2^{nd} Order Boundary Value Problem and Application of Finite Difference Method

JEFFREY PABST
UIN: 822004809
AERO 430 – NUMERICAL METHODS
SEPTEMBER 27, 2016

----------------------------------- BOUNDARY VALUE PROBLEM ------------------------------------

Find u = u (x), 0 < x < 1, which satisfies…

1. $\pm u''(x) + k^2u(x) = k^2x \qquad 0 < x < 1$

2. $u(0) = 0 \qquad u(1) = 0$

-- PART A ---

Derive the analytical form of the exact solution, u=u(x), for both cases. Include all the details of the derivation from Boyce & DiPrima. For example, choose $U_{HOM}(x)=e^{\lambda x}$ and find λ for both cases, etc.

1. We begin with the positive case, $u''(x) + k^2u(x) = k^2x \qquad 0 < x < 1$

   We begin by solving the homogenous differential equation.

$$u_{HOM}''(x) + k^2u_{HOM}(x) = 0$$

We assume that $u_{HOM}(x) = e^{\lambda x}$, and it then follows that λ must be a root of the characteristic equation. The characteristic equation is then…

$$\lambda^2 + k^2 = 0$$

Solving this characteristic equation gives us…

$$\lambda = \pm ik$$

Thus the general form of the homogenous solution becomes…

$$u_{HOM}(x) = Ce^{ikx} + De^{-ikx}$$

Now that we have the general form of the homogenous solution, we need to find the general form of the particular solution.

$$u_{PART}''(x) + k^2u_{PART}(x) = k^2x$$

For this to be true, we see that the particular solution must be…

$$u_{PART}(x) = x$$

Now we the have the entirety of our general solution, shown below.

$$u(x) = x + Ce^{ikx} + De^{-ikx}$$

Implementing the two boundary conditions given, we get the following system of equations…

$$\begin{bmatrix} 1 & 1 \\ e^{ik} & e^{-ik} \end{bmatrix}\begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

We can solve for the unknown coefficients using Cramer's Rule, shown below…

$$C = \frac{\begin{vmatrix} 0 & 1 \\ -1 & e^{-ik} \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ e^{ik} & e^{-ik} \end{vmatrix}} \qquad and \qquad D = \frac{\begin{vmatrix} 1 & 0 \\ e^{ik} & -1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ e^{ik} & e^{-ik} \end{vmatrix}}$$

Solving these two expressions gives us…

$$C = \frac{1}{e^{-ik} - e^{ik}} \qquad and \qquad D = \frac{-1}{e^{-ik} - e^{ik}}$$

Plugging back into our general solution gives us…

$$u(x) = x + \frac{e^{ikx}}{e^{-ik} - e^{ik}} - \frac{e^{-ikx}}{e^{-ik} - e^{ik}} = x + \frac{e^{ikx} - e^{-ikx}}{e^{-ik} - e^{ik}}$$

Simplifying…

$$\boxed{u(x) = x - \frac{\sin(kx)}{\sin(k)}}$$

Note: A unique solution only exists when sin(k) ≠ 0, meaning that k ≠ nπ

2. We now solve the negative case, $-u''(x) + k^2u(x) = k^2x \qquad 0 < x < 1$

   We begin by solving the homogenous differential equation.

$$-u_{HOM}''(x) + k^2u_{HOM}(x) = 0$$

We assume that $u_{HOM}(x) = e^{\lambda x}$, and it then follows that λ must be a root of the characteristic equation. The characteristic equation is then…

$$\lambda^2 - k^2 = 0$$

Solving this characteristic equation gives us…

$$\lambda = \pm k$$

Thus the general form of the homogenous solution becomes…

$$u_{HOM}(x) = Ce^{kx} + De^{-kx}$$

Now that we have the general form of the homogenous solution, we need to find the general form of the particular solution.

$$u_{PART}''(x) + k^2 u_{PART}(x) = k^2 x$$

For this to be true, we see that the particular solution must be…

$$u_{PART}(x) = x$$

Now we the have the entirety of our general solution, shown below.

$$u(x) = x + Ce^{kx} + De^{-kx}$$

Implementing the two boundary conditions given, we get the following system of equations…

$$\begin{bmatrix} 1 & 1 \\ e^{ik} & e^{-ik} \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

We can solve for the unknown coefficients using Cramer's Rule, shown below…

$$C = \frac{\begin{vmatrix} 0 & 1 \\ -1 & e^{-k} \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ e^{k} & e^{-k} \end{vmatrix}} \qquad and \qquad D = \frac{\begin{vmatrix} 1 & 0 \\ e^{k} & -1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ e^{k} & e^{-k} \end{vmatrix}}$$

Solving these two expressions gives us…

$$C = \frac{1}{e^{-k} - e^{k}} \qquad and \qquad D = \frac{-1}{e^{-k} - e^{k}}$$

Plugging back into our general solution gives us…

$$u(x) = x + \frac{e^{kx}}{e^{-k} - e^{k}} - \frac{e^{-kx}}{e^{-k} - e^{k}} = x + \frac{e^{kx} - e^{-kx}}{e^{-k} - e^{k}}$$

Simplifying…

$$\boxed{u(x) = x - \frac{\sinh(kx)}{\sinh(k)}}$$

Note: A unique solution always exists for k ≠ 0

--- PART B ---

Use the finite difference method to compute the approximate solution, $u_{\Delta x}(x)$ for
$\Delta x = ½, ¼$, etc. and $k^2 = 1, 10, 100$, etc.

For this part, I ran both the positive and negative cases with iterations that included Δx values of ½, ¼, 1/8, 1/16, etc. and k^2 values of 1, 10, 100, and 1000. The first step was to partition the domain into a number of sub intervals which was governed by our value of Δx. The next step was to express the differential operator $\frac{d^2U_{\Delta x}}{dx^2}$ in discrete form. This was accomplished using finite difference approximations to the differential operators. For this problem, we use the approximation below.

$$U_i'' = \frac{U_{i+1} - 2U_i + U_{i-1}}{\Delta x^2}$$

We can then use this relationship to simplify the original differential equations for both the positive and negative cases. For the positive case, we get…

$$U_{i-1} + U_i(k^2\Delta x^2 - 2) + U_{i+1} = x_i(k^2\Delta x^2)$$

And for the negative case we get…

$$-U_{i-1} + U_i(k^2\Delta x^2 + 2) - U_{i+1} = x_i(k^2\Delta x^2)$$

With the implementation of our boundary conditions, these equations give us a system of equations which we can use to solve the interior equations. For the positive case we get…

$$\begin{bmatrix} k^2\Delta x^2 - 2 & 1 & 0 \\ 1 & k^2\Delta x^2 - 2 & 1 \\ 0 & 1 & k^2\Delta x^2 - 2 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} x_1(k^2\Delta x^2) \\ x_2(k^2\Delta x^2) \\ x_3(k^2\Delta x^2) \end{bmatrix}$$

For the negative case we get…

$$\begin{bmatrix} k^2\Delta x^2 + 2 & -1 & 0 \\ -1 & k^2\Delta x^2 + 2 & -1 \\ 0 & -1 & k^2\Delta x^2 + 2 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} x_1(k^2\Delta x^2) \\ x_2(k^2\Delta x^2) \\ x_3(k^2\Delta x^2) \end{bmatrix}$$

Note that for this problem the matrix on the left hand side will always be tridiagonal. Therefore, we are able to solve the U approximations. In my code, I solved this system using a variation of the Thomas Algorithm for tridiagonal matrices.

Now having solved for our approximate solution, I was able to plot the results versus the exact solution. This was repeated for both the positive and negative case and for all values of Δx and all values of k^2. The results are shown below.

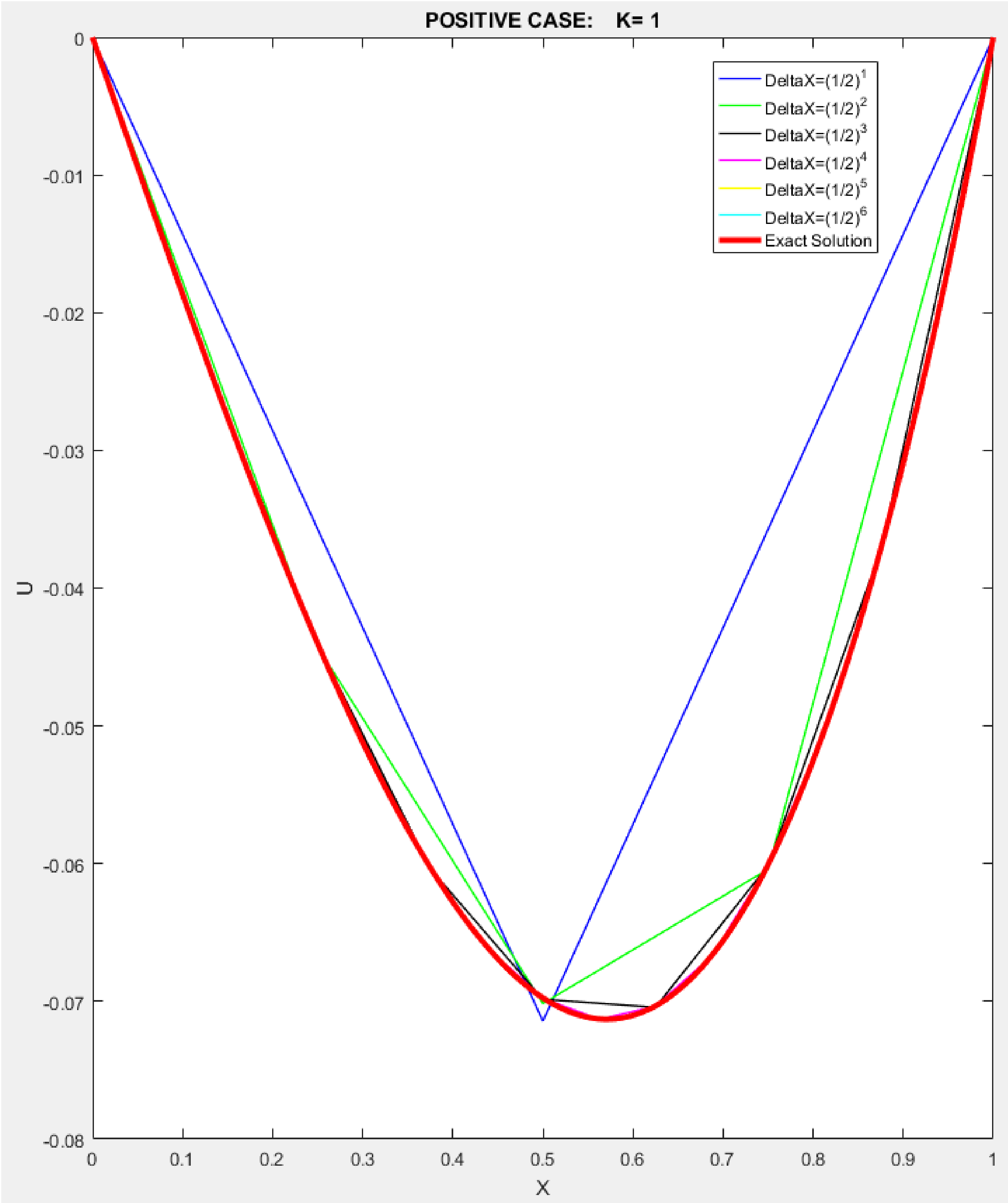

Figure 1: Positive Case, K=1

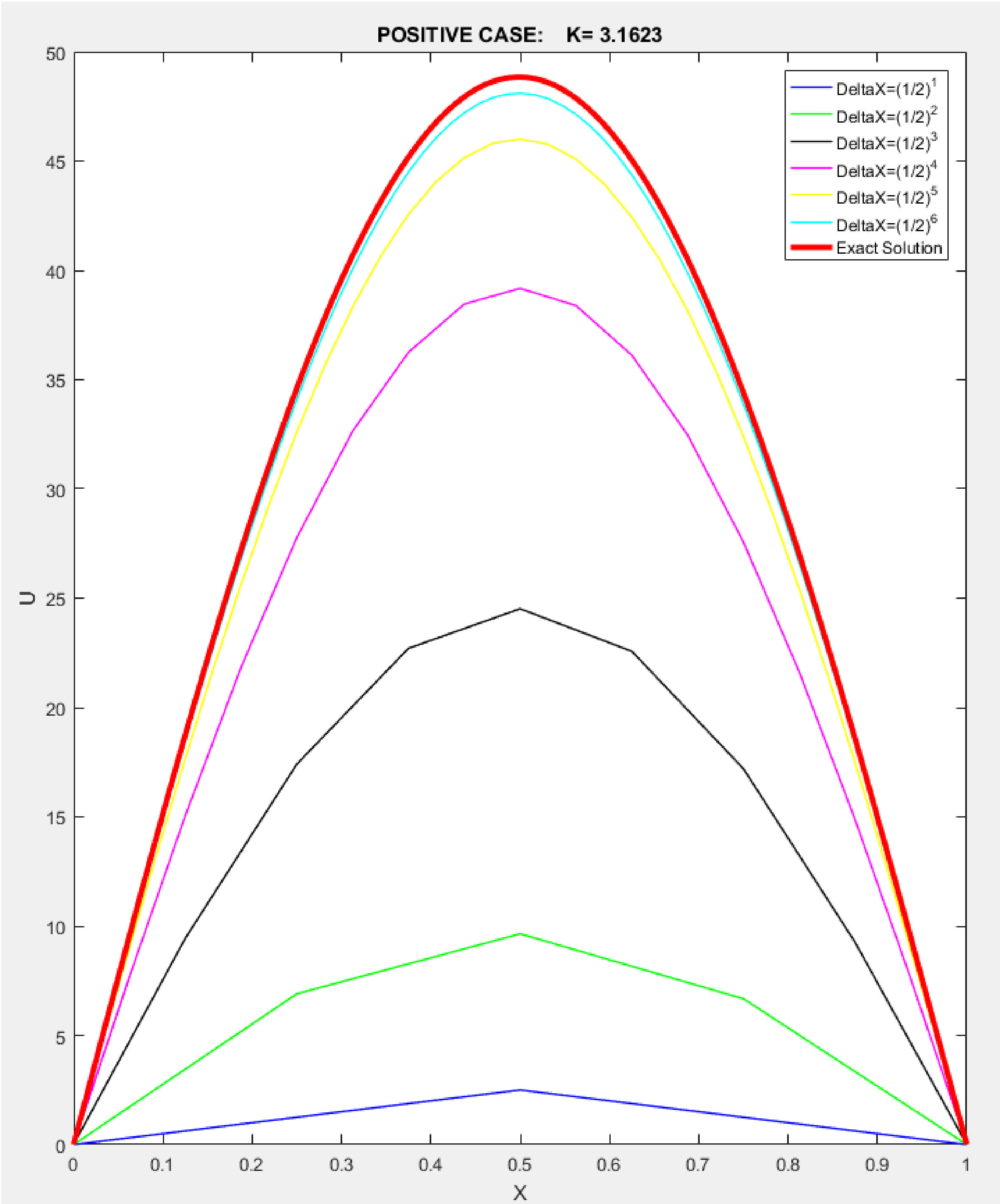

Figure 2: Positive Case, K=sqrt(10)

