

MAT 128 A - Numerical Analysis
Homework 1

1. a) We want to find the limit

$$\lim_{x \rightarrow 0} f(x) = \lim_{x \rightarrow 0} \frac{1 - \cos(x)}{x^2}.$$

We can use l'hospital's rule twice to find the limit. Taking two derivatives of the numerator and denominator we have

$$\begin{aligned} \lim_{x \rightarrow 0} f(x) &= \lim_{x \rightarrow 0} \frac{1 - \cos(x)}{x^2} = \lim_{x \rightarrow 0} \frac{g(x)}{h(x)} \\ &= \lim_{x \rightarrow 0} \frac{g'(x)}{h'(x)} \\ &= \lim_{x \rightarrow 0} \frac{\cos(x)}{2} \\ &= \frac{1}{2} \end{aligned}$$

b) As $x \rightarrow 0$, $f(x) \rightarrow \frac{1}{2}$ as $\mathcal{O}(x^2)$. We see

$$\begin{aligned} \frac{1}{x^2} (1 - \cos(x)) &= \frac{1}{x^2} \left(1 - \left(1 - \frac{1}{2}x^2 + \mathcal{O}(x^4) \right) \right) \\ &= \frac{1}{2} + \mathcal{O}(x^2) \end{aligned}$$

c) Using MATLAB $f(x)$ was computed for $x = 10^{-1}, 10^{-2}, \dots, 10^{-10}$. Results are shown in the table below.

x	$f(x)$
10^{-1}	0.4995834721
10^{-2}	0.4999958333
10^{-3}	0.4999999583
10^{-4}	0.4999999969
10^{-5}	5.0000004137
10^{-6}	5.0004445029
10^{-7}	0.4996003610
10^{-8}	0
10^{-9}	0
10^{-10}	0

The results are close to 0.5 for x from 10^{-1} to 10^{-4} , and evaluate to 0 for values of x less than or equal to 10^{-8} . This is due to the computer evaluating $\cos(x) = 1$ for $x < 10^{-8}$.

d) If the computer can only represent floating point numbers with N decimal digits of accuracy, then $\text{fl}(1 + x^2) = 1$ when $x^2 < 10^{-N}$, where $\text{fl}(y)$ represents the floating point representation of y . When evaluating f , we will see large errors when $|x| \leq 10^{-\sqrt{N}}$ since $\cos(x) = 1 + \mathcal{O}(x^2)$.

e) We can remove the accuracy problem by rearranging $f(x)$.

$$\begin{aligned} f(x) &= \frac{1 - \cos(x)}{x^2} = \frac{1 - \cos(x)}{x^2} \left(\frac{1 + \cos(x)}{1 + \cos(x)} \right) \\ &= \frac{1 - \cos^2(x)}{x^2 (1 + \cos(x))} \\ &= \frac{\sin^2(x)}{x^2 (1 + \cos(x))} \end{aligned}$$

Now when $x \ll 1$, $\sin(x) \approx 0$ and $\cos(x) \approx 1$ and the problem with floating point representation is taken care of. See the table below. There will be a problem when $\cos(x) = -1$, i.e. when $x = (2n + 1)\pi$.

x	$f(x)$
10^{-1}	0.4995834721
10^{-2}	0.4999958333
10^{-3}	0.4999999583
10^{-4}	0.4999999995
10^{-5}	0.4999999999
10^{-6}	0.4999999999
10^{-7}	0.4999999999
10^{-8}	0.5000000000
10^{-9}	0.5000000000
10^{-10}	0.5000000000

2. a) Code to evaluate arctan

```
function ret = evalarctan(x,N)
```

```
% evaluates arctangent using arctan(x) = \sum_0^N \frac{(-1)^n x^{2n+1}}{2n+1}
% inputs: x - point at which to evaluate arctan
%          N - number of terms to use in series
% output: ret - approximation of artcan(x)
```

```
sum = 0;
```

```
for i=0:1:N
```

```
    sum = sum + (-1)^i * x^(2*i+1) / (2*i + 1);
```

```
end
```

```
ret = sum;
```

