

MA318 Introduction to Programming in Maple: III

Creation of LARGE matrices

We will often have to deal with large matrices. It would be tedious to define these by inserting the individual entries by hand. For certain special matrices, *Maple* provides extra functionality to define them:

```
> Z:=matrix(10,10,0);
```

To define diagonal matrices, we can use the command `diag`:

```
> d:=diag(1,2,3,4,5);
```

Thus we can define a identity matrix of given size:

```
> U5:=diag(1,1,1,1,1,1,1);
```

If we want to define a large ‘sparse’ matrix (i.e. with only few entries in special positions) we can proceed as follows:

```
> Tst:=matrix(5,5,0);
```

```
> Tst[2,2]:=1; Tst[2,3]:=2; Tst[2,4]:=3; Tst[5,5]:=1;
```

```
> evalm(Tst);
```

We can construct new matrices from old ones, for instance if the entries of the new matrix are certain function values of the entries of the old one. In this case we can use *Maple*’s `map` - function: In the following example, the function $x \mapsto x + i$ is ‘mapped into’ the matrix A . As a result we get the matrix, whose entries are those of A , with the variable i added.

```
> C:=map(x->x+i,A);
```

$$C := \begin{bmatrix} 1+i & 2+i & 3+i & 4+i \\ 5+i & 6+i & 7+i & 8+i \\ 9+i & 10+i & 11+i & 12+i \end{bmatrix}$$

```
> C:=map(x->x+3,A);
```

$$C := \begin{bmatrix} 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix}$$

We can do the same thing with ‘self written’ functions or procedures. In the following lines we define a procedure, called `f`, which takes a natural number k as input and returns the sum $0 + 1 + 2 + \dots + k$. In a second procedure, this function is then mapped into the matrix A .

```
> f:=proc(k)
```

```
> local u,j;
```

```
> u:=0;
```

```
> for j from 0 to k do u:=u+j; od;
```

```
> RETURN(u);
```

```
> end;
```

```
> g:=proc(A)
```

```
> RETURN(map(f,A));
```

```

> end;
      g := proc(A) RETURN(map(f, A)) end
> A:=matrix(2,3,[1,2,3,4,5,6]);
      A :=  $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ 
> g(A);
       $\begin{bmatrix} 1 & 3 & 6 \\ 10 & 15 & 21 \end{bmatrix}$ 
> g(g(A));
       $\begin{bmatrix} 1 & 6 & 21 \\ 55 & 120 & 231 \end{bmatrix}$ 

```

We can also construct a matrix using a function to specify the entries. For example:

```

> f:=(i,j)-> 2*i+3*j;
      f := (i, j) → 2i + 3j
> matrix(4,5,f);
       $\begin{bmatrix} 5 & 8 & 11 & 14 & 17 \\ 7 & 10 & 13 & 16 & 19 \\ 9 & 12 & 15 & 18 & 21 \\ 11 & 14 & 17 & 20 & 23 \end{bmatrix}$ 

```

Exercises

1. Write a procedure which takes an integer n as input and produces an $n \times n$ matrix where the entry in row i , column j is 0 if $i + j$ is odd and 1 if $i + j$ is even.
2. Given a vector $u = [u_1, u_2, \dots, u_n]$, the corresponding *Vandermonde* matrix is an $n \times n$ matrix V satisfying $V[i, j] = u_i^{j-1}$. For example, if $u = [2, 3, 4]$ then

$$V = \begin{pmatrix} 1 & 2 & 2^2 \\ 1 & 3 & 3^2 \\ 1 & 4 & 4^2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{pmatrix}.$$

Write a procedure which takes a vector u as input and produces the corresponding V as output. Hint: the command `nops` can be used to determine the number of entries in u , i.e., n .

3. Write a procedure that takes a matrix A and a natural number n and returns the matrix B which has in each position the n 'th power of the corresponding entry in A .

can be viewed as ‘matrix - vector’ - equation

$$A \& * x = d; \quad (\text{Eq})$$

where $A = (a_{ik})$ is the ‘coefficient matrix’, $x = (x_1, x_2, \dots, x_n)$ the vector of unknowns and $d = (d_1, d_2, \dots, d_m)$.

