Taylor series approximation. Single/double floating point comparison

Nasser M. Abbasi

May 3,2007 Compiled on January 31, 2024 at 3:50am

This note compares the result of computing the numerical derivative to $\arctan(x)$ at $x = \sqrt{2}$ using Taylor approximation using single floating point and double floating point. This was done using Matlab. With Matlab, we can do single floating point computation using the *single* command. The default in Matlab is to do all the computations in double precision.

The approximation used is $f'(x) = \frac{1}{h}(f(x+h) - f(x))$ with h starting at 1 and halving it at each iteration.

The exact answer to $\frac{d\arctan(x)}{dx}$ evaluated at $x = \sqrt{2}$ is $1/3$. The results below show that using single precision, the numerical derivative keeps getting closer the exact answer up to iteration 12. The best answer is accuracy to 4 decimal places. After iteration 12, subtractive cancellation (loss of significance, L.O.S) become more dominant, and the result starts to become less accurate.

Using double precision, we see that we can go up to iteration 27 before loss of significance kicks in. The best numerical result at this point is accurate to 8 decimal points. Hence the accuracy is twice that of single precision.

The following diagram displays the results table for single precision, with a red box around the line where the numerical results starts to be affected by L.O.S. with the Matlab code used.

Using 32 bits floating point (on Intel PC), we see that the best approximation to derivative of arctan(x) at x=SQRT(2) will occur at k=12, with only 4 decimal points accuracy. The exact answer is 1/3. (0.3333333333……)

k	h	$f(\sqrt{2}+h)$	$f(\sqrt{2})$	$f(\sqrt{2}+h)-f(\sqrt{2})$	$f'(\sqrt{2})$
1	1	1.178097	0.9553166	0.2227806	0.2
2	0.5	1.089384	0.9553166	0.134067	0.
3	0.25	1.029727	0.9553166	0.07441014	0.2
4	0.125	0.9946444	0.9553166	0.0393278	0.3
5	0.0625	0.9755509	0.9553166	0.02023435	0.3
6	0.03125	0.9655817	0.9553166	0.01026511	0.3
7	0.015625	0.9604868	0.9553166	0.005170226	0.3
8	0.0078125	0.9579112	0.9553166	0.00259459	0.3
9	0.00390625	0.9566163	0.9553166	0.001299679	0.3
10	0.001953125	0.9559671	0.9553166	0.0006504655	0.3
11	0.0009765625	0.955642	0.9553166	0.0003253818	0.3
12	0.0004882813	0.9554793	0.9553166	0.0001627207	0.
13	0.0002441406	0.955398	0.9553166	8.136034e-005	0.
14	0.0001220703	0.9553573	0.9553166	4.070997e-005	0.3
15	6.103516e-005	0.9553369	0.9553166	2.032518e-005	0.3
16	3.051758e-005	0.9553268	0.9553166	1.019239e-005	0.3
17	1.525879e-005	0.9553217	0.9553166	5.066395e-006	0.3
18	7.629395e-006	0.9553192	0.9553166	2.563e-006	0.3
19	3.814697e-006	0.9553179	0.9553166	1.251698e-006	0.
20	1.907349e-006	0.9553173	0.9553166	6.556511e-007	0
21	9.536743e-007	0.9553169	0.9553166	2.980232e-007	
22	4.768372e-007	0.9553168	0.9553166	1.788139e-007	
23	2.384186e-007	0.9553167	0.9553166	5.960464e-008	
24	1.192093e-007	0.9553167	0.9553166	5.960464e-008	
25	5.960464e-008	0.9553167	0.9553166	5.960464e-008	
26	2.980232e-008	0.9553166	0.9553166	0	

Figure 1: 32 bit

The following diagram displays the results table for double precision, with a red box around the line where the numerical results starts to be affected by L.O.S. The Matlab code is the same as before, expect we simplify remove the command single wherever it was used.

Using 32 bits floating point (on Intel PC), we see that the best approximation to derivative of arctan(x) at x=SQRT(2) will occur at k=27, with 8 decimal points accuracy. The exact answer is 1/3. (0.3333333333……)