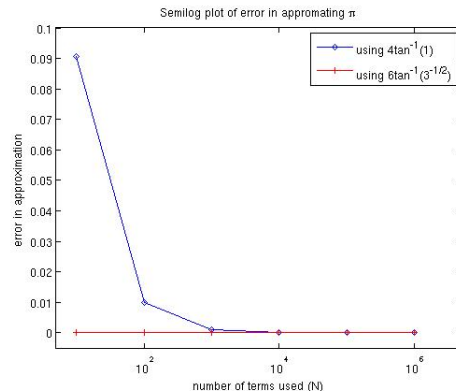
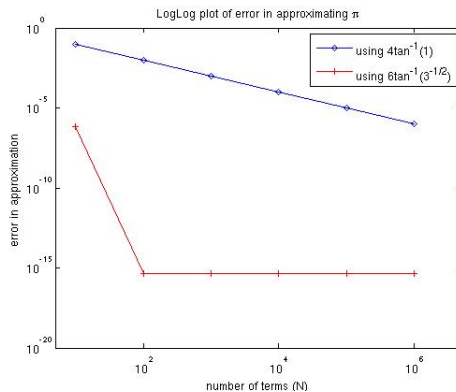
b) We can approximate π by computing $4\tan^{-1}(1)$. Results for various values of N are shown below.

N	$\pi = 4\tan^{-1}(1)$
10^1	3.232315809
10^2	3.151493401
10^3	3.142591654
10^4	3.141692643
10^5	3.141602653
10^6	3.141593653

c) We can approximate π by computing $6\tan^{-1}(3^{-\frac{1}{2}})$. Results for various values of N are shown below.

N	$\pi = 6\tan^{-1}(3^{-\frac{1}{2}})$
10^1	3.141593304
10^2	3.141592653
10^3	3.141592653
10^4	3.141592653
10^5	3.141592653
10^6	3.141592653

d) The errors of the two approximations are shown in log-log and semilog plots below. These plots show that approximating π is done much better when calculating $\pi = 6\tan^{-1}(3^{-\frac{1}{2}})$. This is because the terms in the series $\tan^{-1}(1) = \sum_0^N \frac{(-1)^n 1^{2n+1}}{2n+1}$ decay as $1/n$, which is slow compared to the terms in $\tan^{-1}(3^{-\frac{1}{2}}) = \sum_0^N \frac{(-1)^n (3^{-1/2})^{2n+1}}{2n+1}$ which decay exponentially.



e) The series expression for $\tan^{-1}(x)$ is an alternating series. The bound for the error in the approximating the sum using N terms is just the absolute value of the $N + 1$ term, i.e.

$$\left| \frac{(-1)^{N+1} x^{2(N+1)+1}}{2(N+1)+1} \right|$$

3.

a) With $dt = 0.1s$ and running for 864000 time steps, we have $t = 8.640000000054126 * 10^4$ s.

The absolute error is $\|t - 86400\| = 0.54126 * 10^{-7}$ s.

The relative error is $\frac{\|t-86400\|}{86400} = 6.264 * 10^{-13}$ s.

b) With $dt = 0.125s$ and running for 691200 time steps, we have $t = 86400$ s. The absolute and relative error is zero.

c) The result with $dt = 0.125$ s has no error because $0.125 = \frac{1}{8} = 2^{-3}$ can be represented exactly in binary. 0.1 has no exact binary representation and is subject to computer rounding errors.

4. We want to approximate a function f using a second degree polynomial interpolation. The nodes are equally spaced over the period of the function and we will use the three closest node values in our interpolation, i.e. $(x, x-h, x+h)$. f is given by

$$f(x) = e^{\cos(x)}.$$

We want to find an h that guarantees that the interpolation produces values within a tolerance of 10^{-6} .