Consider the ‘augmented matrix’ matrix (A, d) . It is easy to see that (A, d) can be transformed into row - reduced echelon form (A', d') , using *elementary row operations*. This will be demonstrated in the following examples. How can we read off the solutions of (Eq) from (A', d') ? The recipe is as follows:

1. If a leading one occurs in the last column d' , there is no solution. So assume now that all leading ones occur in columns of A' .
2. Remove the zero rows $r + 1, \dots, m$;
3. ‘Blow the whole matrix up’ to the format $n \times (n + 1)$ by inserting zero vectors as row - vectors in such a way that the leading 1’s come into diagonal position.
4. Now we change the diagonal zeros to -1 .
5. Then the last column d'' is a *special solution* of the original equation $A \& * x = d$. The columns c_1, c_2, \dots, c_{n-r} containing the diagonal -1 ’s are ‘basic solutions’ of the system $A \& * x = 0$.
6. Any solution of $A \& * x = d$ is of the form

$$\lambda_1 c_1 + \lambda_2 c_2 + \dots + \lambda_{n-r} c_{n-r} + d''$$

with scalars $\lambda_1, \lambda_2, \dots, \lambda_{n-r}$.

Let us do an example:

```
> evalm(R);
```

$$\begin{bmatrix} 1 & 2 & 0 & -1 & 0 & 2 \\ 0 & 0 & 1 & 5 & 0 & -2 \\ 0 & 0 & 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

There are 3 leading ones, so $r=3$ and we remove the last row:

```
> R1:=delrows(R,4..4);
```

$$R1 := \begin{bmatrix} 1 & 2 & 0 & -1 & 0 & 2 \\ 0 & 0 & 1 & 5 & 0 & -2 \\ 0 & 0 & 0 & 0 & 1 & 7 \end{bmatrix}$$

Blowing up:

We include the following small procedure, which inserts a given array v as i th row into matrix A :

```
> insertrow:=proc(A,v,i)
> local result, w,j,rows;
> w:=matrix(1,nops(v),v);
> result:=extend(A,1,0,0);
> rows:=rowdim(A)+1;
> result:=copyinto(w,result,rows,1);
> for j from i to rows do result:=swaprow(result,j,rows);od;
> RETURN(result);
> end;
> R2:=insertrow(R1,[0,-1,0,0,0,0],2);
      R2 := result
> evalm(R2);
```

$$\begin{bmatrix} 1 & 2 & 0 & -1 & 0 & 2 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 5 & 0 & -2 \\ 0 & 0 & 0 & 0 & 1 & 7 \end{bmatrix}$$

```
> R3:=insertrow(R2,[0,0,0,-1,0,0],4);
      R3 := result
> evalm(R3);
```

$$\begin{bmatrix} 1 & 2 & 0 & -1 & 0 & 2 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 5 & 0 & -2 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 7 \end{bmatrix}$$

To extract the appropriate columns from R3 we can use the **subvector** - command of *Maple* :

```
> sp:=subvector(R3,1..5,6);
      sp := [2, 0, -2, 0, 7]
> h1:=subvector(R3,1..5,2);
      h1 := [2, -1, 0, 0, 0]
> h2:=subvector(R3,1..5,4);
      h2 := [-1, 0, 5, -1, 0]
```

```

> A:=submatrix(R,1..4,1..5);

```

$$A := \begin{bmatrix} 1 & 2 & 0 & -1 & 0 \\ 0 & 0 & 1 & 5 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```

> evalm(A*sp);

```

$$[2, -2, 7, 0]$$

```

> evalm(A*h1);

```

$$[0, 0, 0, 0]$$

```

> evalm(A*h2);

```

$$[0, 0, 0, 0]$$

Let us do a complete example, starting with a system not yet in row - reduced echelon form:

```

> A:=matrix(4,6,[1,0,2,8,3,3,2,0,3,13,6,4,4,0,-1,5,12,-1,1,0,1,5,3,1]);

```

$$A := \begin{bmatrix} 1 & 0 & 2 & 8 & 3 & 3 \\ 2 & 0 & 3 & 13 & 6 & 4 \\ 4 & 0 & -1 & 5 & 12 & -1 \\ 1 & 0 & 1 & 5 & 3 & 1 \end{bmatrix}$$

```

> d:=vector(4,[72,117,75]);

```

$$d := [72, 117, 75, d_4]$$

we want to specify d4 later...

