AMTH142 Lecture 9

Scilab — Introduction

April 20, 2007

Contents

9.1 What is Scilab? ................................................. 2

9.2 Working with Scilab ......................................... 3
  9.2.1 Matrices ................................................. 3
  9.2.2 Subscripts ............................................... 4
  9.2.3 Numbers .................................................. 5
  9.2.4 Complex Numbers ....................................... 6
  9.2.5 Variables ................................................ 7
  9.2.6 Constants ............................................... 8
  9.2.7 Syntax ................................................... 8

9.3 Vectors and Matrices ....................................... 9
  9.3.1 The Colon Operator ..................................... 9
  9.3.2 Transpose ............................................... 10
  9.3.3 Matrix Arithmetic ..................................... 12
  9.3.4 The Dot Operator ..................................... 15
  9.3.5 Special Matrices ...................................... 17
  9.3.6 Functions ............................................... 18
  9.3.7 Concatenation ......................................... 20
9.1 What is Scilab?

Numerical Software

Numerical software, as distinct from a computer algebra system like Maxima, works exclusively with floating point arithmetic. Thus, except for calculations with small integers, all calculations are approximate. However because arithmetic is performed in hardware it is much faster than arithmetic with large integers and arbitrary precision floating point numbers. Numerical computation is the only practical approach to large scale scientific computation.

Numerical Libraries

Among the very first uses of computers in the late 1940’s was the solution of systems of linear equations in finite precision arithmetic. Over the next couple of decades a large amount of software for for numerical computation was produced, with the best software organized into libraries of functions for specific groups of tasks. This process is continuing with the best known of these packages being linpack and its successor lapack for numerical linear algebra. Originally most software was written in Fortran although C and C++ are becoming more common as implementation languages.

Interactive Systems

There are three major systems:

- Matlab
- Scilab
- Octave

Matlab is a commercial product, the other two are free software. Matlab is also the oldest of the three having been around for more than twenty years. All three are quite similar:

1. The main function of these systems is to provide a convenient interactive user interface to numerical libraries such as lapack mentioned above.

2. Programs written for one system will often require only minor changes to work with another system. All use the same array notation with the dot and colon operators introduced by Matlab.

3. The three have similar numerical capabilities. This is not surprising given that they can be thought of interfaces to standard numerical libraries, but Matlab has superior graphics facilities and Scilab has
the most flexible design. Octave has been designed as a free Matlab “clone”.

9.2 Working with Scilab

Our examples are transcripts of Scilab sessions. The --> is the Scilab prompt at which expressions are entered, and Scilab’s response appears immediately below.

9.2.1 Matrices

One of the great strengths of Scilab is its capabilities in manipulating vectors and matrices. Matrices are the most basic type of data in Scilab, even numbers are considered as 1 x 1 matrices.

Example

The matrix

\[
A = \begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{bmatrix}
\]

is entered in Scilab as follows:

```scilab
-->a = [1 2 3
 -->4 5 6
 -->7 8 9]
a =

! 1.  2.  3. !
! 4.  5.  6. !
! 7.  8.  9. !
```

There are some obvious syntactic differences from Maxima:

- The assignment operator is = in Scilab.
- Statement terminators are not needed in Scilab.

In general matrices are entered between square brackets with spaces separating the elements and newlines separating the rows. Alternatively semicolons can be used to separate rows:

```scilab
-->a = [1 2 3; 4 5 6; 7 8 9]
a =

! 1.  2.  3. !
! 4.  5.  6. !
! 7.  8.  9. !
```
Arithmetic expressions can be entered in matrices:

\[
\begin{array}{c}
\text{\texttt{-->a = [1 3/4}} \\
\text{\texttt{--> (1+2+3) 5^100]}} \\
\text{\texttt{a =}} \\
\end{array}
\]

\[
\begin{array}{ccc}
! & 1. & 0.75 ! \\
! & 6. & 7.889E+69 !
\end{array}
\]

9.2.2 Subscripts

The element in row \(i\) and column \(j\) of the matrix \(a\) is is denoted by \(a(i,j)\).

