

## Chapter 5

# Linear Systems of Differential Equations

### Project 5.1

### Automatic Solution of Linear Systems

Calculations with numerical matrices of order greater than 3 are most frequently carried out with the aid of calculators or computers. For example, in Example 8 of Section 5.1 in the text we needed to solve the linear system

$$\begin{aligned}2c_1 + 2c_2 + 2c_3 &= 0 \\2c_1 - 2c_3 &= 2 \\c_1 - c_2 + c_3 &= 6.\end{aligned}\tag{1}$$

Writing this system in matrix notation  $\mathbf{AC} = \mathbf{B}$  and working with a TI-86 calculator, for instance, we would store the  $3 \times 3$  coefficient matrix  $\mathbf{A}$  and the column vector  $\mathbf{B}$  with the commands

$$\begin{aligned}[[2,2,2] [2,0,-2] [1,-1,1]] &\rightarrow \mathbf{A} \\[[0] [2] [6]] &\rightarrow \mathbf{B}\end{aligned}$$

in which each matrix and each of its successive rows is enclosed within square brackets. Then we calculate and store the solution  $\mathbf{C} = \mathbf{A}^{-1} \mathbf{B}$  with the command

$$\mathbf{A}^{-1} * \mathbf{B} \rightarrow \mathbf{C}$$

which yields the result

$$\begin{aligned}[[ 2] \\[-3] \\[ 1]]\end{aligned}$$

Thus  $c_1 = 2$ ,  $c_2 = -3$ , and  $c_3 = 1$ , as we found using elementary row operations in Example 8 in the text.

Matrix notations and calculations in most computer systems are similar. In the sections below, we illustrate how to enter matrices and perform simple matrix calculations using *Maple*, *Mathematica*, and *MATLAB*. The "investigations" that follow are simple exercises to familiarize you with automatic matrix computation. Applications will appear in the remaining projects in this chapter.

### Investigation A

To practice simple matrix algebra with whatever calculator or computer system is available, you might begin by defining a square matrix  $\mathbf{A}$  of your own selection. Then calculate its inverse matrix  $\mathbf{B}$  and check that the matrix product  $\mathbf{AB}$  is the identity matrix. Do this with several square matrices of different dimensions.

### Investigation B

Use matrix algebra as indicated above (for the computations of Example 8) to solve Problems 31 through 40 of Section 5.1 in the text.

### Investigation C

An interesting  $n \times n$  matrix is the **Hilbert matrix**  $\mathbf{H}_n$  whose  $ij$ th element (in the  $i$ th row and  $j$ th column) is  $1/(i + j - 1)$ . For instance, the  $4 \times 4$  Hilbert matrix is

$$\mathbf{H}_4 = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}. \quad (2)$$

Set up the Hilbert matrices of orders  $n = 3, 4, 5, 6, \dots$ , and calculate their inverse matrices. The results will look more interesting than you might anticipate.

(1) Show that the system  $\mathbf{H}_3 \mathbf{x} = \mathbf{b}$  has the exact solution  $\mathbf{x} = [1 \ 1 \ 1]^T$  if

$$\mathbf{b} = \left[ \frac{1}{6} \quad \frac{13}{12} \quad \frac{47}{60} \right]^T \approx [1.83333 \ 1.08333 \ 0.78333]^T,$$

whereas it has the exact solution  $\mathbf{x} = [0.6 \ 2.4 \ 0]^T$  if  $\mathbf{b} = [1.8 \ 1.1 \ 0.8]^T$ . Thus linear systems with coefficient matrix  $\mathbf{H}_3$  can be quite "unstable" with respect to roundoff errors in the constant vector  $\mathbf{b}$ . (This example is essentially the one given by Steven H. Weintraub on page 324 of the April 1986 issue of the *American Mathematical Monthly*.)

