Approximation of the Modified Bessel I0 function

John Trenholme - 7 Aug 2000 - Maple V - "fitI0.mws"
Revisions to 3 Dec 2001

```
> restart; kernelopts( version); Digits := 16;
```

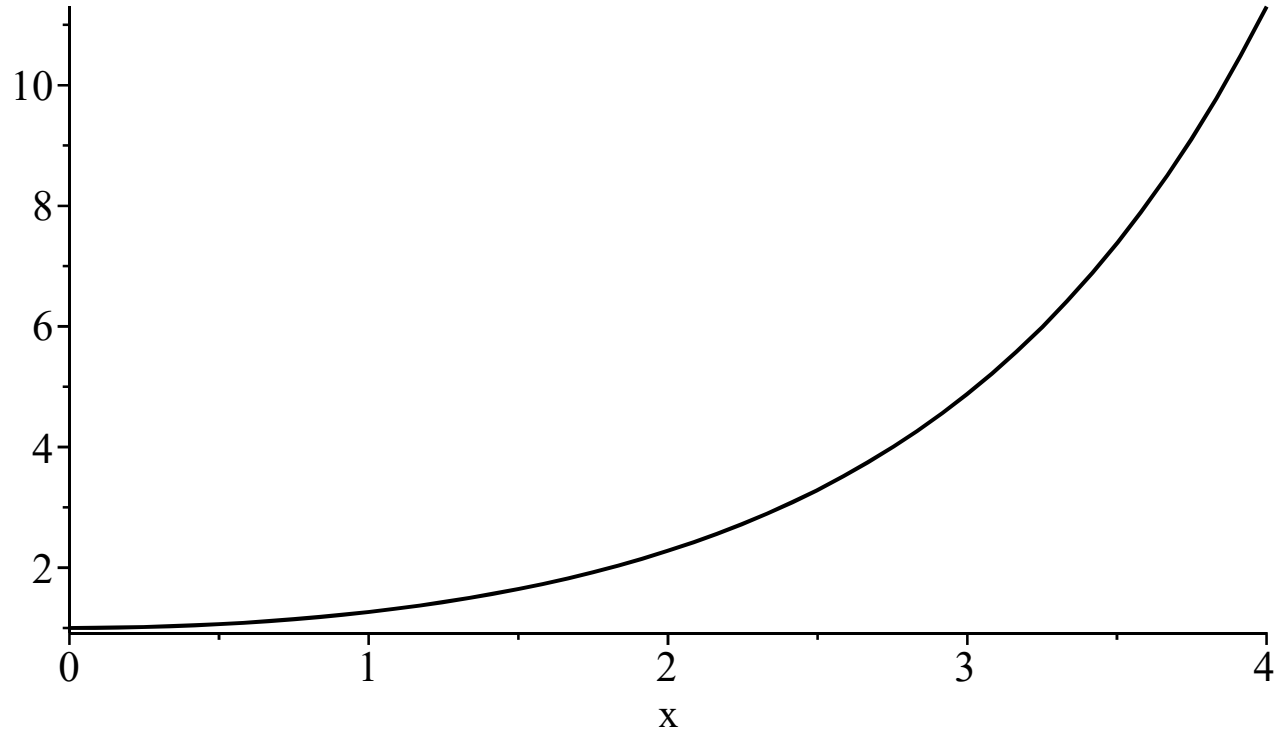
Maple 7.00, IBM INTEL NT, May 28 2001 Build ID 96223

$$Digits := 16$$

Behavior of the I0 function

The function is known to Maple:

```
> plot( BesselI( 0, x), x = 0 .. 4, color = black);
```



Behavior as $x \to 0$ (note that I0(-x) = I0(x)):

```
> series( BesselI( 0, z), z, 9);
```

$$1 + \frac{1}{4} z^2 + \frac{1}{64} z^4 + \frac{1}{2304} z^6 + \frac{1}{147456} z^8 + O(z^9)$$

Explicit coefficients from AMS55 9.6.12:

```
> I0lo := ( z, m) -> sum( ( z^2 / 4)^j / (j!)^2, j = 0 .. m);
```

$$I0lo := (z, m) \to \sum_{j=0}^{m} \frac{\left(\frac{1}{4} z^2\right)^j}{j!^2}$$

```
> I0lo( z, 6);
```

$$1 + \frac{1}{4} z^2 + \frac{1}{64} z^4 + \frac{1}{2304} z^6 + \frac{1}{147456} z^8 + \frac{1}{14745600} z^{10} + \frac{1}{2123366400} z^{12}$$

Behavior as $x \to \infty$:

```
> asympt( BesselI( 0, z), z, 4);
```

$$\frac{1}{2}\frac{\sqrt{2}\,e^{z}\sqrt{\frac{1}{z}}}{\sqrt{\pi}}+\frac{1}{16}\frac{\sqrt{2}\,e^{z}\left(\frac{1}{z}\right)^{3/2}}{\sqrt{\pi}}+\frac{9}{256}\frac{\sqrt{2}\,e^{z}\left(\frac{1}{z}\right)^{5/2}}{\sqrt{\pi}}+\frac{75}{2048}\frac{\sqrt{2}\,e^{z}\left(\frac{1}{z}\right)^{7/2}}{\sqrt{\pi}}+O\left(\left(\frac{1}{z}\right)^{9/2}\right)$$

Explicit coefficients from AMS55 9.7.1:

```
> I0hi := ( z, m) -> sum( product( (2 * j - 1)^2, j = 0 .. k) / (
  k! * ( 8 * z)^k), k = 0 .. m) * exp( z) / sqrt( 2 * Pi * z);
```

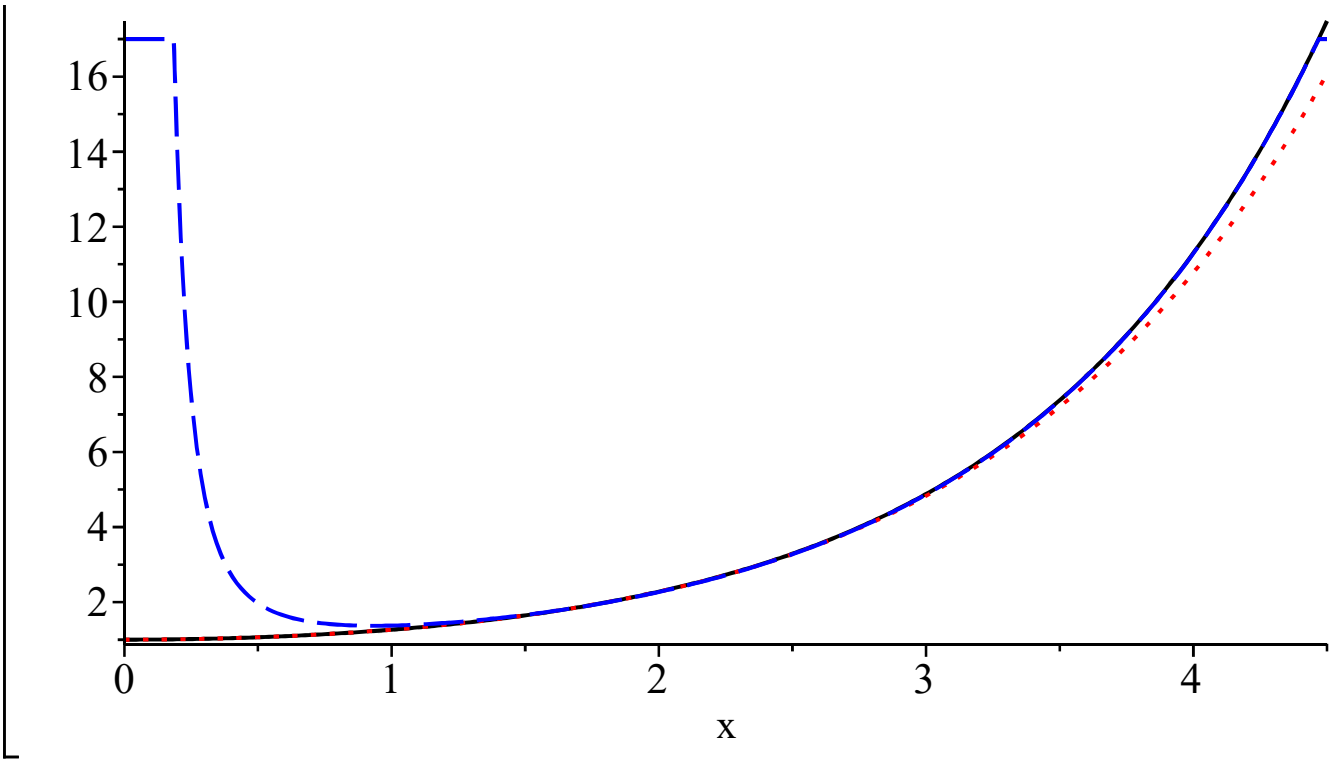
$$I0hi := (z, m) \to \frac{\left(\sum_{k=0}^{m}\frac{\prod_{j=0}^{k}\left((2j-1)^2\right)}{k!\,(8z)^k}\right)e^{z}}{\sqrt{2\pi z}}$$

```
> I0hi( z, 6);
```

$$\frac{1}{2}\frac{\left(1+\frac{1}{8z}+\frac{9}{128z^2}+\frac{75}{1024z^3}+\frac{3675}{32768z^4}+\frac{59535}{262144z^5}+\frac{2401245}{4194304z^6}\right)e^{z}\sqrt{2}}{\sqrt{\pi z}}$$

Compare the function to its low and high limits:

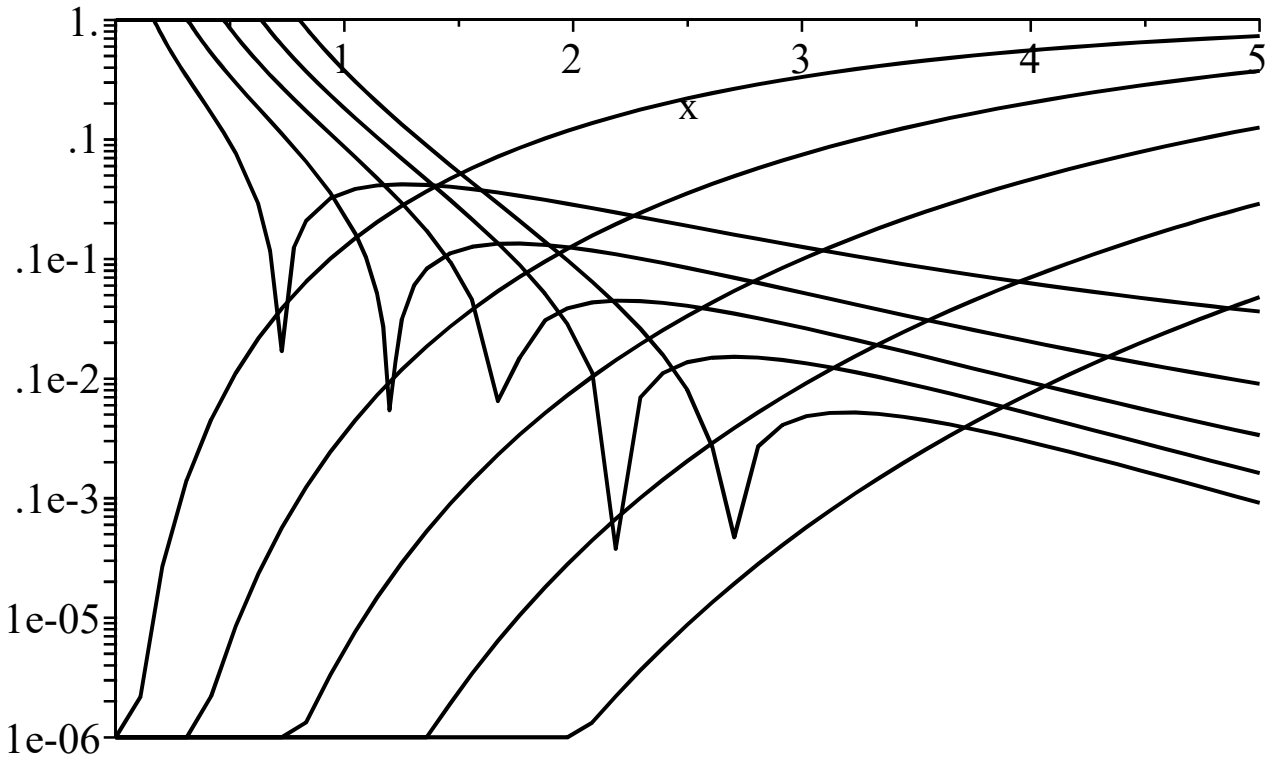
```
> plot( [ BesselI( 0, x), I0lo( x, 3), min( 17, I0hi( x, 3))], x =
  0 .. 4.5, color = [ black, red, blue], linestyle = [ 1, 2, 3],
  numpoints = 300);
```



Splicing low and high ends

We can use the power series for small x, and the asymptotic form for large x, and join them somewhere in the middle. Let us examine the error as a function of x and the number of terms used.

```
> plots[ logplot]( [ seq( max( 1e-6, abs( I0lo( x, N_) / BesselI(
  0, x) - 1)), N_ = 1 .. 5), seq( min( 1, abs( I0hi( x, N_) /
  BesselI( 0, x) - 1)), N_ = 1 .. 5)], x = 0 .. 5, color = black);
```



These curves suggest that we use the zero-error point of the asymptotic curves as the splice point. We can then tweak the low-end power series to match at that point. For example, use the 4-term asymptotic:

```
> xs := fsolve( I0hi( x, 4) = BesselI( 0, x), x = 1 .. 3);
```

$$xs := 2.190327644324161$$

Then tweak I0lo(x, 3) to match at that point. Start from:

```
> I0lo( x, 3);
```

$$1 + \frac{1}{4} x^2 + \frac{1}{64} x^4 + \frac{1}{2304} x^6$$

Find the value of the last coefficient that does the job.

```
> fsolve( 1 + xs^2 / 4 + xs^4 / 64 + xs^6 * a = BesselI( 0, xs), a)
  ;
```

$$0.0004681773415068081$$

Make a function that does the job.

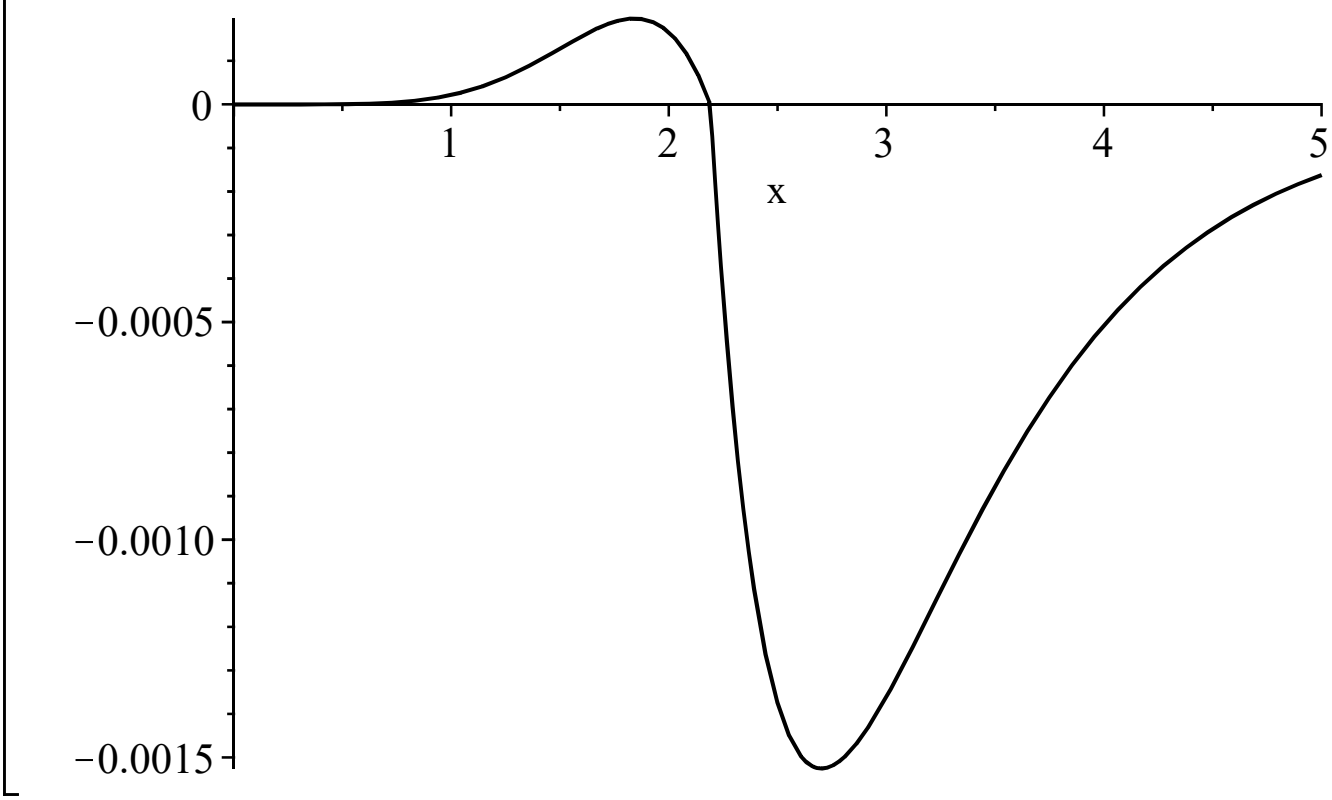
```
> I0s := x -> piecewise( x < 2.190328, 1 + x^2 * ( 0.25 + x^2 * (
  0.015625 + x^2 * 0.0004681773)), exp( x) / sqrt( 2 * Pi * x) * (
  1 + ( 0.125 + ( 0.0703125 + ( 0.07324219 + 0.1121521 / x) / x) /
  x) / x));
```

$$I0s := x \rightarrow piecewise\left(x < 2.190328, 1 + x^2 \left(0.25 + x^2 \left(0.015625 + 0.0004681773\, x^2\right)\right),\right.$$

$$\frac{e^x \left(1 + \frac{0.125 + \frac{0.0703125 + \frac{0.07324219 + \frac{0.1121521}{x}}{x}}{x}}{x}\right)}{\sqrt{2\,\pi\,x}}$$

Show the error of the spliced approximation. It's not very symmetric.

```
> plot( I0s( x) / BesselI( 0, x) - 1, x = 0 .. 5, color = black);
```



Maybe we need a different approach. Let's adjust the last coefficient in both approximations. Define a low end with an adjustable coefficient.

```
> fal := ( x, ba) -> 1 + x^2 * ( 0.25 + x^2 * ( 0.015625 + x^2 *
  ba));
```

$$fal := (x, ba) \rightarrow 1 + x^2 \left(0.25 + x^2 \left(0.015625 + x^2\, ba\right)\right)$$

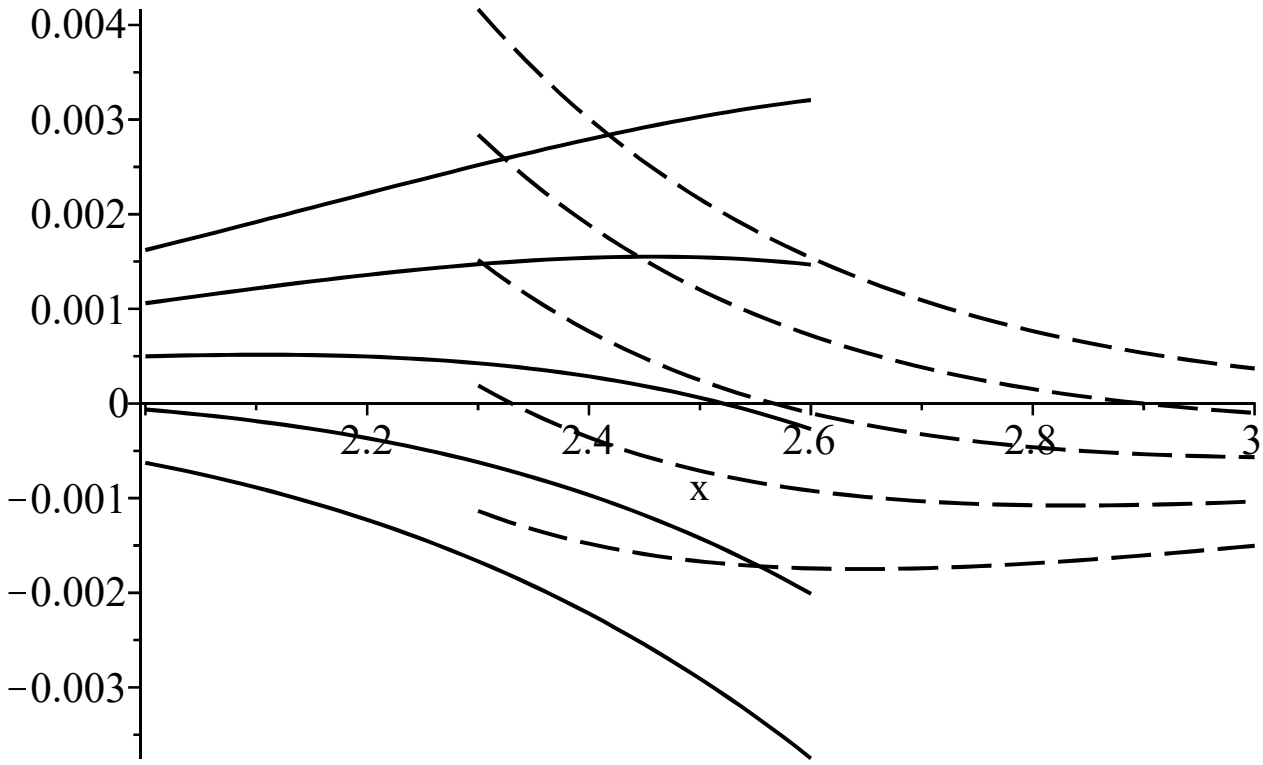
Define a high end with an adjustable coefficient.

```
> fah := ( x, ca) -> exp( x) / sqrt( 2 * Pi * x) * ( 1 + ( 0.125 +
  ( 0.0703125 + ( 0.07324219 + ca / x) / x) / x) / x);
```

$$fah := (x, ca) \rightarrow \frac{e^x \left(1 + \frac{0.125 + \frac{0.0703125 + \frac{0.07324219 + \frac{ca}{x}}{x}}{x}}{x}\right)}{\sqrt{2\,\pi\,x}}$$

Show the behavior of the two approximations as a function of the adjustable parameters.

```
> plots[ display]( [
  plot( [ seq( fal( x, ba_) / BesselI( 0, x) - 1, ba_ = [ .00044,
  .00046, .00048, .0005, .00052])], x = 2 .. 2.6, color = black),
  plot( [ seq( fah( x, ca_) / BesselI( 0, x) - 1, ca_ = [ .1, .14,
  .18, .22, .26])], x = 2.3 .. 3, color = black, linestyle = 3)]);
```



