Reduction alg, high precision

This computes the reduced nodes/weights in extended precision, using 60 digits

```
In[1]:= formMatrixForReductionHalfLine[r_, b_] :=
         Outer[Times, Conjugate[Sqrt[b]], Sqrt[b]] * Outer[1 / (#1 + #2) &, Conjugate[r], r]
```

```
In[2]:= computeConEigenvalue[mat_, tol_] := Module[
          {matForReduction, eigenValues, eigenVectors, smallEigenvalueIndex, v, error},
          matForReduction = Conjugate[mat] . mat;
          {eigenValues, eigenVectors} = Eigensystem[matForReduction ];
          smallEigenvalueIndex =
           Position[(# < tol) & /@ Sqrt[Abs[eigenValues]], True, 1, 1] // Flatten // First;
          v = eigenVectors[[smallEigenvalueIndex]];
          error = matForReduction . v - eigenValues[[smallEigenvalueIndex]] v // Abs // Max;
          {error, v, eigenValues, smallEigenvalueIndex}]
```

```
In[3]:= findRootsRationalFuncHalfLine[gamma0_, w0_, v_] :=
         Module[{r1, b1, v1, w, x, f, a, delta, b, A, eigs, roots, rationalFunc},
          w = Conjugate[Sqrt[w0]] * Conjugate[v];
          x = -Conjugate[gamma0];
          (*f= w/(x-z)//Total;*)
          a = -w / Total[w];
          delta = x;
          b = x;
          A = DiagonalMatrix[delta] + Outer[Times, a, b];
          eigs = Eigenvalues[A];
          eigs = Select[eigs, Re[#] > 10^(-32) &];
          (*eigs=Delete[eigs,Position[Abs[eigs],Min[Abs[eigs]]]];*)
          (*roots =z/.( SetPrecision[f,300]//Together//Numerator//NSolve[#,z]&//N);
          roots=Select[roots,Abs[#]<1&]//Sort//Select[#,Abs[#]>10^-8&]&;*)
          eigs]
```

```
In[4]:= formMatrixForWeightsHalfLine[γNew_] :=
          Outer[1 / (#1 + #2)^3 &, Conjugate[γNew], γNew]
```

```
In[5]:= formRhsForWeightsHalfLine[γ_, γNew_, w_] :=
          Outer[1 / (#1 + #2)^3 &, Conjugate[γNew], γ] . w
```

```
In[394]:= solveForWeightsHalfLine[γ_, γNew_, w_] := Module[{matrixForWeights, rhsForWeights},
           matrixForWeights = formMatrixForWeightsHalfLine[γNew];
           rhsForWeights = formRhsForWeightsHalfLine[γ, γNew, w];
           Block[{$MinPrecision = $MachinePrecision},
            LinearSolve[matrixForWeights, rhsForWeights]]]
```

In[7]:=
```
computeExpReductionHalfLine[gamma_, w_, tol_] :=
 Module[{wH, gammaH, matrixForReduction , v, rationalFunc, newNodes,
   newWeights, dataNew, dataOld, error, eigenValues, smallEigenvalueIndex},
  {wH, gammaH} = {w, gamma} // SetPrecision[#, 60] &;
  matrixForReduction = formMatrixForReductionHalfLine[gammaH, wH];
  {error, v, eigenValues, smallEigenvalueIndex} =
   computeConEigenvalue[matrixForReduction, tol] // SetPrecision[#, 60] &;
  newNodes = findRootsRationalFuncHalfLine[gammaH, wH, v];
  newWeights = solveForWeightsHalfLine[gammaH, newNodes, wH];
  {newNodes, newWeights}]
```

code for sampling and using Hankel formulation for reduction

In[396]:=
```
computeSingularValueDecomp[mat_, tol_] :=
 Module[{U, V, W, singularValues, smallSingValueIndex, smallSingValue, v},
  {U, W, V} =
   Block[ {$MinPrecision = $MachinePrecision}, SingularValueDecomposition[mat]];
  singularValues = Table[W[[i, i]], {i, 1, Length[W]}];
  smallSingValue = Select[singularValues, Abs[#] < tol &, 1];
  smallSingValueIndex =
   Position[(# < tol) & /@ singularValues, True, 1, 1] // Flatten // First;
  v = Transpose[Conjugate[U]][[smallSingValueIndex]];
  {smallSingValueIndex, smallSingValue , v, singularValues}]
```