```

> Ag:=augment(A,d);

```

$$Ag := \begin{bmatrix} 1 & 0 & 2 & 8 & 3 & 3 & 72 \\ 2 & 0 & 3 & 13 & 6 & 4 & 117 \\ 4 & 0 & -1 & 5 & 12 & -1 & 75 \\ 1 & 0 & 1 & 5 & 3 & 1 & d_4 \end{bmatrix}$$

Now we bring Ag into row-echelon normal form:

```

> Ag1:=addrow(Ag,1,2,-2);

```

$$Ag1 := \begin{bmatrix} 1 & 0 & 2 & 8 & 3 & 3 & 72 \\ 0 & 0 & -1 & -3 & 0 & -2 & -27 \\ 4 & 0 & -1 & 5 & 12 & -1 & 75 \\ 1 & 0 & 1 & 5 & 3 & 1 & d_4 \end{bmatrix}$$

> Ag2:= addrow(Ag1,1,3,-4);

$$Ag2 := \begin{bmatrix} 1 & 0 & 2 & 8 & 3 & 3 & 72 \\ 0 & 0 & -1 & -3 & 0 & -2 & -27 \\ 0 & 0 & -9 & -27 & 0 & -13 & -213 \\ 1 & 0 & 1 & 5 & 3 & 1 & d_4 \end{bmatrix}$$

> Ag3:=addrow(Ag2,1,4,-1);

$$Ag3 := \begin{bmatrix} 1 & 0 & 2 & 8 & 3 & 3 & 72 \\ 0 & 0 & -1 & -3 & 0 & -2 & -27 \\ 0 & 0 & -9 & -27 & 0 & -13 & -213 \\ 0 & 0 & -1 & -3 & 0 & -2 & -72 + d_4 \end{bmatrix}$$

> Ag4:=mulrow(Ag3,2,-1);

$$Ag4 := \begin{bmatrix} 1 & 0 & 2 & 8 & 3 & 3 & 72 \\ 0 & 0 & 1 & 3 & 0 & 2 & 27 \\ 0 & 0 & -9 & -27 & 0 & -13 & -213 \\ 0 & 0 & -1 & -3 & 0 & -2 & -72 + d_4 \end{bmatrix}$$

> Ag5:=addrow(Ag4,2,1,-2);

$$Ag5 := \begin{bmatrix} 1 & 0 & 0 & 2 & 3 & -1 & 18 \\ 0 & 0 & 1 & 3 & 0 & 2 & 27 \\ 0 & 0 & -9 & -27 & 0 & -13 & -213 \\ 0 & 0 & -1 & -3 & 0 & -2 & -72 + d_4 \end{bmatrix}$$

> Ag6:=addrow(Ag5,2,3,9);

$$Ag6 := \begin{bmatrix} 1 & 0 & 0 & 2 & 3 & -1 & 18 \\ 0 & 0 & 1 & 3 & 0 & 2 & 27 \\ 0 & 0 & 0 & 0 & 0 & 5 & 30 \\ 0 & 0 & -1 & -3 & 0 & -2 & -72 + d_4 \end{bmatrix}$$

> Ag7:=addrow(Ag6,2,4,1);

$$Ag7 := \begin{bmatrix} 1 & 0 & 0 & 2 & 3 & -1 & 18 \\ 0 & 0 & 1 & 3 & 0 & 2 & 27 \\ 0 & 0 & 0 & 0 & 0 & 5 & 30 \\ 0 & 0 & 0 & 0 & 0 & 0 & -45 + d_4 \end{bmatrix}$$

> Ag8:=mulrow(Ag7,3,1/5);

$$Ag8 := \begin{bmatrix} 1 & 0 & 0 & 2 & 3 & -1 & 18 \\ 0 & 0 & 1 & 3 & 0 & 2 & 27 \\ 0 & 0 & 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & -45 + d_4 \end{bmatrix}$$

> Ag9:=addrow(Ag8,3,1,1);

$$Ag9 := \begin{bmatrix} 1 & 0 & 0 & 2 & 3 & 0 & 24 \\ 0 & 0 & 1 & 3 & 0 & 2 & 27 \\ 0 & 0 & 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & -45 + d_4 \end{bmatrix}$$

> Ag10:=addrow(Ag9,3,2,-2);

$$Ag10 := \begin{bmatrix} 1 & 0 & 0 & 2 & 3 & 0 & 24 \\ 0 & 0 & 1 & 3 & 0 & 0 & 15 \\ 0 & 0 & 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & -45 + d_4 \end{bmatrix}$$