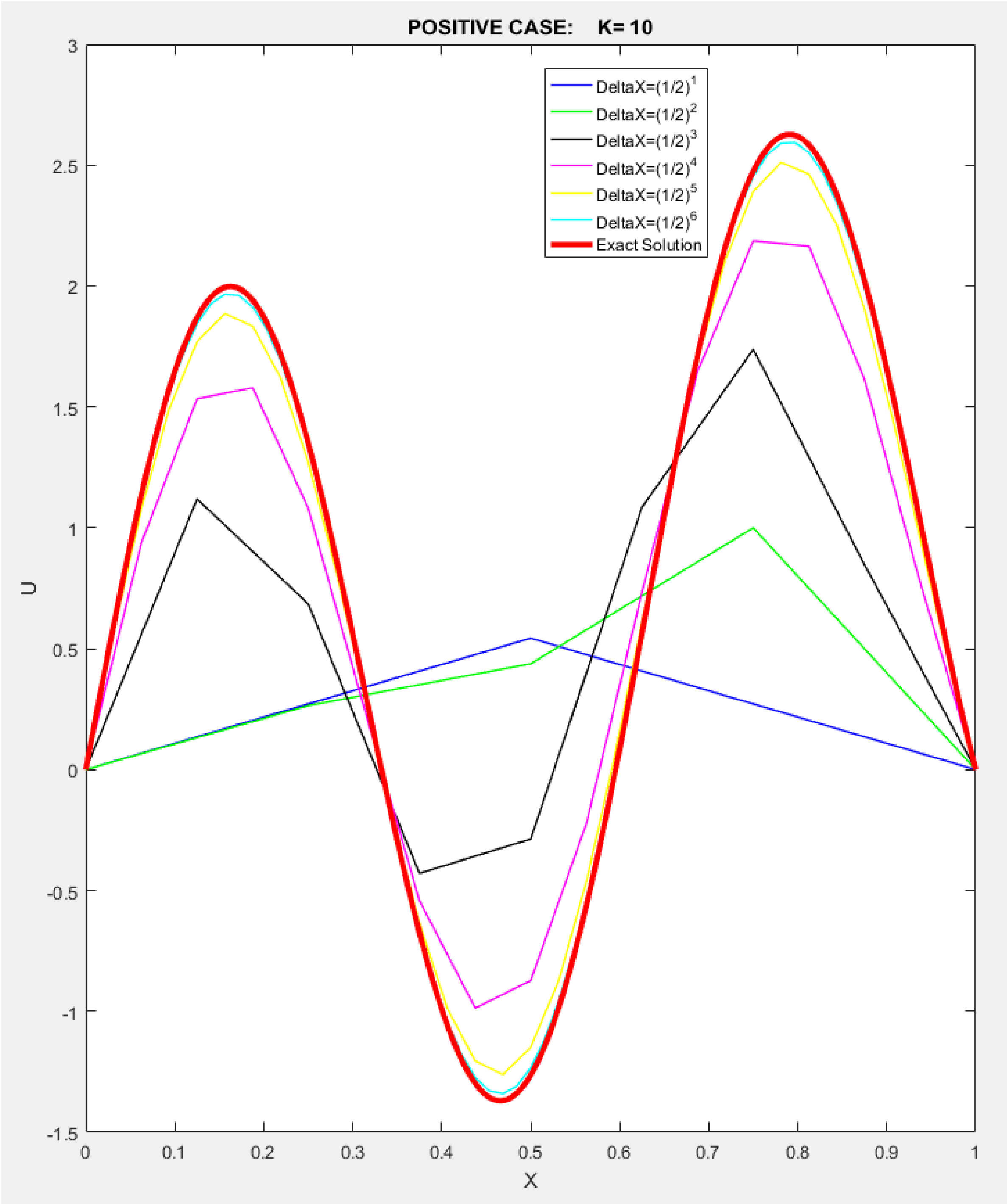

Figure 3: Positive Case, K=10

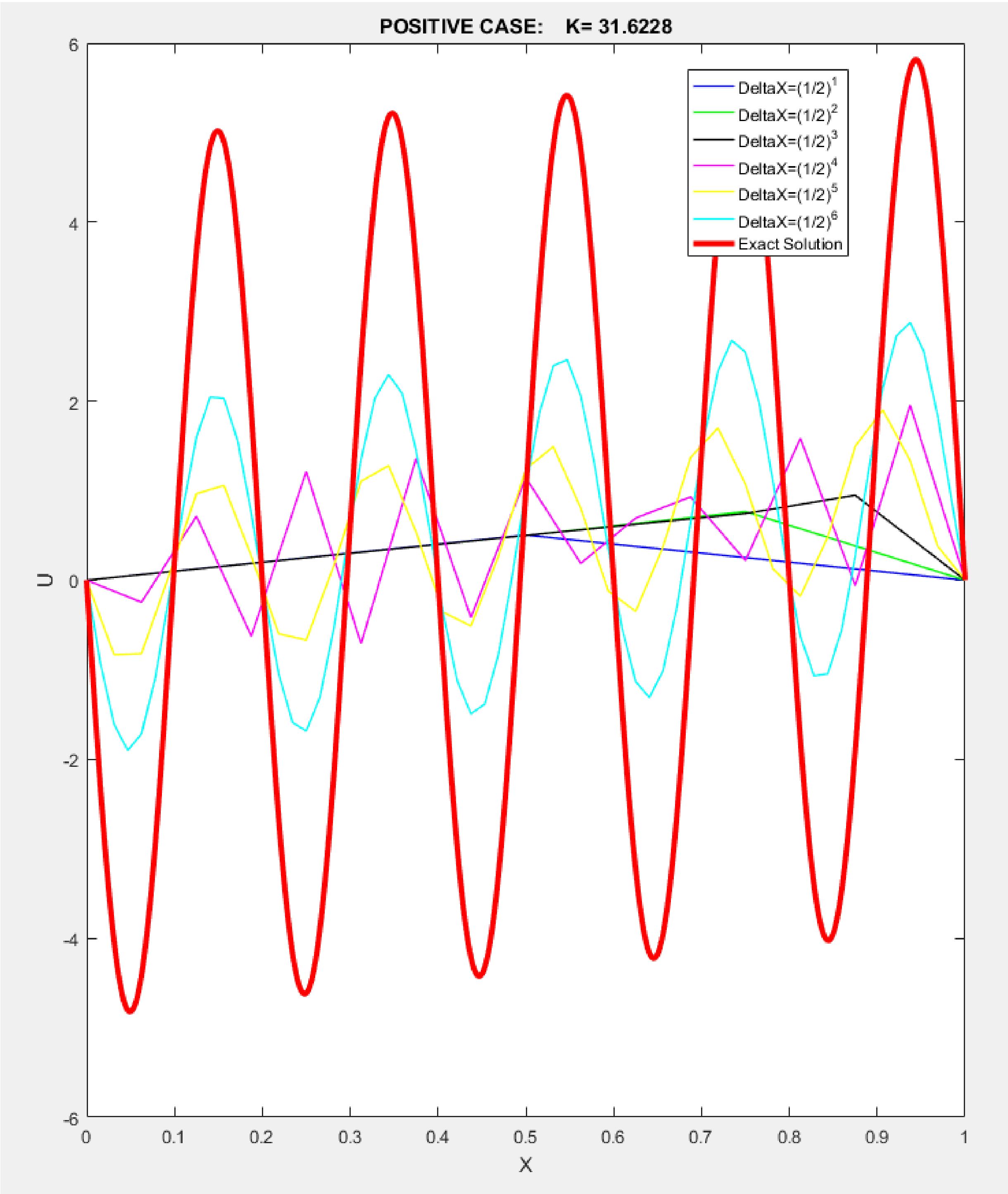

Figure 4: Positive Case, K=sqrt(1000)

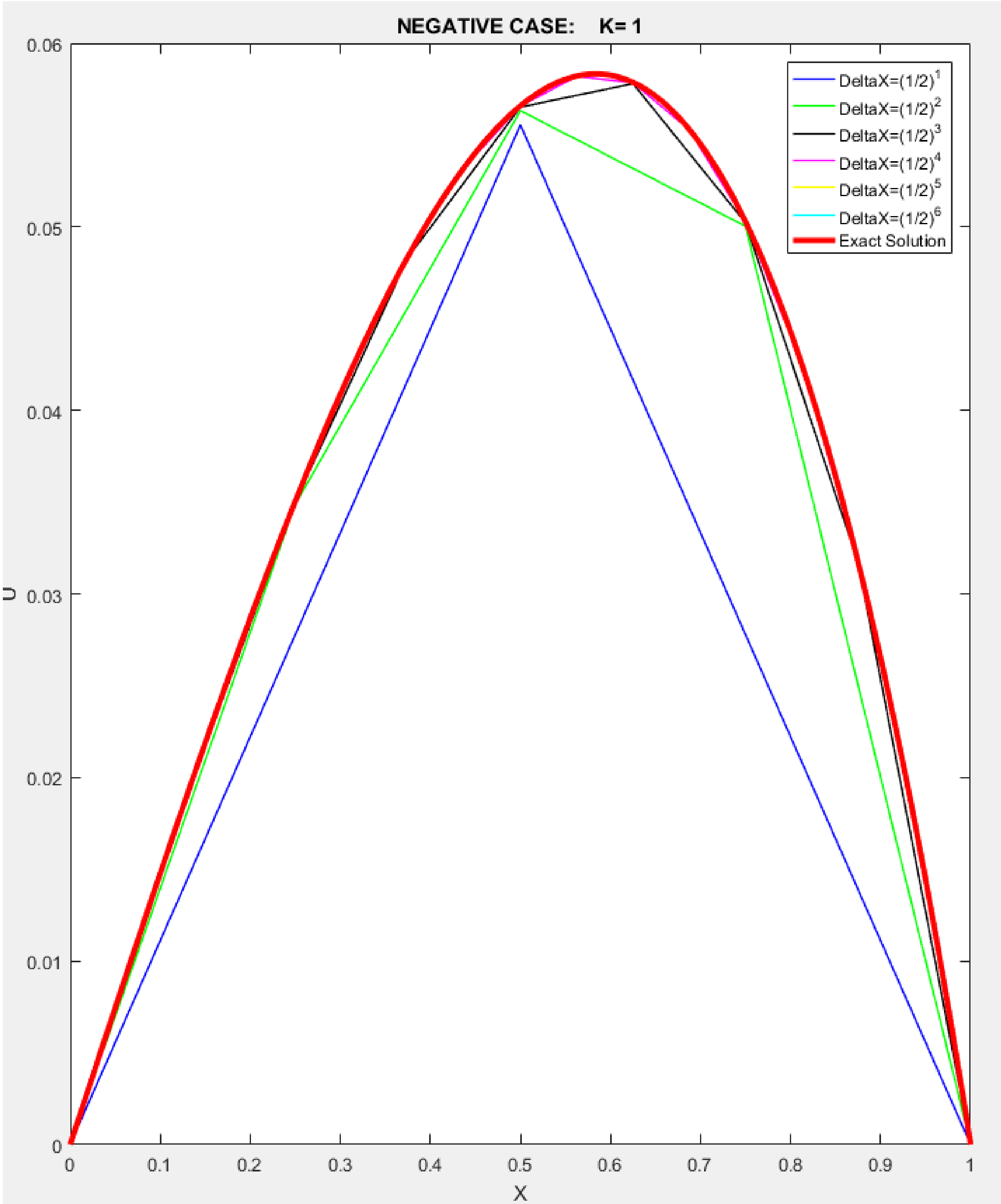

Figure 5: Negative Case, K=1

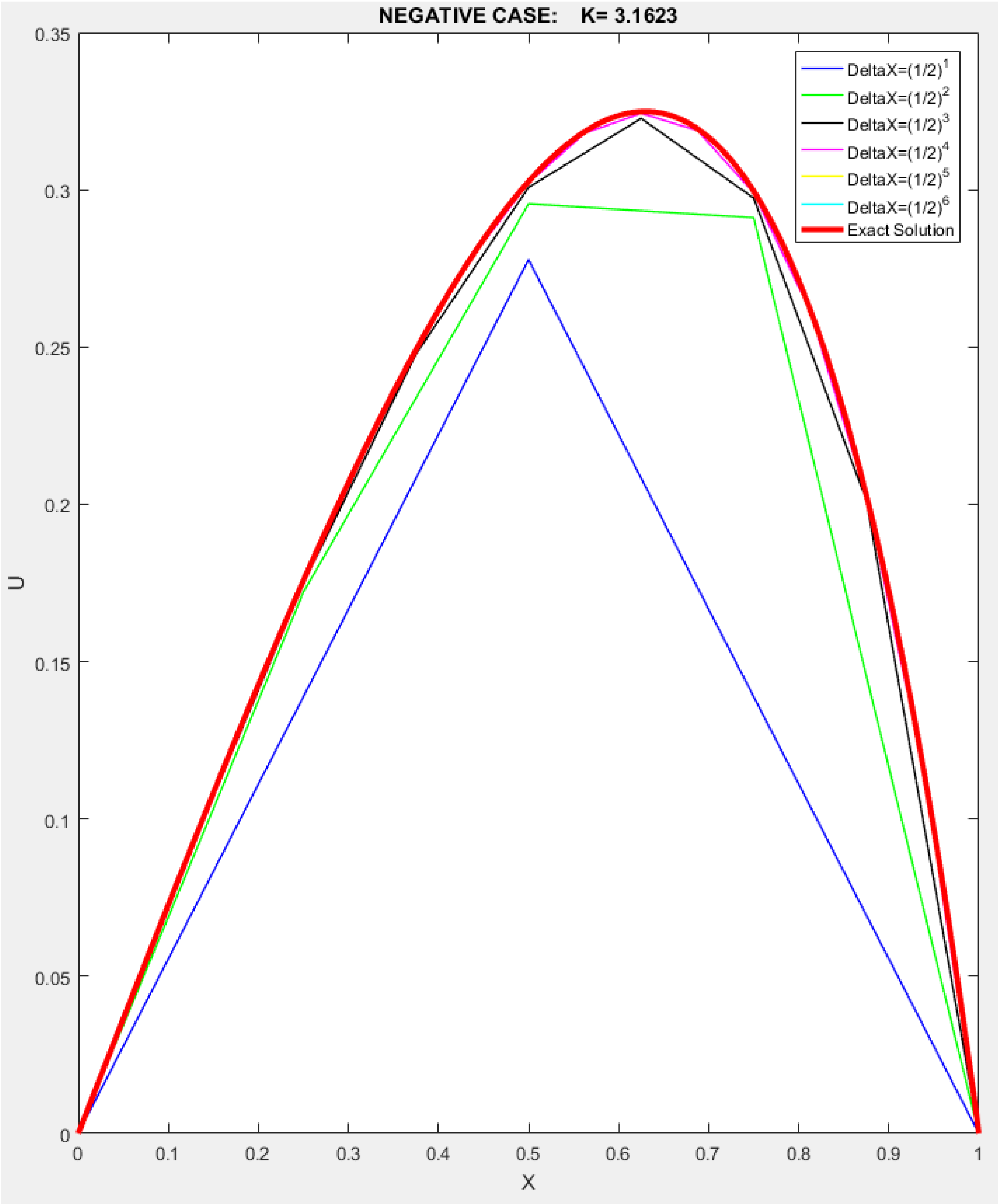

Figure 6: Negative Case, K=sqrt(10)