In[397]:=
```
computeRootsFromEigenVec[u_] :=
 Module[{polynomialForRoots0, rulesForNodes0 , nodes0, error0, accurateNodes0},
  polynomialForRoots0 = u * z ^ (Range[Length[u]] - 1) // Total;
  rulesForNodes0 =
   Block[{$MinPrecision = $MachinePrecision}, NSolve[polynomialForRoots0, z]];
  nodes0 = (rulesForNodes0 // Flatten) /. (z → stuff_) → stuff;
  error0 = polynomialForRoots0 /. rulesForNodes0;
  {error0, nodes0}]
```

In[395]:=
```
solveForWeights[nodes_, sampledFunc_, M_] :=
 Module[{matrixForWeights0, V0, rhsForWeights0 },
  matrixForWeights0 =
   (1 - #^ (2 * M + 1)) / (1 - #) &[Outer[Times, Conjugate[nodes ], nodes]];
  V0 = Map[#^Prepend[Range[2 * M], 0] &, nodes] // Conjugate;
  rhsForWeights0 = V0 . sampledFunc;
  Block[{$MinPrecision = $MachinePrecision},
   LinearSolve[matrixForWeights0, rhsForWeights0]]]
```

In[11]:=
```
computeExpApproximationUsingHankel[data_, tol_] :=
 Module[{M, hankel, v, nodesSmall, error, nodes, weights, expApproxOfData},
  M = (Length[data] - 1) / 2;
  hankel = Table[data[[i + j + 1]], {i, 0, M}, {j, 0, M}];
  v = computeSingularValueDecomp[hankel, tol][[3]];
  nodes = computeRootsFromEigenVec[v][[2]] // Select[#, Abs[#] < 1 &] &;
  weights = solveForWeights[nodes, data, M] // Chop;
  expApproxOfData = Array[Total[weights nodes^#] &, 2 * M + 1, 0] // Chop;
  error = data - expApproxOfData;
  {nodes, weights, error}]
```

In[394]:=
```
computeExpApproximationUsingHankel[r_, b_, M_, tol_] :=
 Module[{data, hankel, v, nodesSmall, error, nodes, weights, expApproxOfData},
  data = Table[Total[b r^(k - 1)], {k, 1, 2 * M + 1}];
  hankel = Table[data[[i + j + 1]], {i, 0, M}, {j, 0, M}];
  v = Block[ {$MinPrecision = $MachinePrecision},
    computeSingularValueDecomp[hankel, tol][[3]]];
  nodes = computeRootsFromEigenVec[v][[2]] // Select[#, Abs[#] < 1 &] &;
  weights = solveForWeights[nodes, data, M] // Chop;
  expApproxOfData = Array[Total[weights nodes^#] &, 2 * M + 1, 0] // Chop;
  error = data - expApproxOfData // Abs // Max;
  {nodes, weights, error}
```

In[840]:=
```
LInfinitySolve[A_, b_] := Module[{m, n, Aall, ball, c, x, bds, onesT},
  {m, n} = Dimensions[A];
  onesT = {Table[1, {m}]};
  Aall = Join[Transpose[Join[onesT, Transpose[-A]]],
    Transpose[Join[onesT, Transpose[A]]]];
  ball = Join[-b, b];
  c = Join[{1}, Table[0, {n}]];
  bds = Table[{-Infinity, Infinity}, {n + 1}];
  x = LinearProgramming[c, Aall, ball, bds];
  Take[x, -n]
 ]
```

