Numerical Analysis
Exercises

Christian Rossi

Academic Year 2023-2024

Abstract

The topics of the course are:

- Floating-point arithmetic: different sources of the computational error; absolute vs relative errors; the floating point representation of real numbers; the round-off unit; the machine epsilon; floating-point operations; over- and under-flow; numerical cancellation.

- Numerical approximation of nonlinear equations: the bisection and the Newton methods; the fixed-point iteration; convergence analysis (global and local results); order of convergence; stopping criteria and corresponding reliability; generalization to the system of nonlinear equations (hints).

- Numerical approximation of systems of linear equations: direct methods (Gaussian elimination method; LU and Cholesky factorizations; pivoting; sparse systems: Thomas algorithm for tridiagonal systems); iterative methods (the stationary and the dynamic Richardson scheme; Jacobi, Gauss-Seidel, gradient, conjugate gradient methods (hints); choice of the preconditioner; stopping criteria and corresponding reliability); accuracy and stability of the approximation; the condition number of a matrix; over- and under-determined systems: the singular value decomposition (hints).

- Numerical approximation of functions and data: Polynomial interpolation (Lagrange form); piecewise interpolation; cubic interpolating splines; least-squares approximation of clouds of data.

- Numerical approximation of derivatives: finite difference schemes of the first and second order; the undetermined coefficient method.

- Numerical approximation of definite integrals: simple and composite formulas; midpoint, trapezoidal, Cavalieri-Simpson quadrature rules; Gaussian formulas; degree of exactness and order of accuracy of a quadrature rule.

- Numerical approximation of ODEs: the Cauchy problem; one-step methods (forward and backward Euler and Crank-Nicolson schemes); consistency, stability, and convergence (hints).

Contents

Chapter 1

Introduction to MATLAB

1.1 Main MATLAB operators

Assignment operator:

```
% Print output
a = 1
% Does not print output
b = 2;
```

The active variables can be found in the workspace and the value can be checked on the command window with:

```
% Value of all variables
whos
% Value of a
whos a
```

If you want to save the file:

```
% Save the command history
diary file_name.txt
% Save the whole workspace
save file_name
% Save only the variable a
save file_name_only_a a
% Load only the variable a
load file_name_only_a
% Load the whole workspace
load file_name
```

It is possible to clear variables with the following commands:

```
% Clear only the variable a
clear a
% Clear the whole workspace
clear all
```

1.2 Vector and matrices

Most of the entities in MATLAB are matrices, even real numbers. The matrices can be defined in the following ways:

```
% Row vector definition
c = [1 2 3]
% Column vector definition
c = [1; 2; 3]
% Vectorn transposition
c = [1 2 3]'
% 2D matrix definition
D = [ 1 2 3;
        4 5 6;
        7 8 9 ]
```

It is also possible to define various types of matrices:

```
% Zeros vector/matrix
A = zeros(row_length,column_length)
% Ones vector/matrix
A = ones(row_length,column_length)
% Identity matrix
A = eye(row_length,column_length)
% Diagonal matrix
d = [1:4]
D = diag(d)
% Set a not principal diagonal
D = diag(d, diagonal_index)
% Select only upper o lower trinagular
Ml = tril(M)
Mu = triu(M)
% Access an element in vector
C(1)
% Access an element in matrix
```

```
C([2,3]);
% Access a part of the matrix
Q(rows,columns)
% Access the element in position (n,m)
Q(end, end)
% Dimension of a matrix
length(a);
numel(b);
size(a);
```

The operations on vectors are done in the following way:

```
% Given two row vectors a and b
% Vector sum
a + b
% Vector difference
a - b
% Scalar product
a * b'
dot(a,b)
% Tensor product
a' * b
% Elementwise product
a .* b
% Elementwise division
a ./ b
% Elementwise exponentiation
a .^ 2
```

The operations on matrices are done in the following way:

```
% Givcen two matrices A and B (both 3x2)
% Matrix sum
A + B
% Matrix difference
A - B
% Matrix product
K * L'
% Elementwise product
A .* B
% Elementwise division
A ./ B
% Elementwise exponentiation
```

```
A .^ 2
% Power matrix (useful only square)
A ^ 2
% Other useful values of the matrices
% Determinant
det(A)
% Trace
trace(A)
% Inverse of small matrix
inv(A)
% Given a column vector b ths olutio of Ax=b
A \ b
```