h	$f\left(\sqrt{2}+h\right)$	$f\left(\sqrt{2}\right)$	$f\left(\sqrt{2}+h\right)-f\left(\sqrt{2}\right)$	$f'($
1	1.17809724509617	0.955316618124509	0.222780626971663	
0.5	1.08938363393987	0.955316618124509	0.134067015815356	
0.25	1.02972677195646	0.955316618124509	0.0744101538319478	
0.125	0.994644389826101	0.955316618124509	0.0393277717015921	
0.0625	0.975550948454817	0.955316618124509	0.0202343303303073	
0.03125	0.965581699976702	0.955316618124509	0.0102650818521931	
0.015625	0.960486822895021	0.955316618124509	0.005170204770512	
0.0078125	0.957911223411024	0.955316618124509	0.00259460528651456	
0.00390625	0.956616307445695	0.955316618124509	0.00129968932118607	
0.001953125	0.955967060828989	0.955316618124509	0.000650442704479559	
0.0009765625	0.955641989159854	0.955316618124509	0.000325371035345134	
0.00048828125	0.955479341084496	0.955316618124509	0.000162722959986428	
0.000244140625	0.955397988967775	0.955316618124509	8.13708432652049e-005	
0.0001220703125	0.955357305887297	0.955316618124509	4.0687762787428e-005	
.103515625e-005	0.955336962591234	0.955316618124509	2.03444667244979e-005	
0517578125e-005	0.95532679050421	0.955316618124509	1.01723797002462e-005	
2587890625e-005	0.955321704350945	0.955316618124509	5.08622643524692e-006	
2939453125e-006	0.955319161246873	0.955316618124509	2.54312236402932e-006	
4697265625e-006	0.955317889687978	0.955316618124509	1.27156346863e-006	
3486328125e-006	0.955317253906815	0.955316618124509	6.35782305913324e-007	
7431640625e-007	0.955316936015805	0.955316618124509	3.17891295953388e-007	
7158203125e-007	0.955316777070193	0.955316618124509	1.58945683725875e-007	
8579101563e-007	0.95531669759736	0.955316618124509	7.94728507447218e-008	
9289550781e-007	0.955316657860937	0.955316618124509	3.97364275928069e-008	
6447753906e-008	0.955316637992724	0.955316618124509	1.98682144070261e-008	
3223876953e-008	0.955316628058617	0.955316618124509	9.93410731453537e-009	
1611938477e-008	0.955316623091563	0.955316618124509	4.96705365726768e-009	
8059692383e-009	0.955316620608036	0.955316618124509	2.48352682863384e-009	
9029846191e-009	0.955316619366273	0.955316618124509	1.24176346982807e-009	
4514923096e-009	0.955316618745391	0.955316618124509	6.20881679402885e-010	
2574615479e-010	0.95531661843495	0.955316618124509	3.10440895212594e-010	
1287307739e-010	0.95531661827973	0.955316618124509	1.55220392095146e-010	
3064365387e-010	0.95531661820212	0.955316618124509	7.7610251558724e-011	
5321826935e-010	0.955316618163314	0.955316618124509	3.88050702682108e-011	
6609134674e-011	0.955316618143912	0.955316618124509	1.94025906452566e-011	
8304567337e-011	0.955316618134211	0.955316618124509	9.70123981147708e-012	
9152283669e-011	0.95531661812936	0.955316618124509	4.85067541688977e-012	
5761418343e-012	0.955316618126935	0.955316618124509	2.42528219729365e-012	
7880709171e-012	0.955316618125722	0.955316618124509	1.21269660979806e-012	
8940354586e-012	0.955316618125116	0.955316618124509	6.06292793747798e-013	
4701772928e-013	0.955316618124813	0.955316618124509	3.0320190802513e-013	
7350886464e-013	0.955316618124661	0.955316618124509	1.51545442861334e-013	
3675443232e-013	0.955316618124585	0.955316618124509	7.58282325818982e-014	
6837721616e-013	0.955316618124547	0.955316618124509	3.78586051397178e-014	
3418860808e-014	0.955316618124528	0.955316618124509	1.89848137210902e-014	
1709430404e-014	0.955316618124519	0.955316618124509	9.43689570931383e-015	
0854715202e-014	0.955316618124514	0.955316618124509	4.77395900588817e-015	
5427357601e-015	0.955316618124512	0.955316618124509	2.33146835171283e-015	
7136788005e-015	0.955316618124511	0.955316618124509	1.22124532708767e-015	
5683940025e-015	0.95531661812451	0.955316618124509	5.55111512312578e-016	
8419700125e-016	0.95531661812451	0.955316618124509	3.33066907387547e-016	
9209850063e-016	0.955316618124509	0.955316618124509	1.11022302462516e-016	
4604925031e-016	0.955316618124509	0.955316618124509	1.11022302462516e-016	
2302462516e-016	0.955316618124509	0.955316618124509	1.11022302462516e-016	
1512312578e-017	0.955316618124509	0.955316618124509	0	
5756156289e-017	0.955316618124509	0.955316618124509	0	
7878078145e-017	0.955316618124509	0.955316618124509	0	
9390390723e-018	0.955316618124509	0.955316618124509	0	
4695195361e-018	0.955316618124509	0.955316618124509	0	
2347597681e-018	0.955316618124509	0.955316618124509	0	

Figure 2: 64 bit

Source code listing

```
% Nasser M. Abbasi.  Do computation using 32 bit
% Computing derivative of arctan(x) at =sqrt(2) as a function
% of changing h in Taylor approximation.

h = single(1);
M = 26;
X = single(sqrt(2));
f = @(x) single(atan(x));

F1 = f(X);
S  = zeros(25,6,'single');

for k = 1:M
    F2 = f(X+h);
    d  = single(F2-F1);
    r  = single(d/h);
    S(k,1) = k;
    S(k,2) = h;
    S(k,3) = F2;
    S(k,4) = F1;
    S(k,5) = d;
    S(k,6) = r;
    h = single(h/2);
end

format long g
S
```