Global Variables

In[ ]:=
```
prec = 100;

Nmax = 100;

x1 = 1.0;
x2 = 10.0;
x3 = 20.0;

xmax = 20.0;
xmin = -20.0;
xsize = 100;
dx = (xmax - xmin) / (xsize - 1);

Xvector = N[Range[xmin, xmax, dx], prec];
                 ·· intervalo de valores
Nvector = Range[0, Nmax, 1];
                 intervalo de valores
```

Wavefuntion_Wolfram_Mathematica_1

In[ ]:=
```
WavefunctionMathematica1[n_, x_, prec_] :=
  Module[{nPrec, xPrec, norm, H, wavefunction},
  módulo de código
   SetPrecision[n, prec];
   define precisão
   SetPrecision[x, prec];
   define precisão
   norm = (2^(-0.5 * n)) * (Gamma[n + 1]^(-0.5)) * (Pi^(-0.25));
                                 função gama de Euler          número pi
   H = HermiteH[n, x];
       polinômios de Hermite
   wavefunction = SetPrecision[norm * Exp[-0.5 * x^2] * H, prec];
                                define precisão          exponencial
   wavefunction];
```

Wavefuntion_Wolfram_Mathematica_2

```
In[•]:= WavefunctionMathematica2[n_, x_, prec_] :=
          Module[{wavefunction, xsize, i}, SetPrecision[x, prec];
          módulo de código                 define precisão
           wavefunction = Table[SetPrecision[0, prec], {n + 1}, {Length[x]}];
                          tabela  define precisão               comprimento
           wavefunction[[1]] = SetPrecision[Pi^(-1 / 4) Exp[-(x^2) / 2], prec];
                               define precisão  número pi  exponencial
           wavefunction[[2]] = SetPrecision[(2 x wavefunction[[1]]) / Sqrt[2], prec];
                               define precisão                    raiz quadrada
           For[i = 3, i ≤ n + 1, i++, wavefunction[[i]] = 2 x (wavefunction[[i - 1]] / Sqrt[2 (i - 1)]) -
           para cada                                                  raiz quadrada
               Sqrt[(i - 2) / (i - 1)] wavefunction[[i - 2]];];
               raiz quadrada
           wavefunction[[n + 1]]];
```

Wavefuntion_Wolfram_Mathematica_3

```
In[•]:= WavefunctionMathematica3[n_, x_, prec_] :=
          Module[{wavefunction, i}, SetPrecision[x, prec];
          módulo de código          define precisão
           wavefunction = Table[SetPrecision[0, prec], {n + 1}, {Length[x]}];
                          tabela  define precisão               comprimento
           wavefunction[[1]] = SetPrecision[Pi^(-1 / 4) Exp[-(x^2) / 2], prec];
                               define precisão  número pi  exponencial
           For[i = 1, i ≤ n, i++,
           para cada
            wavefunction[[i + 1]] =
             SetPrecision[2 x (wavefunction[[i]] / Sqrt[2 (i)]) -
             define precisão                          raiz quadrada
                Sqrt[(i - 1) / i] wavefunction[[i - 1]], prec]];
                raiz quadrada

           wavefunction];
```

