CS 7A - Fall 2015 - Approximating transcendental functions. Due 9/25/15 (extended)

Background Theory

Transcendental functions are functions that, while they cannot be written as polynomials, can be approximated by polynomials to any desired accuracy. For example,

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!} \approx \sum_{n=0}^{N} \frac{(-1)^n x^{2n+1}}{(2n+1)!}$$

So that $\sin(1) \approx 1 - \frac{1}{6} + \frac{1}{120} = 0.148\overline{6}$ is a good approximation, but $\sin(1) \approx 1 - \frac{1}{6} + \frac{1}{120} - \frac{1}{5040} \approx 0.84146825$ is better. Actually $\sin(1) \approx 0.84147098$

The polynomial approximation for e^x, where $e \approx 2.71828182845904523536028747135 27$ is even simpler:

$$e^x \approx \sum_{n=0}^{N} \frac{x^n}{n!}$$

So that $e^1 \approx 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} = 2.708\overline{3}$ is ok, but $e^1 \approx 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} = 2.71\overline{6}$ is better, and so on.

1. Write a C++ function to approximate $\sin(x)$ by its polynomial approximation to the nearest ten-thousandth. This should work for any input up to 1e10. Write a main() function to test your sin() function.

2. Write a C++ function to approximate e^x by its polynomial approximation to the nearest ten-thousandth. This should work for any input up to 1e10. Write a main() function to test your e^x function.

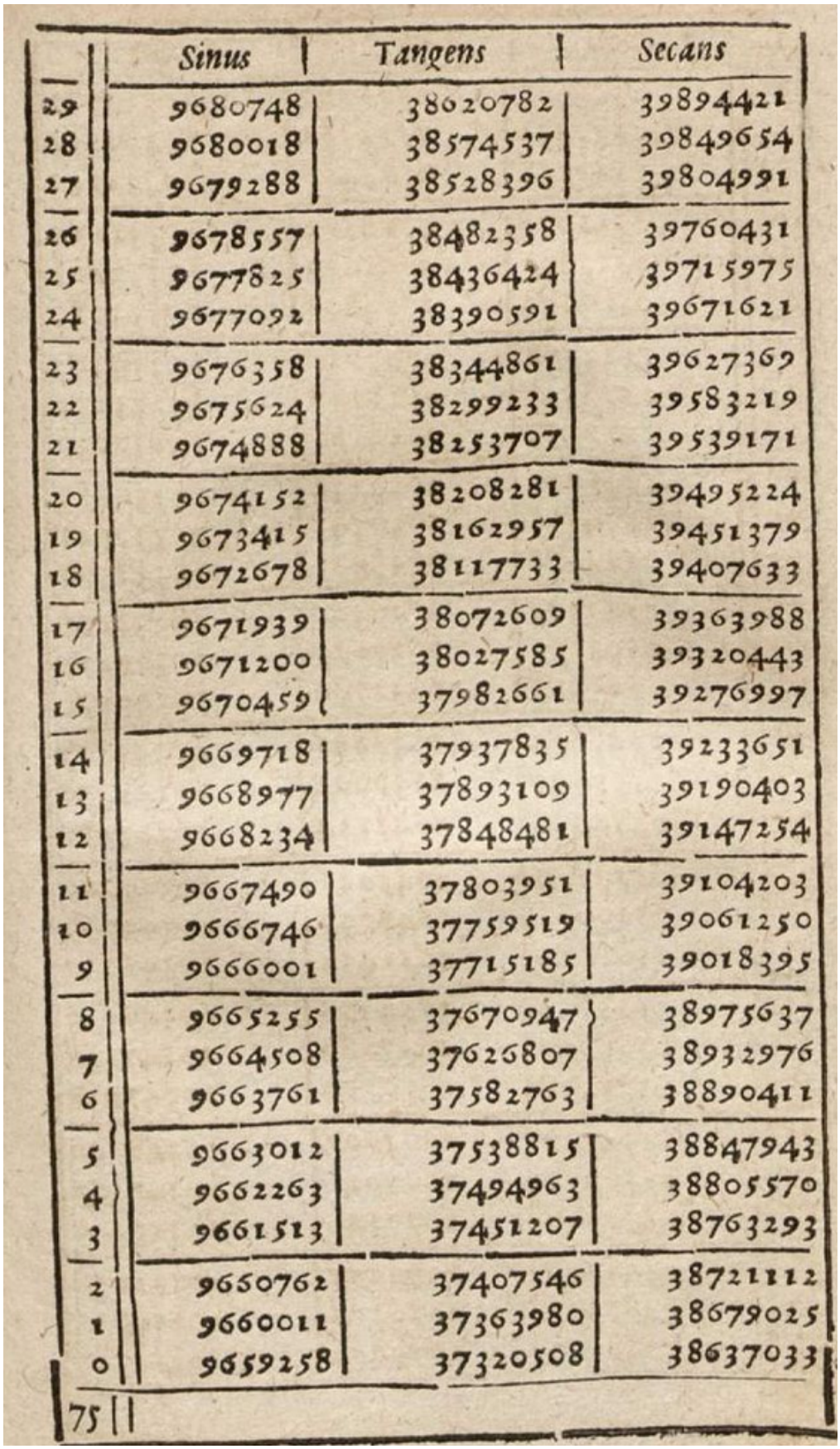
3. The polynomial approximation for $\ln(x+1)$ is