Specify a splice point and manually adjust it for equal-ripple error in the plot below.

```
> xa := 2.5625;
```

$$xa := 2.5625$$

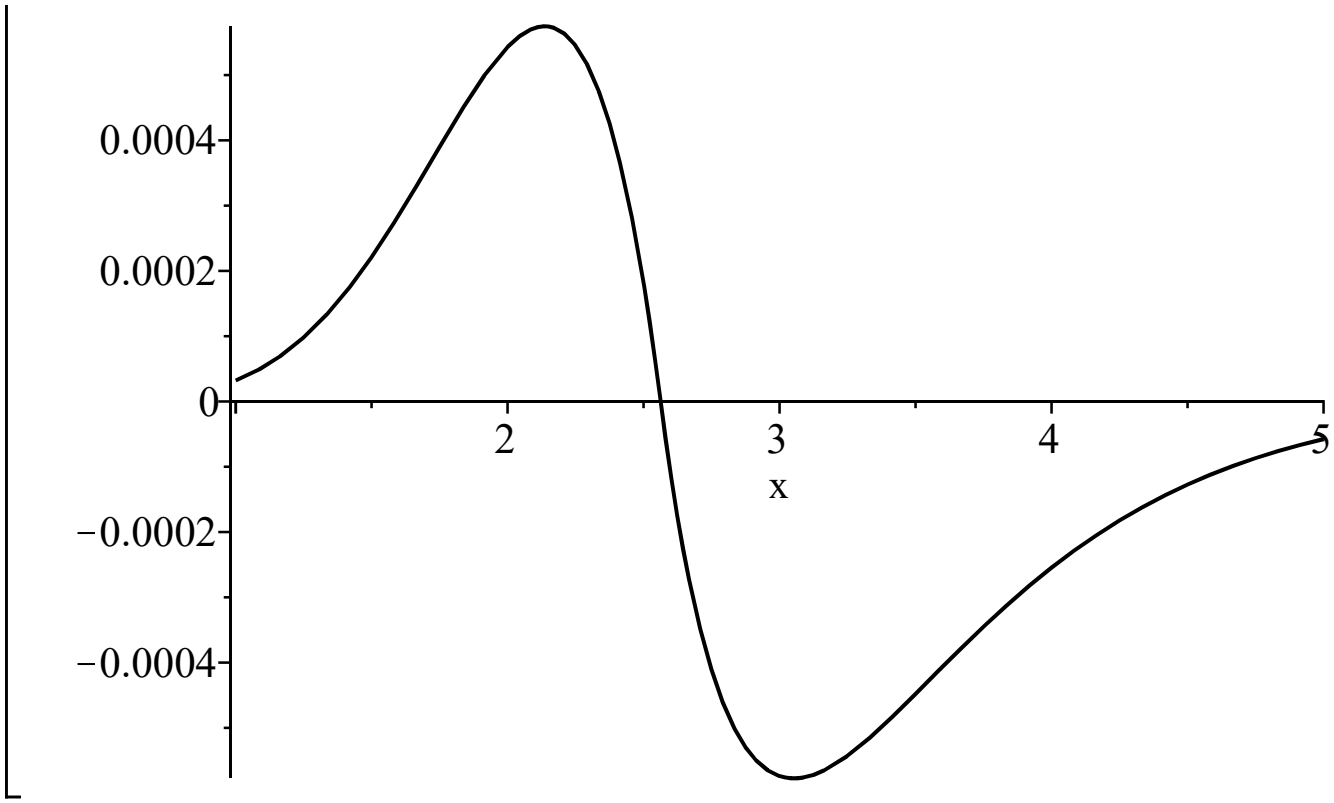
Solve for the values of the coefficients that give equal value and slope at the specified splice point.

```
> unassign( 'ba', 'ca');
  sa := fsolve( { fal( xa, ba) = fah( xa, ca), subs( x = xa, diff(
  fal( x, ba), x) = diff( fah( x, ca), x))}, { ba, ca});
```

$$sa := \{ca = 0.1794150554907968, ba = 0.0004816178586690358\}$$

Plot the resulting relative error curve. We could do much better than this by adjusting more of the coefficients and thereby giving more zeros of the error curve, but we will take another path (see later sections).

```
> assign( sa);
  plot( piecewise( x < xa, fal( x, ba), fah( x, ca)) / BesselI( 0,
  x) - 1, x = 1 .. 5, color = black);
```



Put in finite-precision coefficients, and solve for splice point.

```
> fsolve( 1 + x^2 * ( 0.25 + x^2 * ( 0.015625 + x^2 * 0.0004816179)
  ) = exp( x) / sqrt( 2 * Pi * x) * ( 1 + ( 0.125 + ( 0.0703125 + (
  0.07324219 + 0.1794151 / x) / x) / x) / x), x = 2.5 .. 2.6);
```

$$2.562064800746740$$

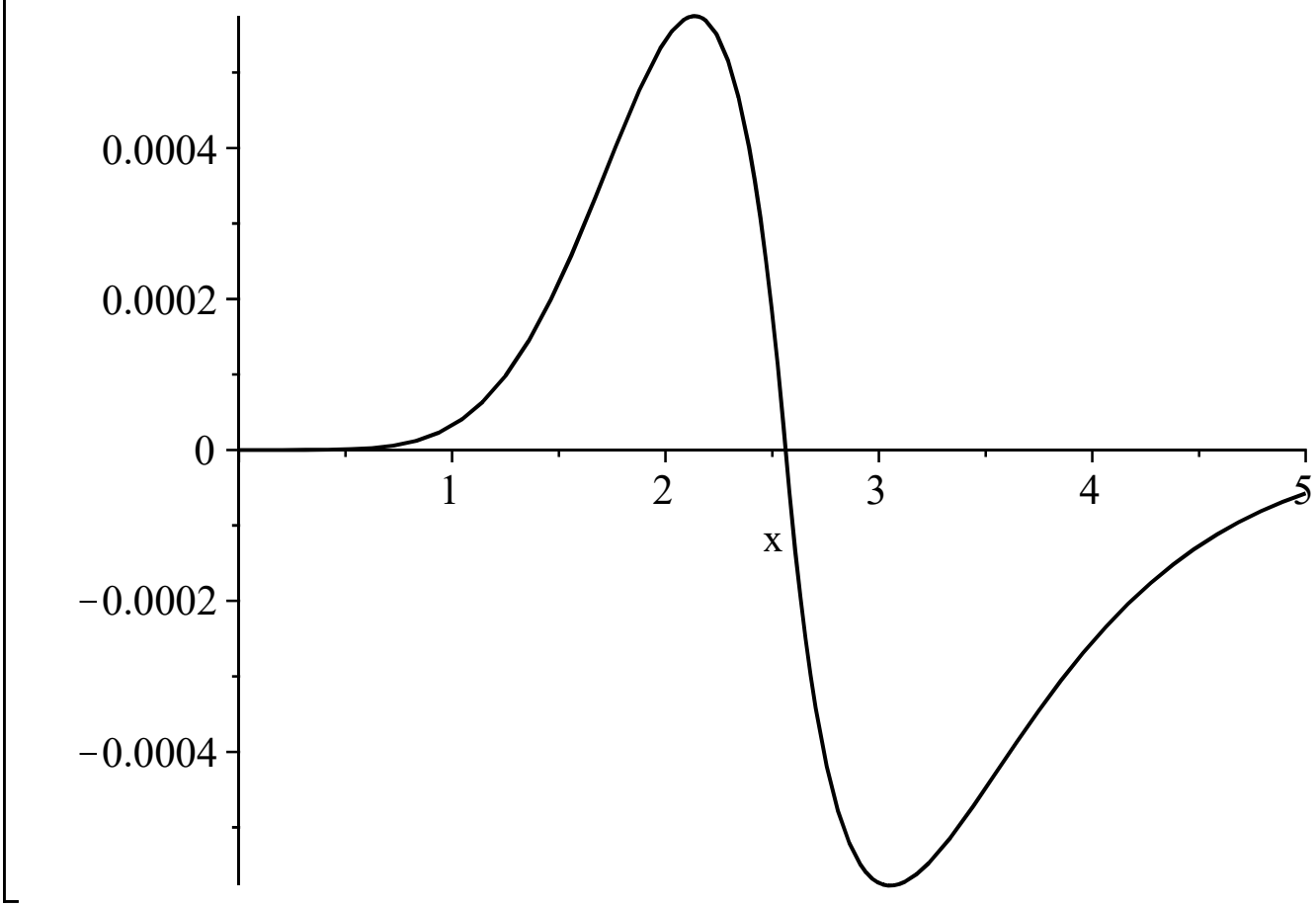
Define the final spliced approximation.

```
> I0sa := x -> piecewise( x < 2.562065, 1 + x^2 * ( 0.25 + x^2 * (
  0.015625 + x^2 * 0.0004816179)), exp( x) / sqrt( 2 * Pi * x) * (
  1 + ( 0.125 + ( 0.0703125 + ( 0.07324219 + 0.1794151 / x) / x) /
  x) / x));
```

$$I0sa := x \to piecewise\left(x < 2.562065,\ 1 + x^2 \left(0.25 + x^2 \left(0.015625 + 0.0004816179\, x^2\right)\right),\ \frac{e^x \left(1 + \dfrac{0.125 + \dfrac{0.0703125 + \dfrac{0.07324219 + \dfrac{0.1794151}{x}}{x}}{x}}{x} \right)}{\sqrt{2\,\pi\, x}} \right)$$

Plot the relative error of the spliced approximation.

```
> plot( I0sa( x) / BesselI( 0, x) - 1, x = 0 .. 5, color = black);
```



A Simple Approximation

Suppose we decide to write an approximation of the form:

```
> exp( x) / sqrt( 2 * Pi * f( x));
```

$$\frac{1}{2}\,\frac{e^{x}\sqrt{2}}{\sqrt{\pi f(x)}}$$

The exact value of f(x) is:

```
> f := x -> solve( exp( x) / sqrt( 2 * Pi * f) = BesselI( 0, x), f)
  ;
```

$$f := x \to solve\left(\frac{e^{x}}{\sqrt{2\,\pi f}} = \mathrm{BesselI}(0,x), f\right)$$

```
> f( x);
```

$$\frac{1}{2}\,\frac{\left(e^{x}\right)^{2}}{\pi\,\mathrm{BesselI}(0,x)^{2}}$$

Get limiting form for f as $x \to 0$:

```
> fLo = series( f( x), x, 5);
```

$$fLo = \frac{1}{2}\,\frac{1}{\pi} + \frac{1}{\pi}\,x + \frac{3}{4}\,\frac{1}{\pi}\,x^2 + \frac{1}{6}\,\frac{1}{\pi}\,x^3 - \frac{17}{192}\,\frac{1}{\pi}\,x^4 + O(x^5)$$

Get limiting form for f as $x \to \infty$, or try to. Maple balks.

```
> fHi = asympt( f( x), x, 5);
Error, (in asympt) unable to compute series
```

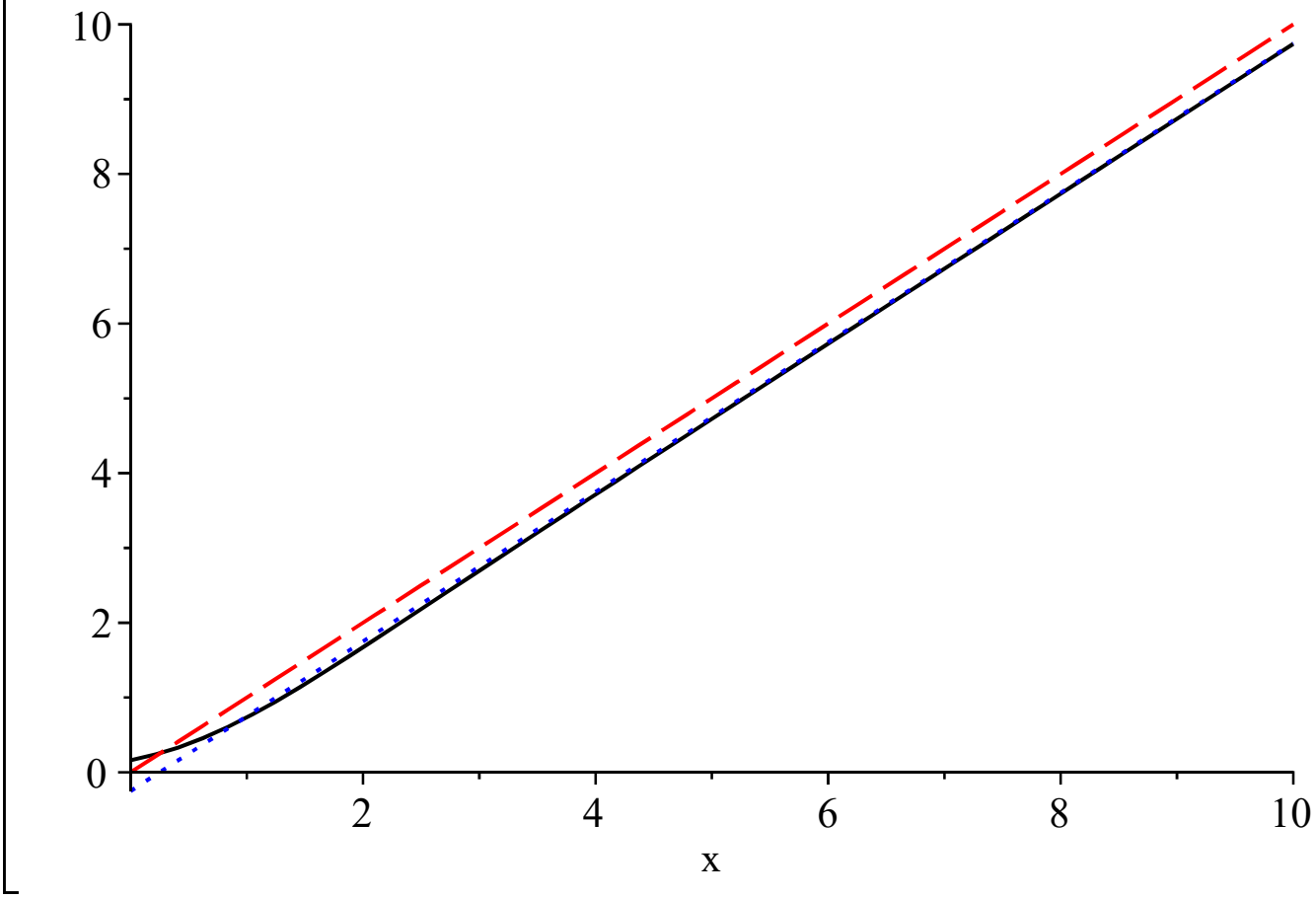
Use the asymptotic series for I0(x) we got previously. That does the trick.

```
> fHi = asympt( ( exp( x) / I0hi( x, 10))^2 / ( 2 * Pi), x);
```

$$fHi = x - \frac{1}{4} - \frac{3}{32}\,\frac{1}{x} - \frac{13}{128}\,\frac{1}{x^2} - \frac{341}{2048}\,\frac{1}{x^3} - \frac{2931}{8192}\,\frac{1}{x^4} + O\left(\frac{1}{x^5}\right)$$

Compare f(x) to the asymptotic value f = x and the somewhat better $f = x - \frac{1}{4}$:

```
> plot( [ f(x), x, x - 1/4], x = 0 .. 10, color=[black,red,blue],
  linestyle=[1,3,2]);
```



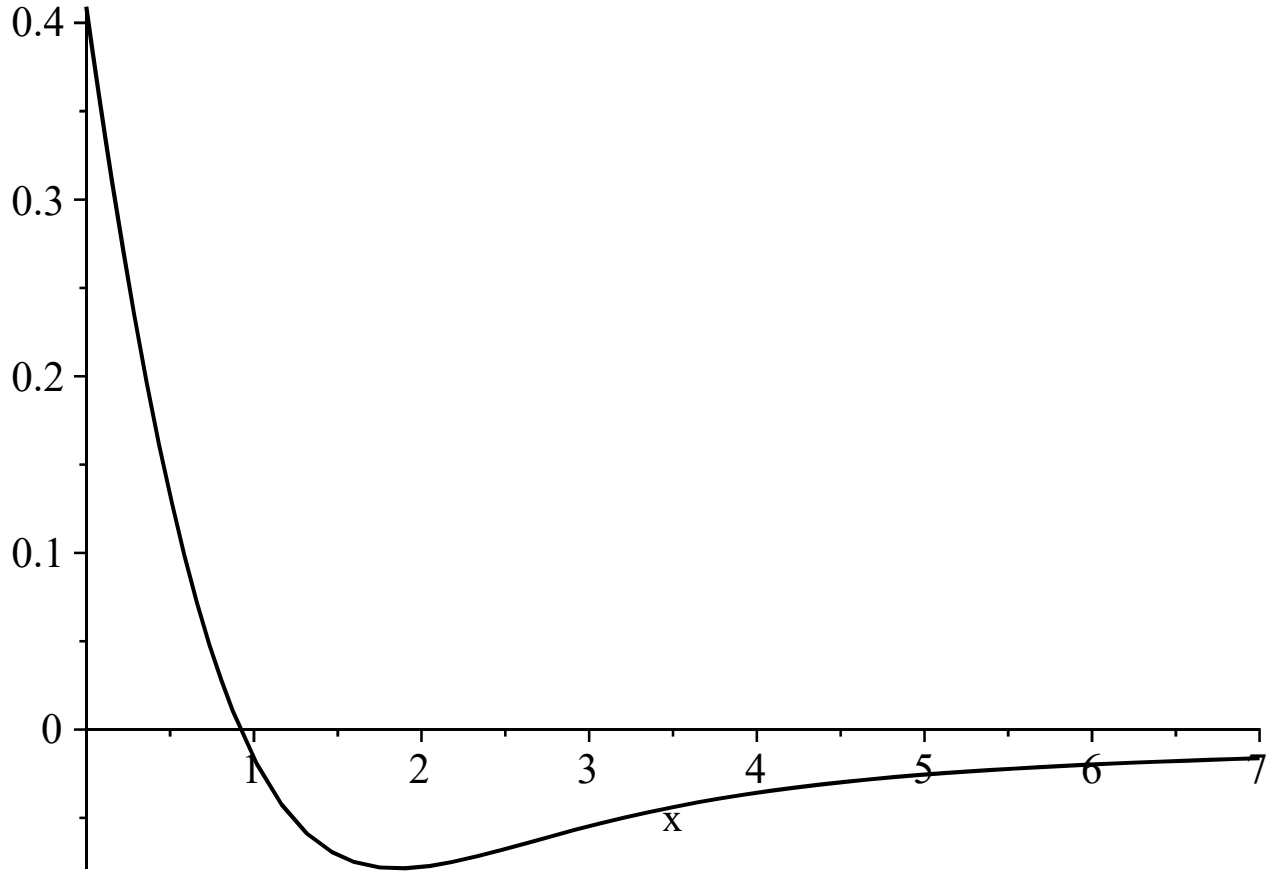
Use a fairly simple form to approximate f by adding a correction to the asymptotic expansion $f = x - \frac{1}{4}$:

```
> f1 = x - 1/4 + q;
```

$$f1 = x - \frac{1}{4} + q$$

The correction changes sign:

```
> plot( f(x) - x + 1/4, x = 0 .. 7, color=black);
```



Add two polynomial-style terms so the sign can switch:

```
> f1 := x -> x - 1/4 + c1/(x+c2) + c3/(x+c4)^2;
```

$$f1 := x \to x - \frac{1}{4} + \frac{c1}{x + c2} + \frac{c3}{(x + c4)^2}$$

Get the first coefficient by matching value at x = 0:

```
> unassign('c1','c2','c3','c4'); solve( f(0) = f1(0), {c1}); assign
  (%);
```

$$\left\{ c1 = \frac{1}{4}\ \frac{c2\left(2\,c4^2 + \pi\,c4^2 - 4\,c3\,\pi\right)}{\pi\,c4^2} \right\}$$

Get the other coefficients by matching at manually selected other points, adjusting them for an equal-relative-error fit:

```
> xvals:= {0.36,1.42,4.5};
```

$$xvals := \{0.36, 1.42, 4.5\}$$

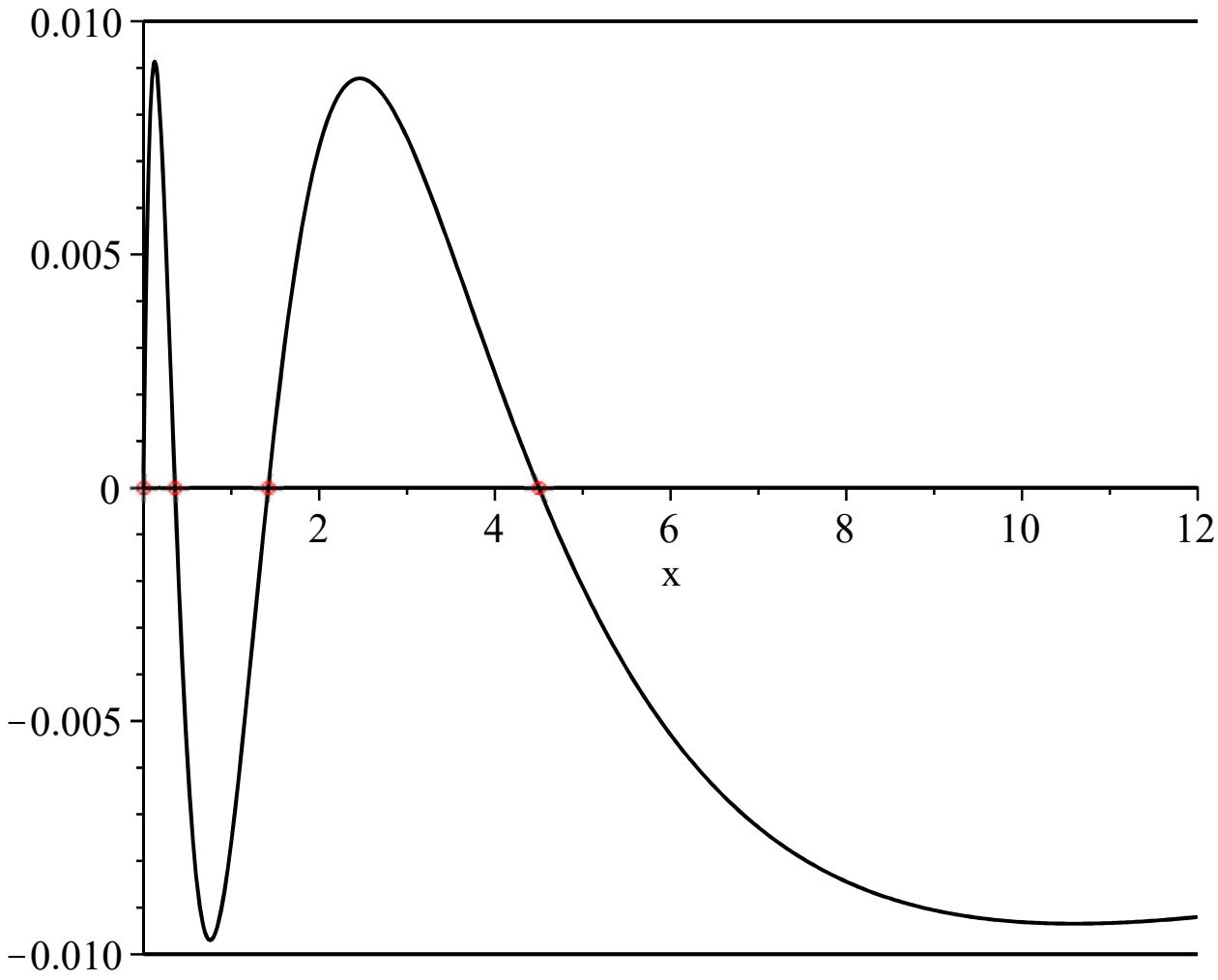
```
> unassign('c2','c3','c4'); fsolve( { seq( f(u) = f1(u), u = xvals)
  }, {c2,c3,c4}); assign(%); 'c1' = evalf( c1);
```

$$\{c2 = 2.922375205835818,\ c4 = 8.517976183027503,\ c3 = -402.5307157954558\}$$

$$c1 = 17.40865851378003$$

Show relative error with exact coefficients:

```
> plots[display]( [ plot( [ exp(x)/sqrt(2*Pi*f1(x)) / BesselI(0,x)
  - 1, 0, -0.01, 0.01], x = 0 .. 12, color=black, numpoints=300),
  plot( [ [0,0], seq( [x,0], x=xvals)], style=point, symbol=circle)
  ]);
```



