CS4220 PS 3

Due: Weds, February 11, 2015

1. (Ascher and Greif, Section 2.5, Problem 10) $f_1(x+\delta) = \cos(x+\delta) - \cos(x)$ can be transformed into another form, $f_2(x+\delta)$, using the trigonometric formula:

$$\cos(\phi) - cos(\psi) = -2\sin\left(\frac{\phi+\psi}{2}\right)\sin\left(\frac{\phi-\psi}{2}\right).$$

Solution: Since

$$f_1(x+\delta) = \cos(x+\delta) - \cos(x),$$

(a) We can use the trigonometric identity to get

$$f_2(x+\delta) = -2\sin\left(\frac{2x+\delta}{2}\right)\sin\left(\frac{\delta}{2}\right).$$

(b) We can find both of the following limits by using the l'Hôpital's rule:

$$\lim_{\delta\to 0}\frac{f_1(x,\delta)}{\delta} = \lim_{\delta\to 0}\frac{-\sin(x+\delta)}{1} = -\sin(x).$$

$$\begin{aligned}\lim_{\delta\to 0}\frac{f_2(x,\delta)}{\delta} &= \lim_{\delta\to 0}\frac{-\cos\left((2x+\delta)/2\right)\sin(\delta/2) - \sin\left((2x+\delta)/2\right)\cos(\delta/2)}{1} \\ &= 0 - \sin(x) = -\sin(x). \end{aligned} \quad (1)$$

(c) Let

$$g_1(x,\delta) = f_1(x,\delta)/\delta + \sin(x),$$
$$g_2(x,\delta) = f_2(x,\delta)/\delta + \sin(x),$$

for $x = 3$, $\delta = 1\times 10^{-11}$. As shown in the derivation above, $g_1(x,\delta)$ and $g_2(x,\delta)$ should both approach zero because $\delta = 10^{-11}$ is very small.

(d) The code is listed below.

```
%
%Testing two mathematically equivalent functions
%in Chapter2 Problem 10 of Ascher and Greif:
%
f1 = @(x,delta) cos(x+delta) - cos(x);
f2 = @(x,delta) -2*sin((2*x+delta)/2)*sin(delta/2);
delta=1.0E-11;
x=3;
g1 = f1(x, delta)/delta + sin(x);
g2 = f2(x, delta)/delta + sin(x);

disp(sprintf('g1= %10.8e,    g2= %10.8e', g1, g2));
```

The numerical results of g_1 and g_2 in MATLAB are very different. g_1 is based on f_1, which has a subtraction of two numbers very close to each other (the cancellation error), whereas g_2 does not. Consequently, g_2, the implementation avoiding cancellation errors, is much more accurate than g_1 when δ is small:

```
x= 3; delta= 1.0E-13;
g1= 1.21683932e-04,   g2= 4.96824804e-14
```

2. (Ascher and Greif, Section 2.5, Problem 11)

a we can derive the expression as follows:

$$\begin{aligned}
\ln(x-\sqrt{x^2-1}) &= \ln\left(\frac{x-\sqrt{x^2-1}}{1}\cdot\frac{x+\sqrt{x^2-1}}{x+\sqrt{x^2-1}}\right)\\
&= \ln\left(\frac{x^2-(x^2-1)}{x+\sqrt{x^2-1}}\right)\\
&= \ln\left(\frac{1}{x+\sqrt{x^2-1}}\right)\\
&= \ln\left(\left[x+\sqrt{x^2-1}\,\right]^{-1}\right) = -\ln(x+\sqrt{x^2-1}).
\end{aligned}$$

b For numerical computation, $-\ln(x+\sqrt{x^2-1})$ is a more suitable expression than $\ln(x-\sqrt{x^2-1})$, because the latter expression the subtraction may potentially trigger the problem of numerical cancellation errors, whereas the former one is free of such problems.

The following examples can illustrate the potential pitfalls:

Let $y1 = \ln(x-\sqrt{x^2-1})$ and $y2 = -\ln(x+\sqrt{x^2-1})$.

```
x=1.00000000e+05, y1= -1.22060738e+01,   y2= -1.22060726e+01
x=1.00000000e+07, y1= -1.68054314e+01,   y2= -1.68112428e+01
x=1.00000000e+08, y1=       -Inf,        y2= -1.91138279e+01
x=1.00000000e+10, y1=       -Inf,        y2= -2.37189981e+01
```

As $x \geq 10^8$, $y1$ no longer provides correct numerical values due to the numerical cancellation error, whereas $y2$ does not have such a problem.