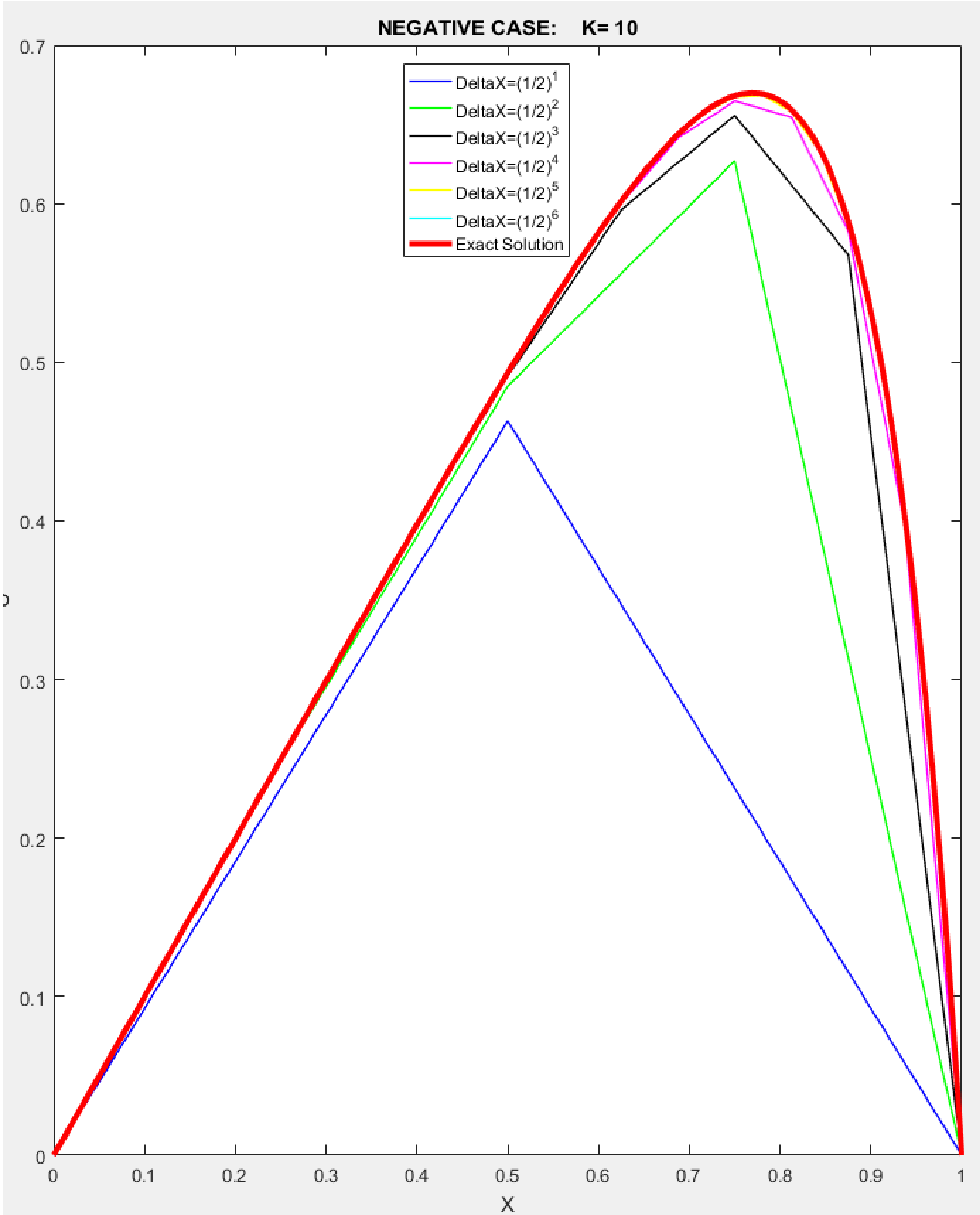

Figure 7: Negative Case, K=10

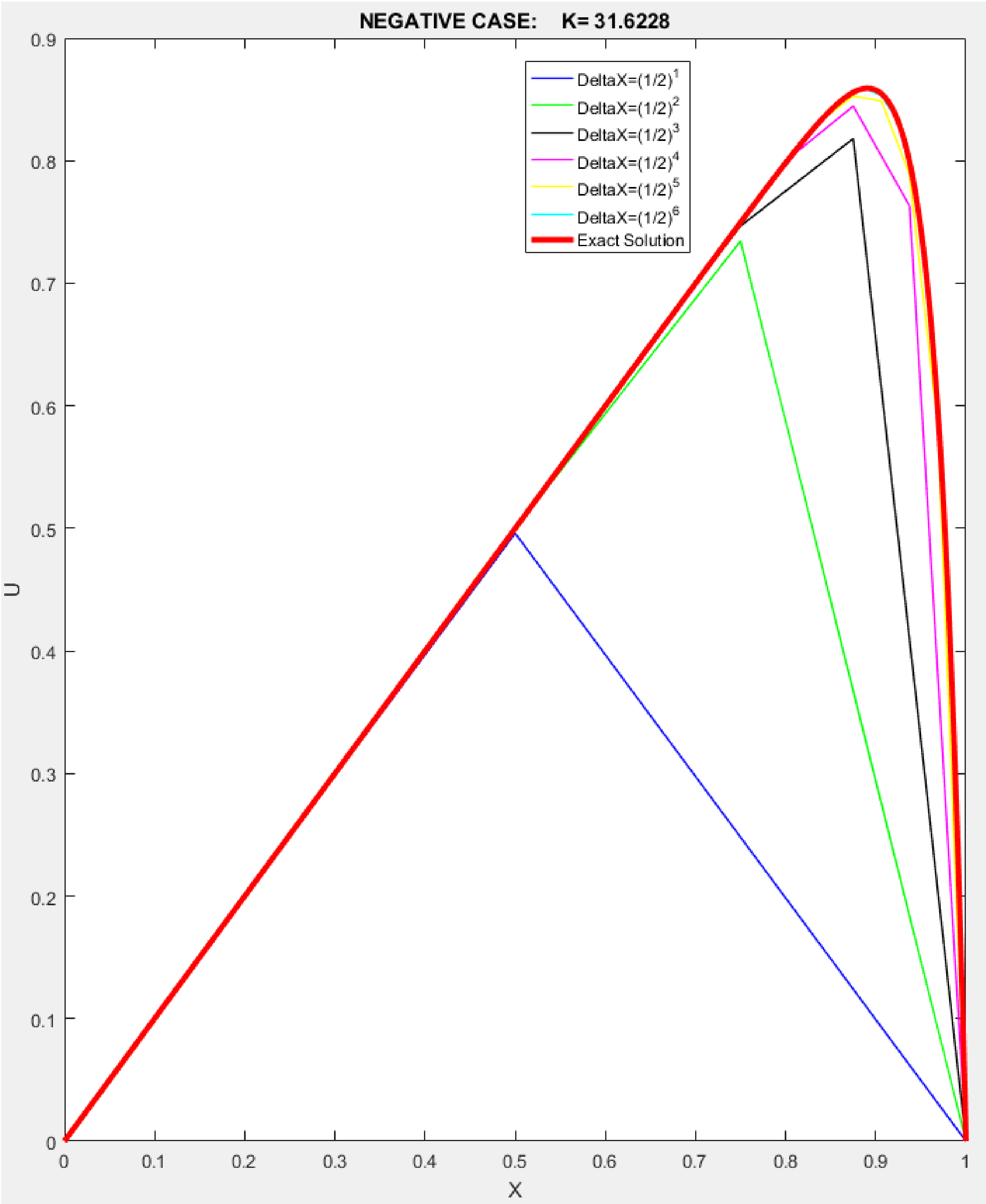

Figure 8: Negative Case, K=sqrt(1000)

--- PART C ---

Compute the finite difference value $u'_{\Delta x}(1)$ for $u'(1)$, % relative error

To calculate the finite difference value $u'_{\Delta x}(1)$ we used the approximation below.

$$U_{\Delta x}'(1) = \frac{U_N - U_{N-1}}{\Delta x}$$

To find the exact value for the derivative at x=1, I found the analytical solutions for the derivatives of the original differential equations. For the positive case, we get…

$$U'(x) = 1 - \frac{k cos(kx)}{\sin(k)}$$

And for the negative case we get…

$$U'(x) = 1 - \frac{k cosh(kx)}{\sinh(k)}$$

Now knowing both the approximate value and the exact value, we can calculate the percent relative error using the following formula.

$$|e_{REL}'(1)| = \frac{|U'(1) - U_{\Delta x}'(1)|}{|U'(1)|} * 100$$

Again, this value was calculated for both the positive and negative cases and for all values of Δx and k^2. I then plotted the percent relative error versus the Δx on a log-log plot.

For this case of percent relative error, we derived in class that the rate of convergence should be equal to 1. To support this, in each case, I also plotted a linear fit with a slope of 1 through the last few data points (represented by the orange line). Because the lines coincide almost perfectly, it shows that the percent relative error always converged to have a slope of 1, or to put it more academically, the rate of convergence was indeed equal to 1, thus confirming the accuracy of the code.