In[13]:=
```
interpAtom[d_, n_, x_] := LaguerreL[n, 1 / 2, x^2] 1 / Sqrt[Pi d] Exp[-x^2 / d]
```

Computation of Gaussian approximation of $\frac{e^{-\frac{x^2}{4}}}{2\sqrt{\pi}}$ in the paper

Compute (near) optional rational approximation of the Gaussian $\frac{e^{-\frac{x^2}{4}}}{2\sqrt{\pi}}$; note that there is an additional step to get the imaginary part of the poles to be aligned with the integers

In[14]:=
```
d0 = 4;
```

In[15]:=
```
gaussian = interpAtom[d0, 0, x]
```

Out[15]= $\frac{e^{-\frac{x^2}{4}}}{2\sqrt{\pi}}$

In[16]:=
```
gaussianFT = Integrate[gaussian Exp[I k x], {x, -Infinity, Infinity}]
```

Out[16]= e^{-k^2}

In[672]:=
```
M = 100;
R = 10;
pts = R Range[0, 2 * M - 1] / (2 M);
fVals = gaussianFT /. k → pts // SetPrecision[#, 160] &;
tol = 10^-11;
{nodes, weights, error} =
  computeExpApproximationUsingHankel[fVals, tol] // SetPrecision[#, 160] &;
alphasAtom =  2 M Log[nodes] / R // N // Conjugate // SetPrecision[#, 160] &;
weightsAtom = -(1 / Pi) weights // N // Conjugate // SetPrecision[#, 160] &;
```

In[680]:=
```
error // Abs // Max
```

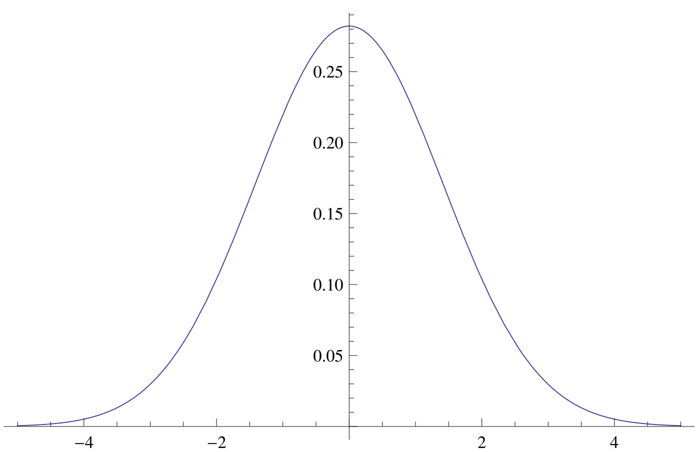
Out[680]= 9.85950557599150824072976628041491260343453830361480396157103385628929595918993707320765444748550892838235308049610043154726546360639091442235938991288762745667 9 × 10^{-11}

In[736]:=
```
alphasAtom ;
```

Plot the error in the rational approximation of the Gaussian $\frac{e^{-\frac{x^2}{4}}}{2\sqrt{\pi}}$

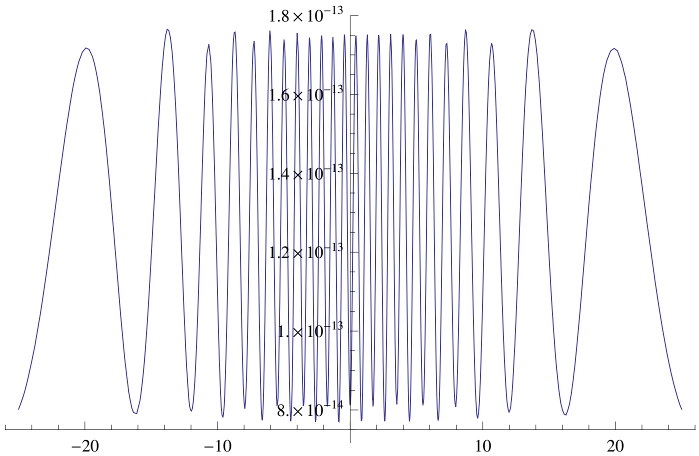
In[647]:=
```
rationalFuncExpr = Total[weightsAtom / (I x + alphasAtom)] // Re;
Plot[rationalFuncExpr, {x, -5, 5}, PlotRange → All]
```

