Numerical Analysis (089180) A.Y. 2022-2023
prof. Simona Perotto Luca Liverotti
simona.perotto@polimi.it luca.liverotti@polimi.it

Lab 1 – Homework Solutions

September 16, 2022

A Homework

Homework 1.1. Find the minimum positive number representable in MATLAB/Octave by implementing an ad hoc procedure. Compare with `realmin`.

Solution Homework 1.1.

```
k = 0;
zero = 1/2;
while (0 + zero) > 0
  zero_old = zero;
  zero = zero / 2;
  k = k + 1;
end

% Remember: realmin is the minimum !normalized! positive floating point number
realmin
% The minimum positive floating point number is instead a !denormalized! number
zero_old % < realmin
k
```

Homework 1.2. a. Use Taylor polynomial approximation to avoid the loss of significance errors in the following function when x approaches 0

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

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

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

Solution Homework 1.2.

```
format long


%  Use Taylor polynomial approximation to avoid the loss of significance
%  errors in the following function when x approaches 0
%
%    f(x) = {1-cos(x)}/{x^2}
%
%  The limit of f(x) = {1-cos(x)}/{x^2} as x -> 0 is well known:
%
%  lim {x -> 0} {1-cos(x)}/{x^2} = {1}/{2}


clear all, close all, clc
k = [1:30]';
```

```
x = 2.^(-k);
f = @(x) (1 - cos(x))./(x.^2);
[k x f(x)]


%  If we try to evaluate f(x) directly with a sequence xk = 2^{-k} that approaches 0, we
notice anomalous behaviour at k = 13, k = 27, k = ...
%  The reason is numerical cancellation of the terms on the numerator. We can use Taylor
expansion of cos(x) for x -> 0 to obtain a better approximation. Indeed
%
%  cos(x) = 1 - {1}/{2} x^2 + {1}/{24} x^4 - {1}/{720} x^6 + o(x^8)
%
%  so that
%
%  f(x) = {1}/{2} - {1}/{24} x^2 + {1}/{720} x^4 + o(x^6)


f_taylor_4 = @(x) 1/2 - x.^2/24 + x.^4/720;

[k x f(x) f_taylor_4(x)]

%  The obtained formula is more stable that the direct evaluation of f(x), and the computed
values approach {1}/{2}.


%  Reformulate the following function g(x) to avoid the loss of
%  significance error in its evaluation for increasing values of x towards +infinity.
%
%  g(x) = x (sqrt{x+1} - sqrt{x}).
%
%  It is well known that
%
%  lim{x -> +infinity} g(x) = +infinity


clear all, close all, clc
format short e

k = [1:20]';
x = 10.^(k);

g = @(x) x.*(sqrt(x+1) - sqrt(x));

[k x g(x)]


%  If we try to evaluate g(x) for a sequence xk = 10^{k} that approaches 0, we notice
anomalous behaviour at k = 16, k = ..., for which the computed value is 0. This happens since,
 in floating point representation, x16 + 1 = x16!
%
%  Rationalization of the expression in bracket solves this numerical cancellation: indeed,
multiplying the numerator and the denominator by sqrt{x+1} + sqrt{x}
%
%  g(x) = {x (sqrt{x+1} - sqrt{x})(sqrt{x+1} + sqrt{x})}/{(sqrt{x+1} + {x})} = {x}/{(sqrt{x+1}
 + sqrt{x})}.



g2 = @(x) x./(sqrt(x+1) + sqrt(x));

[k x g(x) g2(x)]

%  This formula is more stable and does not suffer of numerical cancellation.
```

Homework 1.3. We can compute e^{-x} around $x = 0$ using Taylor polynomials in two ways, either using

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

or using

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

Which approach is the most accurate?

