553.481/681 Numerical Analysis

Homework 1 Solutions
Problem 1. (a) Show that to convert between binary and hexadecimal representations, each hexadecimal digit $b$ is replaced, in order, by the four binary digits $a, a', a'', a'''$ that satisfy

$$b = 2^3a + 2^2a' + 2a'' + a'''$$.

(b) Use part (a) to write a Matlab script `hextobin.m` which converts any $n$-dimensional vector $[h_1, ..., h_n]$ of the values $h_i \in \{0, 1, 2, ..., 9, a, b, ..., f\}$, with the definitions

```matlab
a=10; b=11; c=12; d=13; e=14; f=15;
```

into a vector $[b_1, ..., b_{4n}]$ of the values $b_i \in \{0, 1\}$. Hint: Use Matlab’s `mod` function.

(c) Apply your script from part (b) to find the binary equivalent of the hexadecimal string 401db5f0.

(a) Let $x := \cdots b_2 b_1 b_0, b_{-1} b_{-2} \cdots)_2$ be a binary representation. Note that hexadecimally $x = \sum_i h_i 16^i$ for some $h_i \in \{0, \ldots, 15\}$. We use the representation for the summand $h_i 16^i = b_{4i+3}2^4 + b_{4i+2}2^4 + b_{4i+1}2^4 + b_{4i}2^4$. But this gives us a binary representation so we’re done.

(b) See:

```matlab
a=10; b=11; c=12; d=13; e=14; f=15;
h=[4 0 1 d b 5 f 0];
n=length(h);
bb=[];
for ii=1:n
    z=h(ii);
    z0=mod(z,2);
    z=(z-z0)/2;
    z1=mod(z,2);
    z=(z-z1)/2;
    z2=mod(z,2);
    z3=(z-z2)/2;
    bb=[bb, [z3 z2 z1 z0]];
end
bb=num2str(bb);
bb=bb(~isspace(bb))
```

(c) We find $bb = 0100000000011101101010111110000$. 
Problem 2. In IEEE Standard Floating-Point representation for numbers, a 12-bit binary number is used first to encode the sign \( \sigma \) and exponent \( E \), followed by a 52-bit binary to encode the fraction \( F \). This can also be represented by a 3-digit hexadecimal number, followed by a 13-digit hexadecimal number, as in MATLAB. Consider the following IEEE Standard Floating-Point number in hexadecimal form:

\[ x=4012ad432fc61c97 \]

(a) Use the first 3 hexadecimal digits to determine the sign \( \sigma \) and exponent \( E \) of \( x \) in decimal format.
(b) Convert the remaining 13 hexadecimal digits to a decimal representation of the fraction \( F \) for \( x \).
(c) Calculate the decimal representation of the double-precision number \( x \). You can use the Matlab intrinsic function \texttt{hex2num} to check your answer in (c), but you must explain how the result follows from parts (a) and (b).

(a) We find \((401)_{16} = (01000000001)_2\), so that \( \sigma = 0 \), \( SE = (1000000001)_2 = 1025 \), and \( E = SE - 1023 = 2 \).

(b) We find that \( F = (0.2ad432fc61c97)_{16} = 0.167300402275748 \).

(c) We have that \( x = (-1)^\sigma \times (1 + F) \times 2^E = 4.669201609102990 \). This agrees with \texttt{hex2num} to double precision. Note that this is a double-precision approximation of the Feigenbaum constant. See:

\[ \text{https://en.wikipedia.org/wiki/Feigenbaum_constants} \]
Problem 3. Derive the following bounds for the relative error $\epsilon(x)$ in the rounded floating-point representation:

$$-\frac{1}{2} \beta^{-(t-1)} \leq \epsilon(x) \leq \frac{1}{2} \beta^{-(t-1)}.$$

For simplicity, assume an even base such as $\beta = 2, 10, \text{or } 16$.