The function used to plot a graph are the following:

```
% To plot y=f(x) in [a,b]
x = a:step_length:b;
y = f(x);
figure
plot(x,y,color)
% To add y2=f2(x) in [c,d]
hold on
x2 = c:step_length:d;
y2 = f2(x);
plot(x2,y2,color)
% Show graph's grid
grid on
% Set the axis limit
axis([xmin xmax ymin ymax])
% Set the same scaling for both axis
axis equal
```

To handle functions the commands are:

```
% Define a function handle to g(x)
f = @g(x);
% Evaluation of f in a
f(a)
% Define an anonymous function
% It is useful to modify other functions
f = @(argument-list) expression
```

The operators that u logical values are:

```
% Smaller than
a < b
% Greater than
a > b
% Smaller or equal than
a <= b
% Equal to
a == b
% Different from
a ~= b
% And
(a < b) & (b > c)
% Or
(a < b) | (b > c)
```

The control-flow statement are:

```
% if-then-else statements
if (condition1)
    block1
elseif (condition2)
    block2
else
    block3
end
% for loops
for (index=start:step:end)
    instruction block
end
% while loops
while (condition)
    instruction block
end
```

There are two categories of m-files:

- Scripts: these files contain instructions that are executed in sequence in the command line if the script file is called. The variables are saved in the current workspace.
- Functions: they take some input arguments and return some outputs after a series of instructions are performed. The variables defined in the function are local to the scope of the function itself.

Chapter 2

Laboratory I

Exercise 1

Define the row vector:

$$\bar{v_k} = [1, 9, 25, \ldots, (2k+1)^2] \in \mathbb{R}$$

with $k = 8$ using the following strategies:

1. A for loop to define one by one each element of the vector.
2. The vector syntax to build it in just one shot.

Answer of exercise 1

```
% Point one
k = 8;
A = 0:k;
for i=1:1:(k + 1)
    A(i) = (2 * (i - 1) + 1).^2;
end

% Point two
k = 8;
A = 0:k;
B = 2 * A + 1;
C = B.^2;
```

Exercise 2

Define a function which, for an input value k, returns the corresponding vector v_k as defined in the previous exercise.

Answer of exercise 2

```
function A = vector(k)
    A = 0:k;
    B = 2 * A + 1;
    A = B.^2;
end
```

Exercise 3

Using the function of the previous exercise write another function that returns, for a generic value k, the $2(k+1) \times 2(k+1)$ matrix.

$$m_k = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sqrt[2]{2} & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \sqrt[3]{2} & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \sqrt[4]{2} & 0 & 0 & \cdots & 0 & 9 \\ 0 & 0 & 0 & 0 & \sqrt[5]{2} & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt[6]{2} & \cdots & 0 & 25 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \sqrt[(2k+1)]{2} & 0 \\ 1 & 1 & 9 & 9 & 25 & 25 & \cdots & (2k+1)^2 & (2k+1)^2 \end{bmatrix}$$

Answer of exercise 3

```
A = zeros(k);
B = vector(k);
b = 1;
for i=1:1:(k+1)
    A(i,i) = nthroot(2,i);
    a = mod(i,2);
    % Odd
    if a == 1
        A(k + 1,i) = B(b);
    end
    % Even
    if a == 0
        A(k + 1,i) = B(b);
        A(i, k + 1) = B(b);
        b = b + 1;
    end
end
```

Exercise 4

Compare the results of the following code segments:

```
% Code A
x = 0;
while (x ~= 1)
    x = x + 1/16
end

% Code B
x = 0;
while (x ~= 1)
    x = x + 0.1
end
```

Answer of exercise 4

Code A work as expected: the while loop is repeated 16 times, and the final value of x is 1. Instead, code B does not work as expected, and results in an infinite loop.

Exercise 5

Find the machine epsilon by implementing an ad hoc procedure. Comment and justify the obtained results.

Answer of exercise 5

```
macheps = 1;
while 1.0 + (macheps/2) > 1.0
    macheps = macheps / 2;
end
```

The machine epsilon is the nearest number to one. With this algorithm the *macheps* is the same found by the command *eps*.

