues and eigenvectors.

Type `help sparsfun` to see an overview of the methods available to compute with sparse matrices.

8.2 Modeling Communication Networks

Imagine a network of ten computers. The connectivity of the computers in the network is defined by the so-called *adjacency matrix* A , defined by

$$\begin{aligned} A(i, j) &= 1, \text{ if } i \text{ and } j \text{ are connected;} \\ &= 0, \text{ if there is no connection between } i \text{ and } j. \end{aligned}$$

We arrange our 10 computers as ten equidistant points on the unit circle in the plane. First we define the coordinates on the unit circle:

```
>> dt = 2*pi/10;           % spacing between angles of points
>> t = dt:dt:2*pi;        % range of ten angles
>> x = cos(t); y = sin(t); % compute coordinates
```

We represent the nodes in the network each by a red plus and use their numbers as labels:

```
>> plot(x,y,'r+');        % mark nodes by red plusses
>> axis([-1.5 1.5 -1.5 1.5]) % adjust viewing window
>> hold on                % put more on same plot
>> for i = 1:4             % place upper labels
    text(x(i),y(i)+0.1,int2str(i))
end;
>> text(x(5)-0.1,y(5),int2str(5)); % left label
>> for i = 6:9            % place lower labels
    text(x(i),y(i)-0.1,int2str(i))
end;
>> text(x(10)+0.1,y(10),int2str(10)); % right label
```

Suppose every computer is connected to every other one, then A consist of ones:

```
>> A = ones(10);          % define adjacency matrix
>> gplot(A,[x' y']);      % plot the graph
```

The command `gplot` visualizes the graph. The first argument of `gplot` is the adjacency matrix. The second argument is a matrix with two columns (observe the use of the transpose in `[x' y']`), listing the coordinates of the nodes in the graph.

Suppose we have two servers: computer 5 is connected to all odd labeled nodes and computer 10 is connected to all even labeled nodes. Of course, we connect the servers to each other. From the 100 possible links, we only have 9. Therefore we use the command `spalloc`:

```
>> A = spalloc(10,10,9); % allocate for 9 links
>> for j = 1:9           % enumerate labels
    if mod(j,2) == 1    % odd label
        A(5,j) = 1;    % connected to 5
    else                % even label
        A(10,j) = 1;   % connected to 10
    end                % end of if
```

```

    end;                                % end of for
>> A(5,5) = 0;                          % remove trivial link
>> A(5,10) = 1;                         % connect servers
>> A = A+A';                             % A is symmetric matrix

```

We can visualize the graph as before with **gplot** after marking the nodes and setting the labels.

Every computer is connected to every other one by at most three links. Compute A^2 and A^3 to see the number of possible ways to connect two computers with respectively 2 and 3 links.

8.3 Multi-Dimensional Matrices

Matrices of dimensions higher than two occur for instance from temperature measurements taken at a three-dimensional room. We have already seen **meshgrid** to generate two-dimensional grids. With **ndgrid** we can generate grids of any dimension.

Suppose we have a room where the temperature is distributed like

$$T(x, y, z) = ye^{-x^2 - y^2 - z^2} \quad (1)$$

If our room is the cube $[-2, 2] \times [-2, 2] \times [-2, 2]$, we can take evenly spaced grid points at distance 0.1 for each coordinate. The data is then generated by

```

>> r = -2:0.1:2;                        % define range for each coordinate
>> [x,y,z] = ndgrid(r,r,r);             % generate the grid
>> t = y .* exp(-x.^2 - y.^2 - z.^2); % sample at the grid

```

Observe the use of `.*` and the array power `.^` to perform the calculations componentwise.

To visualize the data, the command **slice** produces a volumetric slice plot. This means that we take slices in the coordinate directions, e.g., fix $y = -1.2$. The equation $y = -1.2$ defines a plane, which slices through the solid (which is in our example a cube). With the values of the temperature distribution, **slice** uses interpolation to color the points in the slicing plane. The command below makes a volumetric plot of the temperature distribution with three slices : $x = -1.2$, $x = 0.8$, and $z = -0.2$.

```

>> slice(y,x,z,t,[ ],[-1.2 .8],[-.2]); % plot three slices

```

It is an interesting exercise to make an animation of a moving slice.

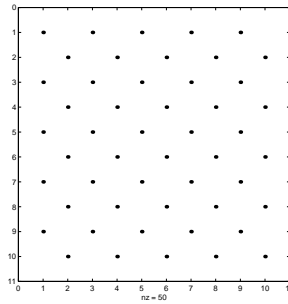
8.4 Assignments

1. Define a square matrix of dimension 12 with the following structure

$$\begin{bmatrix} -2 & 3 & & & & \\ 1 & -2 & 3 & & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 3 \\ & & & & 1 & -2 \end{bmatrix}$$

which has -2 on the diagonal, 1 on the lower, 3 on upper subdiagonal, and zeroes elsewhere.

2. Use **invhilb** to generate the inverse of the Hilbert matrix of dimension 10. Make a sparse matrix by setting the negative elements to zero. (*Hint: use **abs** and **+**.*) Then do **spy** on the result to create



3. The connections in a network of ten computers are defined as follows: computer i is connected to computer $i + 3$ modulo 10. For example, if A is the adjacency matrix, then $A(3, 6) = 1$, $A(6, 9) = 1$, and $A(9, 2) = 1$. Give the MATLAB instructions to define this adjacency matrix and to display the graph structure. How many links does it take at most to go from one computer to another one?
4. Define a function `plotgraph` taking the adjacency matrix A as only input argument. Extract the number of nodes from A with `size(A,1)`. Arrange the nodes equidistantly on the unit circle in the plane and label the nodes with numbers. After defining the coordinates of the nodes, use `gplot`.
5. For the temperature function (1) above, make a MATLAB movie with one vertical slice in the x coordinate, ranging from -2 to 2 , taking incremental steps of 0.1 .

References

- [1] D.J. Higham and N.J. Higham. *MATLAB Guide*. SIAM, 2000.
- [2] A. Knight. *Basics of MATLAB and beyond*. Chapman & Hall/CRC, 2000.