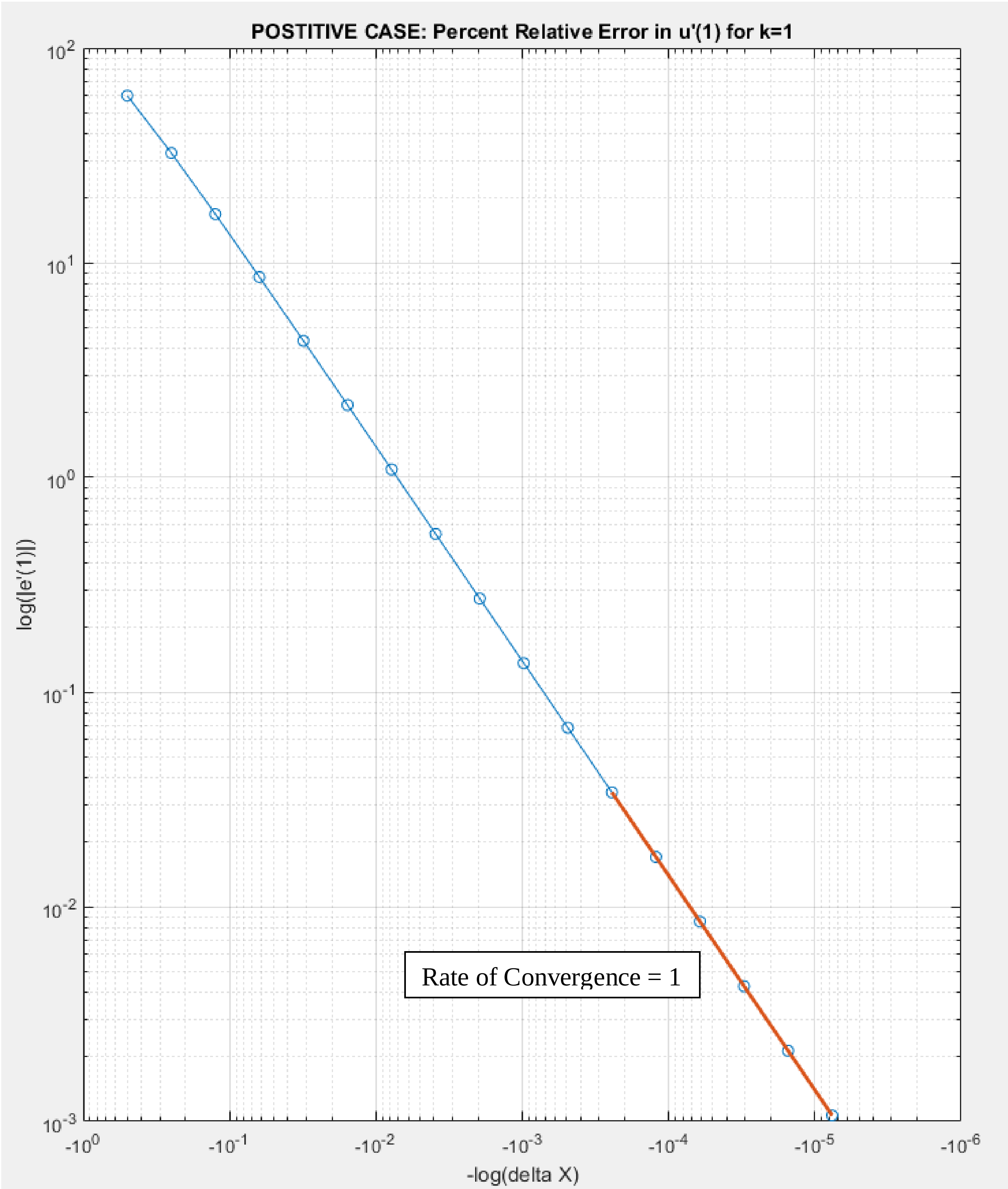

Figure 9: Percent Relative Error

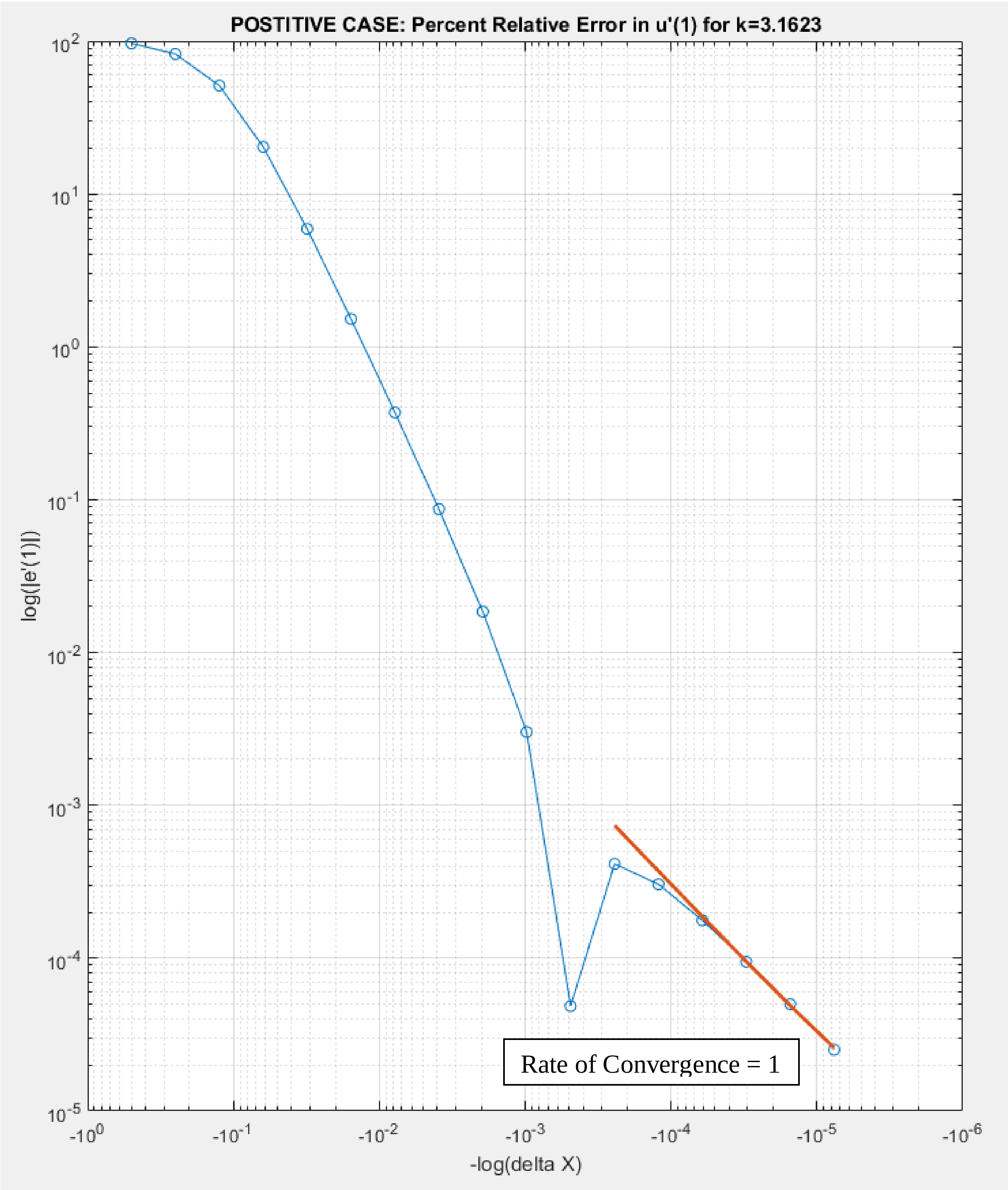

Figure 10: Percent Relative Error

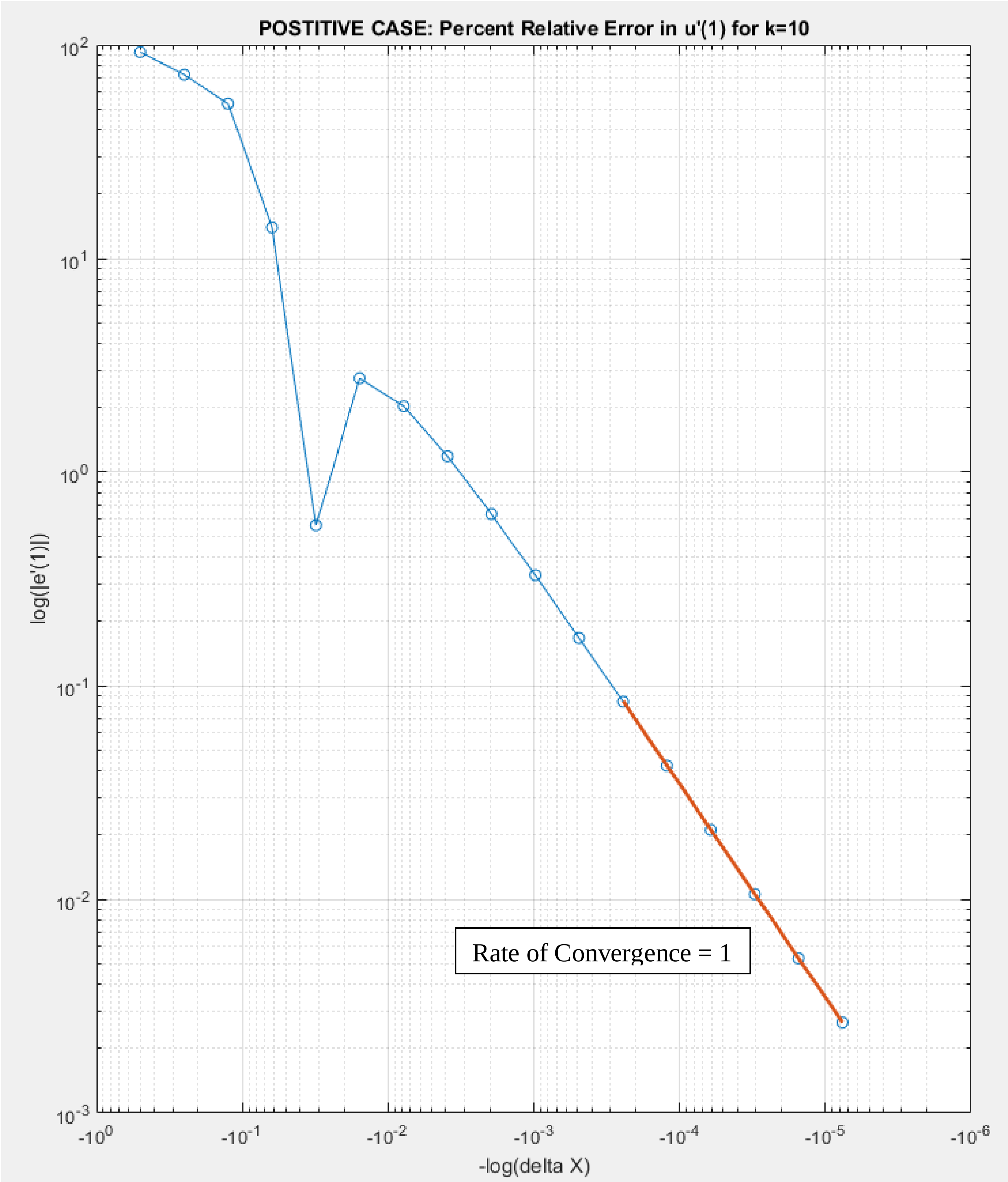

Figure 11: Percent Relative Error

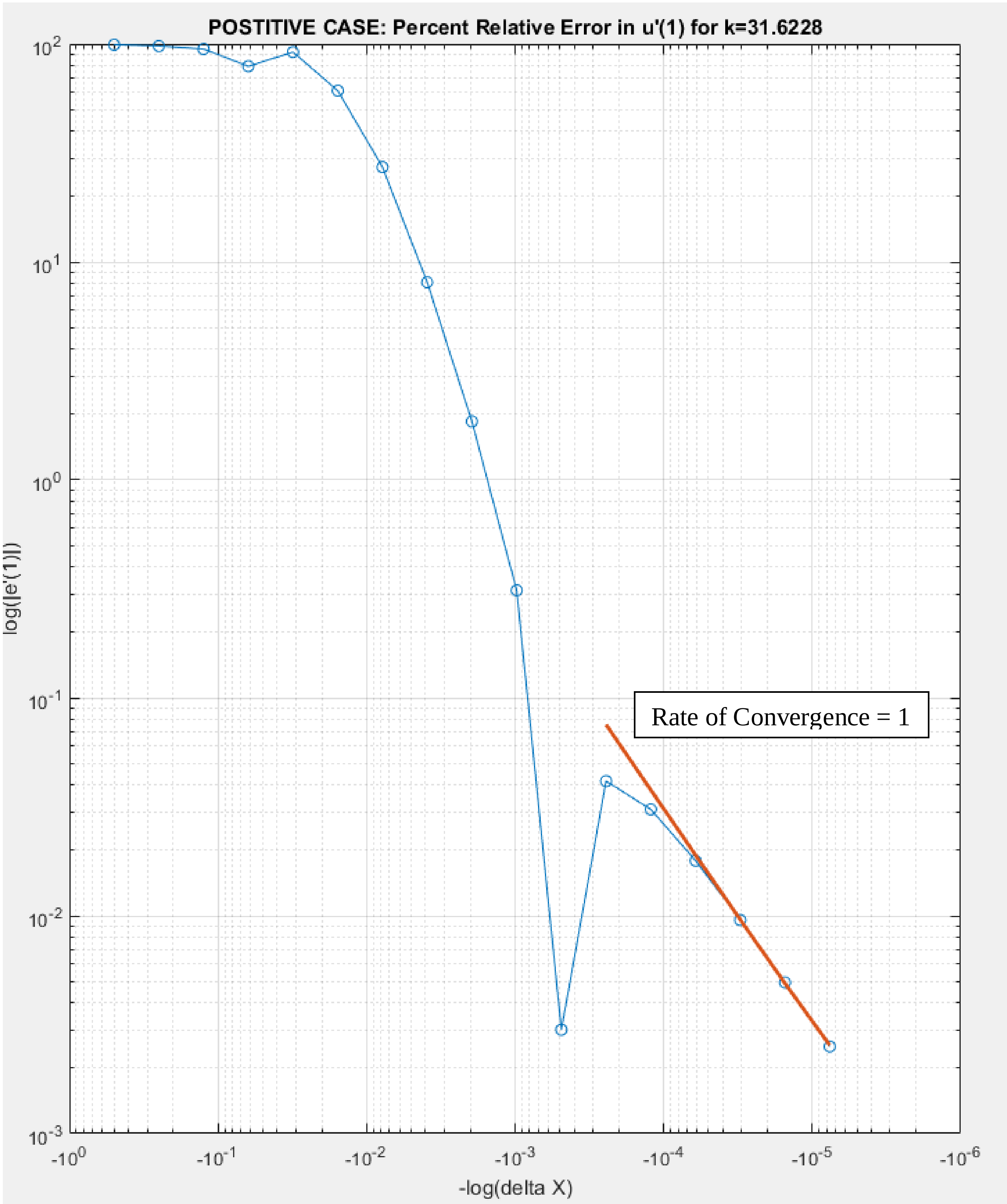

Figure 12: Percent Relative Error

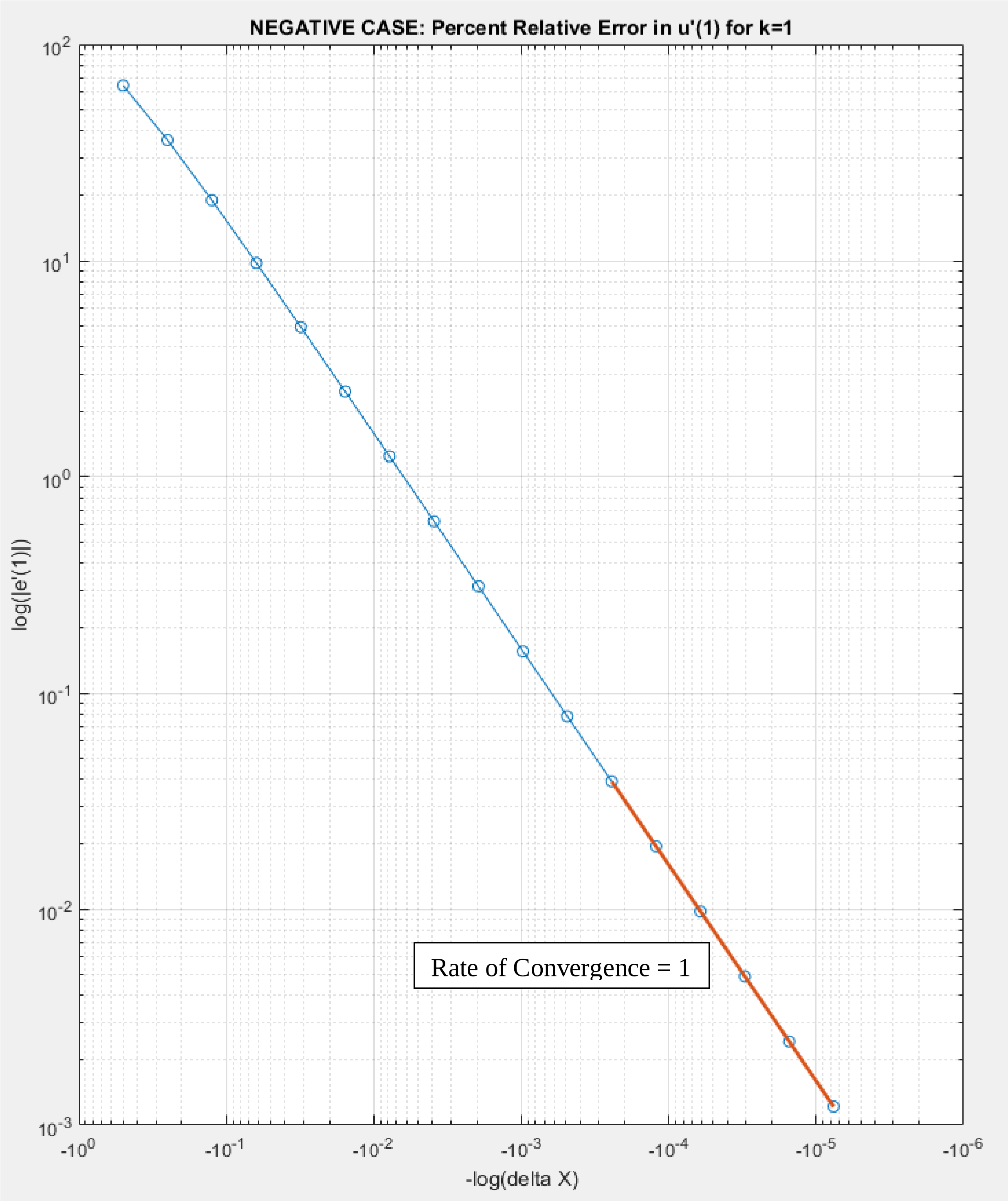

Figure 13: Percent Relative Error

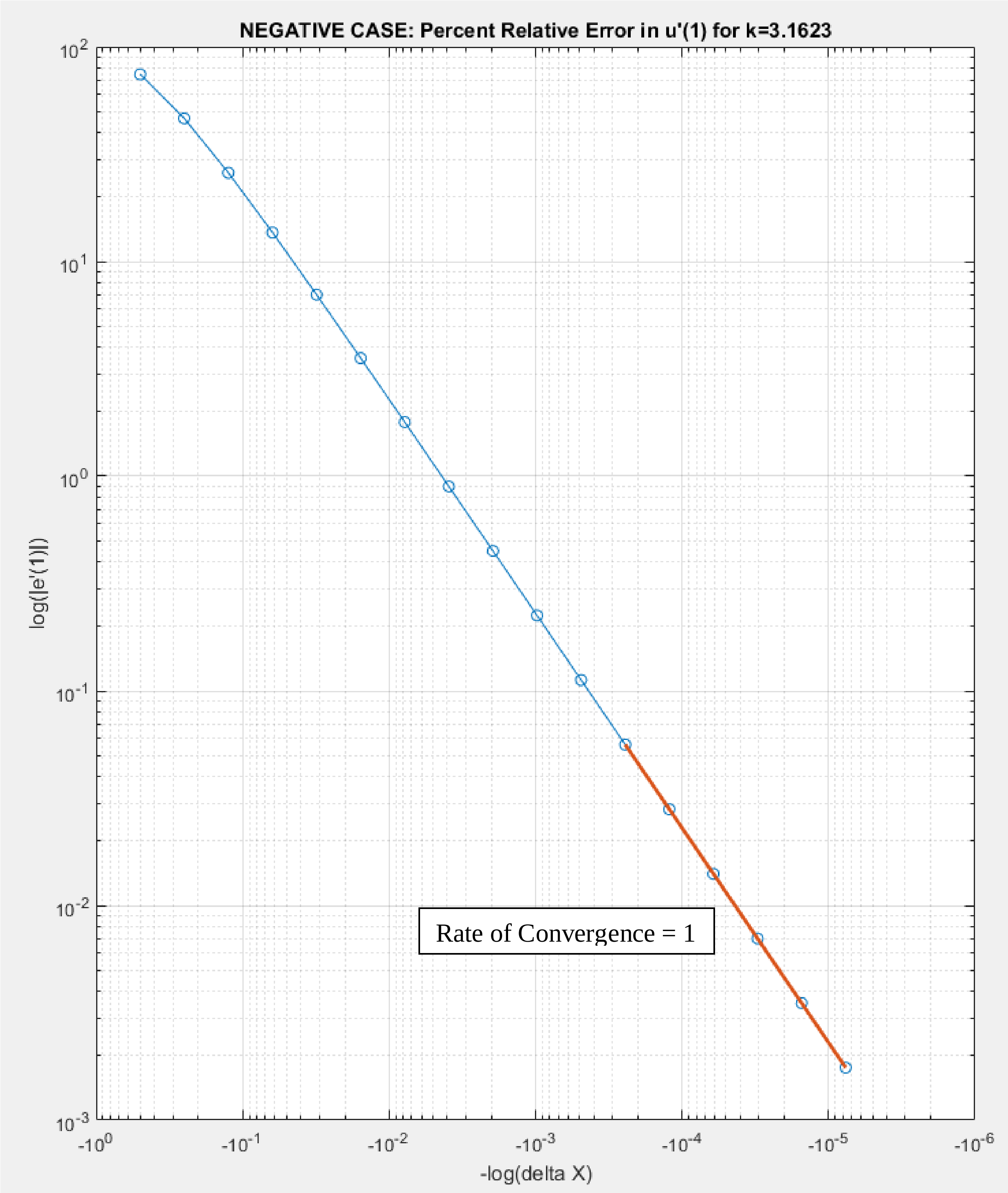

Figure 14: Percent Relative Error

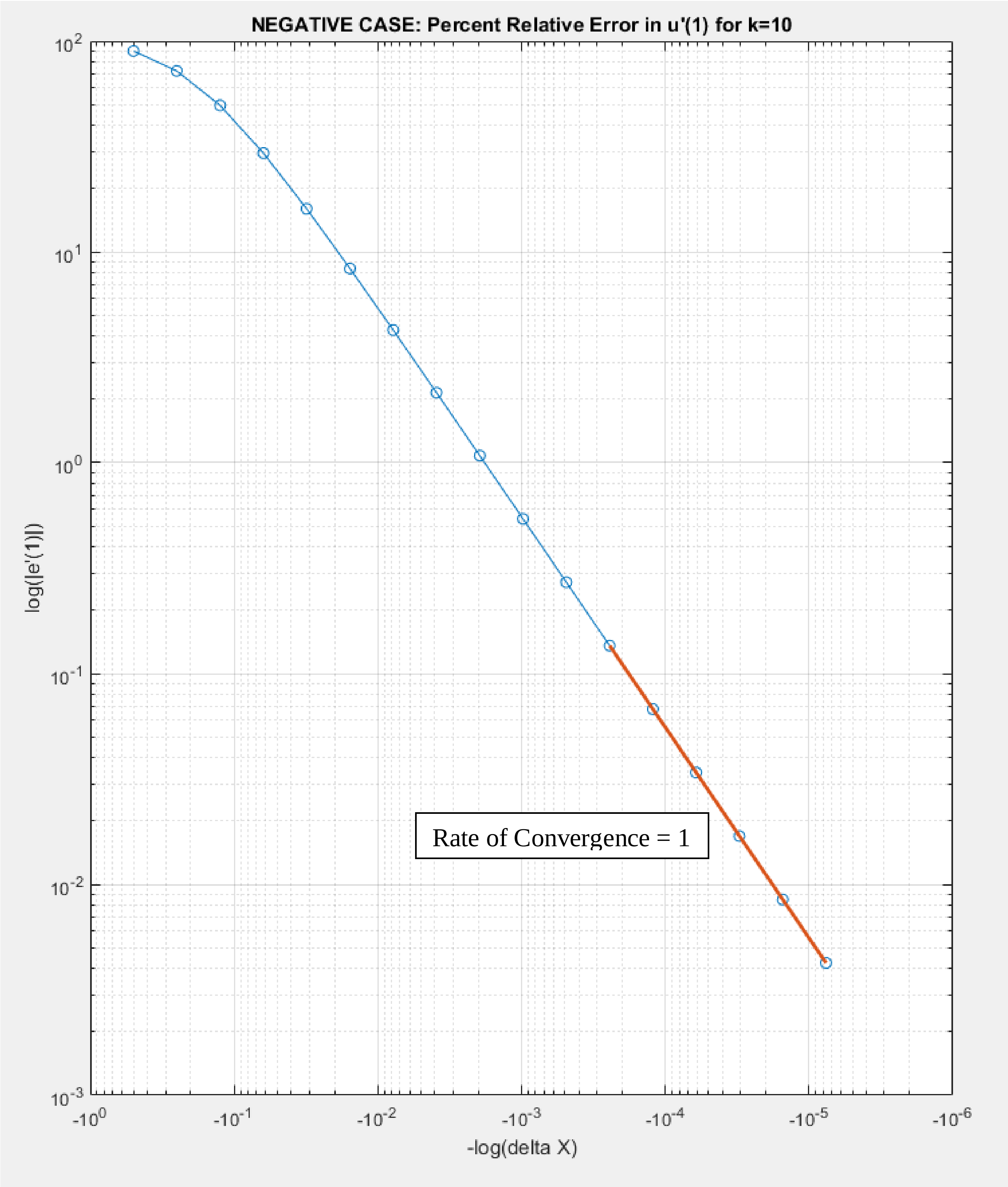

Figure 15: Percent Relative Error

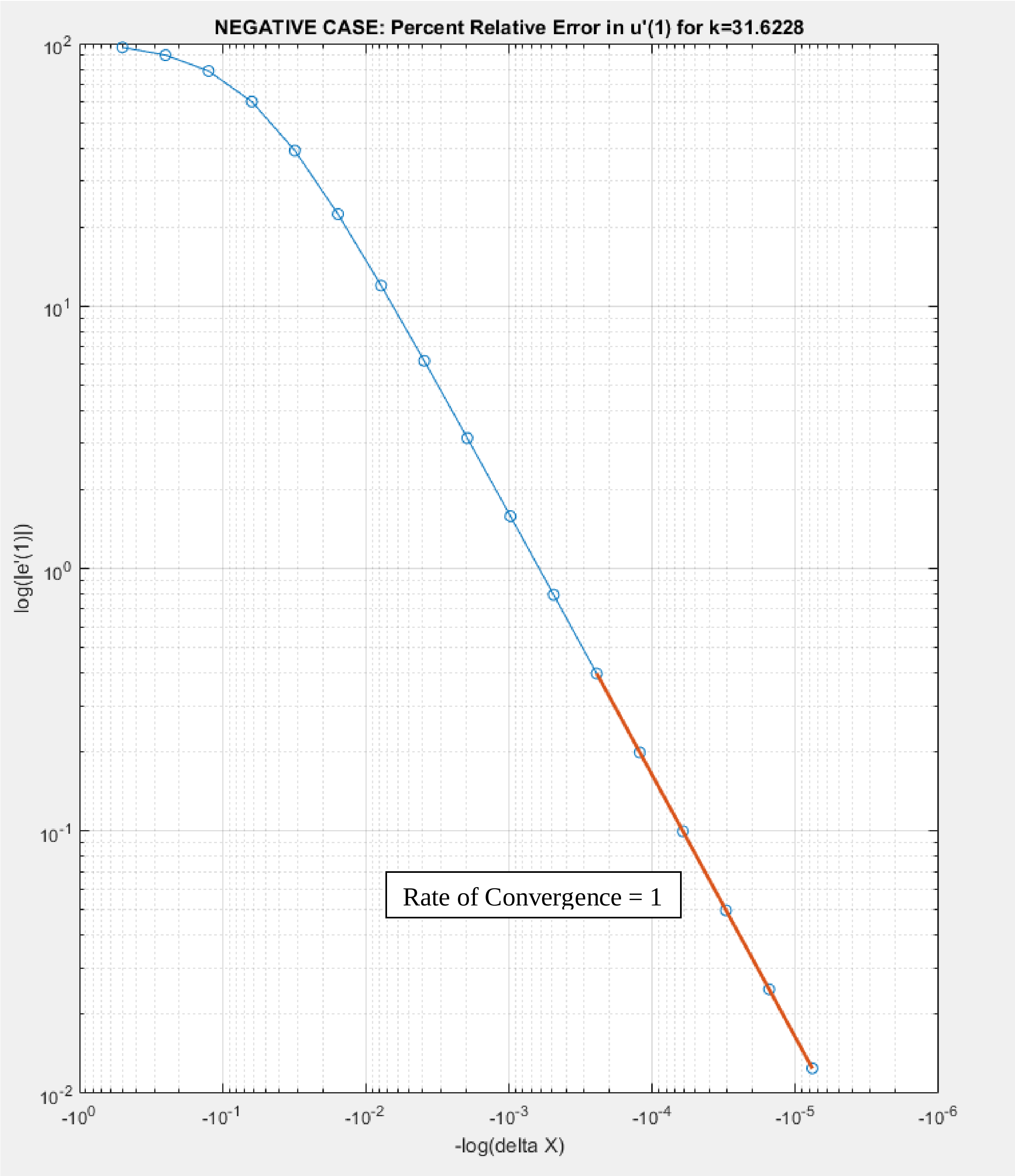

Figure 16: Percent Relative Error

-- DISCUSSION ON ERROR --

As you can see in all cases, the rate of convergence was equal to 1. However, in certain cases such as Figures 10,11 and 12, you can see a jump in the error before it has completely converged. The conclusion from this result is that the convergence for the positive case must not be monotomic. Although at first glance this jump in error can seem like the start of round off error and what would be eventual divergence of the error values, increasing the fineness of the mesh after that point corrected the results and appropriately converged to a slope of 1.

-------------------------- ERROR USING HIGHER ORDER METHODS --------------------------

We used higher order methods to compute even more accurate finite difference approximations for every value of Δx. These values, such as with the original finite difference values, were also plotted on a log-log plot.

For the higher order method, we derived in class that the rate of convergence should be equal to 2. To support this, in each case, I also plotted a linear fit with a slope of 2 through the last few data points (represented by the orange line). Because the lines coincide almost perfectly, it shows that the percent relative error always converged to have a slope of 2, or to put it more academically, the rate of convergence was indeed equal to 2, thus confirming the accuracy of the code.