Solution Homework 1.3.

```
clear all, close all, clc
format long

f_taylor_pos = @(x) 1 - x + 1/2*x.^2 - 1/6*x.^3 + 1/24*x.^4 - 1/120*x.^5;
f_taylor_neg = @(x) 1./(1 + x + 1/2*x.^2 + 1/6*x.^3 + 1/24*x.^4 + 1/120*x.^5);

k = [1:20]';

x_pos = 10.^(-k);
[x_pos f_taylor_pos(x_pos) f_taylor_neg(x_pos)]

x_neg = -10.^(-k);
[x_neg f_taylor_pos(x_neg) f_taylor_neg(x_neg)]


%  No bad behaviour is observed since the 1 in both formulas dominates the sum and numerical
cancellation does not occur.
%  Therefore both expression can be accepted.
%
%  Instead, if the sum were not dominated by the term 1 (or if you didn't notice that),
%  the following combination could have been proposed
%
%  e^{-x} ~ {1}/{1 + x + {1}/{2}x^2 + {1}/{6}x^3 + ...}, x > 0;
%        ~ 1 - x + {1}/{2}x^2 - {1}/{6}x^3 + ... , x <=0.
%
%  The choice of the previous expression for x > 0 should not display any numerical
cancellation because all the terms in the sum are positive. Similarly, the choice for x < 0
grants that the terms -x^{2k+1} are positive, again avoiding any numerical cancellation.
```

Homework 1.4. Consider the following integral

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

a. Give an upper bound for $I_n(\alpha)$, $\forall n \in \mathbb{N}, \alpha > 0$.

b. Prove the following recursive relation between $I_n(\alpha)$ and $I_{n-1}(\alpha)$:

$$\begin{cases} I_n(\alpha) = -\alpha I_{n-1}(\alpha) + \frac{1}{n} \\ I_0(\alpha) = \ln\left(\frac{\alpha+1}{\alpha}\right) \end{cases}$$

c. Employing the previous relation, compute $I_{40}(\alpha = 8)$ and comment the obtained results.

d. Write a numerically stable recursive relation for $I_{40}(\alpha = 8)$.

Solution Homework 1.4.

```
clear all;
format short e

n = 40;

% alpha = 1/8 not requested in the homework
for (alpha = [1/8, 8])
   I(1) = log((alpha+1)/alpha);
   for (k = 1:n)
      I(k+1) = -alpha*I(k) + 1/k;
```

```
   end
   recursion_integral = I(n+1);
   upper_bound = (1/alpha)*(1/(n+1));
   exact_integral = quadl(@(x) (x.^n)./(x+alpha), 0, 1, 1e-16); % Exact value (not requested)
   [recursion_integral, upper_bound, exact_integral]
end


% The recursive approximation of I{40}(alpha = 8) is 1.6389 ... 10^{18}: this result is for
sure incorrect because it violates the upper bound for I40(α = 8).
%
% This is due to the finite precision we use and the error propagation: the initial value I0(
alpha) = ln({alpha+1}/{alpha}) is represented with a certain error eps; denoting by fl(y) the
floating point representation of the number y, we have
%
% fl(I0(alpha)) = I0(alpha) + eps,
% fl(I1(alpha)) = -alpha fl(I0(alpha)) + 1 = -alpha (I0(alpha) + eps) + 1 = I1(alpha) - alpha
 eps,
% ...
% fl(Ik(alpha)) = In(alpha) + (-1)^k alpha^k eps,
%
% Therefore, at the final (n-th) step the error is multiplied by a factor alpha^n, which
result in an amplification of the error if alpha > 1. Instead, for alpha < 1, the error is
damped and the recursive relation is stable (e.g. see the previous test for alpha = 1/8).
%
% Now write a numerically stable recursive relation for I{40}(alpha = 8).
%
% The idea is to transform the factor alpha on the RHS of the recursive relation in a factor
{1}/{alpha}; this can be achieved inverting the recursive relation:
%
%    I{k-1}(alpha) = - {1}/{alpha} I{k}(alpha) + {1}/{k alpha}
%    lim{k -> +infinity} I{k}(alpha) = 0
%
% The final value lim{k -> +infinity} I{k}(alpha) is equal to zero because 0 <= I{k}(alpha)
<= {1}/{alpha} {1}/{k+1} -> 0, as k -> + infinity.


clear all;
n = 40;
big = 1000;

% alpha = 1/8 not requested in the homework
for (alpha = [1/8, 8])
   I(big + 1) = 0;
   for (k = big:-1:n+1)
      I(k) = -1/alpha*I(k+1) + 1/(k*alpha);
   end
   recursion_integral = I(n+1);
   upper_bound = (1/alpha)*(1/(n+1));
   exact_integral = quadl(@(x) (x.^n)./(x+alpha), 0, 1, 1e-16); % Exact value (not requested)
   [recursion_integral, upper_bound, exact_integral]
end

% The recursion is now stable for alpha > 1 (and note that, now, an additional error is
present, because the final condition can be set for an arbitrary large k = k* instead of k = +
infinity).
```