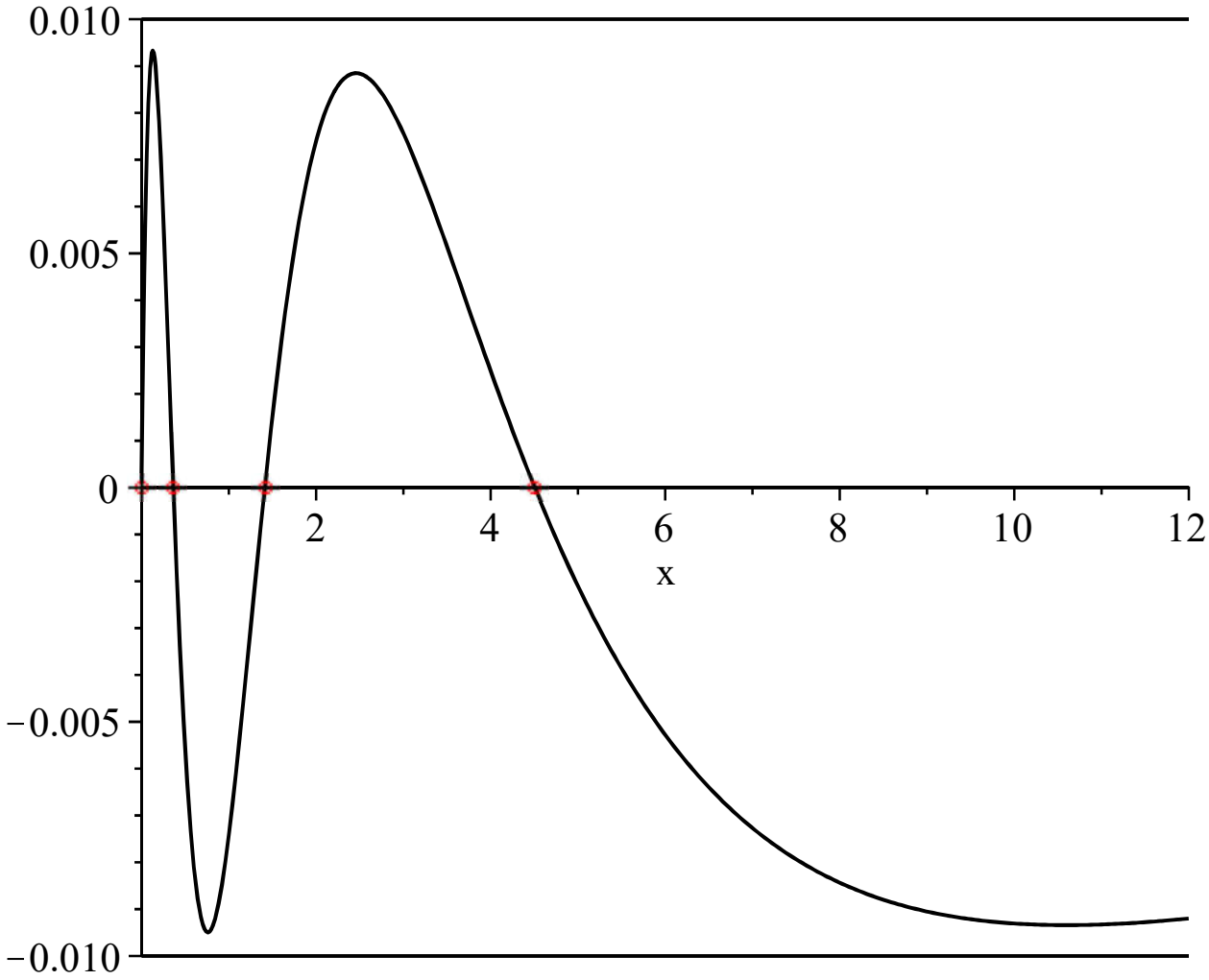
Tweak the coefficients (including 2*Pi) for finite precision:

```
> g1 := x -> exp(x)/sqrt(6.2832*x-1.5708+109.4/(x+2.922)-2530/
  (x+8.518)^2);
```

$$g1 := x \rightarrow \frac{e^{x}}{\sqrt{6.2832\,x - 1.5708 + \frac{109.4}{x + 2.922} - \frac{2530}{(x + 8.518)^{2}}}}$$

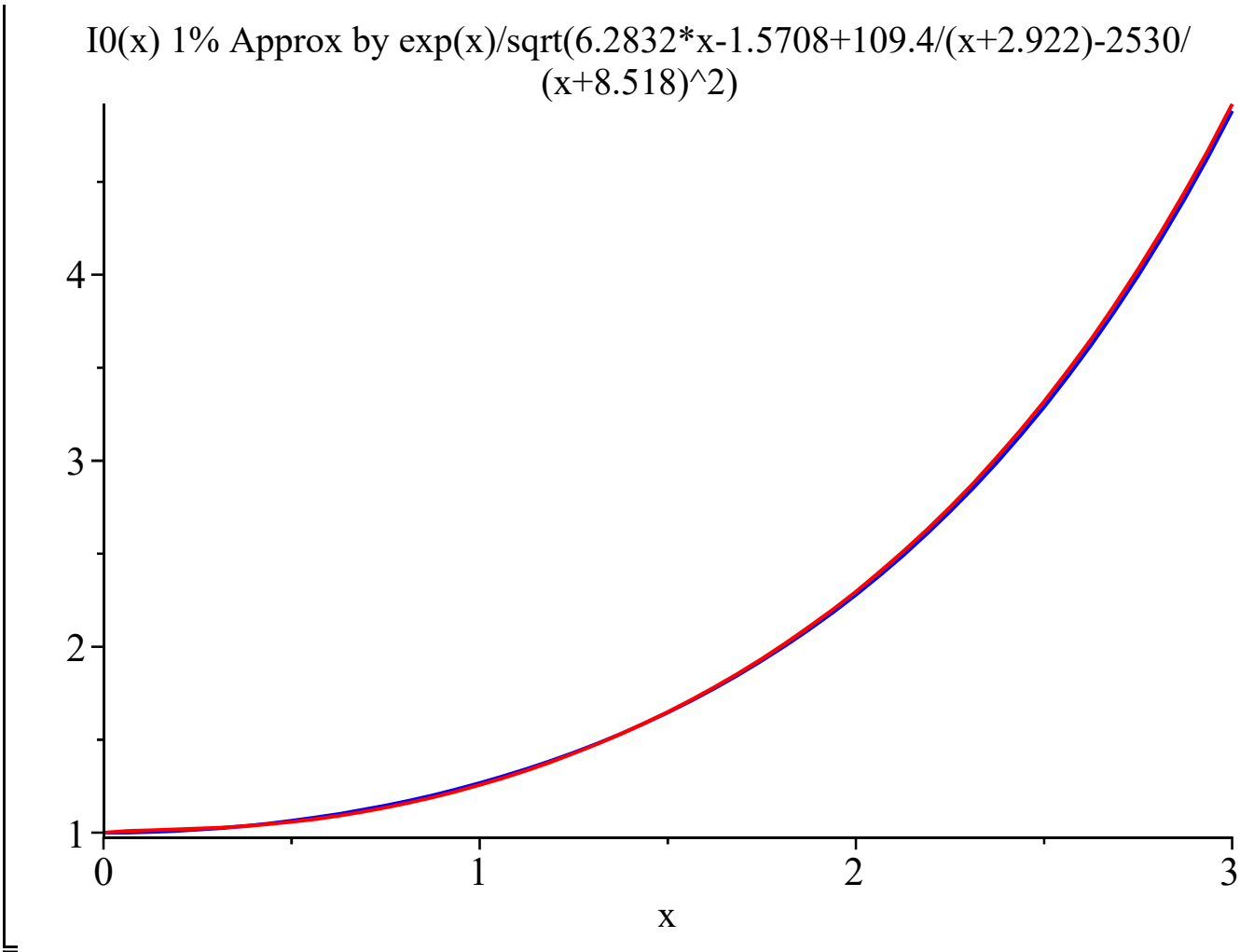
Show relative error with approximate coefficients:

```
> plots[display]( [ plot( [ g1(x) / BesselI(0,x) - 1, 0, -0.01,
  0.01], x = 0 .. 12, color=black, numpoints=300), plot( [ [0,0],
  seq( [x,0], x=xvals)], style=point, symbol=circle)]);
```

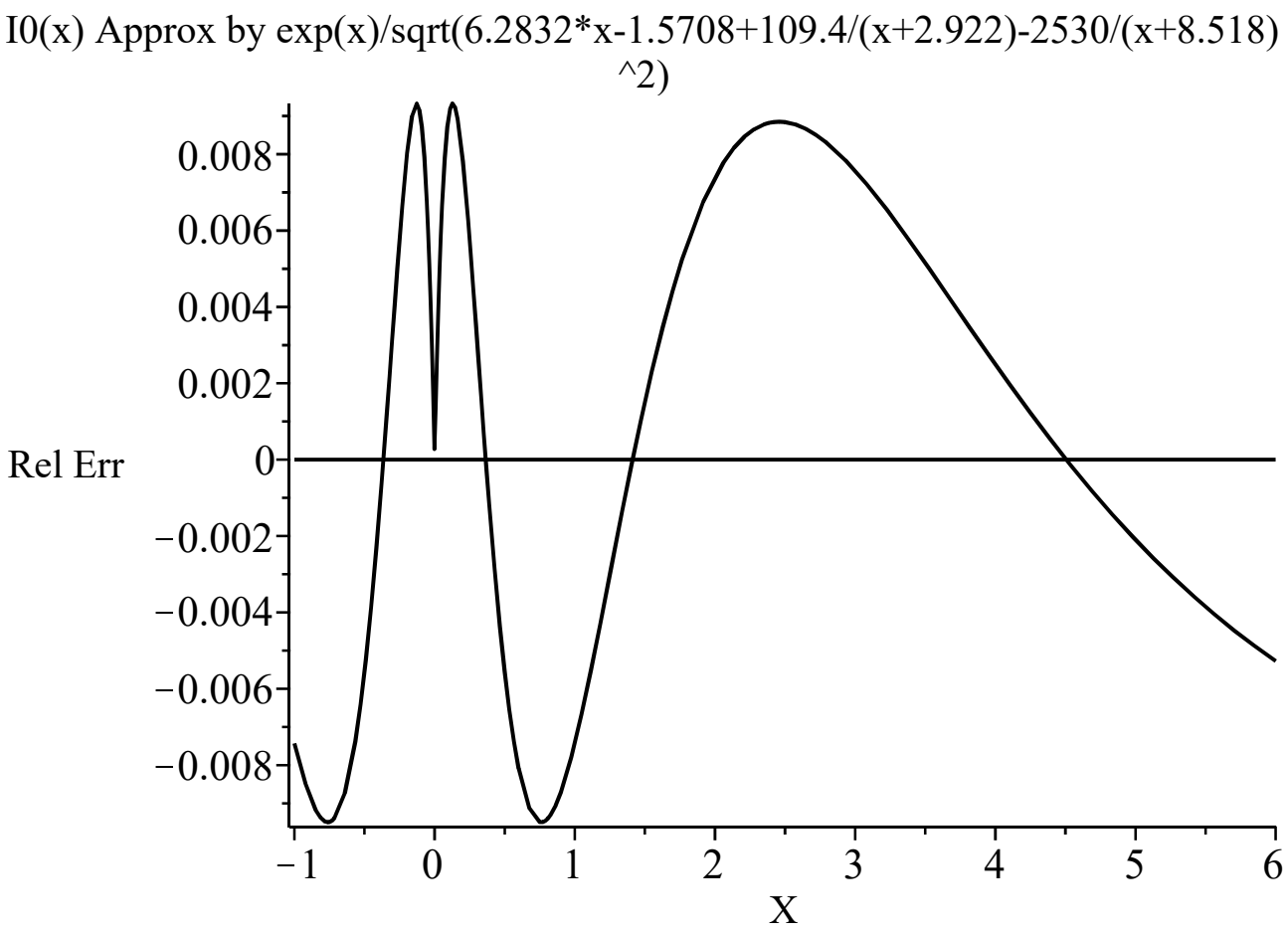


Our final answer, good to plus or minus 1% for $0 < x < \infty$ (but note that I0(-x) = I0(x), so we really use abs(x) in the formula - but beware the derivative discontinuity at x = 0):

```
> plot( [ BesselI(0,x), g1(x)], x = 0 .. 3, color=[blue,red],
  linestyle=[1,18], title="I0(x) 1% Approx by exp(x)/sqrt(6.2832*
  x-1.5708+109.4/(x+2.922)-2530/(x+8.518)^2)");
```



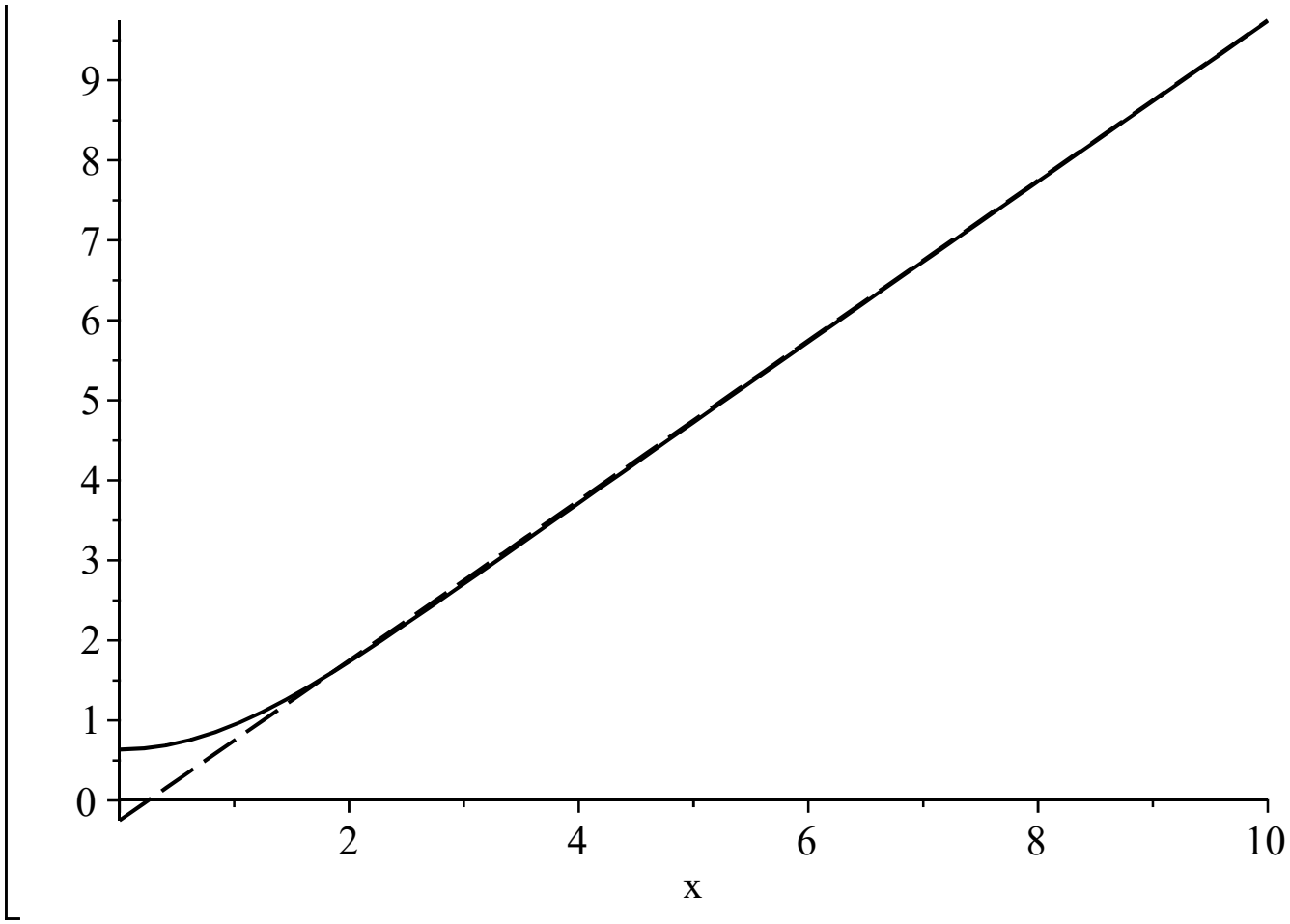

```
> plot( [ g1(abs(x)) / BesselI(0,x) - 1, 0], x = -1 .. 6, color=
  black, title="I0(x) Approx by exp(x)/sqrt(6.2832*x-1.5708+109.4/
  (x+2.922)-2530/(x+8.518)^2)", labels=["X","Rel Err"], axes=frame)
  ;
```



An Improved Approximation (0.0018 relative accuracy)

Try a different tack. We know I0(x) is symmetric around x = 0, so use an exponential that has that same behavior - namely $e^{x} + e^{-x}$(this is really cosh(x)). Then figure out something symmetric in x to put in the denominator. Write $I0(x) = \frac{e^{x} + e^{-x}}{\sqrt{2\,\pi\,Q}}$ and take a look at Q. It's clear that, for large x, Q looks like f, which in turn looks like $x - \frac{1}{4}$, so add that to the plot:

```
> plot( [ ((exp(x)+exp(-x)) / BesselI(0,x))^2 / (2*Pi), x - 1/4], x
  = 0 .. 10, color=black, linestyle=[1,3]);
```



Near x = 0, Q has the form:

```
> series( ((exp(x)+exp(-x)) / BesselI(0,x))^2 / (2*Pi), x, 11);
```

$$\frac{2\cdot 1}{\pi} + \frac{1}{\pi}x^2 - \frac{1}{48}\frac{1}{\pi}x^4 - \frac{17}{1440}\frac{1}{\pi}x^6 + \frac{5903}{1290240}\frac{1}{\pi}x^8 - \frac{142441}{116121600}\frac{1}{\pi}x^{10} + O(x^{11})$$

Somehow, we want to transition this behavior into the $x - \frac{1}{4}$ form at large x, while maintaining the symmetry around x = 0. Let's use the square root of something that goes as $4\,\pi^2\,x^2$ for large x. Perhaps a Pade form in x^2 will work. Let's use a [6,4] form so we will have 3 free coefficients after matching at low and high ends:

```
> unassign('p0','p2','p4','p6','q2','q4');
  f2 := x -> (exp(x)+exp(-x)) / ((p0+p2*x^2+p4*x^4+p6*x^6) / (1+q2*
  x^2+q4*x^4))^(1/4);
```

$$f2 := x \to \frac{e^x + e^{-x}}{\left(\frac{p0 + p2\,x^2 + p4\,x^4 + p6\,x^6}{1 + q2\,x^2 + q4\,x^4}\right)^{1/4}}$$

Match value and coefficient of x^2 in the the expansion at the low end (x = 0):

```
> series( f2(x), x, 3) = I0lo(x,2);
```

$$\frac{2\cdot 1}{p0^{1/4}} + \left(-\frac{1}{2}\,\frac{p2 - p0\,q2}{p0^{5/4}} + \frac{1}{p0^{1/4}}\right)x^2 + O(x^4) = 1 + \frac{1}{4}x^2 + \frac{1}{64}x^4$$

```
> p0 := solve( 2*1/(p0^(1/4)) = 1, p0);
```

$$p0 := 16$$

```
> p2 := simplify( solve( -1/2*(p2-p0*q2)/(p0^(5/4))+1/(p0^(1/4)) =
  1/4, p2));
```

$$p2 := 16\,q2 + 16$$

Match the expansion at the high end to eliminate another coefficient:

```
> asympt( f2(x) / exp(x), x, 3) = I0hi(x,2) / exp(x);
```

$$\frac{\sqrt{\frac{1}{x}}}{\left(\frac{p6}{q4}\right)^{1/4}} + \frac{1}{4}\,\frac{(-p4\,q4 + p6\,q2)\left(\frac{1}{x}\right)^{5/2}}{\left(\frac{p6}{q4}\right)^{1/4} q4\,p6} + O\left(\left(\frac{1}{x}\right)^{9/2}\right) = \frac{1}{2}\,\frac{\left(1 + \frac{1}{8\,x} + \frac{9}{128\,x^2}\right)\sqrt{2}}{\sqrt{\pi\,x}}$$

```
> p6 := solve( sqrt(2*Pi) = (p6/q4)^(1/4), p6);
```

$$p6 := 4\,\pi^2\,q4$$

Matched at both ends, here is our approximate form with 3 remaining free coefficients:

```
> f2(x);
```

$$\frac{e^x + e^{-x}}{\left(\frac{16 + (16\,q2 + 16)\,x^2 + p4\,x^4 + 4\,\pi^2\,q4\,x^6}{1 + q2\,x^2 + q4\,x^4}\right)^{1/4}}$$

Check that we match at low and high ends:

```
> evalf( series( f2(x), x, 5) = I0lo(x,2));
```

$$1.000000000000000 + 0.2500000000000000\,x^2 + (-0.01562500000000000\,p4 + 0.2500000000000000\,q4 + 0.2500000000000000\,q2 + 0.07291666666666664)\,x^4 + O(x^5)$$

$$= 1. + 0.2500000000000000\,x^2 + 0.01562500000000000\,x^4$$

```
> evalf( asympt( f2(x)/exp(x), x, 2) = I0hi(x,0)/exp(x));
```

$$0.3989422804014328\,\sqrt{\frac{1}{x}} + O\left(\left(\frac{1}{x}\right)^{5/2}\right) = \frac{0.3989422804014326 \cdot 1}{\sqrt{x}}$$