3.(Ascher and Greif, Section 4.6.16)

(a) Suppose that A is an orthogonal matrix. What are its singular values?

(b) Is the SVD of a given matrix A unique in general?

Solution: We write the SVD of A as $A = U\Sigma V^T$. Because A is orthogonal,

$$A^T A = I = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^T V^T U\Sigma V^T = \Sigma^T \Sigma.$$

This shows that all the singular values of A must be equal to 1.

We can find a counterexample where the singular value decomposition is not unique. Also note that polar decomposition and SVD are equivalent. Polar decomposition is not unique unless the operation is invertible, therefore SVD is not unique.

4. Definitions Let $\hat{x} = 32$ be regarded as an approximation to the positive solution for $f(x_*) = x_*^2 - 1000 = 0$. What are the absolute error, the relative error and the residual error?

Solution: They are defined in Chapters 1 and 2 of [Ascher and Greif].

- Abolute error: $|\hat{x} - x_*|$ =0.37722.
- Relative error: $\dfrac{|\hat{x} - x_*|}{|x_*|}$ =1.1929E-2.
- Residual error: $|f(\hat{x})|$ =24.

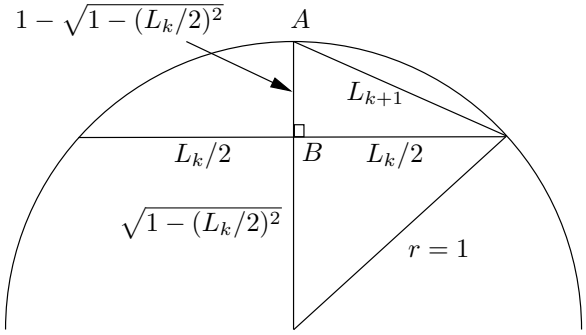
5: Pi, see! The following routine estimates π by recursively computing the semiperimeter of a sequence of 2^{k+1}-gons embedded in the unit circle:

```
1  N = 4;
2  L(1) = sqrt(2);
3  s(1) = N*L(1)/2;
4  for k = 1:30
5    N = N*2;
6    L(k+1) = sqrt( 2*(1-sqrt(1-L(k)^2/4)) );
7    s(k+1) = N*L(k+1)/2;
8  end
9
10 semilogy(1:length(s), abs(s-pi));
11 ylabel('|s_k-\pi|');
12 xlabel('k')
```

Plot the absolute error $|s_k - \pi|$ against k on a semilog plot. Explain why the algorithm behaves as it does, and describe a reformulation of the algorithm that does not suffer from this problem.

Solution: The given routine for estimating π is based on the semi-perimeters of a sequence of regular 2^{k+1}-polygons, $k = 1, 2, \ldots$, inscribed in the unit circle. As

illustrated in the figure, L_k is the length of one side of the 2^{k+1} polygon, which is related to the side length L_{k+1} of the next-level polygon, and the simple relation is based on the Pythagorean theorem:



$$\begin{aligned} L_{k+1}^2 &= \left(1-\sqrt{1-\left(\frac{L_k}{2}\right)^2}\right)^2+\left(\frac{L_k}{2}\right)^2 \\ &= 2-2\sqrt{1-\left(\frac{L_k}{2}\right)^2} \\ &= 2\left(1-\sqrt{1-\left(\frac{L_k}{2}\right)^2}\right) \end{aligned} \tag{2}$$

The square root of Eq.(2) is implemented 'as is' in line 6 of the given code.

When we look at the sides of polygons at two consecutive levels, it becomes obvious that as k grows the following two quantities in Eq.(2) get very close to each other rather quickly (or side $\overline{AB}$ in the figure gets very small very fast):

$$1 \simeq \sqrt{1-(L_k/2)^2} \quad \text{(as } k \text{ becomes larger)},$$

resulting in large numerical cancellation errors in calculating $1-\sqrt{1-(L_k/2)^2}$ in Eq.(2). The problem of the cancellation errors can be seen in Fig.(1). For the data points labeled as "bad algorithm", as $k \geq 15$, the absolute error $|s_k - \pi|$ stops going lower as it should, it actually starts going larger as k increases.