Let $x = \sigma(0.a_1a_2a_3\cdots)\beta^e$. With rounding we have

$$\hat{x} = \begin{cases} 
\sigma(0.a_1a_2a_3\cdots a_t)\beta^e & \text{if } a_{t+1} < \beta/2 \\
\sigma(0.a_1a_2a_3\cdots [a_t+1])\beta^e & \text{else}
\end{cases}.$$

Case I: $0 \leq a_{t+1} < \beta/2$

$$x - fl(x) = \sigma(0.0...0a_{t+1}...a_t)\beta^e$$

$$\frac{x - fl(x)}{x} = \frac{(0.0...0a_{t+1}a_{t+2}a_{t+3}...a_t)\beta}{(0.a_1a_2...a_t)\beta} = \frac{(a_{t+1}a_{t+2}a_{t+3}...a_t)\beta^e}{(a_1a_2...a_t)\beta}.$$

Let’s bound the numerator and denominators (both are positive). Since $\beta$ is an even integer, if $a_{t+1} < \frac{\beta}{2}$ then $a_{t+1} \leq \frac{\beta}{2} - 1$.

Let’s use this on the numerator

$$(a_{t+1}a_{t+2}a_{t+3}...a_t)\beta = a_{t+1} + (0.a_{t+2}a_{t+3}...a_t)\beta \leq a_{t+1} + 1 \leq \frac{\beta}{2}.$$

We bound the denominator from below, $(a_1.a_2...a_t)\beta > 1$. So we have

$$\frac{x - fl(x)}{x} \leq \frac{1}{2} \beta^{-t+1}.$$

Case II: $\frac{\beta}{2} \leq a_{t+1} < \beta$. Here, write down the relative error

$$\frac{fl(x) - x}{x} = \frac{(0.0...01)\beta - (0.0...0a_{t+1}a_{t+2}...a_t)\beta}{(0.a_1a_2...a_t)\beta} = \frac{1 - (0.a_{t+1}a_{t+2}...a_t)\beta}{(0.a_1a_2...a_t)\beta}.$$

Again we bound the positive numerator and the positive denominator. To bound the numerator, we notice that since $\frac{\beta}{2} \leq a_{t+1} < \beta$, hence

$$(0.a_{t+1}a_{t+2}...a_t)\beta = \frac{a_{t+1}}{\beta} + ... \geq \frac{a_{t+1}}{\beta} \geq \frac{1}{2}.$$

or, equivalently, $1 - (0.a_{t+1}a_{t+2}...a_t)\beta \leq \frac{1}{2}$. For the denominator, using that $a_1 > 1$, we deduce that $(0.a_1a_2...a_t)\beta \geq \frac{1}{2}$. So our inequality becomes

$$\frac{fl(x) - x}{x} \leq \frac{1}{2} \beta^{-t+1}.$$
Problem 4. Consider the function 

\[ f(x) = \frac{1}{2}(1 - \cos x) \]

(a) Calculate the condition number \( K(x) \) for evaluating the function at \( x \). In particular, calculate \( K(0) \) and explain, on this basis, whether the problem of evaluating \( f(x) \) for \( |x| \ll 1 \) is well-conditioned.

(b) Evaluate \( f(x) \) in Matlab for \( x = 10^{-8} \) using the given expression. How many significant figures do you obtain? If your answer is not accurate to double precision, find another expression for \( f(x) \) which can yield double-precision accuracy at \( x = 10^{-8} \) and state the double-precision result obtained. Justify carefully your result.