Once again, manually adjust three x values at which the approximate and exact function values are matched to get the three coefficients. Set the x values for good behavior and equal relative error at the error extrema:

```
> xvals:= [1.925,4.7,19.45];
```

$$xvals := [1.925, 4.7, 19.45]$$

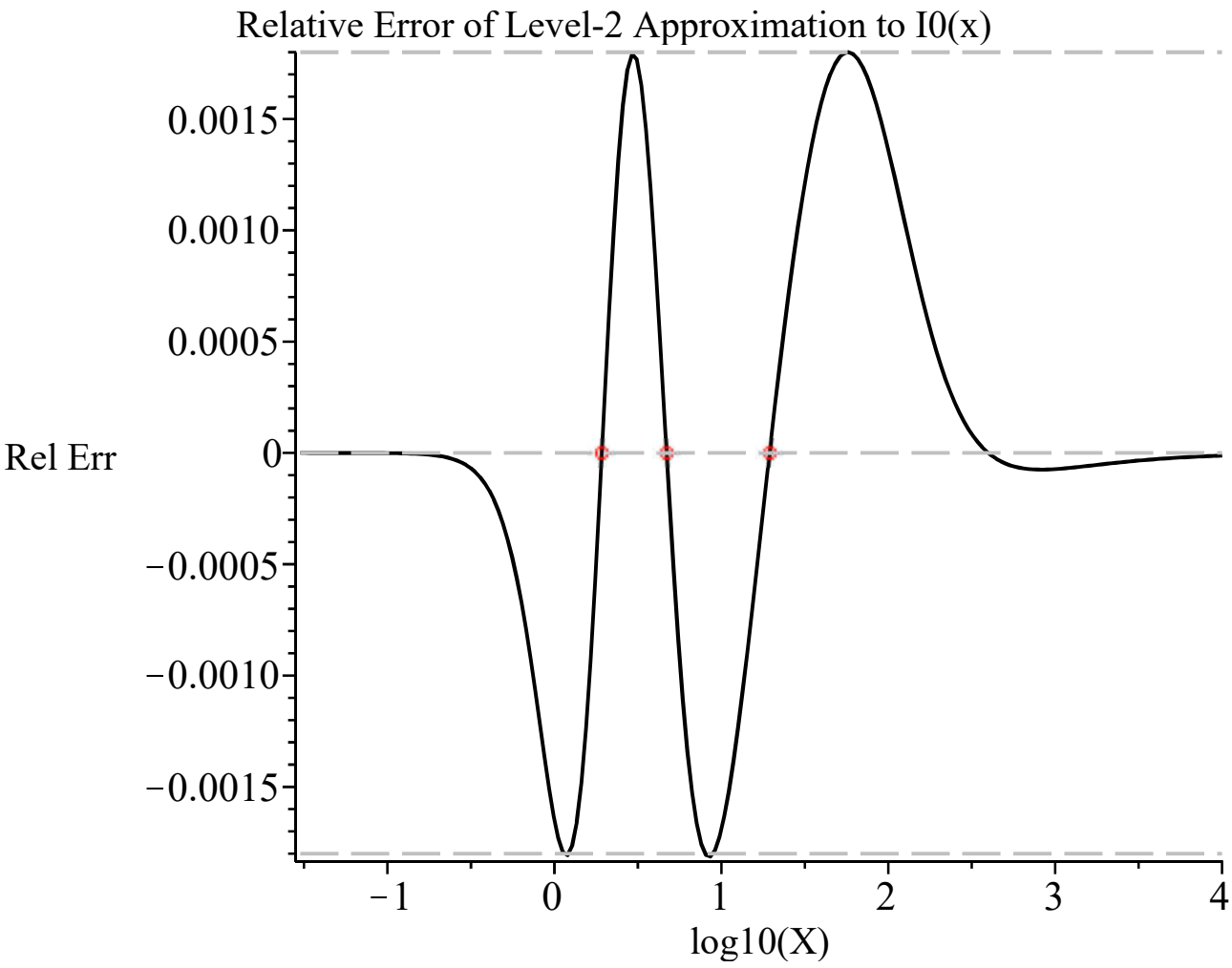
```
> unassign('p4','q2','q4'); fsolve( { seq( BesselI(0,u) = f2(u), u
  = xvals)}, {p4,q2,q4}); assign(%);
```

$$\{q2 = 0.1848272656481030,\ q4 = 0.00001738201482251659,\ p4 = 7.151179411263451\}$$

Plot error curve. Force use of our own evenly-spaced-log points along the axis, because "semilogplot" is not smart about this:

```
> xA := 0.03: xB := 10000: N := 201:
  xLogs := [seq(evalf(log10(xA)+(log10(xB)-log10(xA))*(j-1)/(N-1)),
```

```
j=1..N)]:
plots[display]( [ plot( [ map( z -> [z, evalf( f2(10.0^z) /
BesselI(0,10.0^z) - 1)],xLogs)], color=black, labels=["log10(X)",
"Rel Err"], title="Relative Error of Level-2 Approximation to I0
(x)", titlefont=[TIMES,BOLD,16], axes=framed), plot( [ seq( [z,
0], z=map(log10,xvals))], style=point, symbol=circle), plot(
[-0.0018,0,0.0018], x=xLogs[1]..xLogs[nops(xLogs)], color=gray,
linestyle=3)]);
```



Trim the precision in the coefficients:

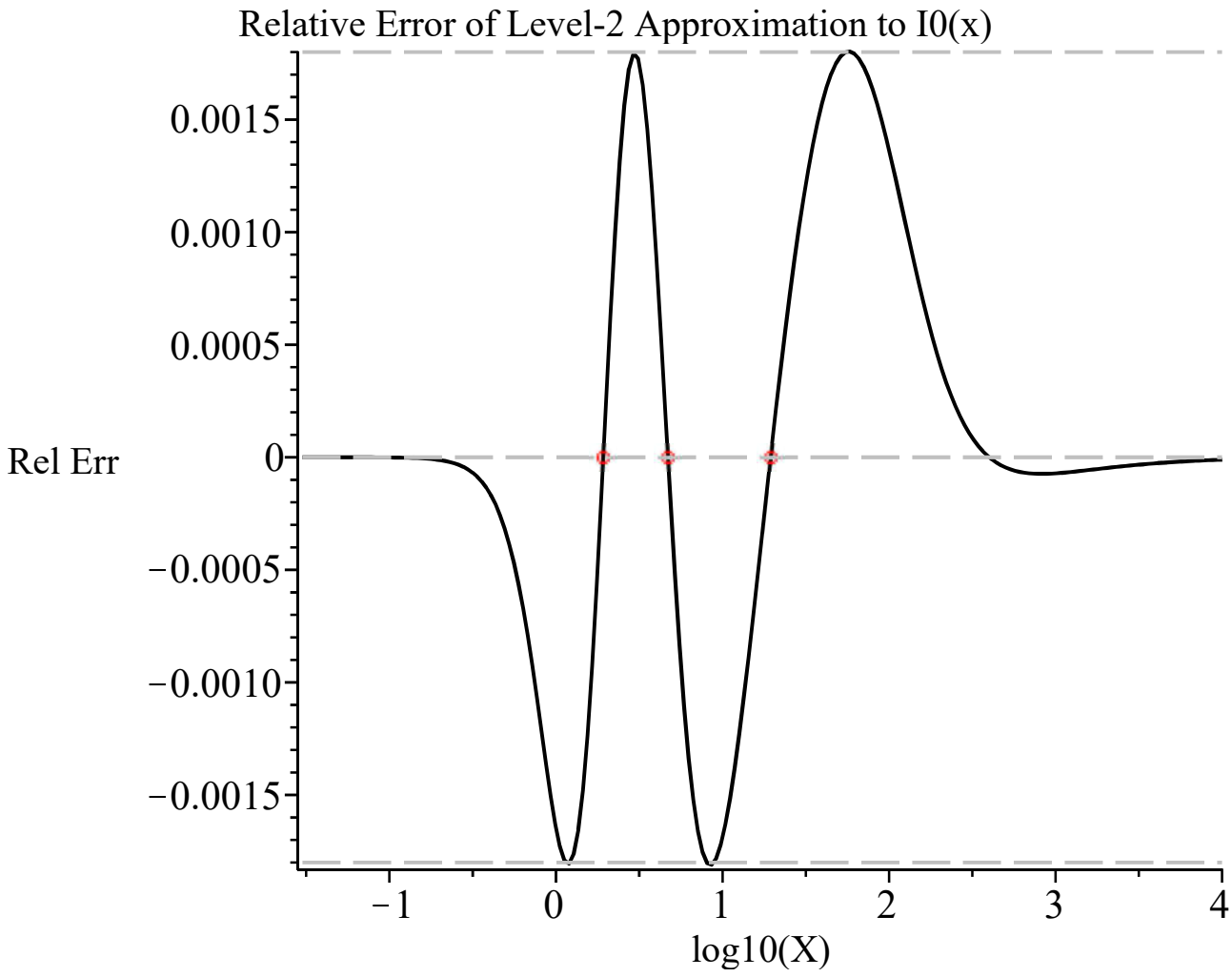
```
> evalf( f2(x));
```

$$\frac{e^{x}+e^{-1.x}}{\left(\frac{16.+18.95723625036965\,x^{2}+7.151179411263451\,x^{4}+0.0006862144399684407\,x^{6}}{1.+0.1848272656481030\,x^{2}+0.00001738201482251659\,x^{4}}\right)^{1/4}}$$

```
> g2 := x -> (exp(x)+exp(-x)) / ((16.0+x^2*(18.957+x^2*(7.1512+x^2*
  0.00068621))) / (1.0+x^2*(0.18483+x^2*0.000017382)))^(1/4);
```

$$g2 := x \rightarrow \frac{e^{x} + e^{-x}}{\left(\frac{16.0 + x^2 \left(18.957 + x^2 \left(7.1512 + 0.00068621\, x^2 \right) \right)}{1.0 + x^2 \left(0.18483 + 0.000017382\, x^2 \right)} \right)^{1/4}}$$

```
> plots[display]( [ plot( [ map( z -> [z, evalf( g2(10.0^z) /
  BesselI(0,10.0^z) - 1)],xLogs)], color=black, labels=["log10(X)",
  "Rel Err"], title="Relative Error of Level-2 Approximation to I0
  (x)", titlefont=[TIMES,BOLD,16], axes=framed), plot( [ seq( [z,
  0], z=map(log10,xvals))], style=point, symbol=circle, thickness=
  3), plot( [-0.0018,0,0.0018], x=xLogs[1]..xLogs[nops(xLogs)],
  color=gray, linestyle=3)]);
```



A Higher-Accuracy Approximation (6e-5 relative accuracy)

We will now get a symmetric approximation good to a relative accuracy of 6e-5. Write a first approximation that is symmetric and has the correct behavior at x = 0 and x = ∞. Then fix it up with a multiplicative function fix(x). The first approximation is:

```
> (exp(x)+exp(-x))/(16+(2*Pi*x)^2)^(1/4);
```

$$\frac{e^{x}+e^{-x}}{\left(16+4\,\pi^{2}\,x^{2}\right)^{1/4}}$$

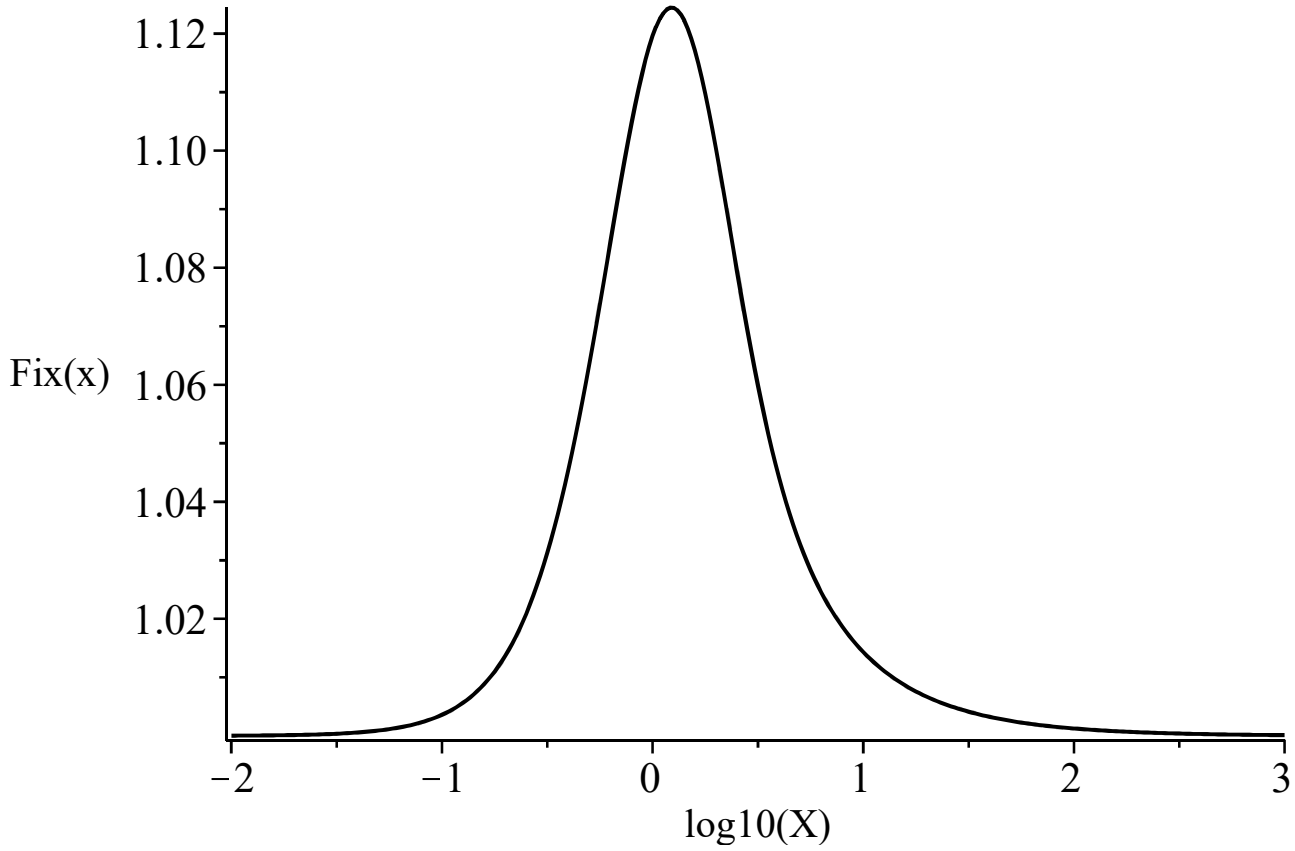
This is equal to 1 at x = 0 and has the correct asymptotic behavior for x going to both ∞ and $-\infty$. The fixup function is then:

```
> fix := x -> BesselI(0,x)*(16+(2*Pi*x)^2)^(1/4)/(exp(x)+exp(-x));
```

$$fix := x \rightarrow \frac{\mathrm{BesselI}(0,x)\left(16+4\,\pi^{2}\,x^{2}\right)^{1/4}}{e^{x}+e^{-x}}$$

This is a fairly well-behaved bump that rises from unity to a value of 1.12 around x = 1.2:

```
> xA := 0.01: xB := 1000: N := 301:
  xLogs := [seq(evalf(log10(xA)+(log10(xB)-log10(xA))*(j-1)/(N-1)),
  j=1..N)]:
  plot( [ map( z -> [z, evalf( fix(10.0^z))], xLogs)], labels=
  ["log10(X)","Fix(x)"], color=black, axes=framed);
```



Write log-spaced values of fix(x) out to a text file for TableCurve2D:

```
> fd := fopen("c:\\Temp\\fixup.txt", WRITE):
  for j from 1 to nops( xLogs) do
    fprintf( fd, "%.8g, %.8g\n", xLogs[j], evalf( fix( 10.0^xLogs
  [j])));
  od:
  fclose( fd);
```

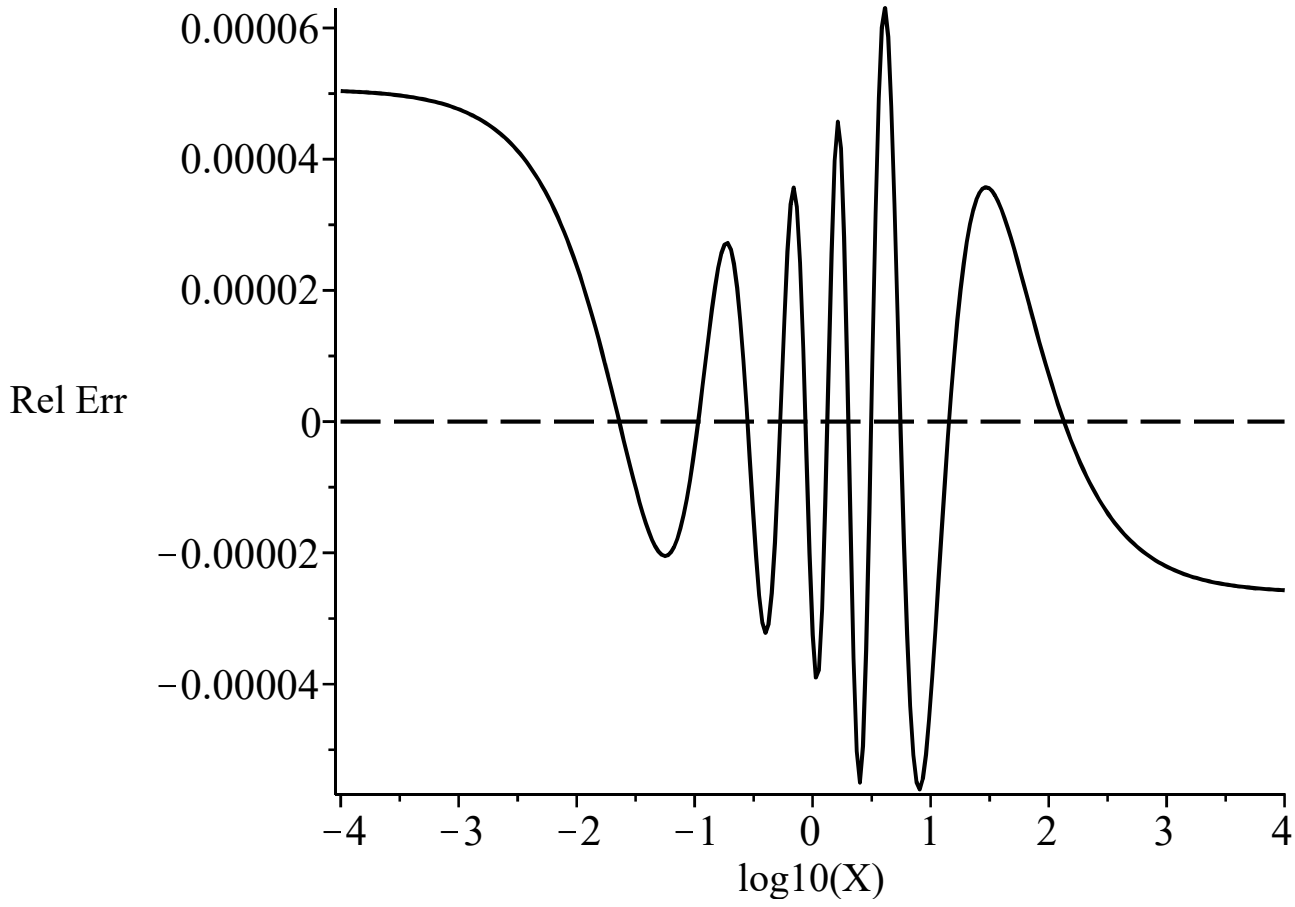
TableCurve2D produces a nice fit:

```
> fix55 := x -> (1.0000507+x*(0.49797944+x*(0.88376019+x*(1.7698894
  -x*(0.67396198-x*0.52169071)))))/(1+x*(0.50107809+x*(0.47307199+
  x*(1.8177196-x*(0.74132953-x*0.52170433)))));
```

$$fix55 := x \to (1.0000507 + x\,(0.49797944 + x\,(0.88376019 + x\,(1.7698894 - x\,(0.67396198 - 0.52169071\,x))))) \Big/ (1 + x\,(0.50107809 + x\,(0.47307199 + x\,(1.8177196 - x\,(0.74132953 - 0.52170433\,x)))))$$

Plot the relative error of that approximation:

```
> xA := 0.0001: xB := 10000: N := 301:
  xLogs := [seq(evalf(log10(xA)+(log10(xB)-log10(xA))*(j-1)/(N-1)),
  j=1..N)]:
  plot( [ map( z -> [z, fix55(10.0^z) / fix(10.0^z) - 1], xLogs), [
  [xLogs[1],0],[xLogs[nops(xLogs)],0]]], labels=["log10(X)","Rel
  Err"], color=black, linestyle=[1,3],axes=frame);
```



Let's improve on this and force zero slope at the origin. Define a Pade function that has the correct asymptotic behaviors:

```
> fixZ := x -> (1.0+x*(pa+x*(pb+x*(pc+x*(pd+x*pe))))) / (1.0+x*(pa+
  x*(qb+x*(qc+x*(qd+x*pe)))));
```

$$fixZ := x \to \frac{1.0 + x\,(pa + x\,(pb + x\,(pc + x\,(pd + x\,pe))))}{1.0 + x\,(pa + x\,(qb + x\,(qc + x\,(qd + x\,pe))))}$$