Out[648]=



In[682]:=
```
rationalFuncExpr = Total[weightsAtom / (I x + alphasAtom)] // Re;
plotRationalApproxGaussianOpt =
 Plot[rationalFuncExpr - gaussian // Abs, {x, -25, 25}, PlotRange → All]
```

Out[683]=



Now, construct the rational approximation of rational approximation of the Gaussian $\frac{e^{-\frac{x^2}{4}}}{2\sqrt{\pi}}$ where the imaginary part of the poles are integers

In[737]:=
```
alphaReMin = alphasAtom // Re // Abs // Min // SetPrecision[#, 160] &;
alphaReMax = alphasAtom // Re // Abs // Max // SetPrecision[#, 160] &;
alphaReAv = (alphaReMin + alphaReMax) / 2 // SetPrecision[#, 160] &;
```

In[748]:=
```
alphasAtom // Length
```

Out[748]= 13

In[745]:=
```
alphasAtom // N // SetPrecision[#, 40] &
```

Out[745]=
```
{-4.315321510875024024755930440733209252357 +
  7.097812295140816019056728691793978214264 ⅈ,
 -4.315321510875024024755930440733209252357 -
  7.097812295140816019056728691793978214264 ⅈ,
 -4.421723015913975096680132992332801222801 +
  5.600740225255292692452258052071556448936 ⅈ,
 -4.421723015913975096680132992332801222801 -
  5.600740225255292692452258052071556448936 ⅈ,
 -4.493645084662802879904575092950835824013 +
  4.337823602355635799199262692127376794815 ⅈ,
 -4.493645084662802879904575092950835824013 -
  4.337823602355635799199262692127376794815 ⅈ,
 -4.543988823306032820426025864435359835625 +
  3.185709552739200756121817903476767241955 ⅈ,
 -4.543988823306032820426025864435359835625 -
  3.185709552739200756121817903476767241955 ⅈ,
 -4.577702118253584195883831853279843926430 +
  2.095480373265162565843411357491277158260 ⅈ,
 -4.577702118253584195883831853279843926430 -
  2.095480373265162565843411357491277158260 ⅈ,
 -4.597172083549668109014874062268063426018 +
  1.039869147232639345901361593860201537609 ⅈ,
 -4.597172083549668109014874062268063426018 -
  1.039869147232639345901361593860201537609 ⅈ,
 -4.603546784437746453022555215284228324890}
```

Redefining poles, finding residues through convex optimization

In[747]:=
```
gamma = -alphaReMin + I Range[-11, 11];
```

In[690]:=
```
N1 = 800;
xPts1 = 60 Range[-N1, N1] / (2 N1);
N2 = 60;
xPts2 = 90 Range[0, N2] / N2 + 15;
xPts3 = -Reverse[xPts2];
xPts = Join[xPts3, xPts1, xPts2];
tau = Re[gamma];
theta = Im[gamma];
```

In[768]:=
```
xPts1;
```

In[852]:=
```
b = gaussian /. x → xPts // SetPrecision[#, 160] & // N;
xPts = N[xPts, 40];
A = ConstantArray[0, {Length[xPts], 2 * Length[tau]}];
A[[1 ;; Length[xPts], 1 ;; Length[tau]]] =
  Table[(xPts[[i]] - theta[[j]]) / ((xPts[[i]] - theta[[j]])^2 + tau[[j]]^2),
    {i, 1, Length[xPts]}, {j, 1, Length[tau]}] // SetPrecision[#, 160] &;
A[[1 ;; Length[xPts], Length[tau] + 1 ;; 2 * Length[tau]]] =
  Table[- tau[[j]] / ((xPts[[i]] - theta[[j]])^2 + tau[[j]]^2),
   {i, 1, Length[xPts]}, {j, 1, Length[tau]}];
```