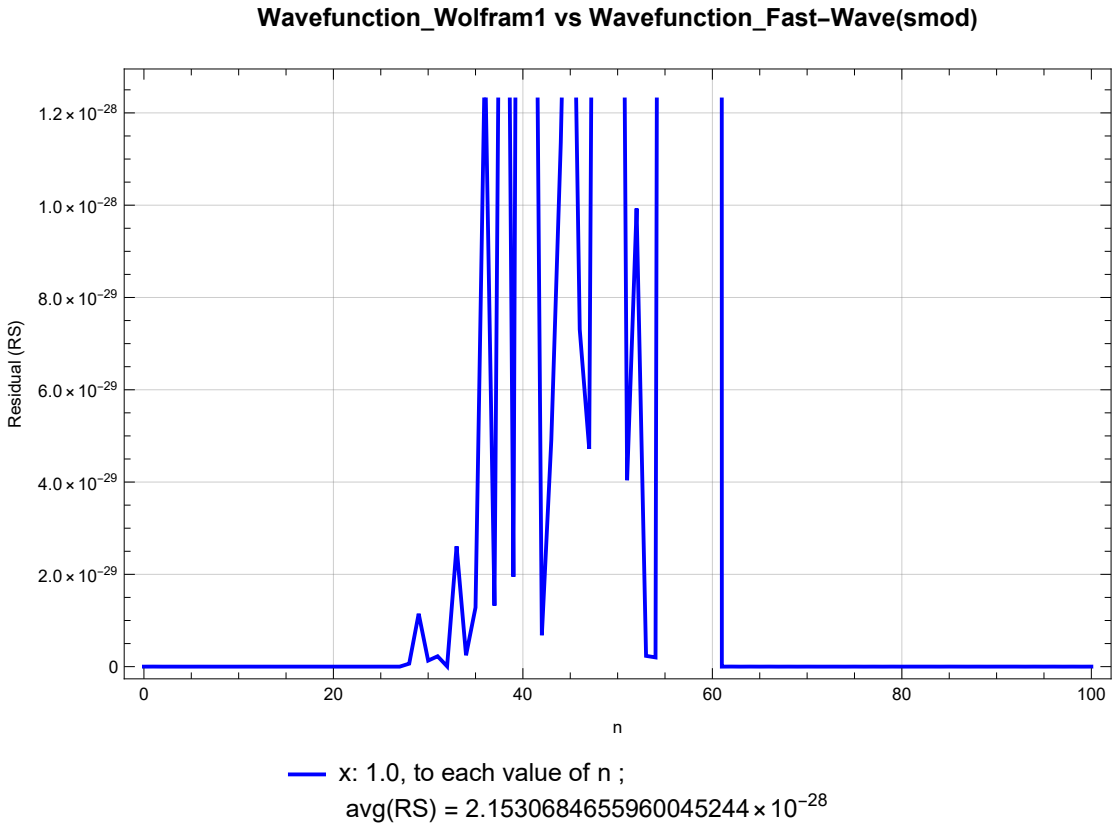
Tests

★ Single-Mode and Onedimensional Function with x = 1.0

In[•]:=
```
FastWaveSMOdN100x1 = SetPrecision[Normal[ExternalEvaluate["Python",
                     define precisão  normal  execução externa
      "from fast_wave.wavefunction import wavefunction_smod;import numpy
        as np;N_max = 100;x1 = 1.0;np.array([wavefunction_smod(n,x1)
        for n in range(N_max+1)]);"]], prec];

(*WolframSMOdN100x1 = WavefunctionMathematica3[Nmax,x1,prec];*)
WolframSMOdN100x1 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
 para cada
 AppendTo[WolframSMOdN100x1, WavefunctionMathematica1[i – 1, x1, prec]];]
 adiciona a

Residual = (WolframSMOdN100x1 – FastWaveSMOdN100x1)^2;


ListLinePlot[
 gráfico de linha de uma lista de valores
 Transpose[{Nvector, Residual}],
 transposição
 PlotStyle → {Thick, Blue},
 estilo do gráfico  espesso  azul
 Frame → True,
 quadro  verdadeiro
 FrameLabel → {"n", "Residual (RS)"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Style["Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave(smod)\n", 13, Bold],
  estilo                                                          negrito
 GridLines → Automatic,
 grade de linhas  automático
 PlotLegends → Placed[{"x: 1.0, to each value of n ; \n avg(RS) = " <>
 legenda do gráfico  situado
      ToString[SetPrecision[Mean[Residual], 20],
      converte···  define precisão  média
       TraditionalForm]}, Below],
       forma tradicional  abaixo
 ImageSize → Large ]
 tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[•]=

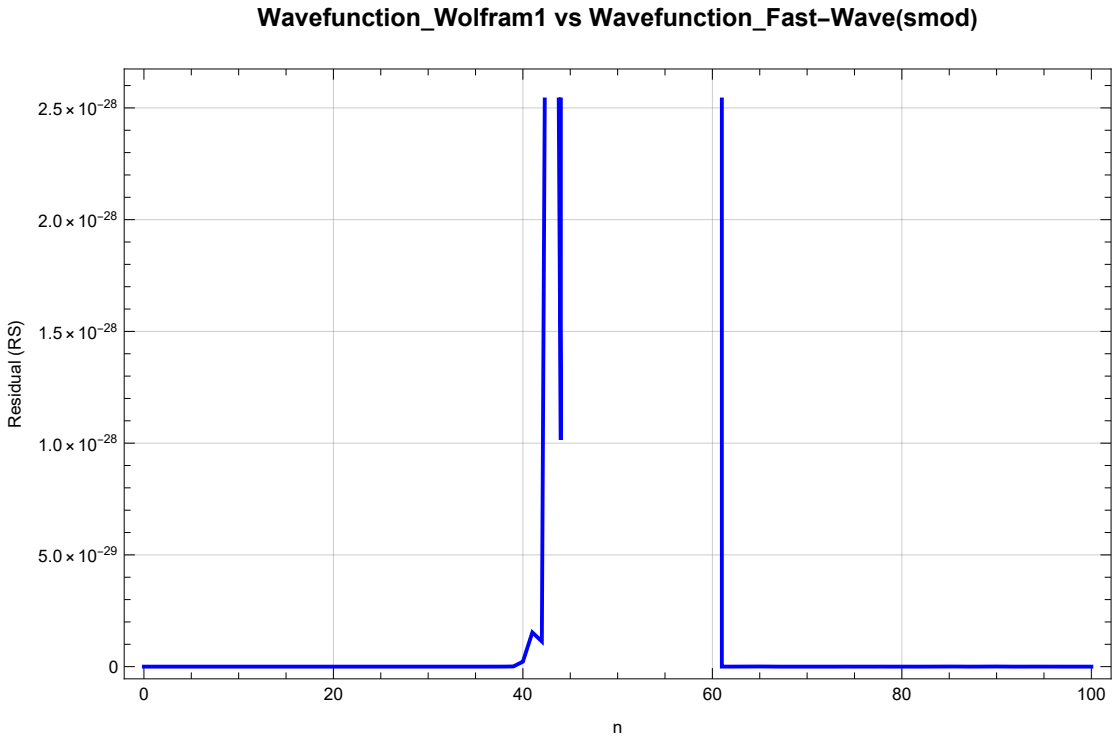

Wavefunction_Wolfram1 vs Wavefunction_Fast–Wave(smod)
Residual (RS)
1.2 × 10^-28
1.0 × 10^-28