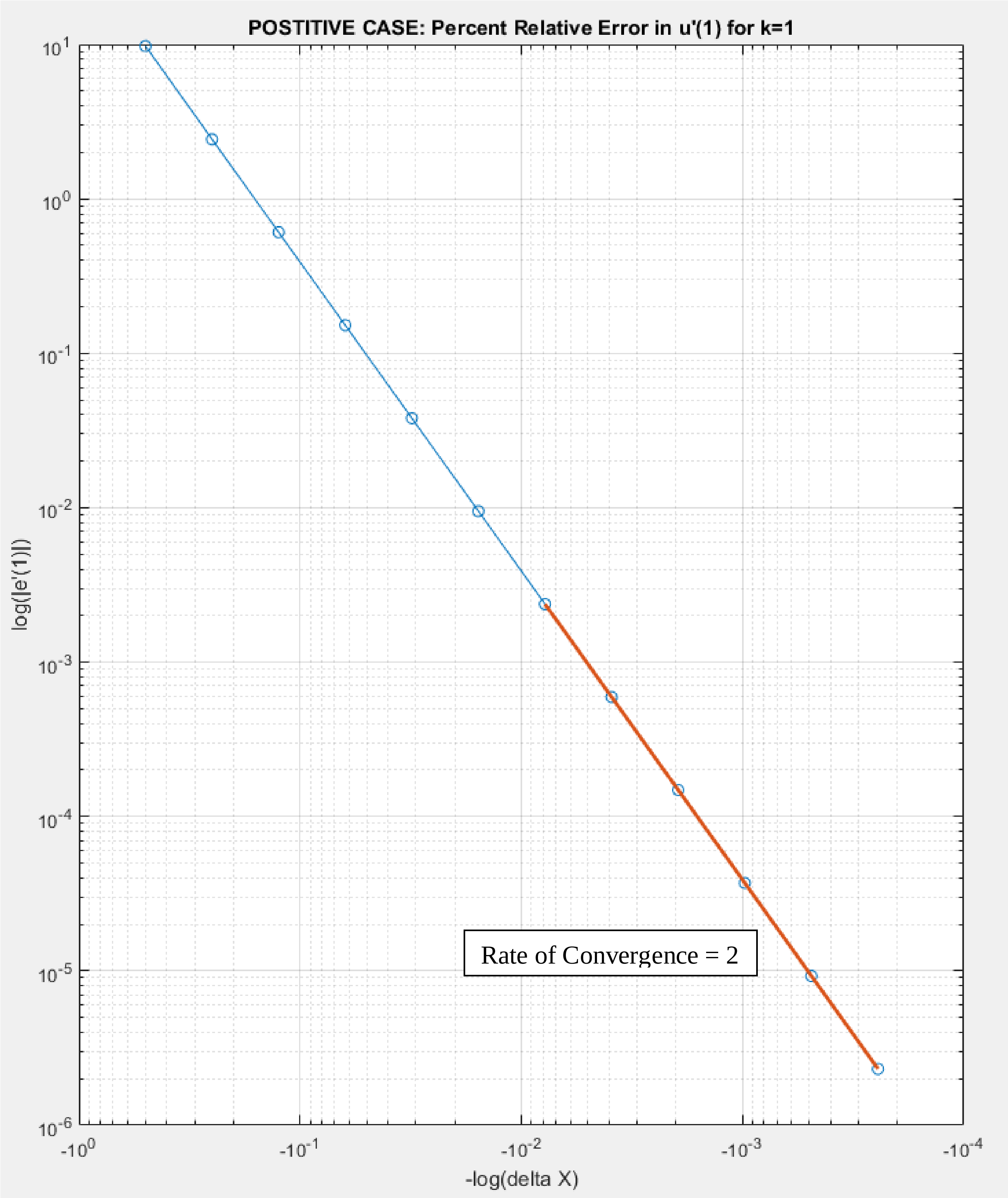

Figure 17: Error Using Higher Order Methods

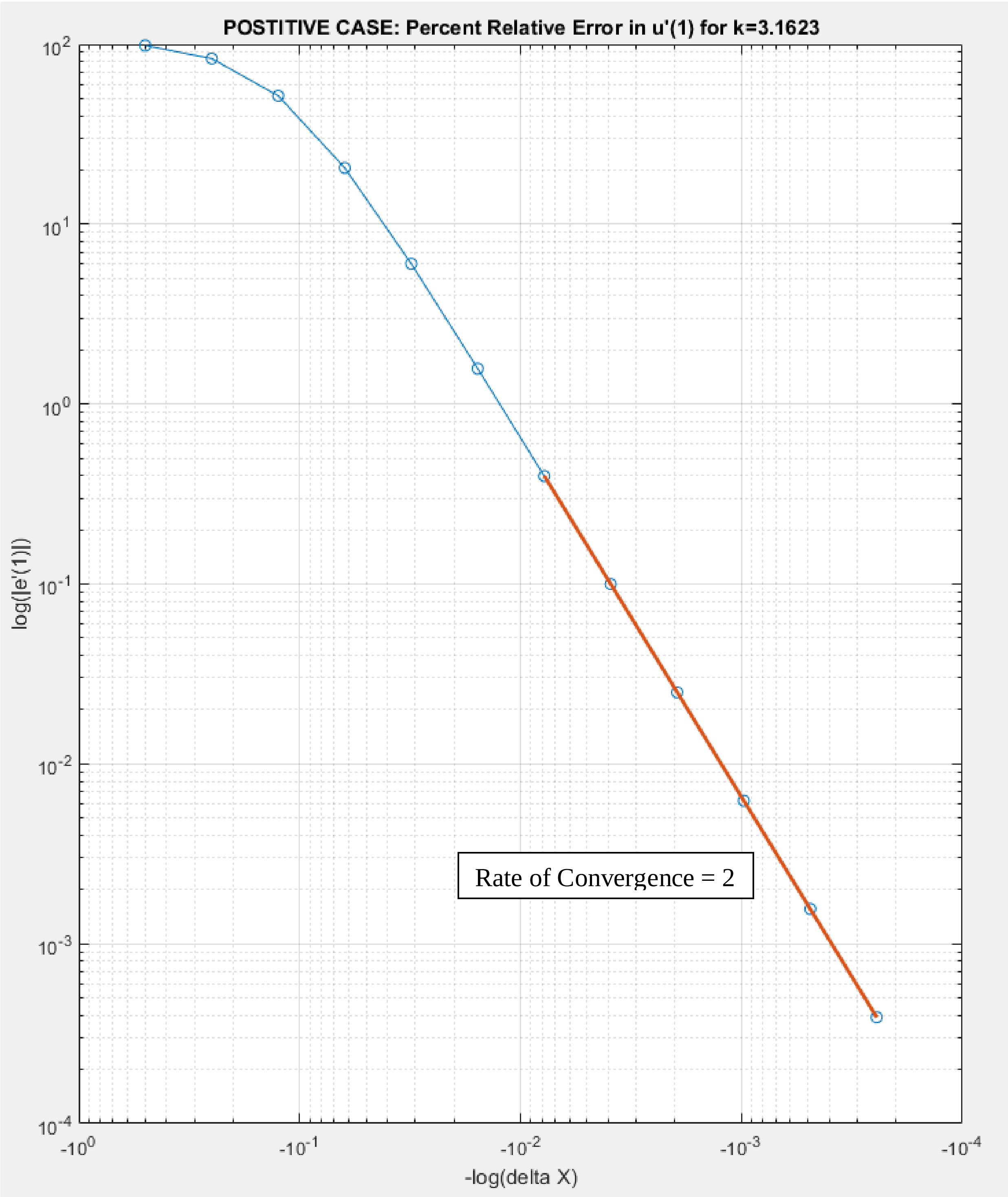

Figure 18: Error Using Higher Order Methods

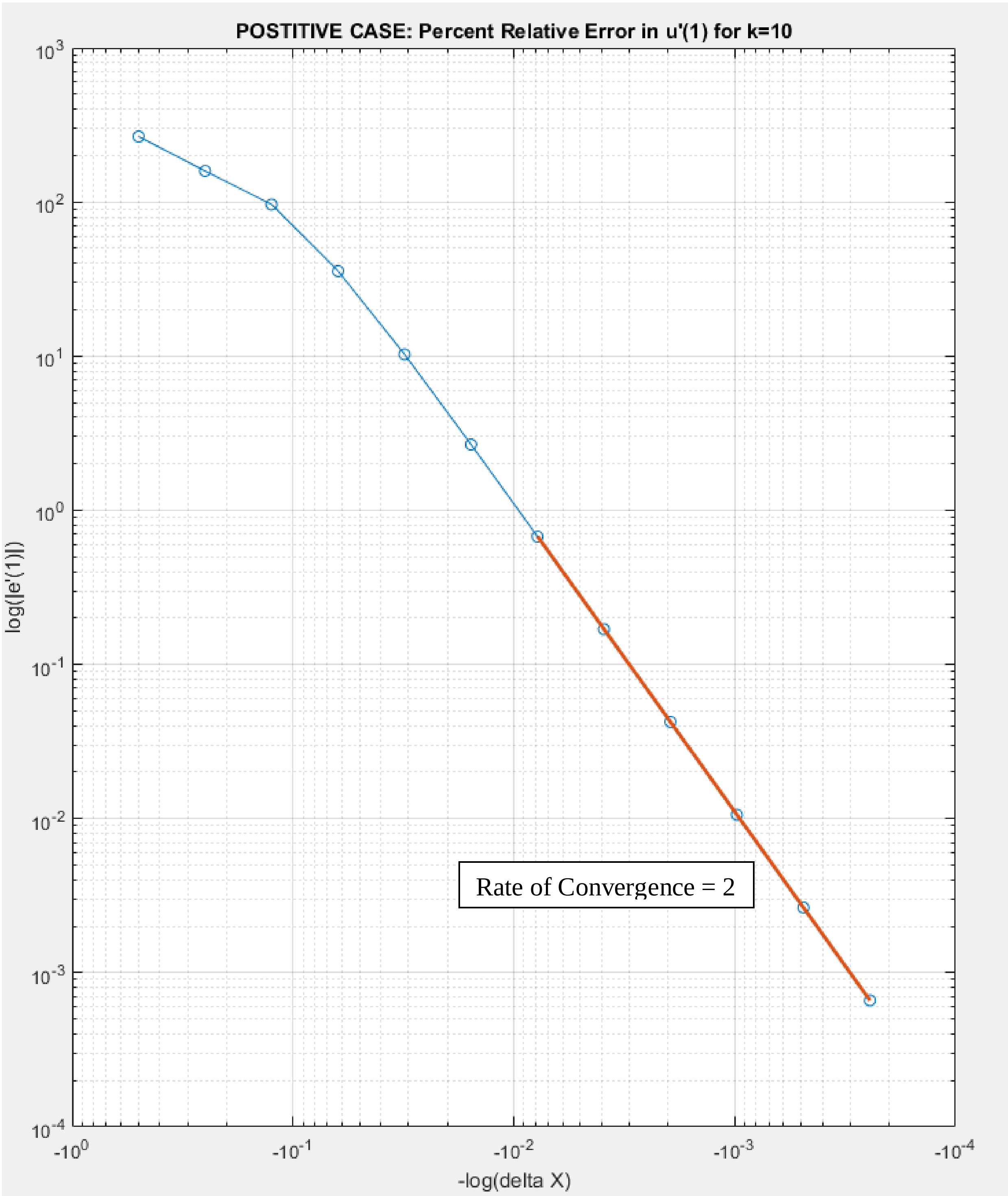

Figure 19: Error Using Higher Order Methods

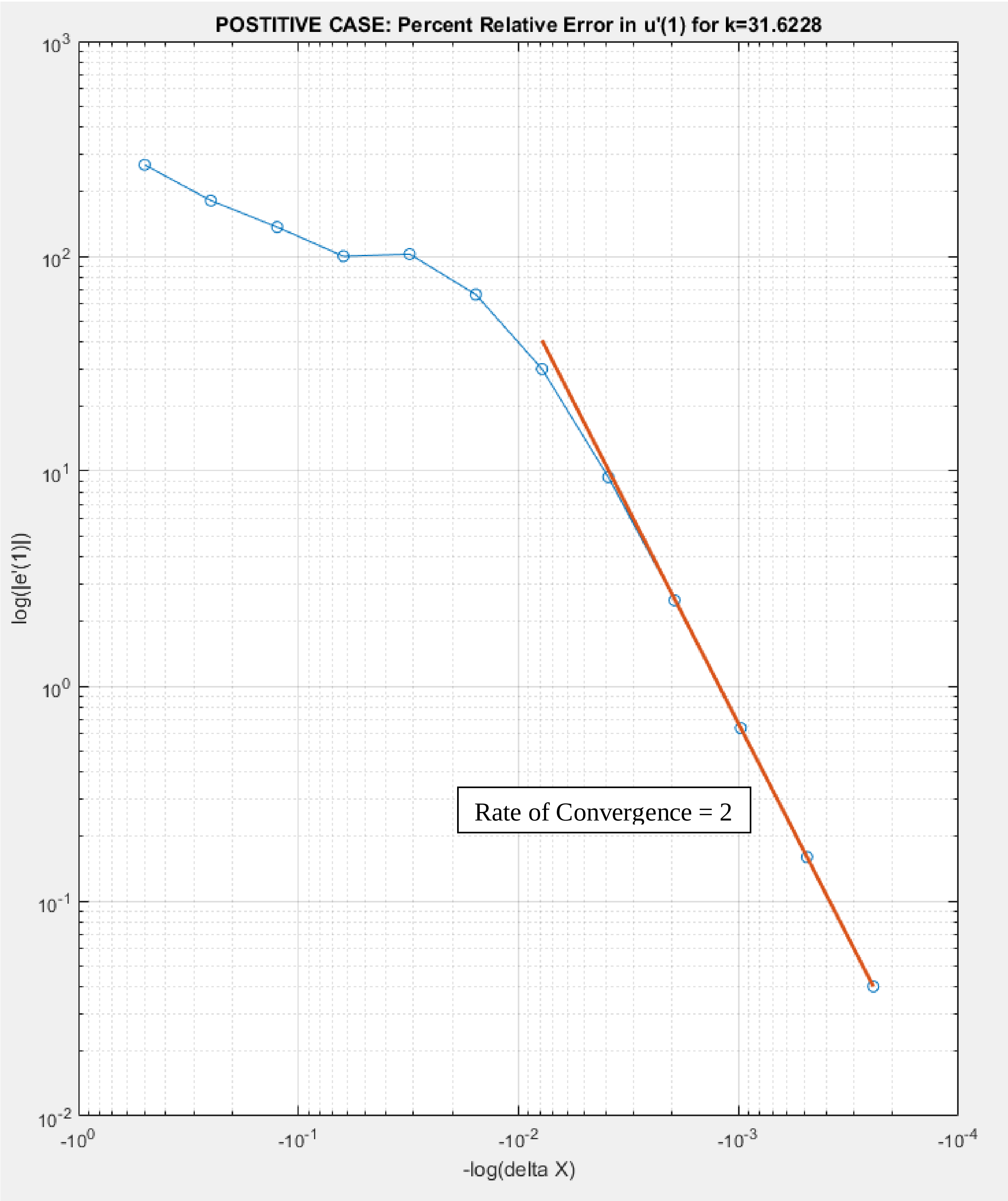

Figure 20: Error Using Higher Order Methods

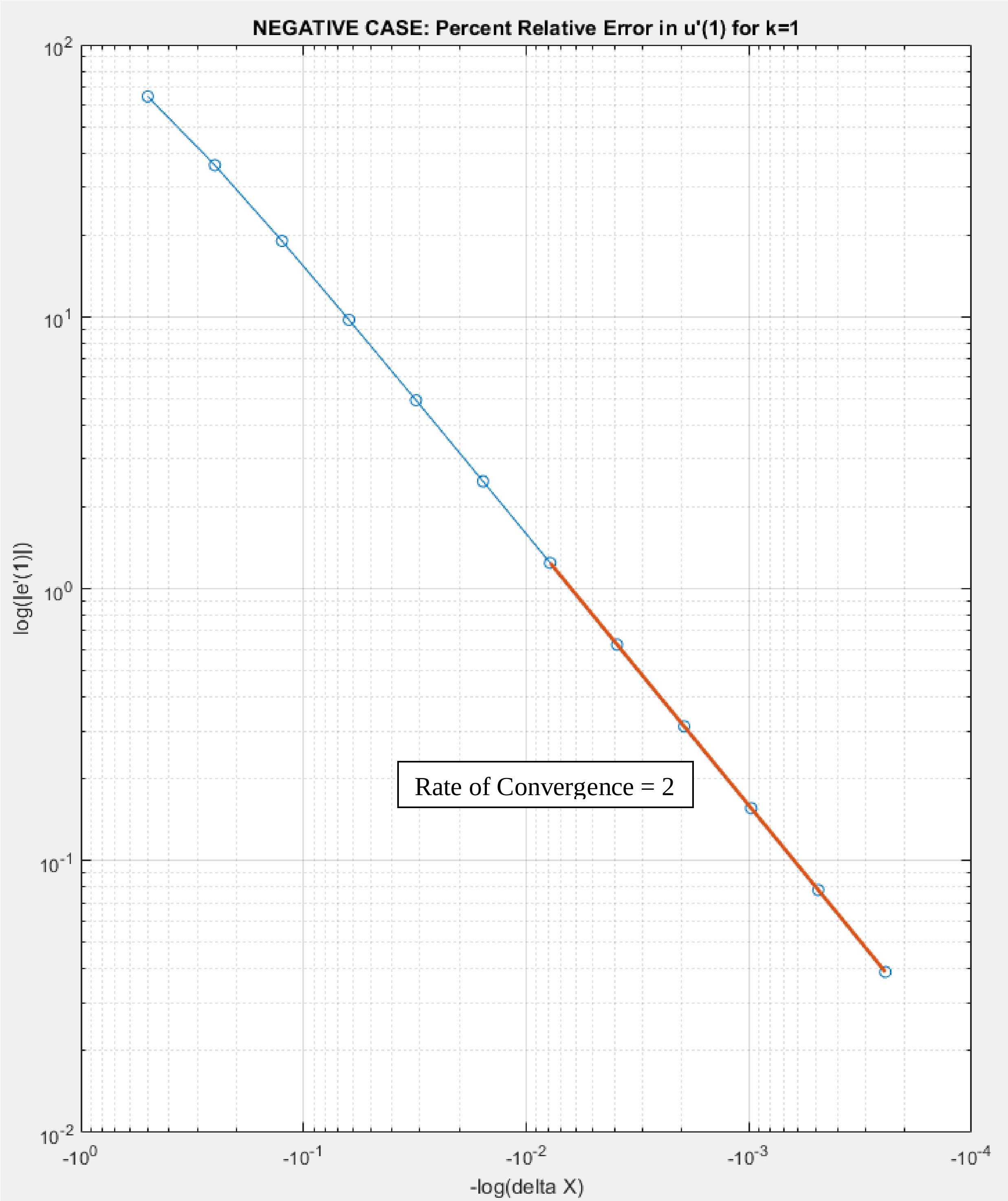

Figure 21: Error Using Higher Order Methods

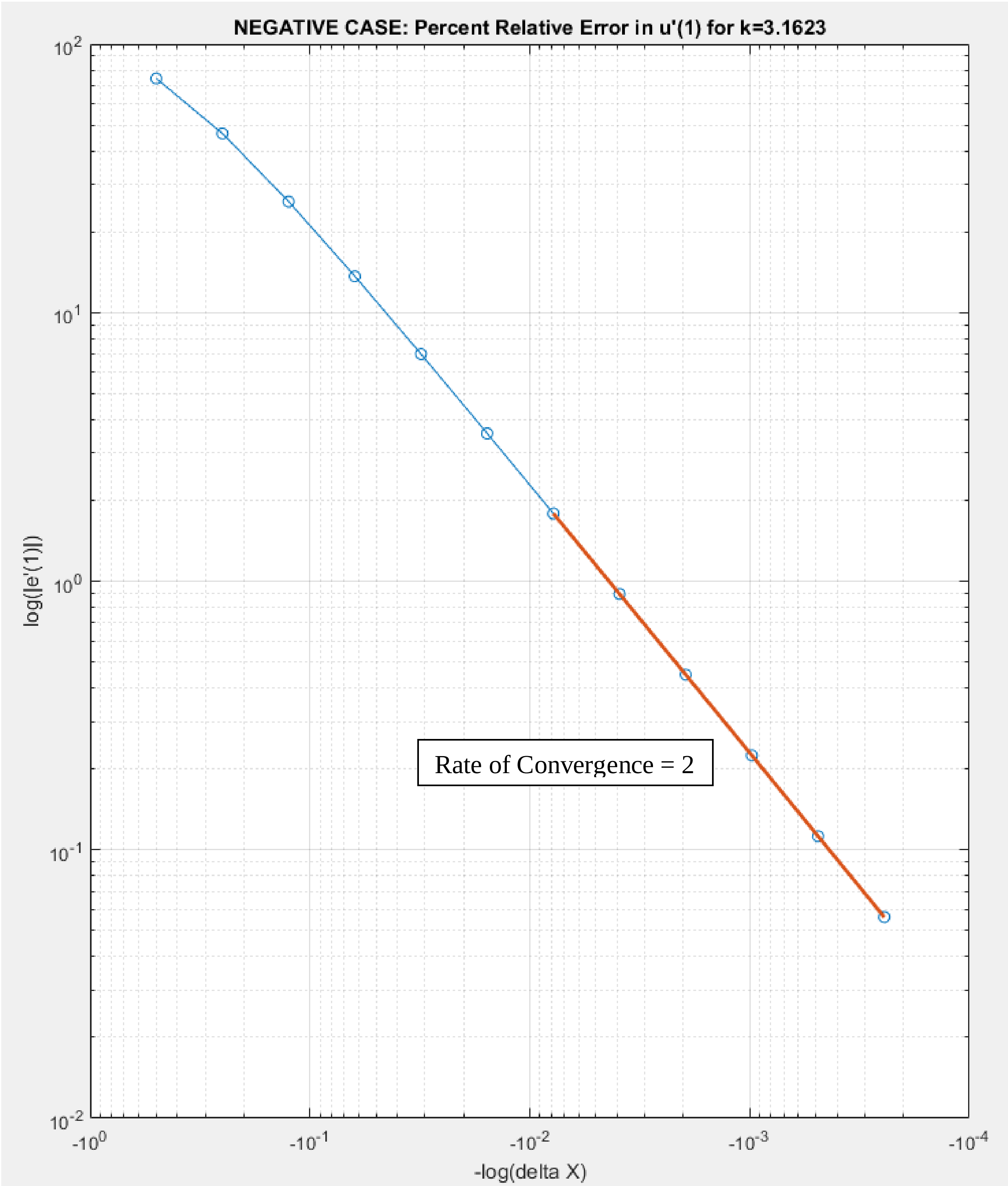

Figure 22: Error Using Higher Order Methods

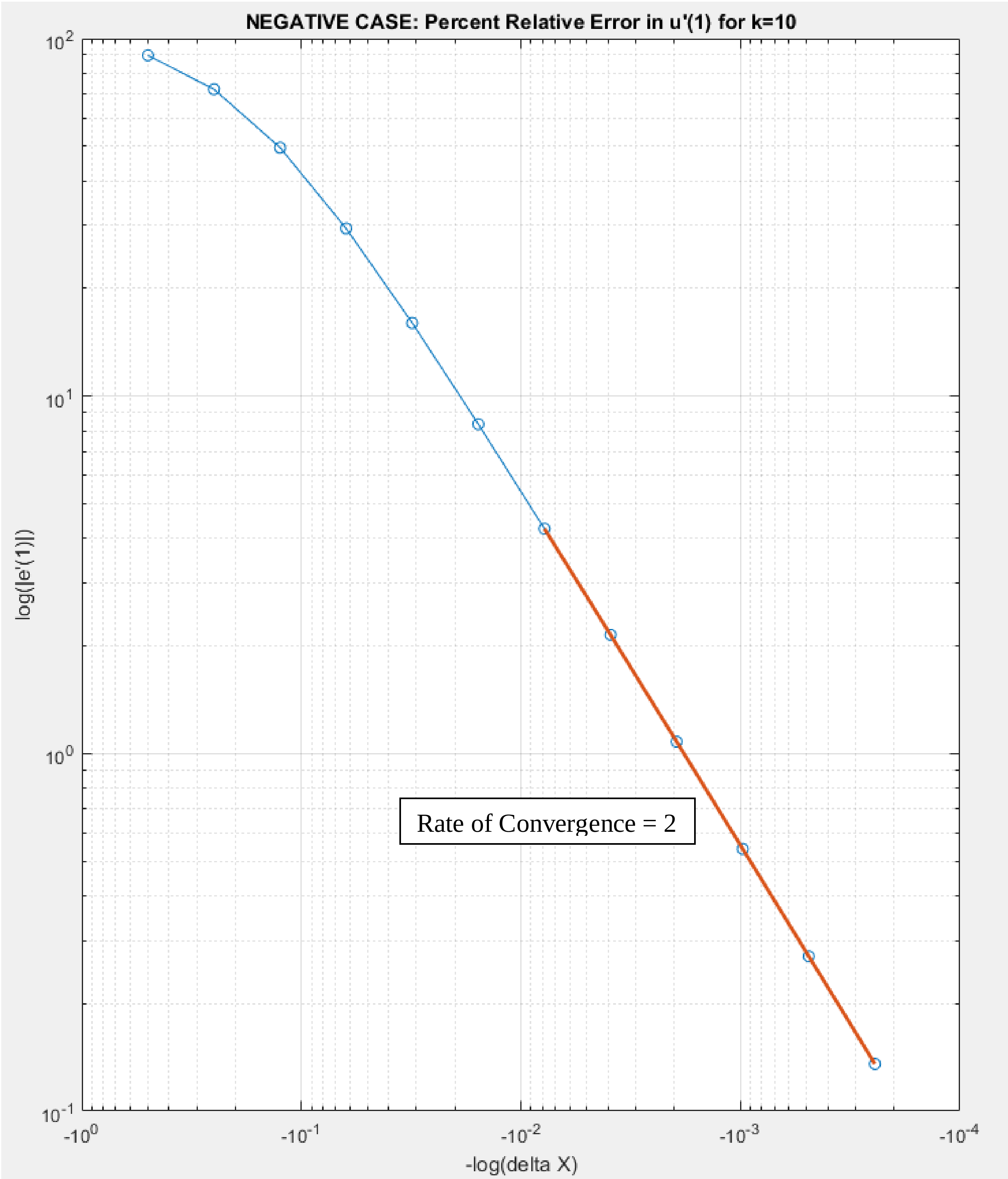

Figure 23: Error Using Higher Order Methods

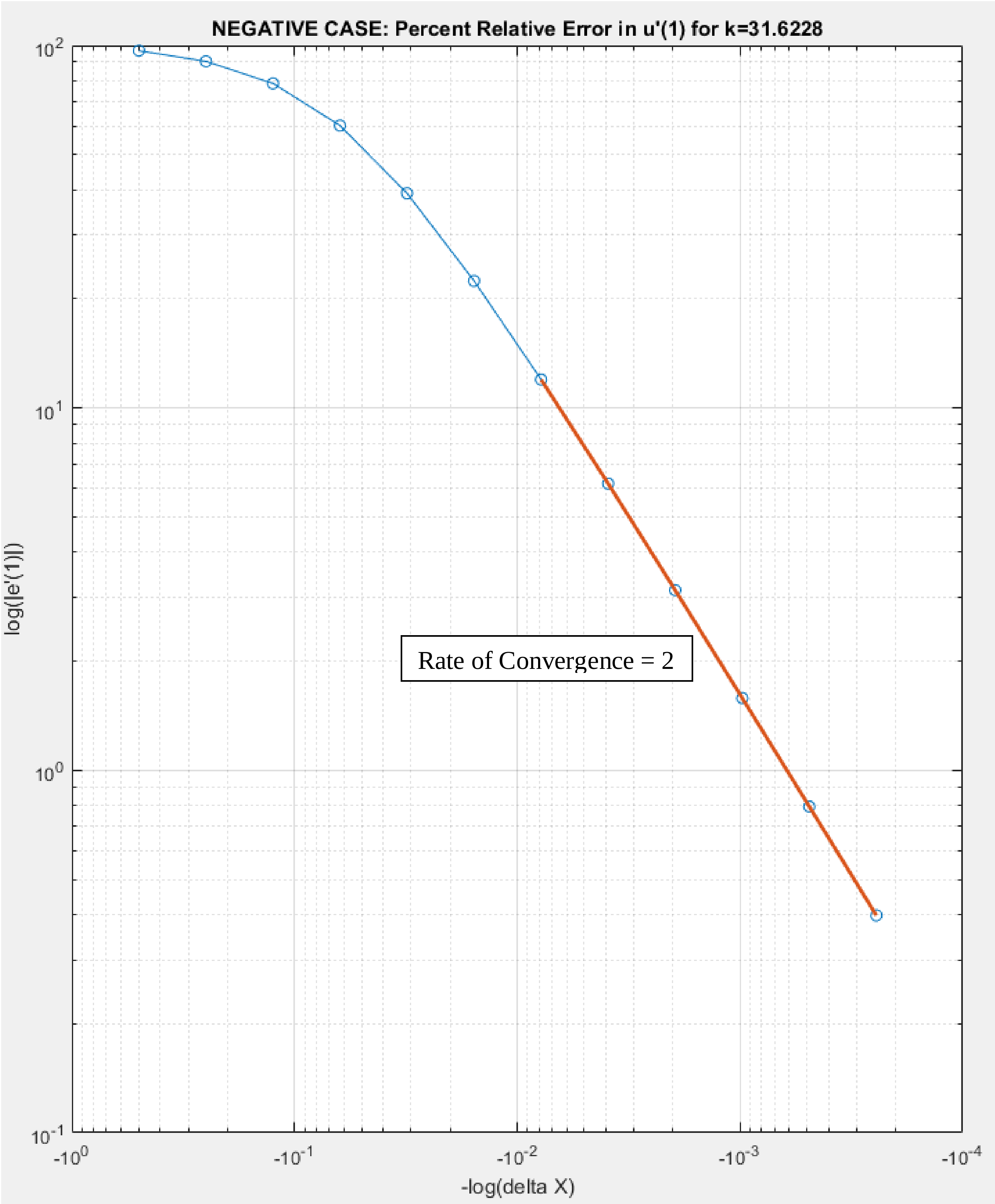

Figure 24: Error Using Higher Order Methods

--------------------------------------- DISCUSSION ON ERROR ---------------------------------------

As you can see in every case, the error follows the expected trends of decreasing as the make the meshes finer and finer. Also, you can see that in every case the error has the correct rate of convergence equal to 2. This supports the accuracy of the code and confirms that we have done our calculations correctly and that our code is working as intended. Although the error plots for the original finite difference method had a couple cases where the error plots had some anomalies, for this method, Figure 20 is the only one that shows even a small hiccup. Again, I would attribute this to the fact that the convergence for the positive case must not be monotomic. Aside from that small inconsistency, everything else looks as expected.

------------------------- RICHARDSON EXTROPOLATION QUANTITY -------------------------

For each Δx, I also calculated the Richardson Extrapolation Quantity for the derivative of U at x=1.

$$EV = \frac{(IV^2 - LAV * MAV)}{2 * IV - LAV - MAV}$$

Where EV stands for extrapolated value, LAV for least accurate value, IV for intermediate value, and MAV for most accurate value.

In this case, I based the Richardson Extrapolation off of the higher order methods that were discussed earlier. This way I could be getting the most accurate and precise values out of what was available to me.

After getting our R.E.Q. values, I also calculated the beta values for each case. Because we used the higher order methods, we would expect the R.E.Q. beta values to be very close to 2, because the rate of convergence for those higher order methods was 2.

For each value of Δx, I took the finite difference values, the higher order finite difference values, the R.E.Q. values, the beta values, the exact relative error, and estimated relative error and documented them in tables below.

-- TABLES --

POSITIVE CASE						
k=1						
ΔX	U'ΔX(1) Finite	U'ΔX(1) Higher Order	R.E.Q. for U'(1)	R.E.Q. Beta Value	Exact Relative Error (%)	Estimated Relative Error (%)
0.25	0.142857143	0.392857143			60.08544411	