In[857]:=
```
w1 = LInfinitySolve[A, b];
```

In[858]:=
```
w1 = N[w1, 50]
```

Out[858]= $\{-2.93636\times10^{-8}, 9.14812\times10^{-7}, -6.60004\times10^{-7}, -0.0000113597, 0.0000314006, 0.000473274, -0.0178393, 0.12327, 0.187712, -3.20347, 6.12376, 0., -6.12376, 3.20347, -0.187712, -0.12327, 0.0178394, -0.000473215, -0.0000314381, 0.0000113455, 6.67813\times10^{-7}, -9.14197\times10^{-7}, 2.90794\times10^{-8}, 1.08464\times10^{-7}, -5.77936\times10^{-9}, -3.70384\times10^{-6}, 2.68146\times10^{-6}, -0.0000146716, 0.00107554, -0.00381713, -0.121242, 0.977499, -1.34329, -4.07241, 9.4427, -4.07241, -1.34329, 0.977499, -0.121242, -0.00381704, 0.00107549, -0.0000147048, 2.70093\times10^{-6}, -3.69969\times10^{-6}, -7.90834\times10^{-9}, 1.08467\times10^{-7}\}$

Here a are residues corresponding to new poles

In[859]:=
```
a = w1[[Length[w1] / 2 + 1 ;; Length[w1]]] + I w1[[1 ;; Length[w1] / 2]] //
    SetPrecision[#, 160] &;
a =
  -a;
```

In[861]:=
```
a // SetPrecision[#, 20] &
```

Out[861]=
$$\{-1.0846354751290588104\times10^{-7}+2.9363558093024306997\times10^{-8}\,i,\\
5.77935685299374656\times10^{-9}-9.1481151228521658961\times10^{-7}\,i,\\
3.7038358208237924102\times10^{-6}+6.6000447404469444982\times10^{-7}\,i,\\
-2.6814569755414357050\times10^{-6}+0.000011359697331911185961 7\,i,\\
0.000014671580056339240006-0.000031400644103252651400\,i,\\
-0.0010755441939931374185 9-0.00047327406930292107703\,i,\\
0.0038171266363603499444+0.017839292571877041899\,i,\\
0.12124171418932352950-0.12326964235131572523\,i,\\
-0.97749882474018490175-0.18771224971482819432\,i,\\
1.3432857754271598782+3.2034709703859047814\,i,\\
4.0724088186063580608-6.1237563578921685448\,i,\\
-9.4426992314956841312+0.\times10^{-20}\,i,\ 4.0724088069324508865+6.1237563189376063022\,i,\\
1.3432858438021879621-3.2034710111648174724\,i,\\
-0.97749875271985153802+0.18771233382364543862\,i,\\
0.12124162750886421924+0.12326973063490963278\,i,\\
0.0038170449673168098 91-0.017839369918773727991\,i,\\
-0.0010754853243729657098 8+0.00047321522033559951237\,i,\\
0.000014704798043942080216+0.000031438120209716725927\,i,\\
-2.7009271345856269452\times10^{-6}-0.000011345497752773350762 2\,i,\\
3.6996919184365413322\times10^{-6}-6.6781271186322433357\times10^{-7}\,i,\\
7.90833808793635723\times10^{-9}+9.1419674399960741488\times10^{-7}\,i,\\
-1.0846676890919319454 3\times10^{-7}-2.9079438997683162566\times10^{-8}\,i\}$$