(2) Show that the system  $\mathbf{H}_4 \mathbf{x} = \mathbf{b}$  has the exact solution  $\mathbf{x} = [1 \ 1 \ 1 \ 1]^T$  if

$$\mathbf{b} = \left[ \frac{25}{12} \quad \frac{77}{60} \quad \frac{57}{60} \quad \frac{319}{420} \right]^T \approx [2.08333 \ 1.28333 \ 0.95000 \ 0.75952]^T,$$

whereas it has the exact solution  $\mathbf{x} = [1.28 \ -1.8 \ 7.2 \ -2.8]^T$  if

$$\mathbf{b} = [2.08 \ 1.28 \ 0.95 \ 0.76]^T.$$

## Using *Maple*

In order to carry out matrix computations the *Maple linalg* package must first be loaded,

```
with(linalg):
```

A particular matrix can be entered either with a command of the form

```
A := matrix( 3,3, [2,2,2, 2,0,-2, 1,-1,1]);
```

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

where the first two arguments prescribe the numbers of rows and columns and the third argument is a vector listing the elements of  $\mathbf{A}$  row-by-row, or with one of the form

```
A := matrix( [[2,2,2], [2,0,-2], [1,-1,1]] );
```

in which the individual row vectors of  $\mathbf{A}$  are prescribed. The inverse matrix  $\mathbf{A}^{-1}$  is then calculated with the command

```
B := inverse(A);
```

$$B := \begin{bmatrix} \frac{1}{8} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & 0 & -\frac{1}{2} \\ \frac{1}{8} & -\frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

The function **evalm** (for **evaluate matrices**) is used for matrix multiplication, with **&\*** denoting the matrix multiplication operator itself. Thus we can verify that  $\mathbf{B}$  actually is the inverse matrix of  $\mathbf{A}$  with the calculation

```
evalm( B &* A );
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Having defined the right-hand side constant (column) vector

```
b := matrix( 3,1, [0,2,6] );
```

$$b := \begin{bmatrix} 0 \\ 2 \\ 6 \end{bmatrix}$$

in (1), we can then calculate the solution vector  $\mathbf{c} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{Bb}$  with the command

```
c := evalm( B &* b );
```

$$c := \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix}$$

For higher-dimensional linear systems the computation of the inverse matrix is not so efficient as immediate application of *Maple's* **linsolve** function:

```
c := linsolve( A, b );
```

$$c := \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix}$$

A matrix can also be defined by prescribing its  $ij$ th element as a function of the row index  $i$  and the column index  $j$ . For instance, the  $3 \times 3$  Hilbert matrix is defined by

```
H3 := matrix( 3,3, (i,j) -> 1/(i+j-1) );
```

$$H3 := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

The **linalg** package also includes an explicit Hilbert matrix function. For instance, the command

```
H4 := hilbert(4);
```

generates the  $4 \times 4$  Hilbert matrix displayed in (2). Finally, the **diag** function can be used to generate the most ubiquitous of all special matrices — the identity matrices such as

```
I3 = diag(1,1,1);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

## Using *Mathematica*

A particular matrix is entered in *Mathematica* as a list of row vectors, each row vector itself being a list of elements, as in the command

```
A = { {2,2,2}, {2,0,-2}, {1,-1,1} }
{{2, 2, 2}, {2, 0, -2}, {1, -1, 1}}
```

The **MatrixForm** function can be used to display **A** in standard matrix form:

```
A // MatrixForm
  2    2    2
  2    0   -2
  1   -1    1
```

The inverse matrix  $\mathbf{A}^{-1}$  is then calculated with the command

```
B = Inverse[A]
  1    1    1    1    1    1    1    1
  {-, -, -}, {-, 0, -(-)}, {-, -(-), -}
  8    4    4    4    2    8    4    4
```

The period `.` denotes the matrix multiplication operator in *Mathematica*. Thus we can verify that **B** actually is the inverse matrix of **A** with the calculation

```
B . A // MatrixForm
  1    0    0
  0    1    0
  0    0    1
```

Having defined the right-hand side constant (column) vector

```
b = { {0}, {2}, {6} }
{{0}, {2}, {6}}
```

in (1), we can then calculate the solution vector  $\mathbf{c} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{Bb}$  with the command

```
c = B . b
{{2}, {-3}, {1}}
```

For higher-dimensional linear systems the computation of the inverse matrix is not so efficient as immediate application of *Mathematica's* **LinearSolve** function:

```
c = LinearSolve[ A, b ]  
{ {2}, {-3}, {1} }
```

The computation

```
A . c - b  
{ {0}, {0}, {0} }
```

then verifies that **c** is a solution of the equation **Ax = b**.