$$\ln(x+1) \approx \sum_{n=1}^{N} \frac{(-1)^{n+1} x^n}{n}$$

   Write a function for approximating $\ln(x)$ to the nearest ten-thousandth and test it. Note that we need to confine the domain to $-1 < x \leq 2$.

4. Use your functions to populate vectors which can be printed out to produce tables for $\sin(x)$, e^x and $\ln(x)$ like these (except with $\sin(x)$, e^x and $\ln(x)$:
   Note that it's helpful to have the "`setw()`","`right`", "`setprecision()`" and "`fixed`" commands from the `iomanip` library as illustrated in the code below:

```
1  void show_table(vector<double> vd) {
       /// top row header
3      int SP = 8;
       cout << "␣␣␣␣␣" << left << setw(SP) << 0 << setw(SP) << 1
5           << setw(SP) << 2 << setw(SP) << 3 << setw(SP) << 4 << setw(SP)
            << 5 << setw(SP) << 6 << setw(SP) << 7 << setw(SP) << 8
7           <<  9 << endl;
       for(int i = 0; i < 628; ++i) {
9          cout << right << setw(SP) << setprecision(3) << fixed << vd[i];
           //if((i+1)%10==0) cout << endl;
11     }
   }
```



	Sinus	Tangens	Secans
29	9680748	38020782	39894421
28	9680018	38574537	39849654
27	9679288	38528396	39804991
26	9678557	38482358	39760431
25	9677825	38436424	39715975
24	9677092	38390591	39671621
23	9676358	38344861	39627369
22	9675624	38299233	39583219
21	9674888	38253707	39539171
20	9674152	38208281	39495224
19	9673415	38162957	39451379
18	9672678	38117733	39407633
17	9671939	38072609	39363988
16	9671200	38027585	39320443
15	9670459	37982661	39276997
14	9669718	37937835	39233651
13	9668977	37893109	39190403
12	9668234	37848481	39147254
11	9667490	37803951	39104203
10	9666746	37759519	39061250
9	9666001	37715185	39018395
8	9665255	37670947	38975637
7	9664508	37626807	38932976
6	9663761	37582763	38890411
5	9663012	37538815	38847943
4	9662263	37494963	38805570
3	9661513	37451207	38763293
2	9650762	37407546	38721112
1	9660011	37363980	38679025
0	9659258	37320508	38637033

75