Finally, here is the final rational approximation of $\frac{e^{-\frac{x^2}{4}}}{2\sqrt{\pi}}$

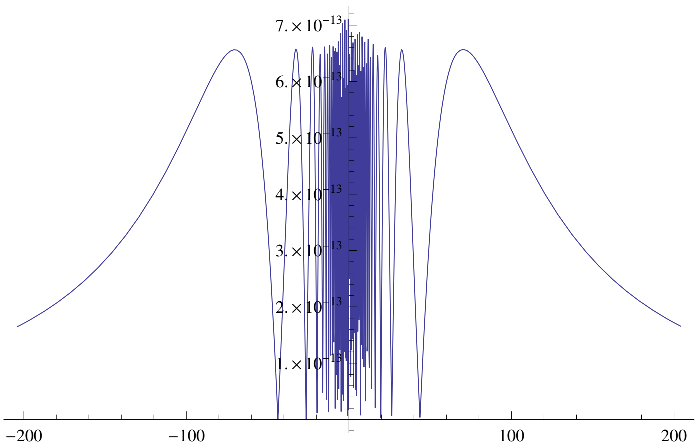
In[862]:=
```
gamma // SetPrecision[#, 40] &
```

```
Out[862]= {-4.315321510875024024755930440733209252357 -
    11.00000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 -
    10.00000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 -
    9.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 -
    8.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 -
    7.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 -
    6.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 -
    5.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 -
    4.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 -
    3.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 -
    2.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 -
    1.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357,
   -4.315321510875024024755930440733209252357 +
    1.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 +
    2.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 +
    3.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 +
    4.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 +
    5.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 +
    6.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 +
    7.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 +
    8.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 +
    9.000000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 +
    10.00000000000000000000000000000000000000 ⅈ,
   -4.315321510875024024755930440733209252357 +
    11.00000000000000000000000000000000000000 ⅈ}
```

In[863]:=
```
a // SetPrecision[#, 20] &
```

Out[863]=
$\{-1.0846354751290588104\times10^{-7}+2.9363558093024306997\times10^{-8}\,i,$
$5.77935685299374656\times10^{-9}-9.1481151228521658961\times10^{-7}\,i,$
$3.7038358208237924102\times10^{-6}+6.6000447404469449820\times10^{-7}\,i,$
$-2.6814569755414357050\times10^{-6}+0.000011359697331911185961 7\,i,$
$0.0000146715800563392400006-0.0000314006441032526514 00\,i,$
$-0.0010755441939931374185 9-0.0004732740693029210770 3\,i,$
$0.0038171266363603499 44+0.0178392925718770418 99\,i,$
$0.12124171418932352950-0.12326964235131572523\,i,$
$-0.97749882474018490175-0.18771224971482819432\,i,$
$1.3432857754271598782+3.2034709703859047814\,i,$
$4.0724088186063580608-6.1237563578921685448\,i,$
$-9.4426992314956841312+0.\times10^{-20}\,i,\ 4.0724088069324508865+6.1237563189376063022\,i,$
$1.3432858438021879621-3.2034710111648174724\,i,$
$-0.97749875271985153802+0.18771233382364543862\,i,$
$0.12124162750886421924+0.12326973063490963278\,i,$
$0.0038170449673168098 91-0.0178393699187737279 91\,i,$
$-0.0010754853243729657098 8+0.0004732152203355995123 7\,i,$
$0.0000147047980439420802 16+0.0000314381202097167259 27\,i,$
$-2.7009271345856269452\times10^{-6}-0.0000113454977527733507622\,i,$
$3.6996919184365413322\times10^{-6}-6.6781271186322433 57\times10^{-7}\,i,$
$7.90833808793635723\times10^{-9}+9.1419674399960741488\times10^{-7}\,i,$
$-1.0846676890919319454 3\times10^{-7}-2.9079438997683162566\times10^{-8}\,i\}$