A matrix can also be defined by prescribing its *ij*th element as a function of the row index *i* and the column index *i*. For instance, the  $3 \times 3$  Hilbert matrix is defined by

```
H3 = Table[ 1/(i+j-1), {i,1,3},{j,1,3} ]  
{ {1, 1/2, 1/3}, {1/2, 1/3, 1/4}, {1/3, 1/4, 1/5} }
```

```
H3 // MatrixForm
```

```
1 1/2 1/3  
1/2 1/3 1/4  
1/3 1/4 1/5
```

Finally, the **DiagonalMatrix** function can be used to generate the most ubiquitous of all special matrices — the identity matrices such as

```
I3 = DiagonalMatrix[ {1,1,1} ]  
{ {1, 0, 0}, {0, 1, 0}, {0, 0, 1} }
```

Actually, the identity matrix has its own function (of the dimension):

```
IdentityMatrix[3] // MatrixForm  
1 0 0  
0 1 0  
0 0 1
```

## Using MATLAB

A particular matrix can be entered either with a command of the form

```
A = [ [2 2 2]; [2 0 -2]; [1 -1 1] ]
A =
     2     2     2
     2     0    -2
     1    -1     1
```

where the row vectors of **A** are separated by semicolons, or with one of the simplest possible form

```
A = [ 2  2  2
       2  0 -2
       1 -1  1 ]
```

in which the matrix is "built" just as it looks. The inverse matrix  $\mathbf{A}^{-1}$  is then calculated with the command

```
B = inv(B)
B =
    0.1250    0.2500    0.2500
    0.2500         0   -0.5000
    0.1250   -0.2500    0.2500
```

The ordinary MATLAB multiplication operator **\*** suffices to multiply matrices of appropriate dimensions. Thus we can verify that **B** actually is the inverse matrix of **A** with the calculation

```
B*A
ans =
     1     0     0
     0     1     0
     0     0     1
```

Having defined the right-hand side constant (column) vector

```
b = [0; 2; 6]
b =
     0
     2
     6
```

in (1), we can then calculate the solution vector  $\mathbf{c} = \mathbf{A}^{-1} \mathbf{b} = \mathbf{Bb}$  with the command

```
c = B*b
```

```

c =
    2
   -3
    1

```

For higher-dimensional linear systems the computation of the inverse matrix is not so efficient as immediate application of MATLAB's "linear solve" function in the form of the operator `\` that denotes "matrix *left* division":

```

c = A\b
c =
    2
   -3
    1

```

A matrix can also be defined with a double loop prescribing its  $ij$ th element as a function of the row index  $i$  and the column index  $j$ . For instance, the  $3 \times 3$  Hilbert matrix  $\mathbf{H}_3$  is defined by

```

for i = 1 : 3
    for j = 1 : 3
        H3(i,j) = 1/(i+j-1);
    end
end

```

using the notation  $\mathbf{A}(i, j)$  for the  $ij$ th element of the matrix  $\mathbf{A}$ . After specifying "rational formatting" of output, we can display the result as

```

format rat
H3
H3 =
    1          1/2        1/3
    1/2        1/3        1/4
    1/3        1/4        1/5

```

MATLAB also includes an explicit Hilbert matrix function. For instance, the command

```

H4 = hilb(4)

```

generates the  $4 \times 4$  Hilbert matrix  $\mathbf{H}_4$  displayed in (2).

Finally, the `diag` function can be used to generate the most ubiquitous of all special matrices — the identity matrices such as

```

I4 = diag([1 1 1 1])

```

```
I4 =  
    1    0    0    0  
    0    1    0    0  
    0    0    1    0  
    0    0    0    1
```

Actually, the identity matrix has its own MATLAB function (of the dimension):

```
I5 = eye(5)  
I5 =  
    1    0    0    0    0  
    0    1    0    0    0  
    0    0    1    0    0  
    0    0    0    1    0  
    0    0    0    0    1
```