Ag10 is in row-echelon normal form. So this tells us that the original system only has a solution if $d_4=45$.

So now we assume $d_4=45$ and apply the earlier recipe:

> C:= evalm(delrows(Ag10,4..4));

$$C := \begin{bmatrix} 1 & 0 & 0 & 2 & 3 & 0 & 24 \\ 0 & 0 & 1 & 3 & 0 & 0 & 15 \\ 0 & 0 & 0 & 0 & 0 & 1 & 6 \end{bmatrix}$$

Blowing up:

> B:=evalm(insertrow(C,[0,-1,0,0,0,0,0],2));

$$B := \begin{bmatrix} 1 & 0 & 0 & 2 & 3 & 0 & 24 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 & 15 \\ 0 & 0 & 0 & 0 & 0 & 1 & 6 \end{bmatrix}$$

> B:=evalm(insertrow(B,[0,0,0,-1,0,0,0],4));

$$B := \begin{bmatrix} 1 & 0 & 0 & 2 & 3 & 0 & 24 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 & 15 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 6 \end{bmatrix}$$

> B:=evalm(insertrow(B,[0,0,0,0,-1,0,0],5));

$$B := \begin{bmatrix} 1 & 0 & 0 & 2 & 3 & 0 & 24 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 & 15 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 6 \end{bmatrix}$$

> ss:=subvector(B,1..6,7);

$$ss := [24, 0, 15, 0, 0, 6]$$

> hh1:=subvector(B,1..6,2);

$$hh1 := [0, -1, 0, 0, 0, 0]$$

> hh2:=subvector(B,1..6,4);

$$hh2 := [2, 0, 3, -1, 0, 0]$$

> hh3:=subvector(B,1..6,5);

$$hh3 := [3, 0, 0, 0, -1, 0]$$

So ss is a special solution of $A \cdot x = d$ whereas hh1, hh2 and hh3 form a basis of the solution space for the homogeneous system $A \cdot x = 0$.

Let us check:

> evalm(A*ss);

$$[72, 117, 75, 45]$$

> evalm(A*hh1);

$$[0, 0, 0, 0]$$

> evalm(A*hh2);

$$[0, 0, 0, 0]$$

> evalm(A*hh3);

$$[0, 0, 0, 0]$$

Exercises

5. Solve the equation $Ax = d$ for

$$A := \begin{bmatrix} -110 & -110 & 9 & -330 & 9 & 51 \\ 406 & 406 & -32 & 1218 & -32 & -187 \\ -187 & -187 & 15 & -561 & 15 & 86 \\ 37 & 37 & -3 & 111 & -3 & -17 \end{bmatrix} \text{ and } d := [-297, 1095, -505, 100].$$

6. Let $A := \begin{bmatrix} 1 & -1 & 0 & -1 & 0 & 0 \\ -3 & 3 & 1 & 3 & -1 & 0 \\ -4 & 4 & 0 & 4 & 0 & 1 \\ 11 & -11 & -3 & -11 & 3 & 0 \\ -21 & 21 & 7 & 21 & -7 & 0 \end{bmatrix}$ and $d := [1, -5, -5 - 2c, 17 + c, -35]$.

Find c such that the equation $Ax = d$ has solutions and find them all.