(a) Note \( K(x) = xf'(x)/f(x) = x\sin(x)/[1 - \cos(x)] = x/\tan(x/2) \). For \( |x| \ll 1 \) write

\[ \lim_{x \to 0} K(x) = \lim_{x \to 0} \frac{x}{\tan(x/2)} = \lim_{x \to 0} \frac{x}{\frac{1}{2}x + O(x^3)} = 2. \]

(b) See:

\begin{verbatim}
>> f=@(x) (1-cos(x))/2;
>> x=1e-8;
>> y=f(x)

y =
   0

Since the true answer is certainly not zero, the result has no significant figures.
We try the trig identity \( f(x) = \sin^2(x/2) \):

\begin{verbatim}
>> f=@(x) sin(x/2).^2;
>> y=f(x)

y =
   2.500000000000000e-17

We try the Taylor series approximation \( f(x) = \frac{1}{4}x^2 - \frac{1}{48}x^4 + \cdots \approx \frac{1}{4}x^2 (1 - \frac{1}{8}x^2) \approx \frac{1}{4}x^2/4 \) since \( \frac{1}{8}x^2 < \text{eps}/2 \), the unit round in matlab:

\begin{verbatim}
>> f=@(x) x.^2/4;
>> y=f(x)

y =
   2.500000000000000e-17
\end{verbatim}
\end{verbatim}
Problem 5*. Consider the function

\[ f(x) = \frac{x^{54} - 1}{x - 1} = 1 + x + x^2 + \cdots + x^{53} \]

(a) Calculate the condition number \( K(x) \) for evaluating the function near \( x = -1 \), that is, for \( x = -1 + y \) with \( |y| \ll 1 \). Is the problem well-conditioned? Then answer the same question for the function \( g(y) = f(-1 + y) \) when evaluated for \( |y| \ll 1 \). If the answer is different, explain why.

(b) Evaluate \( f(x) \) in Matlab for \( x = -1 + 10^{-8} \) using the two given expressions. Explain why the results are probably not accurate to double precision. Which of the two expressions do you believe gives a more accurate result, and why?

(c) Find an expression for \( f(x) \) which can yield double-precision accuracy at \( x = -1 + 10^{-8} \) and state the double-precision result obtained. Justify carefully your result and calculate the relative error in the two results from (b).

(a) Write

\[ f(y - 1) = \frac{(1 - y)^{54} - 1}{y - 2} = \frac{-54y + O(y^2)}{y - 2} \approx 27y + O(y^2), \quad f'(y - 1) \approx 27 + O(y). \]

Thus

\[ K_f(y - 1) = \frac{(y - 1)f'(y - 1)}{f(y - 1)} \approx \frac{y - 1}{y} \approx -\frac{1}{y}, \]

and the problem is poorly conditioned for \( |y| \ll 1 \). However

\[ K_g(y) = \frac{yg'(y)}{g(y)} = \frac{27y + O(y^2)}{27y + O(y^2)} \approx 1, \]

which is well-conditioned. The issue is, while \( dx, dy \) are the same, \( x^{-1}dx \ll y^{-1}dy \) when \( |y| \ll 1 \) and \( x \approx -1 \).

(b) Both expressions have loss of significance, but \( f(x) = (x^{54} - 1)/(x - 1) \) has many fewer subtractions of nearly equal quantities than \( f(x) = \sum_k x^k \), so we expect the former is more accurate. See:

```matlab
1 del=1e-8;
2 x=-1+del
3 S1=1;
4 A=1;
5 for i=1:53
6     A=A*x;
7     S1=S1+A;
8 end
9 S1=S1/( -2+del)
10 S2=(x^54-1)/(x-1)
```

We find \( S1 = 2.699999311905899e-07 \) and \( S2 = 2.699999311528107e-07 \).

(c) Note that

\[ (1 - y)^{54} - 1 = \sum_{k=1}^{54} (-1)^k \binom{54}{k} y^k, \quad g(y) = \frac{1}{y-2} \sum_{k=1}^{54} (-1)^k \binom{54}{k} y^k, \]

where the terms in this sum are alternating in sign but rapidly decreasing in magnitude with \( k \). Thus, we expect that this expression will yield a result accurate to double precision.