In[864]:=
```
rationalFuncExpr = Re[ Total[a / (I x + gamma)] ];
plotRationalApproxGaussianSubOpt =
 Plot[rationalFuncExpr - gaussian // Abs, {x, -204, 204}, PlotRange → All]
```

Out[865]=


Constructing rational approximations of cos(x) and sin(x)

Getting the Gaussian approximation of cos(x) and sin(x)

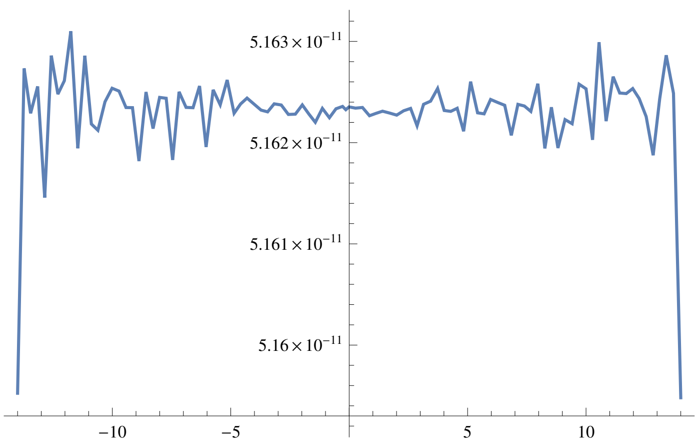
```
gaussianFT = Integrate[Exp[2 Pi I x k] Exp[-x^2 / (d h^2)] / Sqrt[d Pi],
   {x, -Infinity, Infinity}, Assumptions → d > 0 && h > 0];
normalization[h0_, d0_] :=
 gaussianFT /. k → 1 /. d → d0 /. h → h0 // N
```

```
funcExp[x_, d_, N1_, h_] := 1 / Sqrt[d Pi] h / normalization[h, d]
  Sum[Exp[-2 Pi h n I] Exp[-(x + n h) ^2 / (d h^2)], {n, -N1, N1}]
```

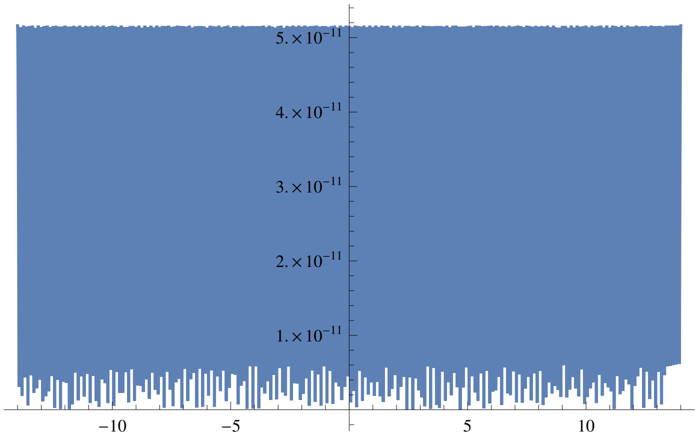
```
funcCos[x_, d_, N1_, h_] := 1 / Sqrt[d Pi] h / normalization[h, d]
  Sum[Cos[2 Pi h n] Exp[-(x + n h) ^2 / (d h^2)], {n, -N1, N1}]
```

```
funcSin[x_, d_, N1_, h_] := 1 / Sqrt[d Pi] h / normalization[h, d]
  Sum[Sin[-2 Pi h n] Exp[-(x + n h) ^2 / (d h^2)], {n, -N1, N1}]
```