Example

\[
\begin{array}{cc}
\text{\texttt{-->a = [1 2 3; 4 5 6; 7 8 9]}} \\
\text{\texttt{a =}} \\
\end{array}
\]

\[
\begin{array}{ccc}
! & 1. & 2. 3. ! \\
! & 4. & 5. 6. ! \\
! & 7. & 8. 9. !
\end{array}
\]

\[
\begin{array}{c}
\text{\texttt{-->a(1,3) \hspace{1cm} ans =}} \\
\end{array}
\]

\[
\begin{array}{c}
3. \\
\end{array}
\]

\[
\begin{array}{c}
\text{\texttt{-->a(3,1) \hspace{1cm} ans =}} \\
\end{array}
\]

\[
\begin{array}{c}
7. \\
\end{array}
\]

Subscripts can also be used to change elements of a matrix:

\[
\begin{array}{c}
\text{\texttt{-->a(2,3) = -100}} \\
\text{\texttt{a =}} \\
\end{array}
\]

\[
\begin{array}{ccc}
! & 1. & 2. 3. ! \\
! & 4. & 5. -100. ! \\
! & 7. & 8. 9. !
\end{array}
\]

For vectors, of either row or column type, only a single index is needed.
--> v = [1 2 3 4]
 v =
 1. 2. 3. 4.

--> v(3)
 ans =
 3.

--> v(3) = 30
 v =
 1. 2. 30. 4.

9.2.3 Numbers

Scilab works with floating point numbers, even integers are represented by floating point values. The scale factor e is used in scientific notation, so

\[ 1.234 \times 10^{-45} \]\n
is written \[ 1.234e-45 \]

Within Scilab floating point numbers are represented with about 16 digits, but not all digits are printed by Scilab. You need to keep this in mind when interpreting results from Scilab.

Example

--> .0000000000000087654321
 ans =
 8.765E-15

--> 12345678
 ans =
 12345678.

--> 123456789
 ans =
 1.235E+08

1There is a bug in many versions of Scilab which fails to print the exponent symbol e for 3 digit exponents.

2The way numbers are printed can be controlled by the format command.
-->3 + 1e-6
ans =

3.000001

-->3 + 1e-8
ans =

3.

This last result may be a bit of surprise. The only digits following the
decimal point that would normally be printed are all zeros and they are
dropped from the printed answer, but not the from the answer itself.

-->3 + 1e-8
ans =

3.

-->ans - 3
ans =

1.000E-08

-->3 + 1e-17
ans =

3.

-->ans - 3
ans =

0.

To understand this last result, note that 16 digits are not enough to
distinguish between 3 + 10^{-17} and 3.

9.2.4 Complex Numbers
Complex numbers are written (but not printed) using %i for $\sqrt{-1}$ so

$$2 - 3i$$

is written

$$2 - 3*%i$$
Example

-->3 + %i
ans =

3. + i

--> (3+%i)/(1+2*%i)
ans =

1. - i

--> (-2 + 2*%i)^{(1/3)}
ans =

1. + i

9.2.5 Variables

Names in Scilab are case sensitive, so A and a are different variables. Variables can be assigned values to be used in subsequent calculations but using an unassigned variable in an expression will cause an error. The who command gives a (rather long) list of names which have been assigned values, many of which have been predefined by Scilab. To find the value associated with a variable just type the name of the variable.

Example

--> a = 3
a =

3.

--> a
a =

3.

--> c
!--error 4
undefined variable : c
9.2.6 Constants

Scilab has a few predefined constants. The ones you need to know about for now are:

\begin{align*}
%i & \quad \text{\(\sqrt{-1}\)} \\
\%pi & \quad \pi \\
\%e & \quad \text{base of natural logarithms} \\
\%T & \quad \text{true} \\
\%F & \quad \text{false}
\end{align*}

Example

\begin{verbatim}
-->%pi
%pi =
3.1415927

-->%e^(%i*%pi)
ans =
- 1. + 1.225E-16i
\end{verbatim}

The last result is not exactly \(-1\) because of \textbf{rounding error}, a topic that we will we discuss in a later lecture.

9.2.7 Syntax

Just a few simple but important points:

1. Comments start with a // and are ignored by Scilab.
2. Terminating a line with a semicolon ; stops the result being printed.
3. To type something that is too long to fit on one line, end the line by three full stops \ldots\ and continue on the next line.

Example

\begin{verbatim}
-->// This is a comment

-->a = 23;

-->a
a =

23.
\end{verbatim}
\[ a = 1 + 2 + 3 + 4 + 5 + 6 + 7 \ldots \]
\[\rightarrow 8 + 9 + 10\]
\[ a = 55. \]

9.3 Vectors and Matrices

9.3.1 The Colon Operator

The colon operator \( : \) has a number of uses in constructing and deconstructing vectors and matrices.