Exercise 6

Given the following code:

```
realmax
a = 1.0e+308;
b = -a;
c = 1.1e+308;
(a + b) + c
(a + c) + b
```

Explain why the second result is Inf.

Answer of exercise 6

The result is Inf due to the error between the real numbers and floating points numbers.

Exercise 7

Consider the following function:

$$f(x) = \frac{e^x - 1}{x}$$

1. Evaluate $f(x)$ for values of x around zero (try with $x_k = 10^{-k}$, $k \in [1, 20]$). What do you obtain? Explain the results.
2. Propose an approach for fixing the problem. (Hint: Use Taylor expansions to get an approximation of $f(x)$ around $x = 0$).
3. How many terms in the Taylor expansion are needed to get double precision accuracy (16 decimal digits) $\forall x \in \left[0, \frac{1}{2}\right]$?

Answer of exercise 7

1. The function in 0 return a value of NaN because it results in an undetermined fraction $(\frac{0}{0})$. With $k = 1$ we obtain the value 1.0517, and with $k = 20$ we obtain the value 0. This means that the function tends to 0 with x small, but the function is not defined.
2. TODO
3. TODO

Exercise 8

The sequence:

$$1, \frac{1}{3}, \frac{1}{9}, \ldots, \frac{1}{3^n}, \ldots$$

can be generated with the following recursive relations:

$$\begin{cases} p_n = \frac{10}{3} p_{n-1} = p_{n-2} \\ p_1 = \frac{1}{3},\ p_0 = 1 \end{cases}$$

$$\begin{cases} q_n = \frac{1}{3} q_{n-1} \\ q_0 = 1 \end{cases}$$

1. Implement the two relations in order to generate the first 100 terms of the sequence.
2. Study the stability of the two algorithms and justify the obtained results.

Answer of exercise 8

Exercise 9

Find the minimum positive number representable in MATLAB/Octave by implementing an ad hoc procedure. Compare with *realmin*.

Answer of exercise 9

Exercise 10

1. Use Taylor polynomial approximation to avoid the loss of significance errors in the following function when x approaches 0:

$$f(x) = \frac{1 - \cos(x)}{x^2}$$

2. Reformulate the following function $g(x)$ to avoid the loss of significance error in its evaluation for increasing values of x towards $+\infty$:

$$g(x) = x\left(\sqrt{x+1} - \sqrt{x}\right)$$

Answer of exercise 10

Exercise 11

We can compute e^{-x} around $x = 0$ using Taylor polynomials in two ways, either using:

$$e^{-x} \approx 1 - x - \frac{1}{2}x^2 + \frac{1}{6}x^3 + \ldots$$

or using

$$e^{-x} = \frac{1}{e^x} \approx \frac{1}{1 - x - \frac{1}{2}x^2 + \frac{1}{6}x^3 + \ldots}$$

Which approach is the most accurate?

Answer of exercise 11

Exercise 12

Consider the following integral:

$$I_n(\alpha) = \int_0^1 \frac{x^n}{x+\alpha}\, dx \qquad \forall n \in \mathbb{N}, \alpha > 0$$

1. Give an upper bound for $I_n(\alpha)$, $\forall n \in \mathbb{N}, \alpha > 0$.
2. Prove the following recursive relation between $In(\alpha)$ and $I_{n-1}(\alpha)$:

$$\begin{cases} I_n(\alpha) = -\alpha I_{n-1}(\alpha) + \dfrac{1}{n} \\ I_o(\alpha) = \ln\left(\dfrac{\alpha+1}{\alpha}\right) \end{cases}$$

3. Employing the previous relation, compute $I_40(\alpha = 8)$ and comment the obtained results.
4. Write a numerically stable recursive relation for $I_40(\alpha = 8)$.

Answer of exercise 12

Exercise 13

Given the following sequence:

$$\begin{cases} x_{n+1} = 2^{n+1}\left[\sqrt{1+\dfrac{x_n}{2^n}}-1\right] \\ x_0 > -1 \end{cases}$$

for which $\lim_{n\to+\infty} x_n = \ln(1+x_0)$

1. Set $x_0 = 1$, compute $x_1, x_2, \ldots, x_{71}$ and explain the obtained results.
2. Transform the sequence in an equivalent one that converges to the theoretical limit.

Answer of exercise 13