8.0 × 10^-29
6.0 × 10^-29
4.0 × 10^-29
2.0 × 10^-29
0
0
20
40
60
80
100
n
x: 1.0, to each value of n ;
avg(RS) = 2.1530684655960045244 × 10^-28

★ Single-Mode and Onedimensional Function with x = 10.0

In[ ]:=
```
FastWaveSMOdN100x2 = SetPrecision[Normal[ExternalEvaluate["Python",
                     define precisão  normal  execução externa
      "from fast_wave.wavefunction import wavefunction_smod;import numpy as
        np;N_max = 100;x2 = 10.0;np.array([wavefunction_smod(n,x2)
        for n in range(N_max+1)]);"]], prec];

(*WolframSMOdN100x2 = WavefunctionMathematica3[Nmax,x2,prec];*)

WolframSMOdN100x2 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSMOdN100x2, WavefunctionMathematica1[i - 1, x2, prec]];]
 adiciona a

Residual = (WolframSMOdN100x2 - FastWaveSMOdN100x2)^2;


ListLinePlot[
gráfico de linha de uma lista de valores
 Transpose[{Nvector, Residual}],
 transposição
 PlotStyle → {Thick, Blue},
 estilo do gráfico  espessc  azul
 Frame → True,
 quadro  verdadeiro
 FrameLabel → {"n", "Residual (RS)"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Style["Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave(smod)\n", 13, Bold],
  estilo                                                          negrito
 GridLines → Automatic,
 grade de linhas  automático
 PlotLegends → Placed[{" x: 10.0, to each value of n ; \n avg(RS) = [" <>
 legenda do gráficc  situado
       ToString[SetPrecision[Mean[Residual], 20],
       converte···  define precisão  média
        TraditionalForm]} <> "]", Below],
        forma tradicional              abaixo
 ImageSize → Large ]
 tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[•]=



x: 10.0, to each value of n ;