Example

The simplest use of the colon operator is to obtain a vector containing all the numbers in some range:

\[\rightarrow n = 1:10\]
\[ n = \]
\[ 1. \ 2. \ 3. \ 4. \ 5. \ 6. \ 7. \ 8. \ 9. \ 10. \]
\[\rightarrow n = 10:-2:0\]
\[ n = \]
\[ 10. \ 8. \ 6. \ 4. \ 2. \ 0. \]

The colon operator is also be used to pick out selected pieces of a matrix. This is important when matrices are used to store data and we want to extract certain parts of the data.

- \( a(i,:) \) is the \( i \)th row of \( a \).
- \( a(:,j) \) is the \( j \)th column of \( a \).
- \( a(:,j:k) \) is the matrix formed from the \( j \)th to \( k \)th columns of \( a \), etc.
Example

```matlab
---> a = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
a =
```

<table>
<thead>
<tr>
<th></th>
<th>16.</th>
<th>3.</th>
<th>2.</th>
<th>13.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.</td>
<td>10.</td>
<td>11.</td>
<td>8.</td>
</tr>
<tr>
<td></td>
<td>9.</td>
<td>6.</td>
<td>7.</td>
<td>12.</td>
</tr>
</tbody>
</table>

```matlab
---> a(2,:) // the second row
ans =
```

|   | 5.  | 10. | 11. | 8.  |

```matlab
---> a(:,3) // the third column
ans =
```

<table>
<thead>
<tr>
<th></th>
<th>2.</th>
<th>!</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11.</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>7.</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>14.</td>
<td>!</td>
</tr>
</tbody>
</table>

```matlab
---> a(:,2:3) // the second to third columns
ans =
```

<table>
<thead>
<tr>
<th></th>
<th>3.</th>
<th>2.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10.</td>
<td>11.</td>
</tr>
<tr>
<td></td>
<td>6.</td>
<td>7.</td>
</tr>
<tr>
<td></td>
<td>15.</td>
<td>14.</td>
</tr>
</tbody>
</table>

```matlab
---> a(1:2,3:4) // the first to second rows and
ans = // third to fourth columns
```

<table>
<thead>
<tr>
<th></th>
<th>2.</th>
<th>13.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11.</td>
<td>8.</td>
</tr>
</tbody>
</table>

9.3.2 Transpose

The quote ' is used to take the transpose of a matrix.
Example

-->a = [1 2 3; 4 5 6; 7 8 9]
  a =

  1.  2.  3.
  4.  5.  6.
  7.  8.  9.

-->a'
  ans =

  1.  4.  7.
  2.  5.  8.
  3.  6.  9.

The distinction between row and column vectors is often important in Scilab and the transpose can be used to convert between one and the other:

-->v = [1 2 3 4]
  v =

  1.  2.  3.  4.

-->w = v'
  w =

  1.
  2.
  3.
  4.

-->x = w'
  x =

  1.  2.  3.  4.

For complex numbers $z'$ is the complex conjugate of $z$ and for complex matrices $a'$ is the conjugate transpose of $a$.

-->z = 3 - 4*%i
  z =

  3. - 4.i
9.3.3 Matrix Arithmetic

The arithmetic operators +, -, *, and ^ work in the expected way with matrices. The division operator / applied to matrices is connected with solving linear equations and will be discussed in detail in a later lecture.

Example

```
-->a = [1 2 3; 4 5 6; 7 8 9]
    a =
          ! 1.  2.  3. !
          ! 4.  5.  6. !
          ! 7.  8.  9. !

-->a+a
    ans =
          ! 2.  4.  6. !
          ! 8. 10. 12. !
          ! 14. 16. 18. !
```
It is important to remember the rules for multiplication of matrices and vectors:

Let \( A \) is \( m \times n \) matrix and \( B \) is a \( p \times q \) matrix. The matrix product \( AB \) is defined if and only if \( n = p \), in which case the result is a \( m \times q \) matrix.

**Example**

```
--> a = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
a =
!
! 16. 3. 2. 13. !
! 5. 10. 11. 8. !
! 9. 6. 7. 12. !
! 4. 15. 14. 1. !

--> v = [1 2 3 4]
v =
!
! 1. 2. 3. 4. !
```
-->\(w = v'\)
\[
w = \\
\quad ! 1. ! \\
\quad ! 2. ! \\
\quad ! 3. ! \\
\quad ! 4. ! \\
\]

--->a*\(w\)
\[
\quad \text{ans} = \\
\quad ! 80. ! \\
\quad ! 90. ! \\
\quad ! 90. ! \\
\quad ! 80. ! \\
\]

