Numerical Analysis and Computational Mathematics

Fall Semester 2021 – CSE Section

Dr. Rafael Vázquez Hernández

Assistant: Davide Pradovera

Session 1 – September 22, 2021

Solutions – Introduction to Matlab® /Octave

Solution I (MATLAB)

We consider the following MATLAB code:

```
c = @( a, b, alpha ) sqrt( a^2 + b^2 - 2 * a * b * cos( alpha ) );
c( 1, 1, pi/3 )
% ans = 1.000
c( 3, 4, pi/2 )
% ans = 5
```

Solution II (MATLAB)

a) We consider the following MATLAB script `logplot.m`:

```
x = linspace( 0, 1000, 100 );
% Alternatively:
% h = ( 1000 - 0 ) / ( 100 - 1 ); x = 0 : h : 1000;
f = @( x )( ( x - log( x + 1 ) ).^4 );
fx = f( x );
figure( 1 );
plot( x, fx );
figure( 2 );
semilogx( x, fx );
figure( 3 );
semilogy( x, fx );
figure( 4 );
loglog( x, fx );
```

b) For the function $f(x) = [x - \log(x+1)]^4$, we have that:

$$\log(f(x)) = 4\log(x - \log(x+1)) \quad \begin{cases} \leq 4\log x \\ \geq 4\log(x-7)\,, \end{cases} \qquad \text{for } x \in [0, 1000],$$

so that the curve $(x, f(X))$ is close to a straight line in the plot with the logarithmic scale on both the axes; its slope is approximately $p = 4$, being $f(x) = \mathcal{O}(x^p)$. Therefore, the plot obtained with the command `loglog` is the most useful to estimate the order of growth of $f(x)$.

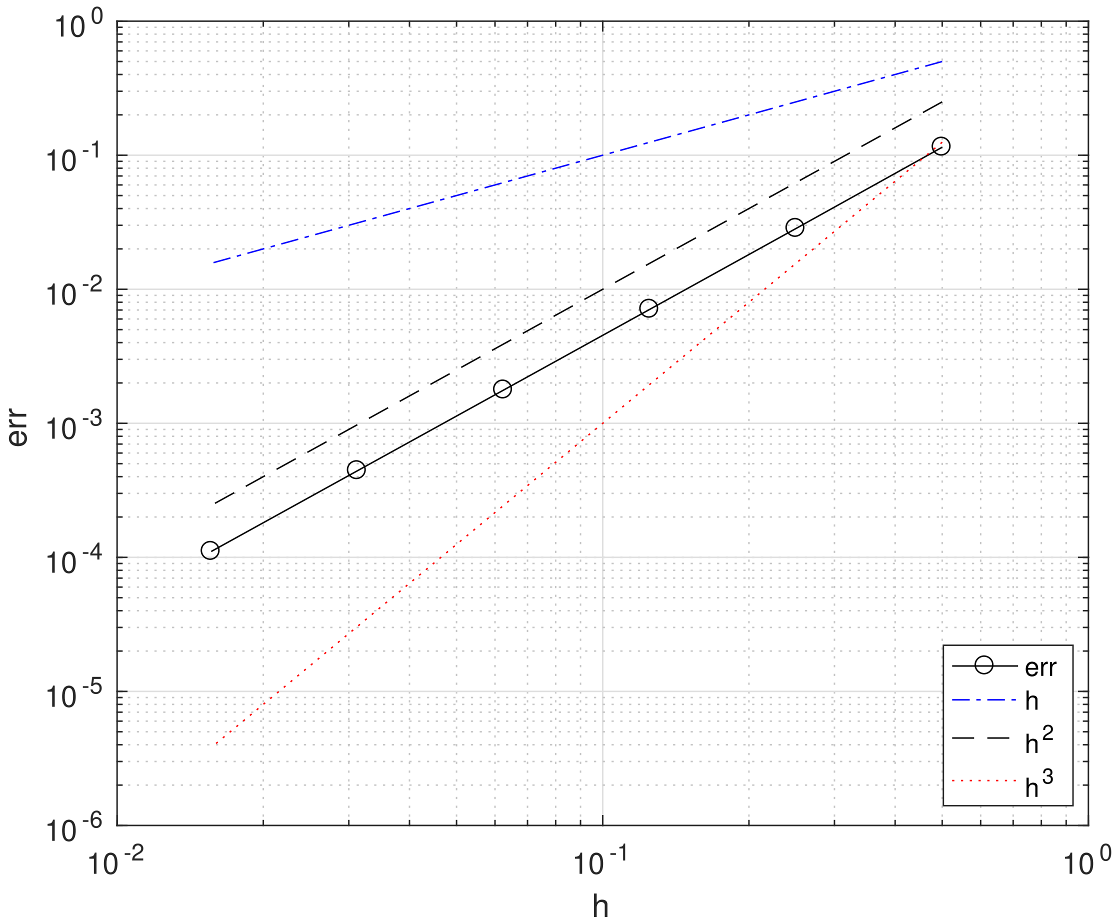
c) We add the following MATLAB commands to the code suggested in a) to obtain the indicated figure:

```
loglog(x,f(x));
axis([10 1000 100 1e12])
grid on
xlabel('x')
ylabel('f(x) = (x - log(x+1))^4') % MATLAB has a LaTeX interpreter
yticks([1e5 1e10])
title('An example of a function that grows very fast:')
```