We can use Theorem 3.3 from the text to see that for parabolic interpolation

$$\begin{aligned} f(x) &= P(x) + \frac{f^{(n+1)}(\zeta(x))}{(n+1)!} (x-x_0)(x-x_1)(x-x_2) \\ &= P(x) + \frac{f'''(\zeta(x))}{6} (h^3) \end{aligned}$$

where $P(x)$ is our interpolating polynomial and the other term is the error. Taking three derivatives of f gives

$$f'''(x) = \sin(x)e^{\cos(x)} [1 + 3\cos(x) - \sin^2(x)]$$

We need to bound $f'''(x)$ and we do this by taking the absolute value and using the triangle inequality and $-1 \leq \sin(x), \cos(x) \leq 1$ to see

$$f'''(x) \leq 4e$$

We want the error to be less than 10^{-6} so we need

$$\begin{aligned} 10^{-6} &> \frac{1}{6} f'''(\zeta(x)) h^3 \\ &> \frac{1}{6} 4e h^3 \end{aligned}$$

This gives us our condition on h

$$h \leq \left(\frac{3}{2e} 10^{-6} \right)^{1/3} \approx 5.51819 * 10^{-6}$$

5. a) Here is the Neville's Method code modified to terminate when successive diagonal entries are less than a user entered tolerance.

```
function ret = neville(x,y,xi,tol)

%neville's method
%inputs : x - vector of nodes
%         y - vector of function values at nodes
%         xi - location to interpolate
%         tol - tolerance used to terminate algorithm
%output : yi - value of interpolation at xi

N = length(x);
p = zeros(N);
p(1,1) = y(1);

for i= 2:N
    p(i,1) = y(i);

    for j=2:i
        num = (xi-x(i))*p(i-1,j-1) - (xi-x(i-j+1))*p(i,j-1);
        den = x(i-j+1) - x(i);
        p(i,j) = num/den;
    end
    %added code to terminate when error tolerance acheived
    if (abs(p(i,i)-p(i-1,i-1)) < tol) break; end
end

ret = p;
```

Using the values in the table provide and the modified code with a specified tolerance of 10^{-6} , we can evaluate $Ei(1.625) = 0.0832168217387078$. This used a 7^{th} degree polynomial.

b) The differences in the diagonal elements of the Neville table are an order of magnitude greater than the actual error.

i	p(i,i) w/o reordering	p(i,i)-p(i-1,i-1) w/o reordering	error
1	2.193839343955e-01	-	1.361671378514870e-01
2	1.067749777377e-02	-2.087064366217e-01	7.253929877026284e-02
3	1.060070334557e-01	9.532953568198e-02	2.279023691172216e-02
4	7.845171204439e-02	-2.755532141136e-02	4.765084499639236e-03
5	8.386051909522e-02	5.408807050835e-03	6.437225511963934e-04
6	8.317063895927e-02	-6.898801359581e-04	4.615758476175791e-05
7	8.321743206827e-02	4.679310900708e-05	6.355242453265353e-07
8	8.321682173870e-02	-6.103295705217e-07	2.519467480477378e-08

c) After reordering the input data to put nodes closer to 1.625 earlier in the ordering, we approximate $Ei(1.625) = 0.08321681815685121$ with a 5^{th} degree

polynomial. We see that the differences in the diagonal elements decay much faster after the reordering. See table below.

i	p(i,i) w/o reordering	p(i,i) w/ reordering
1	2.193839343955e-01	8.630833369754e-02
2	1.067749777377e-02	8.339491137346e-02
3	1.060070334557e-01	8.320201517851e-02
4	7.845171204439e-02	8.321550678609e-02
5	8.386051909522e-02	8.321696859815e-02
6	8.317063895927e-02	8.321681815685e-02
7	8.321743206827e-02	-
8	8.321682173870e-02	-

6. Using the data given in the problem statement we can construct a table with the value of $q(p = 0.6)$ returned by Neville's method using interpolating polynomials of degree 1,2,3,4,5,6. Inspecting the table we see we *must* use linear interpolation as all other interpolations return a negative value of $q(p = 0.6)$. Since q is a probability, we must enforce $0 \leq q \leq 1$.

degree	$q(p = 0.6)$
1	0.1442
2	-0.1055
3	-1.5228
4	-7.0817
5	-17.9428