0.0625	0.241612185	0.366612185	0.357918257	2.006753536	32.49309862	32.49514929
0.015625	0.297581588	0.360081588	0.357908049	2.001655335	16.85514153	16.8552961
0.25	0.327200811	0.358450811	0.357907425	2.000411829	8.579474611	8.579485177
0.0625	0.342418233	0.358043233	0.357907387	2.000102833	4.327698092	4.327698783
0.015625	0.350128846	0.357941346	0.357907384	2.000025716	2.173338296	2.17333834
0.25	0.354009624	0.357915874	0.357907384	2.000006397	1.089041401	1.089041404
0.0625	0.355956382	0.357909507	0.357907384	1.999994106	0.54511376	0.545113759
0.015625	0.356931352	0.357907915	0.357907384	1.999966098	0.272705145	0.272705144
0.25	0.357419235	0.357907517	0.357907384	2.000528664	0.136389639	0.136389643
0.0625	0.357663277	0.357907417	0.357907384	1.983908749	0.068204086	0.068204051
0.015625	0.357785322	0.357907392	0.357907385	2.83104905	0.03410436	0.034104655
0.25	0.357846351	0.357907386	0.357907386	#NUM!	0.017052759	0.017053323
0.0625	0.357876867	0.357907385	0.357907391	-0.094948865	0.008526525	0.008528427
0.25	0.357892125	0.357907384	0.357907429	-0.00227914	0.00426331	0.004275757
0.0625	0.357899755	0.357907384	0.357907406	-0.010001707	0.002131662	0.002137756
0.015625	0.357903569	0.357907384			0.001065871	
EXACT VALUE	0.357907384	0.357907384	0.357907384	2		

POSITIVE CASE						
k=sqrt(10)						
ΔX	U'ΔX(1) Finite	U'ΔX(1) Higher Order	R.E.Q. for U'(1)	R.E.Q. Beta Value	Exact Relative Error (%)	Estimated Relative Error (%)
0.25	-5	-2.5			96.70740625	
0.0625	-26.68831169	-25.43831169	18.53008698	-1.064012173	82.42524634	244.0269099
0.015625	-74.02130796	-73.39630796	-3455.725408	0.020312287	51.2555808	97.85800956
0.25	-120.9963153	-120.6838153	-161.9384052	1.101808907	20.32165765	25.28250778
0.0625	-142.8728247	-142.7165747	-152.4507876	1.706391364	5.915566041	6.282658821
0.015625	-149.5461008	-149.4679758	-151.8930119	1.919927889	1.52108864	1.545108025
0.25	-151.2912156	-151.2521531	-151.8582796	1.979503311	0.371897788	0.373416543
0.0625	-151.724111	-151.7045797	-151.8561093	1.994844789	0.086828071	0.086923264
0.015625	-151.8278569	-151.8180913	-151.8559736	1.998709295	0.018509438	0.018515389
0.25	-151.8513774	-151.8464946	-151.8559652	1.999673735	0.003020765	0.003021155
0.0625	-151.8560384	-151.853597	-151.8559647	1.999902593	4.86047E-05	4.85434E-05
0.015625	-151.8565934	-151.8553727	-151.8559646	2.000014692	0.000414099	0.00041408
0.25	-151.856427	-151.8558167	-151.8559652	1.996013077	0.000304503	0.000304109
0.0625	-151.8562331	-151.855928	-151.855965	2.001322316	0.000176836	0.000176571
0.25	-151.8561083	-151.8559557	-151.8559682	1.675816258	9.46547E-05	9.22851E-05
0.0625	-151.8560406	-151.8559643	-151.8559646	#NUM!	5.00451E-05	5.00408E-05
0.015625	-151.8560029	-151.8559647			2.52061E-05	
EXACT VALUE	-151.8559646	-151.8559646	-151.8559646	2		