--->a*\(v\)
\[
\quad !\text{--error 10} \\
\quad \text{inconsistent multiplication} \\
\]

--->v*\(w\)
\[
\quad \text{ans} = \\
\quad 30. \\
\]

--->\(w*v\)
\[
\quad \text{ans} = \\
\quad ! 1. 2. 3. 4. ! \\
\quad ! 2. 4. 6. 8. ! \\
\quad ! 3. 6. 9. 12. ! \\
\quad ! 4. 8. 12. 16. ! \\
\]

--->v*\(a\)
\[
\quad \text{ans} = \\
\quad ! 69. 101. 101. 69. ! \\
\]

--->w*\(a\)
\[
\quad !\text{--error 10} \\
\quad \text{inconsistent multiplication} \\
\]
9.3.4 The Dot Operator

The dot operator . is used in conjunction with the operators *, / and ^ to perform element by element operations on vectors and matrices. Whenever we use a matrix to represent data, as opposed to an algebraic matrix, we almost always want to use the dot operators .*, ./ and .^ rather than their algebraic counterparts *, / and ^. Of all the features of Scilab (and Matlab and Octave) this is one that causes the most trouble.

Example

```
-->v = [1 2 3 4]
   v =
   ! 1.  2.  3.  4. !

-->v.*v
   ans =
   ! 1.  4.  9.  16. !

-->v.^3
   ans =
   ! 1.  8.  27.  64. !

-->a = [1 2 3; 4 5 6; 7 8 9]
   a =
   ! 1.  2.  3. !
   ! 4.  5.  6. !
   ! 7.  8.  9. !

-->a.*a
   ans =
   ! 1.  4.  9. !
   ! 16.  25.  36. !
   ! 49.  64.  81. !
```
---> a.^3
ans =

! 1.  8.  27.  !
! 64. 125. 216. !
! 343. 512. 729. !

---> a./a
ans =

! 1.  1.  1.  !
! 1.  1.  1.  !
! 1.  1.  1.  !

Note in particular:

1. a*b is the matrix product of a with b; a.*b is the matrix whose components are formed by multiplying corresponding components of a and b.

2. a^3 is the cube of matrix a, that is a*a*a; a.^3 is the matrix whose components are the cubes of the components of a.

3. a./b divides each element of a by the corresponding element of b.

A Tricky Point

As compared with many other systems, Scilab’s notation is usually quite clear and consistent. However Scilab can can confuse a decimal point and a dot operator. This occurs because something like 10. is a valid way to write a number. You might have noticed Scilab sometimes writes numbers like this. This has to do with the distinction in some programming languages between the floating point number 10. and the integer 10, a distinction which is irrelevant in Scilab since it treats all numbers as floating point numbers.

Here is an example; suppose we want to divide 10 by each of the numbers 1, ..., 10. A natural way to do this in Scilab is to define

---> d = [1 2 3 4 5 6 7 8 9 10]
   d =

! 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  !

and then use the ./ operator:
Of course this is not what we wanted. What has happened is that the dot in 10./d has been taken as a decimal point rather than as a dot operator. Once we have realised this we can get the desired result using spaces:

```matlab
disp --> 10 ./ d
disp ans =
```

```
column 1 to 7
! 10. 5. 3.3333333 2.5 2. 1.6666667 1.4285714!
column 8 to 10
! 1.25 1.1111111 1. !
```

### 9.3.5 Special Matrices

The following functions are used to construct simple matrices.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zeros</td>
<td>matrix of zeros</td>
</tr>
<tr>
<td>ones</td>
<td>matrix of ones</td>
</tr>
<tr>
<td>eye</td>
<td>identity matrix</td>
</tr>
<tr>
<td>rand</td>
<td>random matrix</td>
</tr>
</tbody>
</table>

These can be used in two ways; for example

1. If \( m \) and \( n \) are numbers, \( \text{zeros}(m,n) \) is a \( m \times n \) matrix of zeros.

2. If \( a \) is a matrix, \( \text{zeros}(a) \) is a matrix of zeros the same size as \( a \).
Examples

```matlab
---> a = rand(3, 5)
a =

! 0.7783129 0.6856896 0.8415518 0.8784126 0.5618661 !
! 0.2119030 0.1531217 0.4062025 0.1138360 0.5896177 !
! 0.1121355 0.6970851 0.4094825 0.1998338 0.6853980 !

---> b = ones(a)
b =

! 1. 1. 1. 1. 1. !
! 1. 1. 1. 1. 1. !
! 1. 1. 1. 1. 1. !

---> c = eye(a)
c =

! 1. 0. 0. 0. 0. !
! 0. 1. 0. 0. 0. !
! 0. 0. 1. 0. 0. !
```