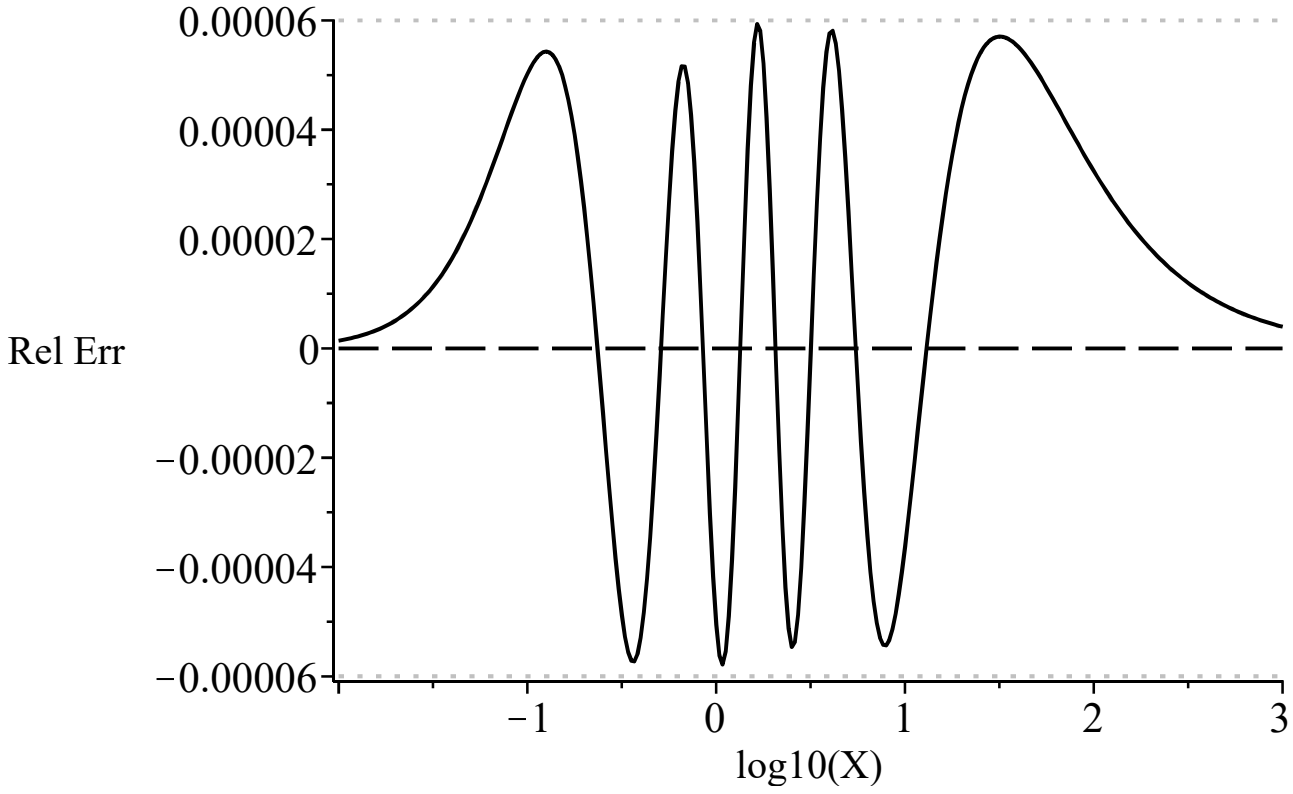
Manually adjust the positions of the zeroes to get equal-ripple behavior:

```
> z := [0.235,0.51,0.85,1.34,2.06,3.17,5.48,13];
  unassign( 'pa','pb','pc','pd','pe', 'qb','qc','qd');
  soln := solve( { seq( evalf( fix(x) = fixZ(x)), x = z) }, {pa,pb,
  pc,pd,pe,qb,qc,qd});
```

$$z := [0.235, 0.51, 0.85, 1.34, 2.06, 3.17, 5.48, 13]$$

$$soln := \{pa = 0.2791988813867496, qb = 0.7721897219048464, qc = 1.495838567914915, pc = 1.459928420517920, qd = -0.6116455388286841, pd = -0.5486342685569403, pb = 1.154442620160509, pe = 0.4883030649430555\}$$

```
> assign(soln);
  xA := 0.01: xB := 1000: N := 301: e := 6.0e-5:
  xLogs := [ seq( evalf( log10( xA) + ( log10( xB) - log10( xA)) *
  ( j - 1) / ( N - 1)), j = 1 .. N)]:
  plot( [ map( z -> [z, fixZ(10.0^z) / fix(10.0^z) - 1], xLogs), [
  [xLogs[1],0],[xLogs[nops(xLogs)],0]], [[xLogs[1],e],[xLogs[nops
  (xLogs)],e]], [[xLogs[1],-e],[xLogs[nops(xLogs)],-e]]], labels=
  ["log10(X)","Rel Err"], color = [ black, black, gray, gray],
  linestyle = [ 1, 3, 2, 2], axes = frame);
```



Trim the precision of the coefficients:

```
> f6x := evalf( fixZ(x), 6);
```

$$f6x := \frac{1.0 + x\,(0.279199 + x\,(1.15444 + x\,(1.45993 + x\,(-0.548634 + 0.488303\,x))))}{1.0 + x\,(0.279199 + x\,(0.772190 + x\,(1.49584 + x\,(-0.611646 + 0.488303\,x))))}$$

```
> fix6 := unapply( f6x, x);
```

$$fix6 := x \rightarrow \frac{1.0 + x\,(0.279199 + x\,(1.15444 + x\,(1.45993 + x\,(-0.548634 + 0.488303\,x))))}{1.0 + x\,(0.279199 + x\,(0.772190 + x\,(1.49584 + x\,(-0.611646 + 0.488303\,x))))}$$

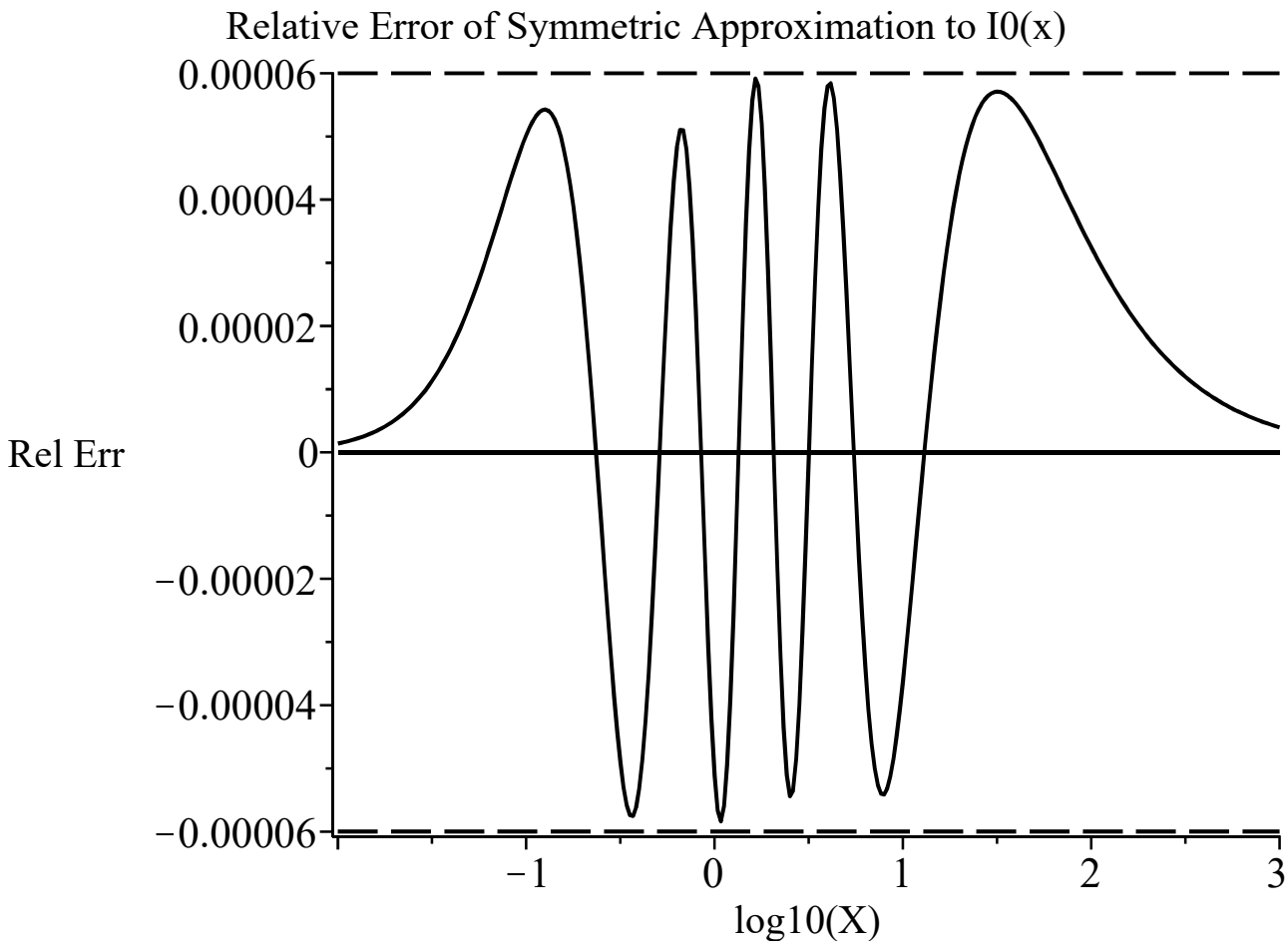
Define the approximate I0 function:

```
> I0ap := x -> fix6(x)*(exp(x)+exp(-x))/(16+(2*Pi*x)^2)^(1/4);
```

$$I0ap := x \rightarrow \frac{fix6(x)\,\left(e^{x} + e^{-x}\right)}{\left(16 + 4\,\pi^{2}\,x^{2}\right)^{1/4}}$$

The relative error of the approximation to I0 mirrors the relative error of the fixup function. Note, however, that we must use abs(x) in fix6(x) (or just use abs(x) throughout):

```
> xA := 0.01: xB := 1000: N := 301: e := 6e-5:
  xLogs := [ seq( evalf( log10( xA) + ( log10( xB) - log10( xA)) *
  ( j - 1) / ( N - 1)), j = 1 .. N)]:
  P := plot( [ map( z -> [z, I0ap(10.0^z) / BesselI(0,10.0^z) - 1],
  xLogs), [[xLogs[1],0],[xLogs[nops(xLogs)],0]], [[xLogs[1],e],
  [xLogs[nops(xLogs)],e]], [[xLogs[1],-e],[xLogs[nops(xLogs)],-e]]
  ], labels = [ "log10(X)", "Rel Err"], title = "Relative Error of
  Symmetric Approximation to I0(x)", titlefont=[TIMES,BOLD,16],
  color=black, linestyle=[1,1,3,3],axes=frame): print( P);
```



Save the present state, write a PostScript file, and restore the state:

```
> mydevice := interface( plotdevice):
  myoutput := interface( plotoutput):
  interface( plotdevice = ps,   plotoptions=`color,portrait,width=
  8in,height=6in,noborder`,
    plotoutput="C:\\Temp\\FitI0plot1.ps");
  print( P);
  interface( plotdevice=mydevice, plotoutput=myoutput);
```

A Two-Part Fit

It looks tough to beat 6e-5 by the above approach. Let's try fitting good low-end and high-end approximations together.

Low end: procedure from Atlas of Functions, adjusted to give single-precision accuracy:

```
> I0fn := proc( x)
    local j, f, r;
    j := 2*( 8 + floor(x));
    f := 1;
    r := x / j;
    for j from j to 2 by -2 do
      f := 1 + f*r;
      r := x / ( x*r + j);
    od;
    evalf( exp(x) / ( 1 + 2*f*r))
  end;
```

```
I0fn := proc(x)
    local j, f, r;
    j := 16 + 2 * floor(x);
    f := 1;
    r := x/j;
    for j from j by -2 to 2 do f := 1 + f*r; r := x/(x*r+j) end do;
    evalf(exp(x) / (1 + 2*f*r))
end proc
```

Test some values:

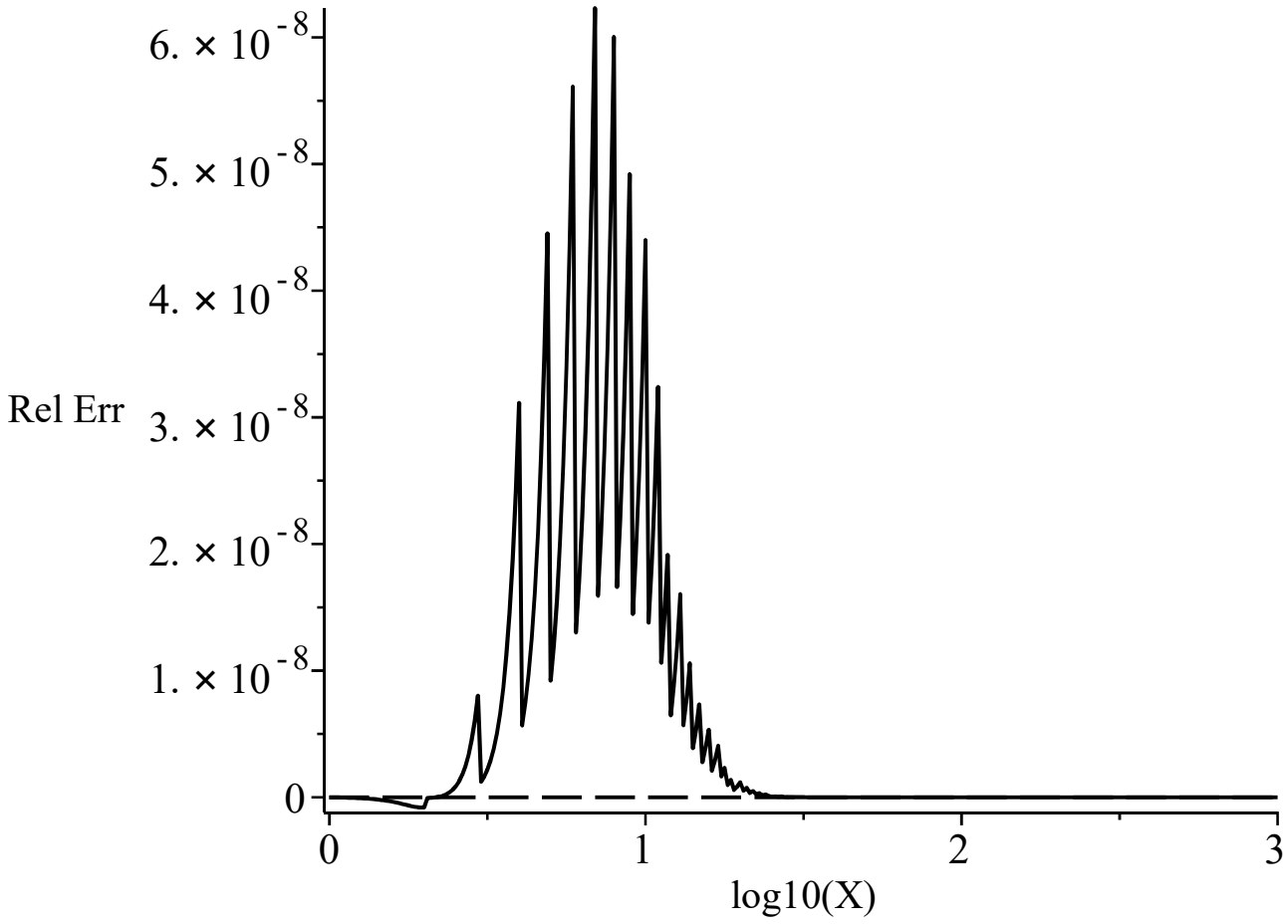
```
> I0fn(3) = evalf( BesselI(0,3));
                    4.880792591346579 = 4.880792585865024
```

```
> I0fn(6.9) = evalf( BesselI(0,6.9));
                    153.6990057349862 = 153.6989964353272
```

The approximation has discontinuities where the term count jumps, but it is very accurate:

```
> xA := 1: xB := 1000: N := 301:
```

```
xLogs := [seq(evalf(log10(xA)+(log10(xB)-log10(xA))*(j-1)/(N-1)),
j=1..N)]:
plot( [ map( z -> [z, I0fn(10.0^z) / BesselI(0,10.0^z) - 1],
xLogs), [[xLogs[1],0],[xLogs[nops(xLogs)],0]]], labels=["log10(X)
","Rel Err"], color=black, linestyle=[1,3],axes=frame);
```



This is good, but slow for large x because of the high iteration count.

For large x, we have $I0 \rightarrow \frac{e^x}{\sqrt{Q(x)}}$ where Q(x) is:

```
> unassign('j','k'); asympt( (exp(x)/I0hi(x,infinity))^2, x, 7);
```

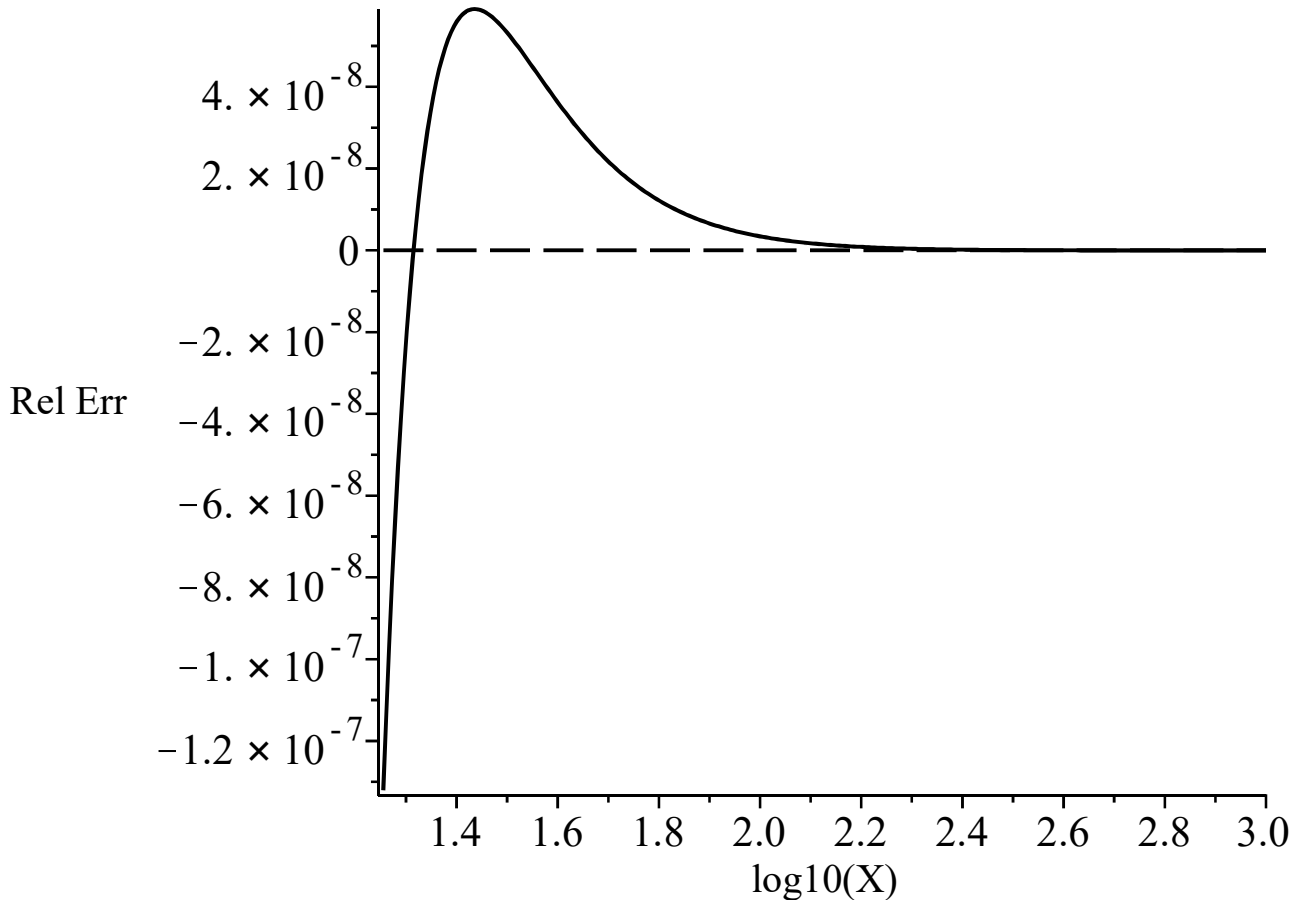
$$undefined\,\pi\,x$$

For large x (x > 21), a simple 4-term form gives better than single-precision accuracy (the last coefficient is an adjusted value):

```
> I0big := x -> exp(x) / sqrt( 2*Pi*x - 1.570796 - 0.5890486/x -
  0.695/x^2);
```

$$I0big := x \rightarrow \frac{e^x}{\sqrt{2\,\pi\,x - 1.570796 - \frac{0.5890486}{x} - \frac{0.695}{x^2}}}$$

```
> xA := 18: xB := 1000: N := 301:
  xLogs := [seq(evalf(log10(xA)+(log10(xB)-log10(xA))*(j-1)/(N-1)),
  j=1..N)]:
  plot( [ map( z -> [z, I0big(10.0^z) / BesselI(0,10.0^z) - 1],
  xLogs), [[xLogs[1],0],[xLogs[nops(xLogs)],0]]], labels=["log10(X)
  ","Rel Err"], color=black, linestyle=[1,3],axes=frame);
```



Define a single-precision combination procedure that switches from one form to the other to be both good and fast for all x:

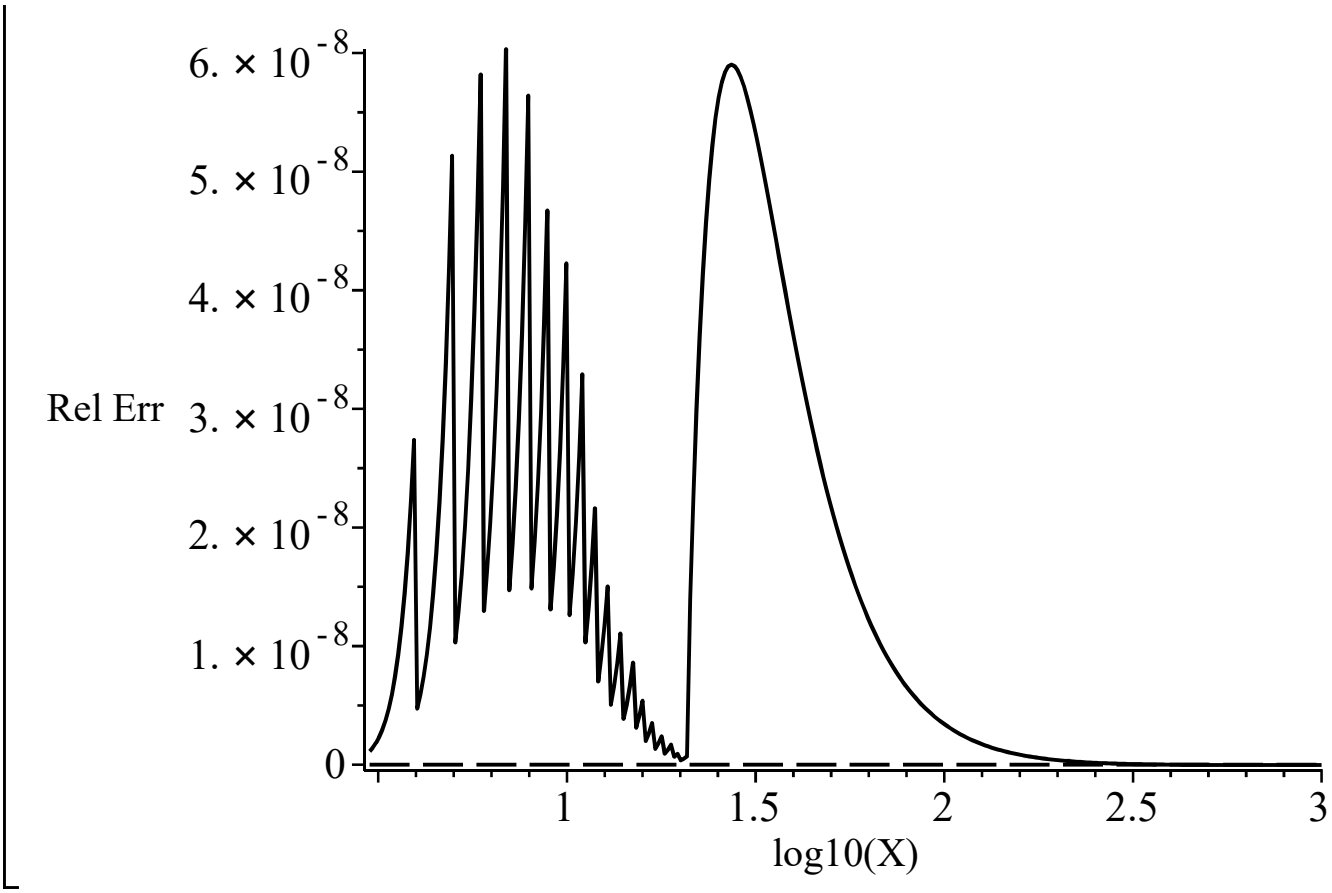
```
> I0sp := proc( x)
    local j, f, r, u;
    u := abs( x);
    if u < 21 then
      j := 2*( 8 + floor(u));
      f := 1;
      r := u / j;
      for j from j to 2 by -2 do
        f := 1 + f*r;
        r := u / ( u*r + j);
      od;
      evalf( exp(u) / ( 1 + 2*f*r))
```

```
    else
      evalf( exp(x) / sqrt( 2*Pi*u - 1.570796 - 0.5890486/u -
  0.695/u^2))
    fi
  end;
```

```
I0sp := proc(x)
    local j, f, r, u;
    u := abs(x);
    if u < 21 then
        j := 16 + 2 * floor(u);
        f := 1;
        r := u / j;
        for j from j by -2 to 2 do f := 1 + f * r; r := u / (u * r + j) end do;
        evalf(exp(u) / (1 + 2 * f * r))
    else
        evalf(exp(x) / sqrt(2 * Pi * u - 1.570796 - 0.5890486 / u - 0.695 / u^2))
    end if
end proc
```

We get a reasonably fast, very accurate, noisy fit - good values, but terrible derivatives:

```
> xA := 3: xB := 1000: N := 301:
  xLogs := [seq(evalf(log10(xA)+(log10(xB)-log10(xA))*(j-1)/(N-1)),
  j=1..N)]:
  plot( [ map( z -> [z, I0sp(10.0^z) / BesselI(0,10.0^z) - 1],
  xLogs), [[xLogs[1],0],[xLogs[nops(xLogs)],0]]], labels=["log10(X)
  ","Rel Err"], color=black, linestyle=[1,3],axes=frame);
```



AMS55 Style Two-piece Approximation

First show AMS55 version (this is due to E. E. Allen) designed for equal-ripple absolute error:

Low-end symmetric polynomial:

```
> I0amsLo := x -> 1 + (x/3.75)^2*(3.5156229 + (x/3.75)^2*(3.0899424
  + (x/3.75)^2*(1.2067492 + (x/3.75)^2*(0.2659732 + (x/3.75)^2*
  (0.0360768 + (x/3.75)^2*0.0045813)))));
```

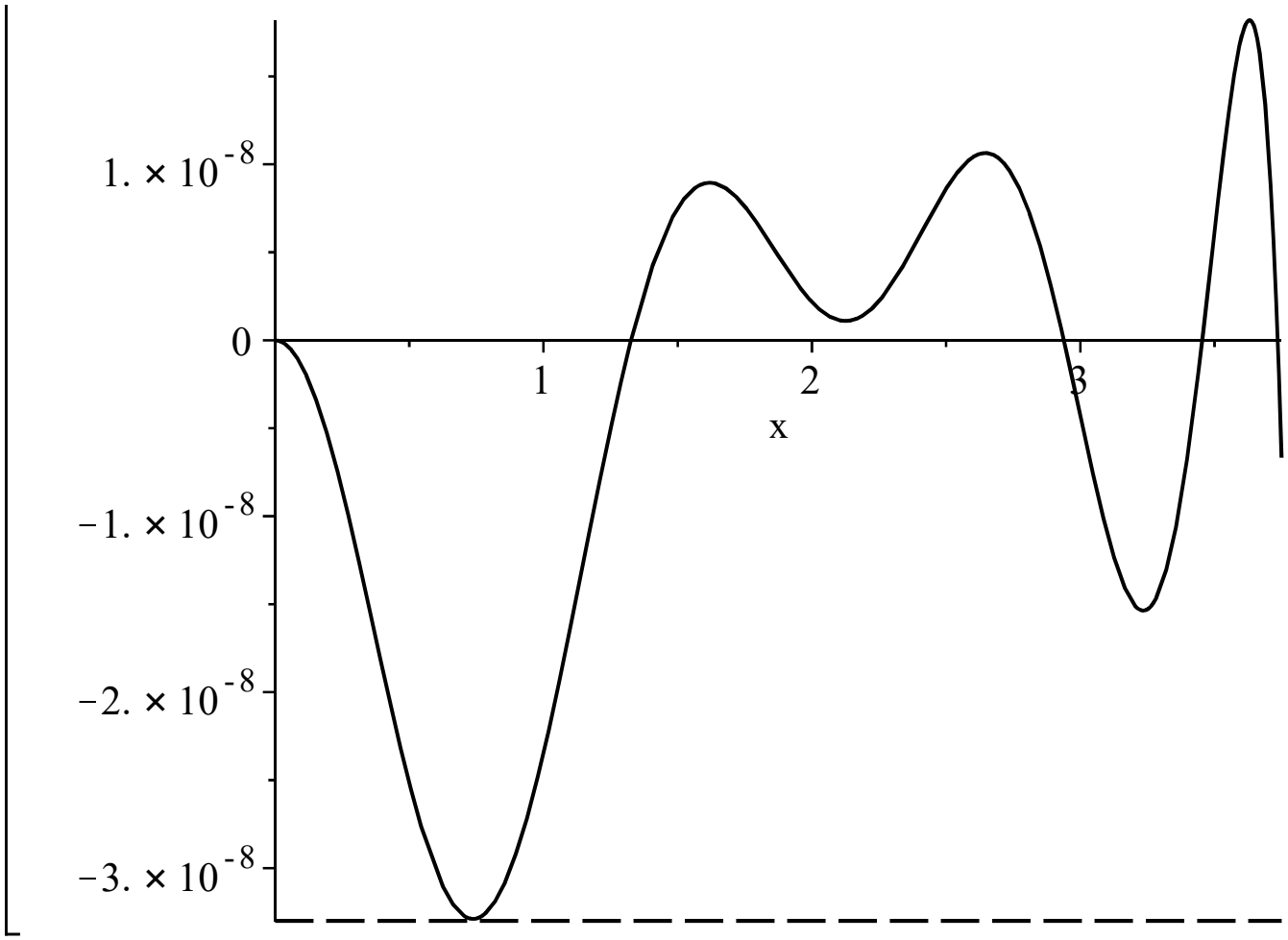
$$I0amsLo := x \to 1 + 0.07111111111111113\,x^2\,(3.5156229 + 0.07111111111111113\,x^2\,(3.0899424 + 0.07111111111111113\,x^2\,(1.2067492 + 0.07111111111111113\,x^2\,(0.2659732 + 0.07111111111111113\,x^2\,(0.0360768 + 0.0003257813333333334\,x^2)))))$$

```
> expand( I0amsLo(x));
```

$$1 + 0.2499998506666667\,x^2 + 0.01562519021037038\,x^4 + 0.0004339397287857342\,x^6 + 0.00000680123430362933\,x^8 + 6.560173590760768\,10^{-8}\,x^{10} + 5.923979120365895\,10^{-10}\,x^{12}$$

The relative error is about 3.3e-8 (recall it was designed for equal absolute error):

```
> plot( [ I0amsLo(x) / BesselI(0,x) - 1, -3.3e-8], x = 0 .. 3.75,
  color=black, linestyle=[1,3]);
```



High end is $\frac{e^x}{\sqrt{x}}$ times a polynomial in 1/x:

```
> I0amsHiP := x -> 0.39894228 + (3.75/x)*(0.01328592 + (3.75/x)*
  (0.00225319 + (3.75/x)*(-0.00157565 + (3.75/x)*(0.00916281 +
  (3.75/x)*(-0.02057706 + (3.75/x)*(0.02635537 + (3.75/x)*
  (-0.01647633 + (3.75/x)*0.00392377))))))):
```

$$I0amsHiP := x \rightarrow 0.39894228 + \frac{1}{x}\left(3.75\left(0.01328592 + \frac{1}{x}\left(3.75\left(0.00225319 + \frac{1}{x}\left(3.75\left(-0.00157565 + \frac{1}{x}\left(3.75\left(0.00916281\right.\right.\right.\right.\right.\right.$$

$$+\frac{3.75\left(-0.02057706+\frac{3.75\left(0.02635537+\frac{3.75\left(-0.01647633+\frac{0.0147141375}{x}\right)}{x}\right)}{x}\right)}{x}$$

$$\Bigg)\Bigg)\Bigg)\Bigg)\Bigg)\Bigg)\Bigg)$$

```
> expand( I0amsHiP(x));
```

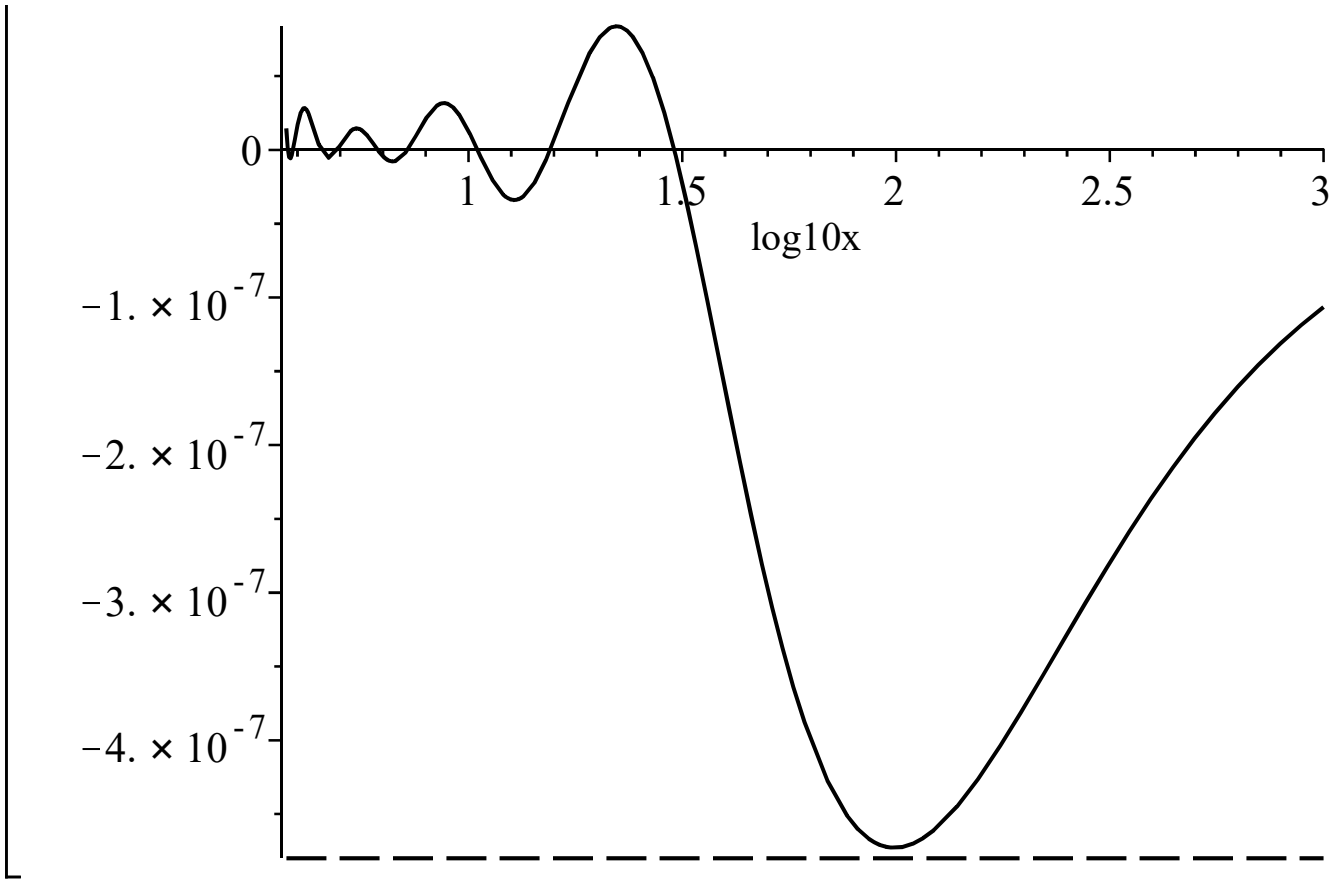
$$0.39894228+\frac{0.0498222000}{x}+\frac{0.031685484375}{x^2}-\frac{0.08309091796875}{x^3}+\frac{1.811981469726562}{x^4}$$
$$-\frac{15.25947747802734}{x^5}+\frac{73.29202548980711}{x^6}-\frac{171.8222318172454}{x^7}$$
$$+\frac{153.4453330636024}{x^8}$$

```
> I0amsHi := x -> I0amsHiP(x)*exp(x)/sqrt(x);
```

$$I0amsHi := x \rightarrow \frac{I0amsHiP(x)\ e^{x}}{\sqrt{x}}$$

The relative error is about 4.8e-7, with a big bulge around x = 100 (again, this was designed for absolute error):

```
> plot( [ I0amsHi(10.0^log10x) / BesselI(0,10.0^log10x) - 1,
  -4.8e-7], log10x = evalf(log10(3.75)) .. evalf(log10(1e3)), color
  = black, linestyle=[1,3]);
```



An Improved AMS-style Approximation

Improve on this using Maple. First define an approximate crossover or splice point:

```
> xS1 := 4;
```

$$xS1 := 4$$

Low End

Get the low-end approximation up to the crossover by fitting a function of $u = x^2$:

```
> tLo := numapprox[minimax]( BesselI(0,sqrt(u)), u = 0 .. xS1^2,
  [3,3], 1 + 10000 / ( 1 + 6 * u), 'e');
  `maximum error`=e;
```

$$tLo := (1.211596702934360 + (0.2725822948805788 + (0.01169864414408521 + 0.0001371117867692495\,u)\,u)\,u) \Big/ (1.211596706229317 + (-0.03031696072773909 + (0.0003468374801733865 - 0.00000184173320628716\,u)\,u)\,u)$$

$$maximum\ error = 0.0000271978825163$$

Normalize denominator:

```
> tLo2 := numapprox[pade]( tLo, u, [3,3]);
```

$$tLo2 := (0.9999999972804834 + 0.2249777450525584\,u + 0.009655559547114687\,u^2 + 0.0001131661930610254\,u^3) \Big/ (1.000000000000000 - 0.02502232019274000\,u + 0.0002862647928887170\,u^2 - 0.00000152008766351016 5\,u^3)$$

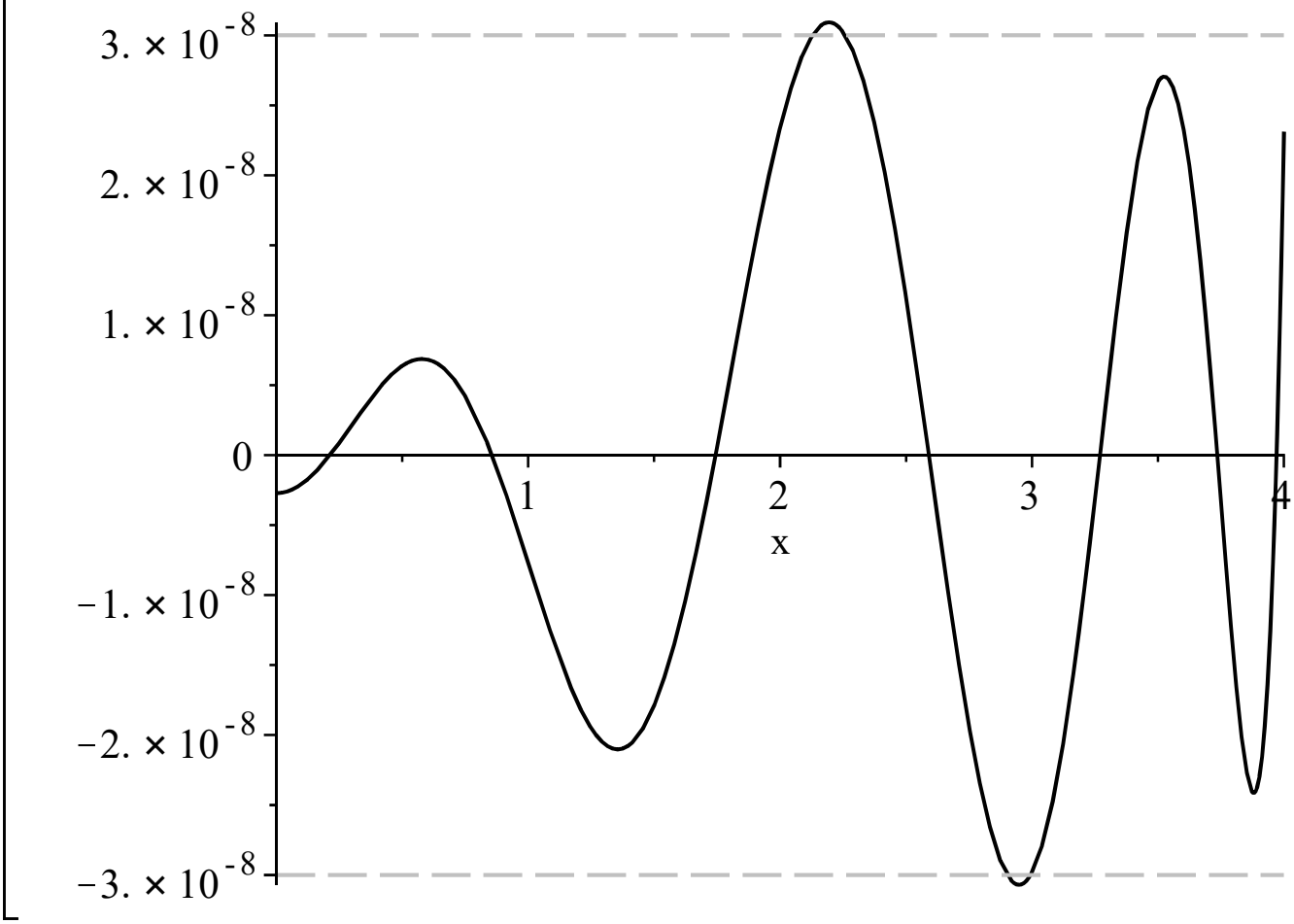
Convert back to x^2 and define as a function of x:

```
> I0mmLo := unapply( subs( u = x^2, tLo2), x);
```

$$I0mmLo := x \to (0.9999999972804834 + 0.2249777450525584\,x^2 + 0.009655559547114687\,x^4 + 0.0001131661930610254\,x^6) \Big/ (1.000000000000000 - 0.02502232019274000\,x^2 + 0.0002862647928887170\,x^4 - 0.000001520087663510165\,x^6)$$

Plot the relative error of the low-end approximation:

```
> plot( [ I0mmLo(x) / BesselI(0,x) - 1, 3e-8, -3e-8], x = 0 .. xS1,
  color=[black,gray,gray], linestyle=[1,3,3]);
```



Adjust the coefficients for lower precision:

```
> evalf( I0mmLo(x), 10);
```

$$\frac{0.9999999973 + 0.2249777451\,x^2 + 0.009655559547\,x^4 + 0.0001131661931\,x^6}{1.000000000 - 0.02502232019\,x^2 + 0.0002862647929\,x^4 - 0.000001520087664\,x^6}$$

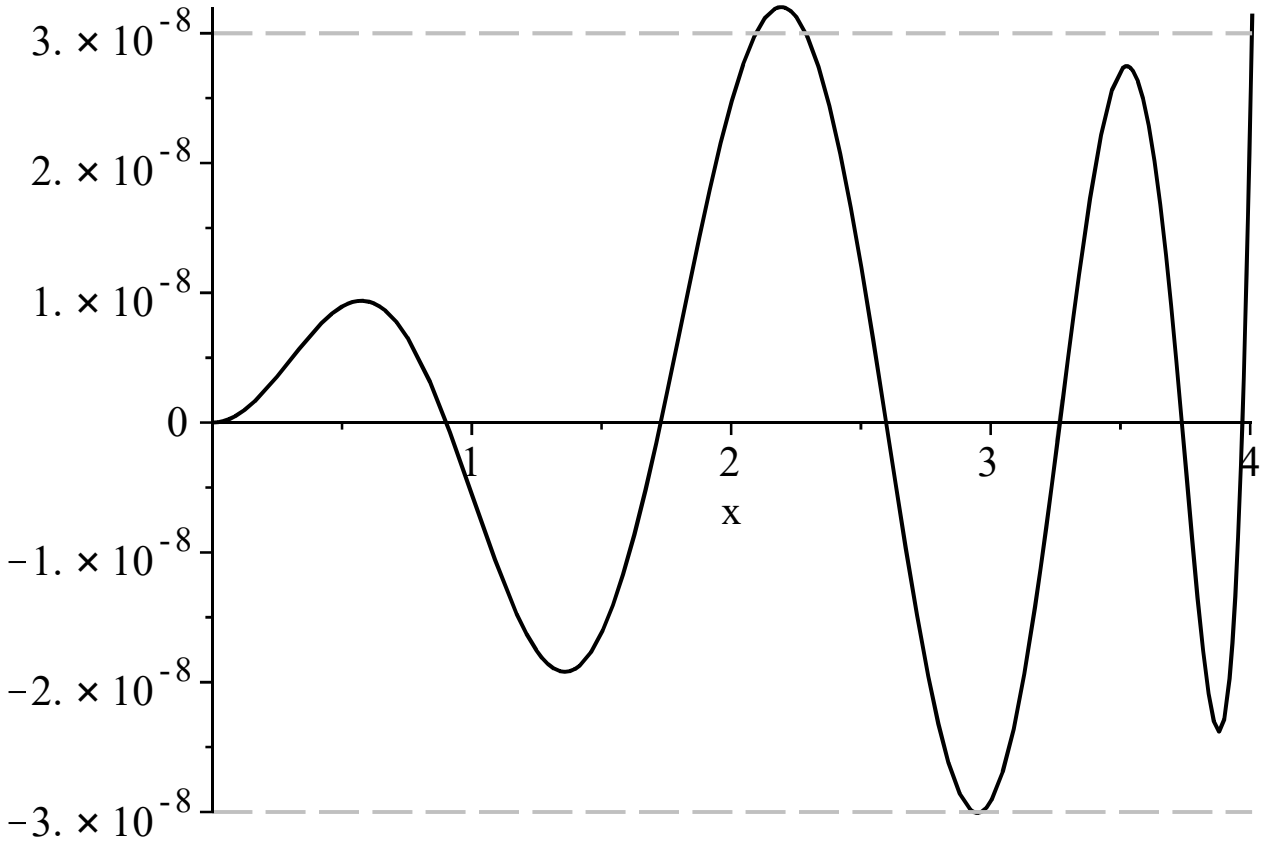
Tweak a bit by hand to force unity at x = 0:

```
> I0mLo := x -> (1 + .2249777450*x^2 + .9655559546e-2*x^4 +
  .1131661930e-3*x^6) / (1 - .2502232020e-1*x^2 + .2862647930e-3*
  x^4 - .1520087664e-5*x^6);
```

$$I0mLo := x \rightarrow \frac{1 + 0.2249777450\,x^2 + 0.009655559546\,x^4 + 0.0001131661930\,x^6}{1 - 0.02502232020\,x^2 + 0.0002862647930\,x^4 - 0.000001520087664\,x^6}$$