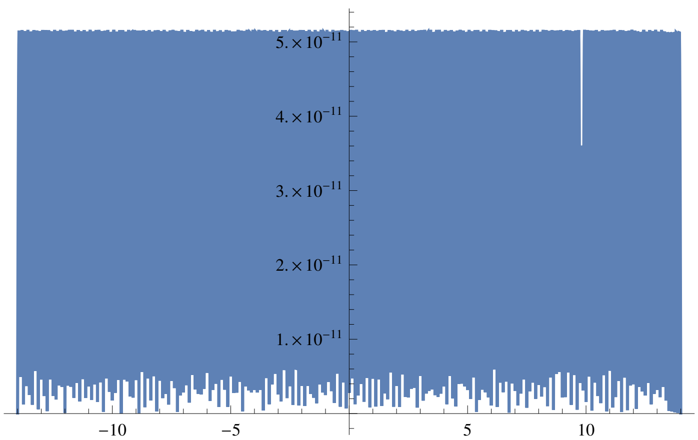
```
h0 = 1 / 5;
M = 80;
d0 = 4;
plotExpErrorGaussian =
 Plot[Exp[2 Pi x I] - funcExp[x, d0, M, h0] // Abs, {x, -14, 14}, PlotRange → All]
```



```
plotExpErrorGaussian =
 Plot[Cos[2 Pi x] - funcCos[x, d0, M, h0] // Abs, {x, -14, 14}, PlotRange → All]
```




```
plotExpErrorGaussian =
 Plot[Sin[2 Pi x] - funcSin[x, d0, M, h0] // Abs, {x, -14, 14}, PlotRange → All]
```



Constructing final rational approximation

```
computeResidues[p_, func_, M_, L_, h_, aGauss_] := Module[{L1, L2, c, ind, res},
  L1 = Max[-L, p - M];
  L2 = Min[L, p + M];
  ind = Range[L1, L2];
  c = Map[func[h #] &, p - ind];
  res = Total[h aGauss[[L + ind + 1]] * c];
  res
 ]
```

Construction for sin(2 pi x)

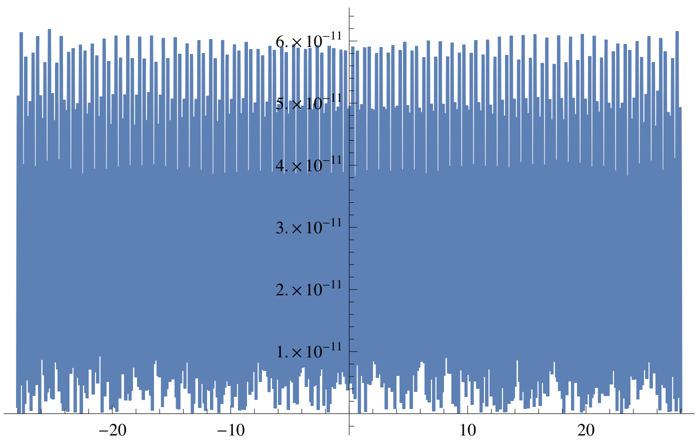
```
d0 = 4;
h0 = 1 / 5;
L = 11;
M = 160;
norm = h0 / normalization[h0, d0];
residuesSin =
  norm Table[computeResidues[p, Sin[-2 Pi #] &, M, L, h0, a], {p, -M - L, M + L}];
```

```
norm
```

```
4.85077
```

```
tau = -alphaReMin;
polesCos = h0 (tau + I Range[-M - L, M + L]);
rationalFuncSin = residuesSin / (I x + polesCos) // Total;
```

```
plotRationalApproxSin =
 Plot[ Sin[2 Pi x] - Re[rationalFuncSin] // Abs, {x, -28, 28}, PlotRange → All]
```



Construction for cos(2 pi x)

```
norm = h0 / normalization[h0, d0];
residuesCos =
  norm Table[computeResidues[p, Cos[2 Pi #] &, M, L, h0, a], {p, -M - L, M + L}];
```

```
rationalFuncCos = residuesCos / (I x + polesCos) // Total;
```

```
plotRationalApproxSin =
 Plot[Cos[2 Pi x] - Re[rationalFuncCos] // Abs, {x, -28, 28}, PlotRange → All]
```