By default `rand` produces random numbers uniformly distributed between 0 and 1. Normally distributed random numbers with mean 0 and variance 1 can be produced as follows:

```matlab
---> rand(3, 5, 'normal')
ans =

! -0.7616491 1.1443051 0.7223316 -0.8498895 -0.6834217 !
! 0.6755537 0.8529775 1.9273333 0.2546697 -0.7209534 !
! 1.4739762 0.4529708 0.6380837 1.5417209 0.8145126 !
```

9.3.6 Functions

Elementary Transcendental Functions

The following functions act element by element on vectors and matrices.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sqrt</code></td>
<td>square root</td>
</tr>
<tr>
<td><code>exp</code></td>
<td>exponential</td>
</tr>
<tr>
<td><code>log</code></td>
<td>logarithm to base e</td>
</tr>
<tr>
<td><code>log10</code></td>
<td>logarithm to base 10</td>
</tr>
<tr>
<td><code>log2</code></td>
<td>logarithm to base 2</td>
</tr>
</tbody>
</table>
### Example

```plaintext
--> a = [1 2 3; 4 5 6; 7 8 9]
   a =

   ! 1.  2.  3. !
   ! 4.  5.  6. !
   ! 7.  8.  9. !

--> sqrt(a)
   ans =

   ! 1.  1.4142136  1.7320508 !
   ! 2.  2.236068   2.4494897 !
   ! 2.6457513  2.8284271   3. !

--> gamma(a)
   ans =

   ! 1.  1.  2. !
   ! 6.  24. 120. !
   ! 720. 5040. 40320. !
```
Elementary Numerical Functions

The following functions also act element by element on matrices:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>absolute value</td>
</tr>
<tr>
<td>sign</td>
<td>sign</td>
</tr>
<tr>
<td>round</td>
<td>round to nearest</td>
</tr>
<tr>
<td>ceil</td>
<td>round up</td>
</tr>
<tr>
<td>floor</td>
<td>round down</td>
</tr>
<tr>
<td>fix</td>
<td>round towards zero</td>
</tr>
<tr>
<td>int</td>
<td>integer part</td>
</tr>
<tr>
<td>real</td>
<td>real part</td>
</tr>
<tr>
<td>imag</td>
<td>imaginary part</td>
</tr>
</tbody>
</table>

Example

```plaintext
---> a = [-1.6 3.4]
a =
! - 1.6   3.4 !

---> round(a)
ans =
! - 2.   3. !

---> ceil(a)
ans =
! - 1.   4. !
```

9.3.7 Concatenation

In expressions such as

```plaintext
a = [1 2; 3 4]
```

the numbers can be replaced by matrices allowing a matrix to be built up from sub-matrices.

Examples

The following is an example of a common and useful construction especially in data analysis:
---> x = (0:0.5:3)'
x =

! 0. !
! 0.5 !
! 1. !
! 1.5 !
! 2. !
! 2.5 !
! 3. !

---> y = [sin(x) cos(x) tan(x)]
y =

! 0. 1. 0. !
! 0.4794255 0.8775826 0.5463025 !
! 0.8414710 0.5403023 1.5574077 !
! 0.9974950 0.0707372 14.10142 !
! 0.9092974 - 0.4161468 - 2.1850399 !
! 0.5984721 - 0.8011436 - 0.7470223 !
! 0.1411200 - 0.9899925 - 0.1425465 !

In this example x is a column vector of length 7. Then sin(x), cos(x) and tan(x) are column vectors of the same size. The matrix resulting from concatenating the three column vectors is then a 7 x 3 matrix whose columns are vectors sin(x), cos(x) and tan(x).

Our next example builds a matrix in a more complicated way:

---> a = [1 2 3 4]
a =

! 1. 2. 3. 4. !

---> b = [10 20 30; 40 50 60]
b =

! 10. 20. 30. !
! 40. 50. 60. !

---> c = [100; 200]
c =

! 100. !
! 200. !
---> \[d = [a; b c]

\[
d = \\
1. 2. 3. 4. \\
10. 20. 30. 100. \\
40. 50. 60. 200. \\
\]

Here is how the parts fit together:

\[
\begin{array}{cccc}
1. & 2. & 3. & 4. \\
\hline
10. & 20. & 30. & 100. \\
40. & 50. & 60. & 200. \\
\end{array}
\]