Plot the relative error of the low-end approximation as tweaked:

```
> plot( [ I0mLo( x) / BesselI( 0, x) - 1, 3e-8, -3e-8], x = 0 ..
  1.002 * xS1, color = [ black, gray, gray], linestyle = [ 1, 3, 3]
  );
```



Let's adjust this for equal-ripple error. Define a form with adjustable coefficients.

```
> tLo3 := x -> ( 1.0 + x^2 * ( ps + x^2 * ( pt + x^2 * pu))) / (
  1.0 + x^2 * ( qs + x^2 * ( qt + x^2 * qu)));
```

$$tLo3 := x \rightarrow \frac{1.0 + x^2\,(ps + x^2\,(pt + x^2\,pu))}{1.0 + x^2\,(qs + x^2\,(qt + x^2\,qu))}$$

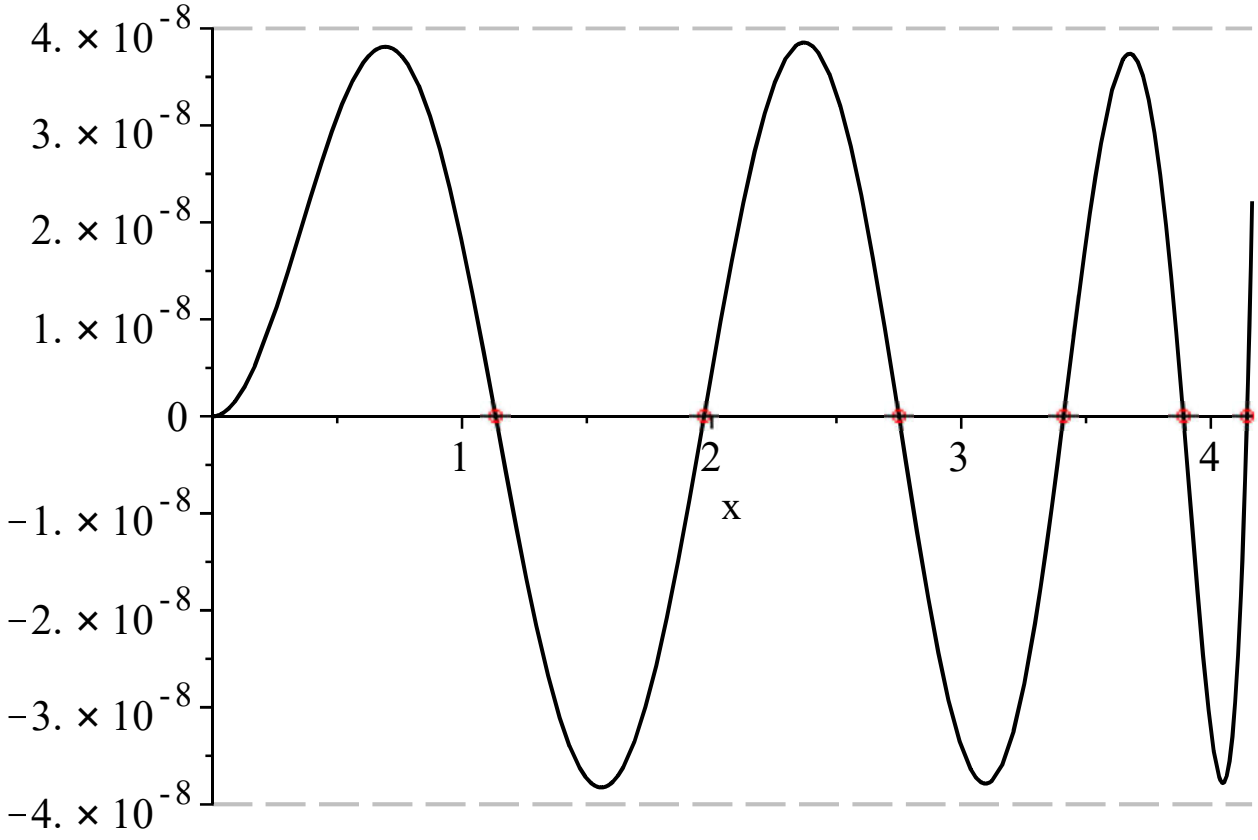
Manually adjust the position of the zeros for equal-ripple behavior.

```
> zeros := [ 1.135, 1.97, 2.75, 3.41, 3.89, 4.145];
  unassign( 'ps','pt','pu','qs','qt', 'qu');
  soln3 := solve( { seq( evalf( tLo3( x) = BesselI( 0, x)), x =
  zeros) }, { ps, pt, pu, qs, qt, qu});
```

$$zeros := [1.135, 1.97, 2.75, 3.41, 3.89, 4.145]$$

$$soln3 := \{pt = 0.009693409864300103,\ pu = 0.0001146169853847578,\ qu = -0.00000145965874012372 4,\ ps = 0.2251516395286773,\ qt = 0.0002808018654710827,\ qs = -0.02484854838406377\}$$

```
> assign( soln3);
  plots[ display]( [
  plot( [ tLo3( x) / BesselI( 0, x) - 1, 4e-8, -4e-8], x = 0 ..
  1.005 * zeros[ 6], color = [ black, gray, gray], linestyle = [ 1,
  3, 3]),
  plot( [ seq( [ z, 0], z = zeros)], style = point, symbol =
  circle, thickness = 3)]);
```



Trim the precision of the coefficients.

```
> evalf( tLo3( x), 10);
```

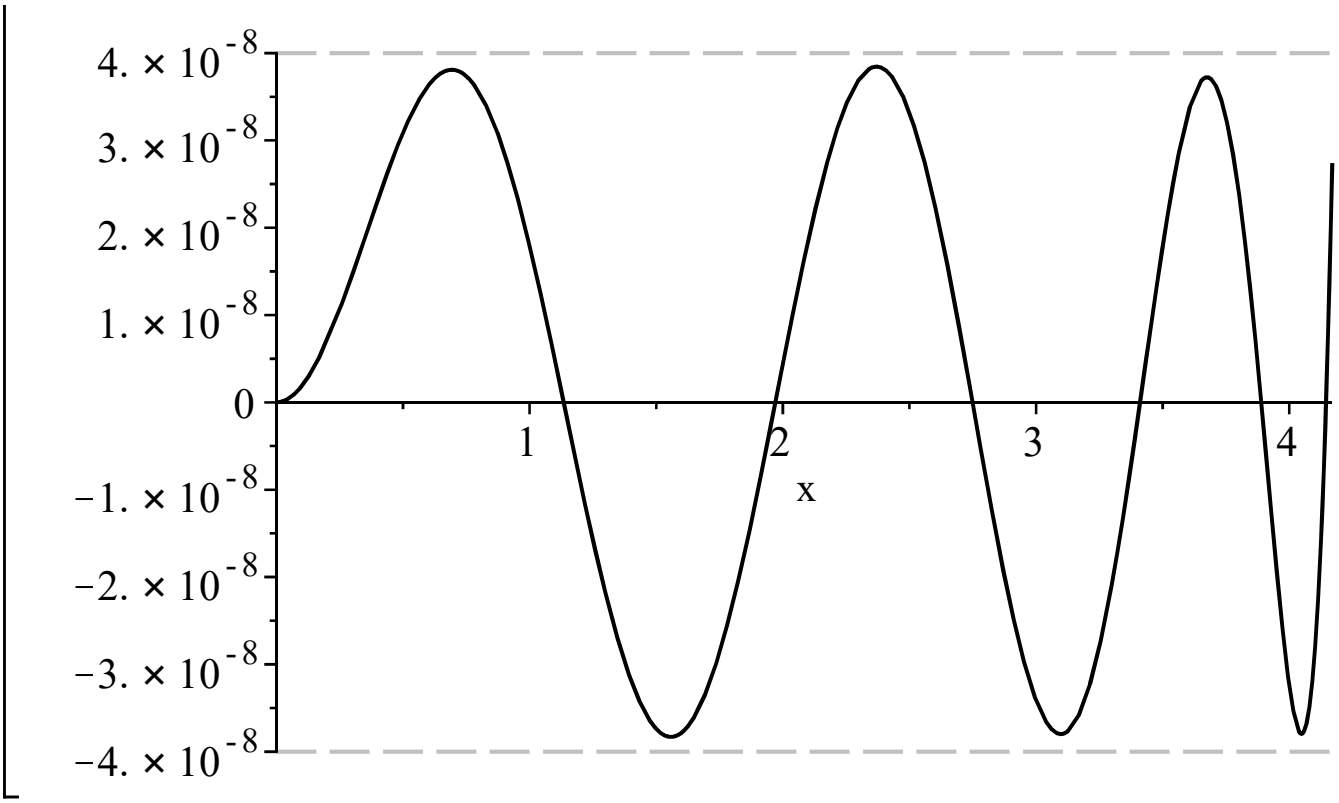
$$\frac{1.0 + x^2 \left(0.2251516395 + x^2 \left(0.009693409864 + 0.0001146169854\, x^2\right)\right)}{1.0 + x^2 \left(-0.02484854838 + x^2 \left(0.0002808018655 - 0.000001459658740\, x^2\right)\right)}$$

Define the low-end approximation.

```
> I0mLo := x -> ( 1.0 + x^2 * ( 0.2251516395 + x^2 * (
  0.9693409864e-2 + 0.1146169854e-3 * x^2))) / ( 1.0 + x^2 * (
  -0.2484854838e-1 + x^2 * ( 0.2808018655e-3 - 0.1459658740e-5*x^2)
  ));
```

$$I0mLo := x \rightarrow \frac{1.0 + x^2 \left(0.2251516395 + x^2 \left(0.009693409864 + 0.0001146169854\, x^2\right)\right)}{1.0 + x^2 \left(-0.02484854838 + x^2 \left(0.0002808018655 - 0.000001459658740\, x^2\right)\right)}$$

```
> plot( [ I0mLo( x) / BesselI( 0, x) - 1, 4e-8, -4e-8], x = 0 ..
  4.17, color = [ black, gray, gray], linestyle = [ 1, 3, 3]);
```



Locate the zeros and extrema of the relative error.

```
> 'zeros' = seq( fsolve( I0mLo( x) = BesselI( 0, x), x = z_ - 0.1 .
  . z_ + 0.1), z_ = [ 0.0, 1.1, 2.0, 2.7, 3.4, 3.9, 4.1]);
```

zeros = (0., 1.134756113338708, 1.970502972780424, 2.749343179943356, 3.410673824966584, 3.889478161325384, 4.145198719861011)

```
> 'extrema' = seq( fsolve( diff( I0mLo( x) / BesselI( 0, x), x) =
  0.0, x = e_ - 0.1 .. e_ + 0.1), e_ = [ 0.7, 1.6, 2.4, 3.1, 3.7,
  4]);
```

extrema = (0.6929382108186166, 1.558315093654736, 2.369916221005930, 3.097790587362600, 3.675540523724998, 4.048445483607183)

High End

Set a crossover or splice point for the high-end approximation. Put it near the low-end crossover, to allow smooth crossover behavior.

```
> xS2 := 3.9;
```

$$xS2 := 3.9$$

Get the high-end approximation. Set the variable u = 1/x to map the infinite range into a finite one. Use Maple's minimax ability, with a weight selected to keep the error reasonable at both ends:

```
> tHi := numapprox[ minimax]( BesselI( 0, 1 / u) * sqrt( 1 / u) /
  exp( 1 / u), u = 0.000001 .. 1 / xS2, [ 4, 4], 1 / u^( 1 / 5),
  'e');
  `maximum error` = e;
```

tHi := (0.7336975155656745 + (-4.417794864112766 + (8.822005047411893

$$+ (5.276054400005697 - 7.378436814421998\,u)\,u)\,u)\,\Big/\,(1.839106943422109 + ($$
$$-11.30366814489624 + (23.39795349621395 + (10.93922451385757$$
$$- 20.65777552418088\,u)\,u)\,u)\,u)$$

$$maximum\ error = 2.017602048201530\ 10^{-8}$$

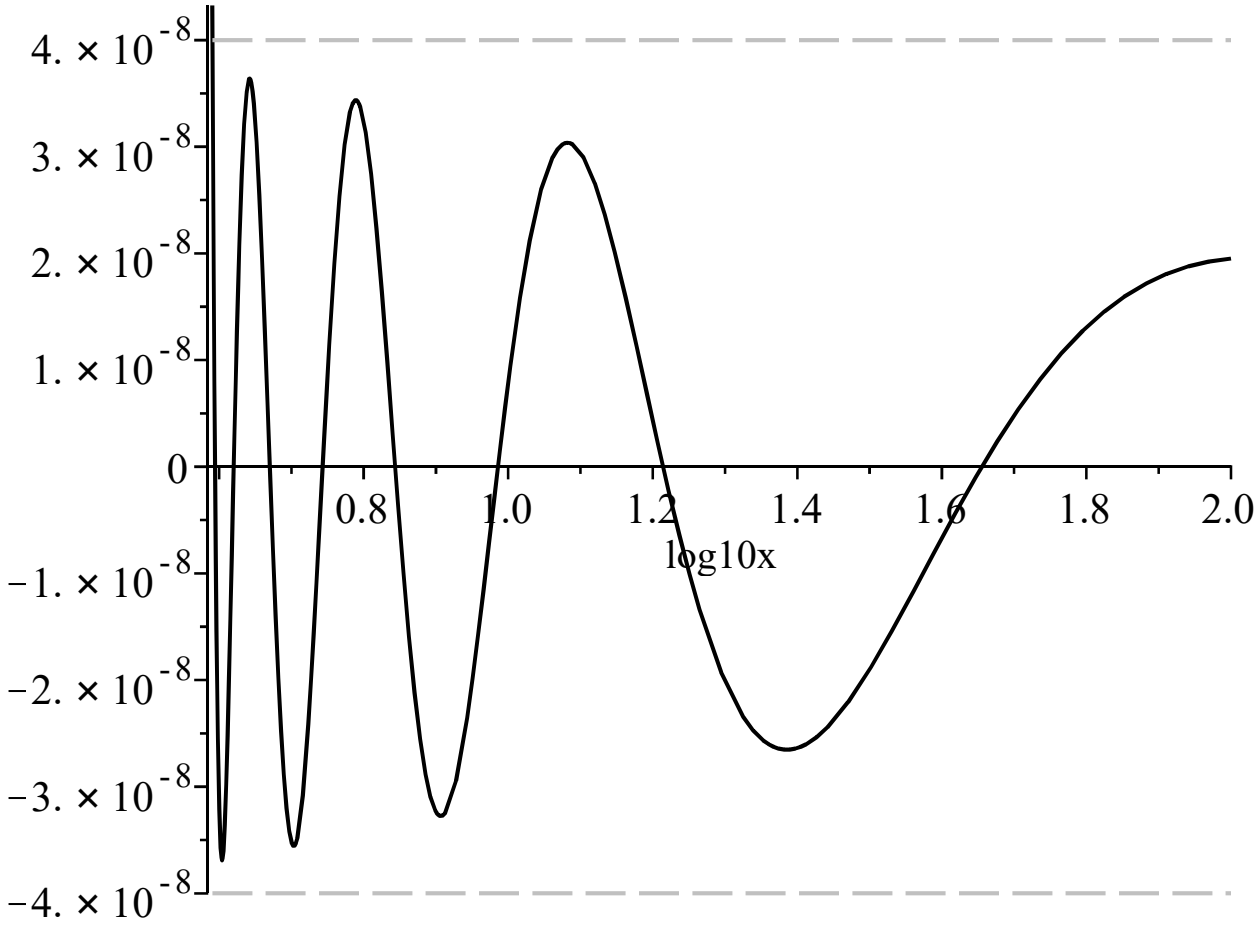
Define approximation with this Pade form:

```
> I0mmHi := x -> subs( u = 1 / x, tHi) * exp( x) / sqrt( x);
```

$$I0mmHi := x \rightarrow \frac{subs\left(u = \frac{1}{x}, tHi\right) e^{x}}{\sqrt{x}}$$

Plot the error of the approximation:

```
> plot( [ I0mmHi( 10.0^log10x) / BesselI( 0, 10.0^log10x) - 1,
  4e-8, -4e-8], log10x = evalf( log10( 0.999 * xS2)) .. evalf(
  log10( 100)), color = [ black, gray, gray], linestyle = [ 1, 3,
  3]);
```



Adjust the expansion coefficients for lower precision:

```
> evalf( numapprox[pade]( tHi, u, [ 4, 4]), 10);
```

$$\frac{0.3989422791 - 2.402141365\,u + 4.796896159\,u^2 + 2.868813260\,u^3 - 4.011967236\,u^4}{0.9999999999 - 6.146281039\,u + 12.72245399\,u^2 + 5.948117657\,u^3 - 11.23250366\,u^4}$$

Tweak manually to get best asymptotic behavior (lead numerator coefficient = $\frac{1}{\sqrt{2\,\pi}}$) and "nice" shape near crossover (coefficient of u in numerator):

```
> hiPade := ( .3989422804 - 2.402141371*u + 4.796896159*u^2 +
  2.868813260*u^3 - 4.011967236*u^4) / (1 - 6.146281039*u +
  12.72245399*u^2 + 5.948117657*u^3 - 11.23250366*u^4);
```

$$hiPade := \frac{0.3989422804 - 2.402141371\,u + 4.796896159\,u^2 + 2.868813260\,u^3 - 4.011967236\,u^4}{1 - 6.146281039\,u + 12.72245399\,u^2 + 5.948117657\,u^3 - 11.23250366\,u^4}$$

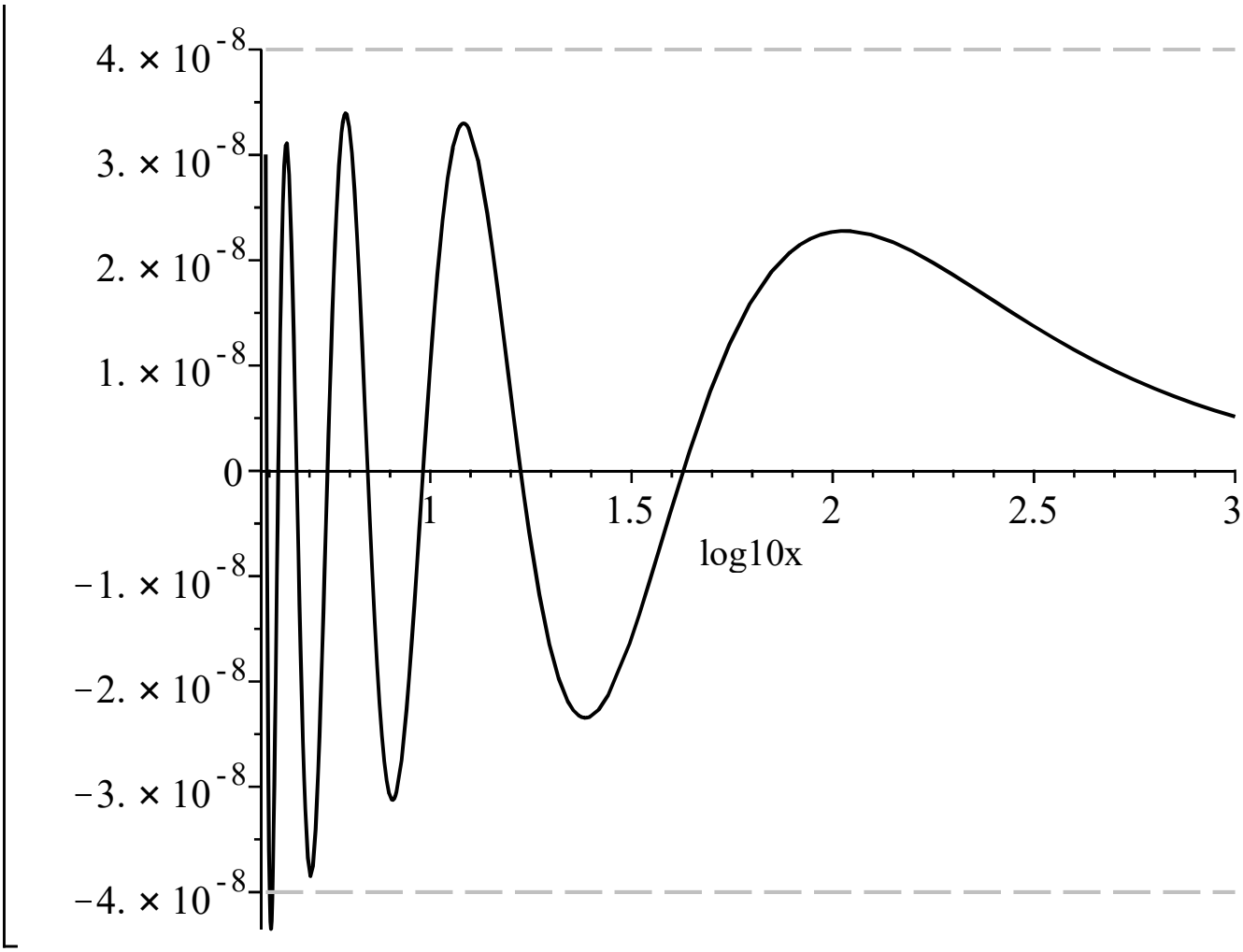
Transfer that form into the definition of I0mHi:

```
> I0mHi := unapply( collect( subs( u = 1 / x, hiPade), x) * exp( x)
  / sqrt( x), x);
```

$$I0mHi := x \rightarrow \frac{\left(0.3989422804 - \frac{2.402141371}{x} + \frac{4.796896159}{x^2} + \frac{2.868813260}{x^3} - \frac{4.011967236}{x^4}\right) e^x}{\left(1 - \frac{6.146281039}{x} + \frac{12.72245399}{x^2} + \frac{5.948117657}{x^3} - \frac{11.23250366}{x^4}\right)\sqrt{x}}$$