POSITIVE CASE						
k=10						
ΔX	U'ΔX(1) Finite	U'ΔX(1) Higher Order	R.E.Q. for U'(1)	R.E.Q. Beta Value	Exact Relative Error (%)	Estimated Relative Error (%)
0.25	-1.086956522	23.91304348			92.46399463	
0.0625	-3.999771115	8.500228885	-13.30269281	0.771391303	72.26908714	69.93262063
0.015625	-6.77940275	-0.52940275	-291.7170175	0.044057863	52.99755374	97.67603453
0.25	-12.41245023	-9.287450229	-15.55960234	1.260831715	13.9429318	20.22643022
0.0625	-14.50471247	-12.94221247	-14.50737815	1.737714505	0.562983738	0.018374654
0.015625	-14.81932213	-14.03807213	-14.42889144	1.927519482	2.744211809	2.705895239
0.25	-14.71677756	-14.32615256	-14.42384887	1.981390217	2.033257503	2.030863574
0.0625	-14.5944202	-14.3991077	-14.42353164	1.995315881	1.18493861	1.18479
0.015625	-14.51506205	-14.4174058	-14.42351178	1.99882697	0.634738624	0.634729383
0.25	-14.47081217	-14.42198404	-14.42351054	1.999706638	0.32794868	0.327948104
0.0625	-14.4475429	-14.42312883	-14.42351046	1.99992624	0.166619935	0.166619898
0.015625	-14.43562208	-14.42341505	-14.42351045	1.999979817	0.083971414	0.083971411
0.25	-14.42959012	-14.4234866	-14.42351045	2.000124278	0.042151071	0.042151082
0.0625	-14.42655625	-14.42350449	-14.42351046	1.999498949	0.021116873	0.021116855
0.25	-14.42503484	-14.42350896	-14.42351046	1.994934437	0.010568782	0.010568759
0.0625	-14.42427302	-14.42351008	-14.42351048	1.896113361	0.005286981	0.005286774
0.015625	-14.42389185	-14.42351038			0.002644237	
EXACT VALUE	-14.42351045	-14.42351045	-14.42351045	2		

POSITIVE CASE						
k=sqrt(1000)						
ΔX	U'ΔX(1) Finite	U'ΔX(1) Higher Order	R.E.Q. for U'(1)	R.E.Q. Beta Value	Exact Relative Error (%)	Estimated Relative Error (%)
0.25	-1.008064516	248.9919355			99.32657032	
0.0625	-3.066133776	121.9338662	-19.91890487	0.92273358	97.95169312	84.60691592
0.015625	-7.590353368	54.90964663	-249.2257236	0.287326037	94.92932333	96.95442619
0.25	-31.2612995	-0.011299503	3.244512603	#NUM!	79.11613142	1063.512962
0.0625	-12.17531397	3.449686028	0.197757623	#NUM!	91.86637597	6256.685016
0.015625	-58.19918567	-50.38668567	3744.027066	-0.020616026	61.12048557	101.5544542
0.25	-108.904149	-104.997899	-174.8713574	0.833151654	27.24742826	37.7232781
0.0625	-137.6043911	-135.6512661	-151.1366184	1.575073757	8.074454199	8.953639067
0.015625	-146.9158945	-145.939332	-149.7814078	1.878818062	1.853976552	1.913130148
0.25	-149.2250041	-148.7367229	-149.6967815	1.968559319	0.311393811	0.315155305
0.0625	-149.6956194	-149.4514788	-149.6914862	1.992064059	0.002997067	0.00276114
0.015625	-149.7532237	-149.6311534	-149.6911551	1.998011191	0.041479171	0.041464418
0.25	-149.7371692	-149.676134	-149.6911344	1.999503063	0.03075406	0.030753143
0.0625	-149.7179006	-149.687383	-149.6911331	1.999871542	0.017881855	0.017881792
0.25	-149.7054543	-149.6901955	-149.691133	2.000006108	0.009567213	0.009567228
0.0625	-149.698528	-149.6908987	-149.691133	2.000132766	0.004940172	0.004940203
0.015625	-149.6948891	-149.6910744			0.002509212	
EXACT VALUE	-149.6911331	-149.6911331	-149.6911331	2		

NEGATIVE CASE						
k=1						
ΔX	U'ΔX(1) Finite	U'ΔX(1) Higher Order	R.E.Q. for U'(1)	R.E.Q. Beta Value	Exact Relative Error (%)	Estimated Relative Error (%)
0.25	-0.111111111	-0.361111111			64.50524389	
0.0625	-0.200147051	-0.325147051	-0.313004151	1.986132597	36.06246312	36.05610322
0.015625	-0.253569196	-0.316069196	-0.313033292	1.996448483	18.99660896	18.99609319
0.25	-0.282544139	-0.313794139	-0.31303516	1.999106646	9.740482255	9.740446118
0.0625	-0.297600022	-0.313225022	-0.313035278	1.999776317	4.930838085	4.930835702
0.015625	-0.305270221	-0.313082721	-0.313035285	1.999944054	2.48057157	2.480571418
0.25	-0.309140895	-0.313047145	-0.313035285	1.999985972	1.244074123	1.244074113
0.0625	-0.311085125	-0.31303825	-0.313035285	1.999997375	0.622984159	0.622984158
0.015625	-0.312059464	-0.313036027	-0.313035285	1.999994916	0.311728854	0.311728854
0.25	-0.31254719	-0.313035471	-0.313035286	2.000081408	0.155923621	0.155923622
0.0625	-0.312791191	-0.313035332	-0.313035285	1.997660297	0.077976609	0.077976601
0.015625	-0.312913227	-0.313035297	-0.313035286	2.065120703	0.038992004	0.038992058
0.25	-0.312974253	-0.313035288	-0.313035286	2.258161723	0.019496927	0.019496975
0.0625	-0.313004769	-0.313035286	-0.313035281	0.178984287	0.009748694	0.009747309
0.25	-0.313020027	-0.313035286	-0.313035287	-0.129160602	0.00487442	0.004874973
0.0625	-0.313027656	-0.313035285	-0.313035265	-0.003968132	0.002437236	0.002430707
0.015625	-0.313031471	-0.313035286			0.001218602	
EXACT VALUE	-0.313035285	-0.313035285	-0.313035285	2		

NEGATIVE CASE						
k=sqrt(10)						
ΔX	U'ΔX(1) Finite	U'ΔX(1) Higher Order	R.E.Q. for U'(1)	R.E.Q. Beta Value	Exact Relative Error (%)	Estimated Relative Error (%)
0.25	-0.555555556	-3.055555556			74.4411179	
0.0625	-1.16461281	-2.41461281	-2.166034334	1.839327642	46.4208373	46.23294784
0.015625	-1.610500084	-2.235500084	-2.173070923	1.951981416	25.90735284	25.88828708
0.25	-1.876706427	-2.189206427	-2.173593517	1.98735301	13.66026706	13.65881375
0.0625	-2.021281112	-2.177531112	-2.17362779	1.996794826	7.008965888	7.008866894
0.015625	-2.096480791	-2.174605791	-2.173629959	1.999195936	3.54933036	3.549323924
0.25	-2.134811553	-2.173874053	-2.173630095	1.999798793	1.785885787	1.785885377
0.0625	-2.154159843	-2.173691093	-2.173630104	1.999949385	0.895748593	0.895748567
0.015625	-2.163879727	-2.173645352	-2.173630104	1.999985452	0.448575755	0.448575754
0.25	-2.168751104	-2.173633916	-2.173630104	1.99989695	0.224463245	0.22446324
0.0625	-2.171189651	-2.173631057	-2.173630104	2.000572987	0.112275464	0.11227547
0.015625	-2.172409639	-2.173630342	-2.173630104	2.004120139	0.056148692	0.056148703
0.25	-2.173019812	-2.173630164	-2.173630104	1.969987195	0.028077086	0.028077065
0.0625	-2.173324943	-2.173630119	-2.173630107	3.858446452	0.01403923	0.014039359
0.25	-2.17347752	-2.173630108	-2.173630179	-0.053813048	0.007019789	0.00702322
0.0625	-2.173553811	-2.173630105	-2.173614406	0.00024379	0.003509935	0.002787727
0.015625	-2.173591956	-2.173630103			0.001755068	
EXACT VALUE	-2.173630104	-2.173630104	-2.173630104	2		

NEGATIVE CASE						
k=10						
ΔX	U'ΔX(1) Finite	U'ΔX(1) Higher Order	R.E.Q. for U'(1)	R.E.Q. Beta Value	Exact Relative Error (%)	Estimated Relative Error (%)
0.25	-0.925925926	-25.92592593			89.7119342	
0.0625	-2.507812276	-15.00781228	-8.141477927	1.373004181	72.13541928	69.19708807
0.015625	-4.542476563	-10.79247656	-8.880062375	1.679962402	49.52803842	48.84634396
0.25	-6.351909193	-9.476909193	-8.989631622	1.887459597	29.42323151	29.34183001
0.0625	-7.558834258	-9.121334258	-8.999282205	1.968387794	16.01295307	16.00625377
0.015625	-8.249221196	-9.030471196	-8.999953914	1.991836458	8.341987134	8.341517364
0.25	-8.617001528	-9.007626528	-8.999997138	1.997941995	4.255539018	4.25550813
0.0625	-8.806594708	-9.001907208	-8.999999859	1.999484411	2.148948137	2.14894616
0.015625	-8.902820617	-9.000476867	-9.00000003	1.999871069	1.079771375	1.07977125
0.25	-8.951291125	-9.00011925	-9.000000041	1.999967746	0.54121018	0.541210172
0.0625	-8.975615781	-9.000029844	-9.000000041	2.000005847	0.270936223	0.270936224
0.015625	-8.987800461	-9.000007492	-9.000000041	1.999736814	0.135550896	0.135550891
0.25	-8.993898388	-9.000001904	-9.000000042	2.002496424	0.067796144	0.067796157
0.0625	-8.996948749	-9.000000507	-9.000000029	1.891646263	0.033903245	0.033903107
0.25	-8.998474279	-9.000000158	-9.000000041	1.991925653	0.016952916	0.01695291
0.0625	-8.999237131	-9.00000007	-8.999999908	0.518379727	0.008476784	0.008475304
0.015625	-8.999618551	-9.000000021			0.004238775	
EXACT VALUE	-9.000000041	-9.000000041	-9.000000041	2		

NEGATIVE CASE						
k=sqrt(1000)						
ΔX	U'ΔX(1) Finite	U'ΔX(1) Higher Order	R.E.Q. for U'(1)	R.E.Q. Beta Value	Exact Relative Error (%)	Estimated Relative Error (%)
0.25	-0.992063492	-250.9920635			96.76037381	
0.0625	-2.937969582	-127.9379696	-14.98626041	1.063116232	90.4059334	80.39557901
0.015625	-6.544628631	-69.04462863	-23.80374889	1.202745357	78.62823246	72.50589114
0.25	-12.20854811	-43.45854811	-29.12687823	1.477823701	60.1324587	58.08494129
0.0625	-18.64737765	-34.27237765	-30.45076811	1.767122475	39.10618266	38.76220928
0.015625	-23.7610346	-31.5735346	-30.60931783	1.925619831	22.40731496	22.37319781
0.25	-26.95687585	-30.86312585	-30.62187761	1.979923238	11.97115728	11.96857294
0.0625	-28.72990985	-30.68303485	-30.62271942	1.994876496	6.181238158	6.181062973
0.015625	-29.66128942	-30.63785192	-30.62277301	1.998712393	3.139777921	3.139766566
0.25	-30.13826486	-30.62654611	-30.62277638	1.999677669	1.582194039	1.582193317
0.0625	-30.37957839	-30.62371902	-30.62277659	1.999919237	0.794174252	0.794174206
0.015625	-30.5009419	-30.62301221	-30.6227766	1.999978307	0.397856494	0.39785649
0.25	-30.56180035	-30.6228355	-30.6227766	1.999997898	0.199120593	0.199120591
0.0625	-30.59227375	-30.62279133	-30.6227766	2.00029934	0.099608385	0.099608378
0.25	-30.60752149	-30.62278028	-30.62277658	1.990768558	0.049816222	0.049816153
0.0625	-30.61514812	-30.62277751	-30.62277652	1.883992064	0.024911147	0.024910884
0.015625	-30.61896209	-30.62277679			0.012456446	
EXACT VALUE	-30.6227766	-30.6227766	-30.6227766	2		

-- DISCUSSION --

As you can see, the general trend is that as Δx decreases, the accuracy of our calculations increases. The finite difference values, the higher order finite difference values, and the R.E.Q. values all convergence to the exact value as you decrease the size of the mesh. The original finite difference method converges the slowest since it is the least accurate of the methods, following by the higher order finite difference method and lastly the R.E.Q. method. The R.E.Q. method converges faster than the higher order method because it is actually using those higher order values to produce an even more accurate approximation.

The exact relative error represents the error between the original finite difference method and the exact values. As you can see, the general trend is always that as you decrease the size of your mesh, your error decreases. This was also proven earlier as we discussed the individual error plots. The estimated relative error represents the difference between the R.E.Q values and the exact value of the derivative. As you can see, earlier on, the R.E.Q. method converges just a little bit faster, represented by slightly smaller error values within the first few mesh sizes.

Also, since the R.E.Q. is calculated using the higher order method values, we would expect the rate of convergence, or beta, to be equal to 2. As you can see, this is indeed the case with our calculations. However, it is interesting to note that within the last few mesh sizes (the finest of them all), the beta value becomes inaccurate as it is dealing with calculations involving numbers with drastically different orders of magnitude. This is not so much a shortcoming of the method as it is just a mathematical obstacle when computing values using extremely small numbers.

-- CONCLUSIONS --

As you can see from all the results, the finite difference approximation results in a very rough representation of the exact function. You can clearly see that when Δx is larger, the approximation is much less accurate and may not best represent the original function. However, as you decrease Δx, making your mesh finer, you can see a trend of increased accuracy and the finite difference approximation fitting closer and closer to the original function. Although for the cases where Δx=.5 were extremely rough approximations, you can see that when we get to Δx=1/16 and beyond the difference between the original function and the approximation is significantly reduced.

For the positive case, you can see that the original differential equation represents a forced vibration. As k increases, the frequency of the oscillation also increases. However, when k is close to a multiple of pi, which we discussed earlier as not being a solution, you can get some weird results which don't reflect what a forced vibration system would normally look like. This is evident in the case for k=sqrt(10), which is close to pi. However, besides those anomalies, all the correct trends are shown and the value of the finite difference method is proven.

Contrary to the positive case, the negative case represents a diffusion problem. The function follows a trend along the line y=x before decreasing to meet the boundary condition at x=1. In this case, as k increases, so does the distance that the function follows y=x before diverging down to y=0. Again as with the positive case, when you start with a large Δx, you don't get the best representation of the original function because the approximation can jump far below the exact function before reaching the boundary condition. However, as Δx is decreased to produce a finer mesh, the approximation becomes more accurate and ends up very clearly depicting the original function.

When looking at the plots of percent relative error versus Δx, the trends between the positive and negative cases are the same. In all cases, as Δx decreases, the error also decreases. This is because we are creating a finer mesh which can more accurately fit to the original function by evaluating approximations at more data points. All of the original error plots for percent relative error all showed a rate of convergence equal to 1 which is the same as what we derived in class. For the higher order methods, the rate of convergence was always equal to 2, which is also what we derived in class. However, it is important to note that a few of the error plots had some anomalous data points. In certain cases, a finer mesh caused a temporary jump in error, however it always corrected and ended up converging to the correct value.