We implement as:

```matlab
1 % first solution (it is generally better to sum from smallest to largest)
2 A=-54*del;
3 S3=A;
4 for i=2:54
5     A=-A*(55-i)*del/i;
6     S3=S3+A;
7 end
8 S3=S3/( -2+del)
10 % second solution
11 A=1;
12 S4=A*del^54;
```
for i=1:53
    A=-A*(55-i)/i;
    S4=S4+A*del^(54-i);
end
S4=S4/(-2+del)
relerr1=S1/S4-1
relerr2=S2/S4-1
err3=S3-S4

The sum is 2.699999298000121e-07.
As expected, the results from (b) do not agree with this one to double precision. The relative error of summing powers is 5.150289528188523e-09, and of direct computation 5.010366566082780e-09.
Problem 6. (a) For any natural number \( n \), calculate the condition number \( K_n(x) \) for evaluating the function \( f_n(x) = x^n \) at a real number \( x \).

(b) Show that \( y_n = f_n(\psi) \) can be evaluated for \( \psi = \frac{1 - \sqrt{5}}{2} \) by the algorithm based on the following iteration:

\[
y_{n+1} = y_n + y_{n-1}, \quad n > 1; \quad y_0 = 1, \quad y_1 = \psi.
\]

**Hint:** Show that the general solution of the above iteration is

\[
y_n = a\psi^n + b\phi^n
\]

with \( \phi = \frac{1 + \sqrt{5}}{2} \) and constants \( a, b \) determined by \( y_0, y_1 \).

(c) Implement in Matlab the algorithm in part (b) to evaluate \( f_n(\psi) \) for \( n = 2, \ldots, 40 \) and calculate the corresponding relative errors, plotting the latter versus \( n \) with logarithmic scale on the ordinate axis. Are the relative errors of the same magnitude as those estimated from the condition number? How rapidly do the relative errors grow? Explain your results in terms of the analysis of part (b).

(a) Directly:

\[
K_n(x) = x f'_n(x)/f_n(x) = n.
\]

(b) Observe that \( \psi, \phi \) are quadratic solutions to \( x^2 - x - 1 = 0 \) or \( x^2 = x + 1 \), so that the general solution of the linear difference equation \( y_{n+1} = y_n + y_{n-1} \) is \( y_n = a\psi^n + b\phi^n \). The initial conditions \( n = 0, 1 \) by inspection give \( a = 1, b = 0 \) and thus \( y_n = f_n(\psi) \).

(c) The relative error of the Fibonacci recurrence grows far more rapidly than the condition number for \( f_n \) would suggest. Recall from (b) that the general solution includes a \( \phi^n \) term which doesn’t appear in our exact solution. The culprit is the initial conditions \( y_0 = 1, y_1 = \psi \). Although 1 is a machine number, the irrational number \( \psi \) is not. Thus, in Matlab, \( \hat{y}_1 = \psi + \phi\delta \) where \( \delta = O(10^{-16}) \). It follows that the solution in Matlab is

\[
\hat{y}_n = \psi^n + \phi^n\delta
\]

with the final “parasitic term” growing and dominating the exact solution as \( n \to \infty \). See:

```matlab
phi=(1+sqrt(5))/2;
psi=(1-sqrt(5))/2;
xold=1;
x=psi;
for ii=2:40
    x=x+xold;
    xold=x;
end
```
xnew = x + xold;
xold = x;
x = xnew;
relerr(ii-1) = x / psi^ii-1;
end

nn = (2:40);
semilogy(nn, abs(relerr), '-b', 'LineWidth', 1)
hold on;
semilogy(nn, nn*eps, '--r', 'LineWidth', 1)
xlabel('n', 'FontSize', 15, 'Interpreter', 'latex')
ylabel('relerr', 'FontSize', 15, 'Interpreter', 'latex')
legend('relerr', 'estimate', 'Location', 'NorthWest')
relerr40 = relerr(39)
relerrest = eps*(phi/psi)^40