With reduced precision, we still get essentially the same error curve:

```
> plot( [ I0mHi( 10.0^log10x) / BesselI( 0, 10.0^log10x) - 1, 4e-8,
  -4e-8], log10x = evalf( log10( xS2)) .. evalf( log10( 1000)),
  color = [ black, gray, gray], linestyle = [ 1, 3, 3]);
```



Locate the zeros and extrema of the relative error.

```
> 'zeros' = seq( fsolve( I0mHi( x) = BesselI( 0, x), x = z_ * 0.9 ..
  z_ * 1.1), z_ = [ 3.9, 4.2, 4.7, 5.6, 7.0, 9.6, 16.8, 42.5]);
```

zeros = (3.923008120426186, 4.190386480162621, 4.654062298548812, 5.552403550661842, 6.987602070655855, 9.596593516679939, 16.76299052273954, 42.51715182380297)

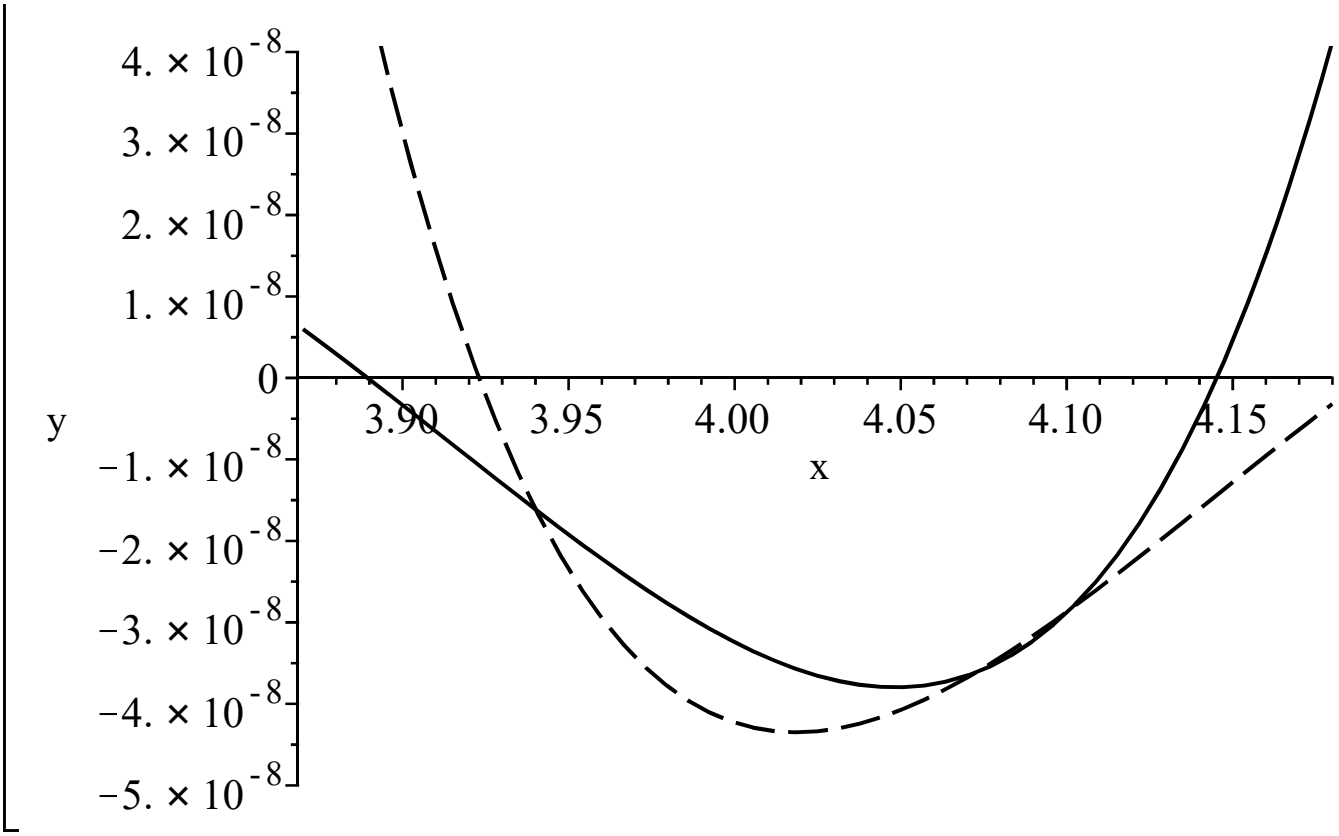
```
> 'extrema' = seq( fsolve( diff( I0mHi( x) / BesselI( 0, x), x) =
  0.0, x = e_ * 0.9 .. e_ * 1.1), e_ = [ 4.0, 4.4, 5.0, 6.2, 8.0,
  12.1, 24.3, 107.1]);
```

extrema = (4.018620439152715, 4.390003195921968, 5.045879539690607, 6.162154419842588, 8.059229594198275, 12.09782733863163, 24.28059977048790, 107.1033291070807)

Joining the Low and High Ends

Display the crossover region between the low and high approximations:

```
> plot( [ I0mLo( x) / BesselI( 0, x) - 1, I0mHi( x) / BesselI( 0,
  x) - 1], x = 3.87 .. 4.18, y = -5e-8 .. 4e-8, color = black,
  linestyle = [ 1, 3]);
```



Determine the actual crossover numerically:

```
> xJoin := evalf( fsolve( I0mLo( x) = I0mHi( x), x = 4.06 .. 4.08))
  ;
```

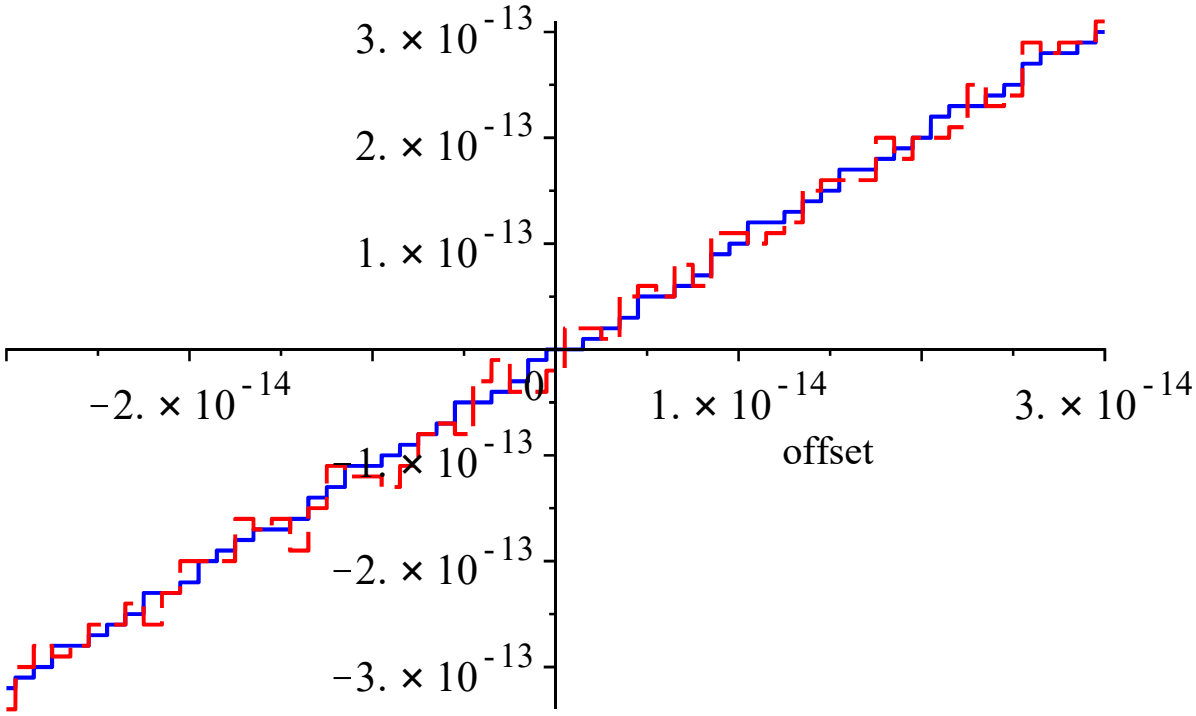
$$xJoin := 4.072619704841123$$

```
> fJoin := evalf( I0mLo( xJoin));
```

$$fJoin := 12.03454796859928$$

Examine the crossover region in detail, to see the 16-digit roundoff noise.

```
> plot( [ I0mLo( xJoin + offset) - fJoin, I0mHi(  xJoin + offset) -
  fJoin], offset = -3.0e-14 .. 3.0e-14, color = [ blue, red],
  linestyle = [ 1, 3], numpoints = 500);
```



Determine that the low and high ends match to hardware accuracy.

```
> evalhf( I0mLo( xJoin) - I0mHi( xJoin));
```

$$0.$$

We can now construct our final algorithm. Recall the low-end and high-end approximations:

```
> convert( I0mLo( x), horner);
```

$$\frac{1.0 + x^2 \left(0.2251516395 + x^2 \left(0.009693409864 + 0.0001146169854\, x^2\right)\right)}{1.0 + x^2 \left(-0.02484854838 + x^2 \left(0.0002808018655 - 0.000001459658740\, x^2\right)\right)}$$

```
> convert( hiPade, horner);
```

$$\frac{0.3989422804 + \left(-2.402141371 + \left(4.796896159 + \left(2.868813260 - 4.011967236\, u\right) u\right) u\right) u}{1 + \left(-6.146281039 + \left(12.72245399 + \left(5.948117657 - 11.23250366\, u\right) u\right) u\right) u}$$

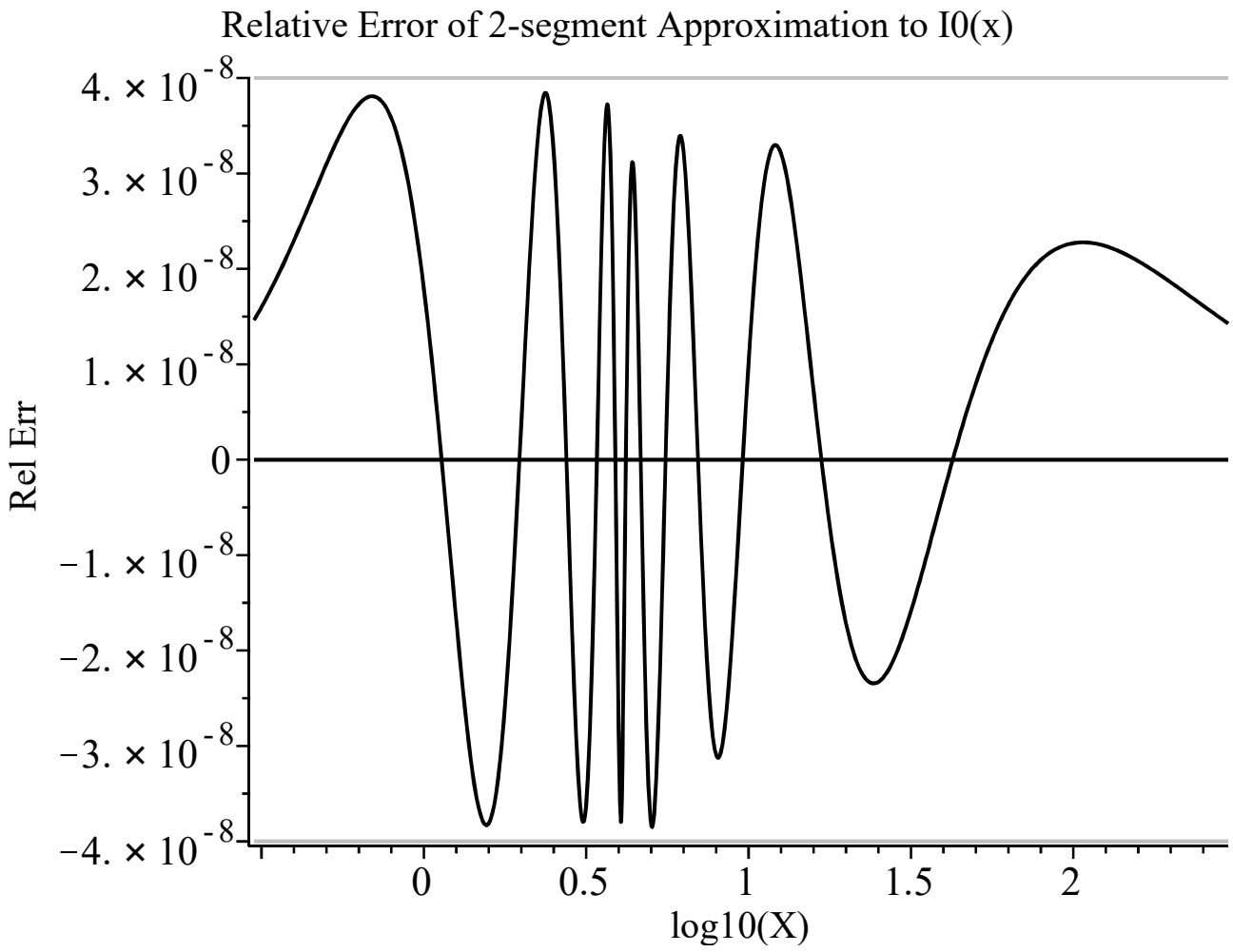
Our final algorithm is:

```
> I0 := proc( x)
    local b, u, v, w;
    if type( x, numeric) then
      u := abs( x); # use same form for +x and -x
      if u <= 4.072619704841123 then # use Pade form in x^2
        w := x * x;
        ( 1.0 + w * ( 0.2251516395 + w * ( 0.9693409864e-2 +
          w * 0.1146169854e-3))) /
        ( 1.0 - w * ( 0.2484854838e-1 - w * ( 0.2808018655e-3 -
          w * 0.1459658740e-5)));
      else # use asymptote & Pade form in 1/|x|
        v := 1 / u;
        exp( u) * sqrt( v) * ( .3989422804 - v * ( 2.402141371 -
```

```
          v * ( 4.796896159 + v * ( 2.868813260 -
          v * 4.011967236)))) /
        ( 1 - v * ( 6.146281039 - v * ( 12.72245399 +
          v * ( 5.948117657 - v * 11.23250366))));
      end if;
    else # if non-numeric argument, return unevaluated form
      'I0'(x)
    end if;
  end:
```

Examine the error over a broad range. It has better than 4e-8 relative accuracy for all X:

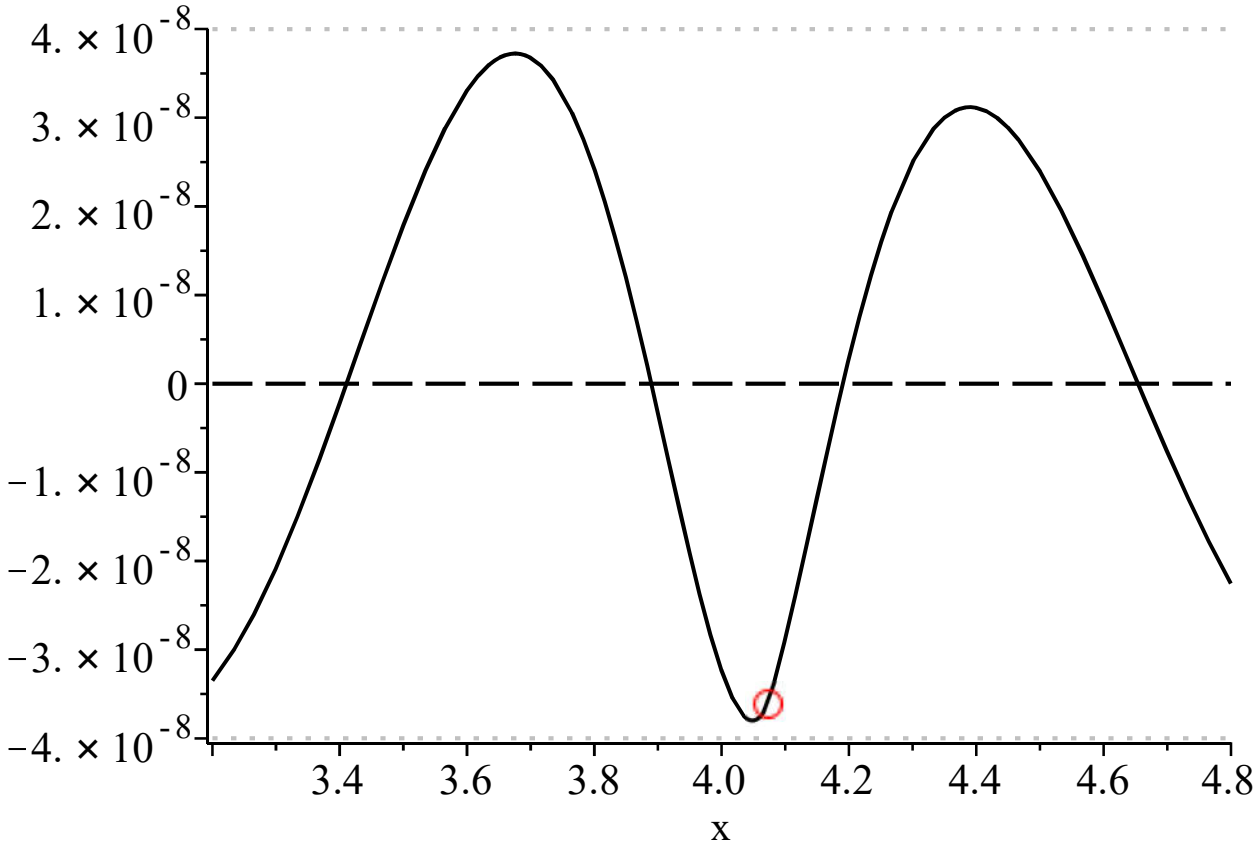
```
> plot( [ I0( 10.0^log10x) / BesselI( 0, 10.0^log10x) - 1, 0, 4e-8,
  -4e-8], log10x = evalf( log10( 0.3)) .. evalf( log10( 300)),
  color = [ black, black, gray, gray], numpoints = 500, axes =
  frame, labels = [ "log10(X)", "Rel Err"], labeldirections = [
  horizontal, vertical], title = "Relative Error of 2-segment
  Approximation to I0(x)", titlefont = [ TIMES, BOLD, 16]);
```



Examine the error over a narrower range, including the crossover region:

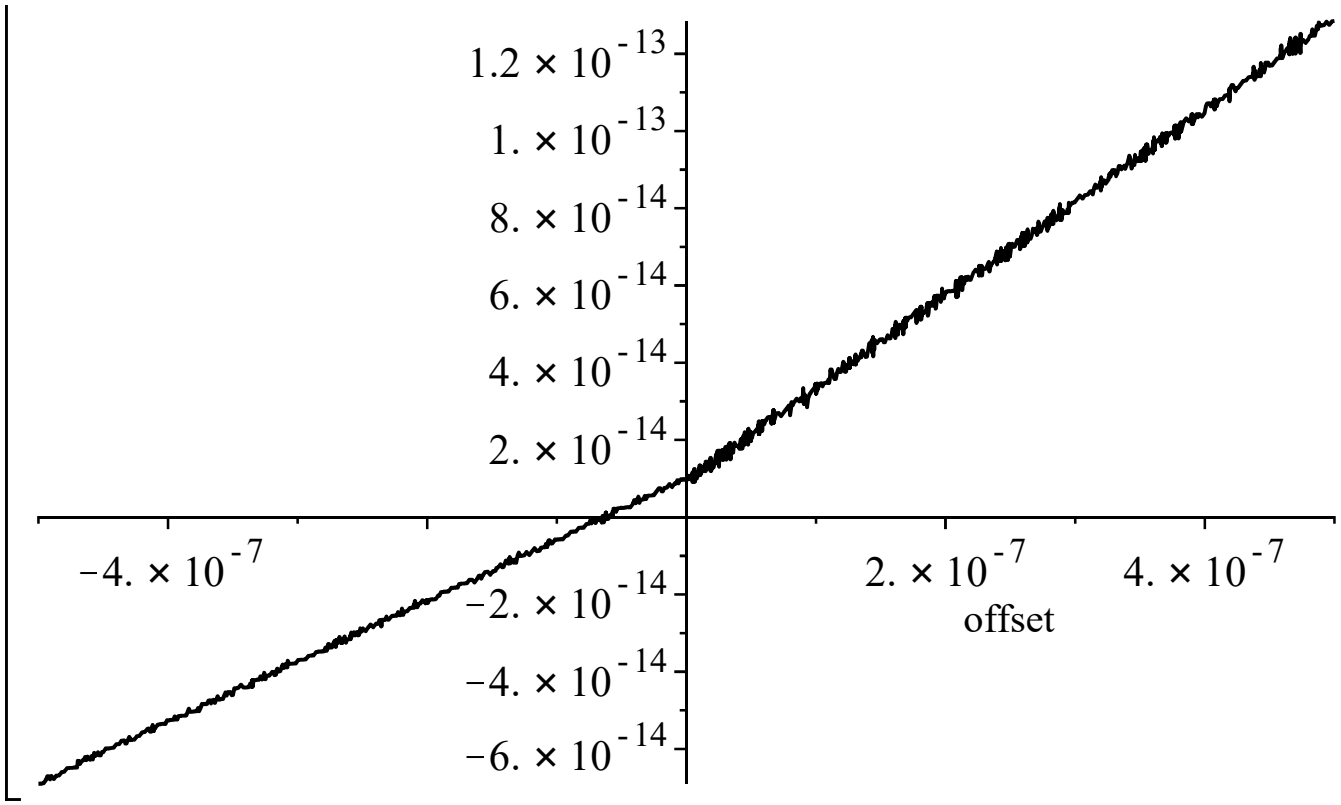
```
> plots[ display]( [
```

```
plot( [ I0( x) / BesselI( 0, x) - 1, 4e-8, 0, -4e-8], x = 3.2 ..
4.8, color = [ black, gray, black, gray], linestyle = [ 1, 2, 3,
2], axes = framed),
plot( [ [ 4.0726197, -3.6155e-8]], style = point, symbol =
circle, symbolsize = 20)]);
```



Look in detail at the crossover region, where value and slope are nearly continuous:

```
> plot( I0( 4.0726197 + offset) / BesselI( 0, 4.0726197 + offset) -
  1 + 3.615535e-8, offset = -0.0000005 .. 0.0000005, color = black,
  numpoints = 500);
```


1.2 × 10^-13
1. × 10^-13
8. × 10^-14
6. × 10^-14
4. × 10^-14
2. × 10^-14
−2. × 10^-14
−4. × 10^-14
−6. × 10^-14
−4. × 10^-7
2. × 10^-7
4. × 10^-7
offset