Now that we have identified where the numerical problem occurs, we can implement an effective but simple change to Eq.(2) in the following:

$$\begin{aligned} 1-\sqrt{1-\left(\frac{L_k}{2}\right)^2} &= \frac{\left(1-\sqrt{1-(L_k/2)^2}\right)\left(1+\sqrt{1-(L_k/2)^2}\right)}{1+\sqrt{1-(L_k/2)^2}} \\ &= \frac{(L_k^2/4)}{1+\sqrt{1-(L_k/2)^2}}. \end{aligned} \tag{3}$$

The difference in this implementation is that we have avoided subtracting two numbers very close to each other and the associated cancellation errors.

Substituting Eq.(3) into Eq.(2) to get

$$L_{k+1} = \sqrt{\frac{(L_k^2/2)}{1+\sqrt{1-(L_k/2)^2}}}. \tag{4}$$

We can change just line 6 in the given MATLAB code accordingly. The absolute errors from the original (bad) implementation versus the robust implementation given above are plotted in the following figure. Unlike the original code, the absolute error of the improved algorithm reaches the machine precision without any problem.

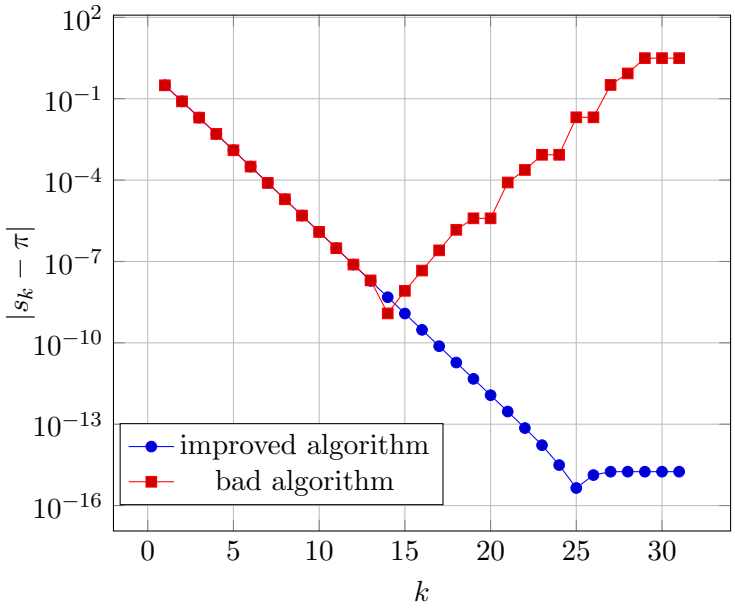


Figure 1: In estimating the value of π, a plot of absolute error $|s_k - \pi|$ versus k, the level of the 2^{k+1}-polygons, in two numerical implementations.

The modified code is listed as follows:

```
N = 4;
kmax = 30;
L = zeros(kmax, 1);
Le =zeros(kmax, 1);
s = zeros(kmax, 1);
se = zeros(kmax, 1);
L(1) = sqrt(2);
Le(1) = L(1);
s(1) = N*L(1)/2;
se(1) = N*Le(1)/2;
for k = 1:kmax
    N = N*2;
    Le(k+1) = sqrt( 2*(1-sqrt(1-Le(k)^2/4)) );
    L(k+1) = sqrt( L(k)^2/(2*( 1 + sqrt( 1-L(k)^2/4))));
    se(k+1)= N*Le(k+1)/2;
    s(k+1) = N*L(k+1)/2;
    disp( sprintf('%d, pi_approx=%20.18f, pi_approx_e=%20.18f',...
        k, s(k+1), se(k+1)));

end
semilogy(1:length(s), abs(s-pi), 1:length(s), abs(se-pi));
ylabel('|s_k -\pi|');
xlabel('k')
%output for plotting
%[(1:length(s))' (abs(s-pi)) (abs(se-pi))]
```