Homework 1.5. 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 $\lim\limits_{n\to+\infty} x_n = \ln(1+x_0)$.

a. Set $x_0 = 1$, compute $x_1,\ x_2,\ \ldots,\ x_{71}$ and explain the obtained results.

b. Transform the sequence in an equivalent one that converges to the theoretical limit.

Solution Homework 1.5.

```
%  Set x0 = 1, compute x1, x2, ... , x{71} and explain the obtained results.

clear all, close all, clc
format long

n_max = 71;
x = zeros(n_max+1, 1);

x(1) = 1;     % x0 set to 1
for n = 0:n_max-1
  x(n+2) = 2^(n+1)*(sqrt(1 + x(n+1)/2^(n)) - 1); % Pay attention to the indexing!!!
end
x(end)
x_lim = log(1 + x(1))

figure
hold on, box on
plot([0:71], x, 'bx-','Linewidth',3)
plot([0:71], x_lim*ones(n_max+1,1)', 'r-','Linewidth',3)
axis([-1 72 -0.05 1.05])
set(gca,'LineWidth',2)
set(gca,'FontSize',16)

%  When computing the sequence with Octave/MATLAB we find that x{71} differs a lot from ln(1+
x0) = ln(2). The plots also shows that xn = 0, for all n >= n* = 52: this is due to numerical
cancellation effects, because at n = n* it holds {x{n*}}/{2^{n*}} < eps, hence in finite
precision arithmetic x{n*} = 0!

x(53)

%%

%  The value of n* can also be computed with the following argument: since (hopefully)
numerical cancellation will occur for large n
%  x{n} ~ ln(1+x0)
%  therefore
%  {x{n}}/{2^{n}} < eps is approximately equivalent to {ln(1+x0)}/{2^{n}} < eps
%  which can be solved for n, finding
%  n > log2({ln(1+x0)}/{eps}


x_0 = 1;
n = log(log(1 + x_0)/eps)/log(2)


%  Transform the sequence in an equivalent one that converges to the theoretical limit.
%
%  After a razionalization, the recurrence can be written as
%  x{n+1} = 2^{n+1} [ sqrt{1+{xn}/{2^{n}}} - 1 ] = {2 xn}{ sqrt{1 + {xn}/{2^{n}}} + 1}


clc, clear all

n_max = 71;
x = zeros(n_max+1, 1);

x(1) = 1;      % x0 set to 1
for n = 0:n_max-1
  x(n+2) = 2*x(n+1)/(sqrt(1 + x(n+1)/2^(n)) + 1);
end
x(end)

%  The error in this case is
x(end) - log(1 + x(1))

figure
hold on, box on
plot([0:n_max], x,'bx-','Linewidth',3)
```

```
plot([0:n_max], log(1+x(1))*ones(n_max+1), 'r-','Linewidth',3)
axis([-1 72 0.67 1.02])
set(gca,'LineWidth',2)
set(gca,'FontSize',14)

%  Using the previous calculations we know that (1 + xn/2^n) approximately 1 for n >= 52, so x
(n+1) = 2*xn/(1 + 1) = xn
```