Solution III (MATLAB)

a) Since we conjecture that the error reads $E_{c,h_i} = C h_i^p$, then we have that $\log E_{c,h_i} = \log C + p \log h_i$, for $h = 1, \ldots, n$. Therefore, when using a logarithmic scale on both axes (*log-log scale*) for plotting the errors E_{c,h_i} vs. h_i, we should obtain a straight line $\log E_{c,h_i}$ of slope p, where p is the order of convergence. In order to graphically determine p, we plot the errors E_{c,h_i} vs. h_i in log-log scale and compare the line obtained with those corresponding to the lines (h_i, h_i^q) for some values of q, e.g. $q = 1, 2, 3$. The value of q for which (h_i, h_i^q) is *parallel* in log-log scale to (h_i, E_{c,h_i}) corresponds to the convergence order of the method p. We use the following MATLAB commands to *graphically* determine p:

```
h = 2.^[ -1 : -1 : -6 ];
err = [ 1.147e-1, 2.840e-2, 7.084e-3, 1.770e-3, 4.425e-4, 1.106e-4 ];
loglog( h, err, '-ok', h, h, '-.b', h, h.^2, '--k', h, h.^3, ':r' );
xlabel( 'h' ); ylabel( 'err' ); grid on
legend( 'err', 'h', 'h^2', 'h^3', 'Location', 'SouthEast' );
```



We deduce that the convergence order of the method is $p = 2$ since (h_i, E_{c,h_i}) is *parallel* to (h_i, h_i^2) in the log-log scale plot.

We remark that in practical cases the conjecture $E_{c,h_i} = Ch_i^p$ can be typically used only for "small" values of $h_i (h_i \to 0)$, for which it suffices to verify that the lines (h_i, E_{c,h_i}) and (h_i, h_i^p) in log-log scale are parallel in the left part of the previous plot.

b) Still using the conjecture $E_{c,h_i} = Ch_i^p$, let us consider the errors $E_{c,h_{i-1}} = Ch_{i-1}^p$ and $E_{c,h_i} = Ch_i^p$ corresponding to $h = h_{i-1}$ and h_i for $i = 2, \ldots, n$, respectively. Then, we have:

$$\frac{E_{c,h_i}}{E_{c,h_{i-1}}} = \left(\frac{h_i}{h_{i-1}}\right)^p, \qquad p = \frac{\log\left(\frac{E_{c,h_i}}{E_{c,h_{i-1}}}\right)}{\log\left(\frac{h_i}{h_{i-1}}\right)} \qquad \text{for } i = 2, \ldots, n.$$

Since in practical cases the conjecture $E_{c,h_i} = Ch_i^p$ holds for "small" values of h_i, we typically select $i = n$ to determine the convergence order of the method p:

$$p = \frac{\log\left(\frac{E_{c,h_n}}{E_{c,h_{n-1}}}\right)}{\log\left(\frac{h_n}{h_{n-1}}\right)}.$$

We use the following MATLAB command to *algebraically* determine that $p = 2$:

```
p = log( err( end ) / err( end - 1 ) ) / log( h( end ) / h( end - 1 ) )
% p = 2.0003
```

Solution IV (MATLAB)

We use the MATLAB commands:

```
M = [ 7 : 10; 12 : -2 : 6 ]
% M =
%
%      7     8     9    10
%     12    10     8     6
```

a) We extract the element in position (1,3) (first row, third column) as:

```
M(1,3)
% ans =
%
%      9
```

b) We extract the second row as:

```
M(2,:)
% ans =
%
%     12    10     8     6
```

c) We extract the first two columns from the matrix as:

```
M(:,1:2)
% ans =
%
%      7     8
%     12    10
```

d) We extract the vector containing all the elements of the second row of the matrix except for the third element as:

```
M(2,[1 2 4])
% ans =
%
%     12    10     6
```

Exercise V (MATLAB)

We use the following MATLAB code:

```
f1 = @(x) ( sqrt( 1 + x ) - 1 ) ./ x;
f2 = @(x) 1 ./ ( sqrt( 1 + x ) + 1 );
f3 = @(x) 0.5 - x / 8 + x.^2 / 16 - 5 / 128 * x.^3;

values_f1 = []; values_f2 = []; values_f3 = [];

for k = 10 : 2 : 16
    values_f1 = [ values_f1, f1( 10^( -k ) ) ];
    values_f2 = [ values_f2, f2( 10^( -k ) ) ];
    values_f3 = [ values_f3, f3( 10^( -k ) ) ];
end

values_f1, values_f2, values_f3

X = 10.^-( 10 : 2 : 16 );

f1(X), f2(X), f3(X)

% ans =
%
%     0.5000    0.5000    0.4885         0
%
%
% ans =
%
%     0.5000    0.5000    0.5000    0.5000
%
%
% ans =
%
%     0.5000    0.5000    0.5000    0.5000
```

We can clearly see that the expression $f(x) = (\sqrt{1+x} - 1)/x$ suffers from severe round-off errors as x approaches machine precision, because we compute the difference of two numbers with similar values, $\sqrt{1+x}$ and x, which causes a cancellation of significant digits (*hint*: type `eps` in the console to visualize the best relative precision that you can get using MATLAB).