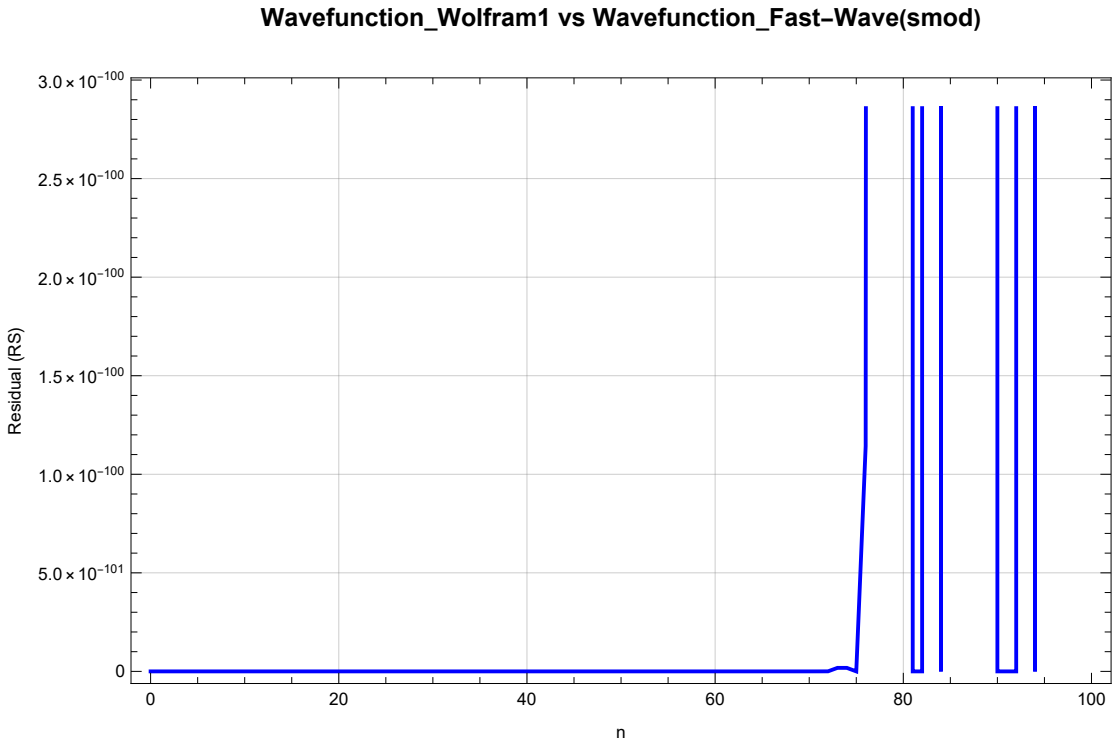
avg(RS) = [$9.3427258602736198176 \times 10^{-17}$]

★ Single-Mode and Onedimensional Function with x = 20.0

In[ ]:=
```
FastWaveSMOdN100x3 = SetPrecision[Normal[ExternalEvaluate["Python",
                     define precisão  normal  execução externa
      "from fast_wave.wavefunction import wavefunction_smod;import numpy as
        np;N_max = 100;x3 = 20.0;np.array([wavefunction_smod(n,x3)
        for n in range(N_max+1)]);"]], prec];

(*WolframSMOdN100x3 = WavefunctionMathematica3[Nmax,x3,prec];*)
WolframSMOdN100x3 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSMOdN100x3, WavefunctionMathematica1[i - 1, x3, prec]];]
 adiciona a

Residual = (WolframSMOdN100x3 - FastWaveSMOdN100x3)^2;


ListLinePlot[
gráfico de linha de uma lista de valores
 Transpose[{Nvector, Residual}],
 transposição
 PlotStyle → {Thick, Blue},
 estilo do gráfico  espessc  azul
 Frame → True,
 quadro  verdadeiro
 FrameLabel → {"n", "Residual (RS)"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Style["Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave(smod)\n", 13, Bold],
  estilo                                                          negrito
 GridLines → Automatic,
 grade de linhas  automático
 PlotLegends → Placed[{" x: 20.0, to each value of n ; \n avg(RS) = [" <>
 legenda do gráficc  situado
       ToString[SetPrecision[Mean[Residual], 20],
       converte···  define precisão  média
        TraditionalForm]} <> "]", Below],
        forma tradicional  abaixo
 ImageSize → Large ]
 tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[•]=



x: 20.0, to each value of n ;

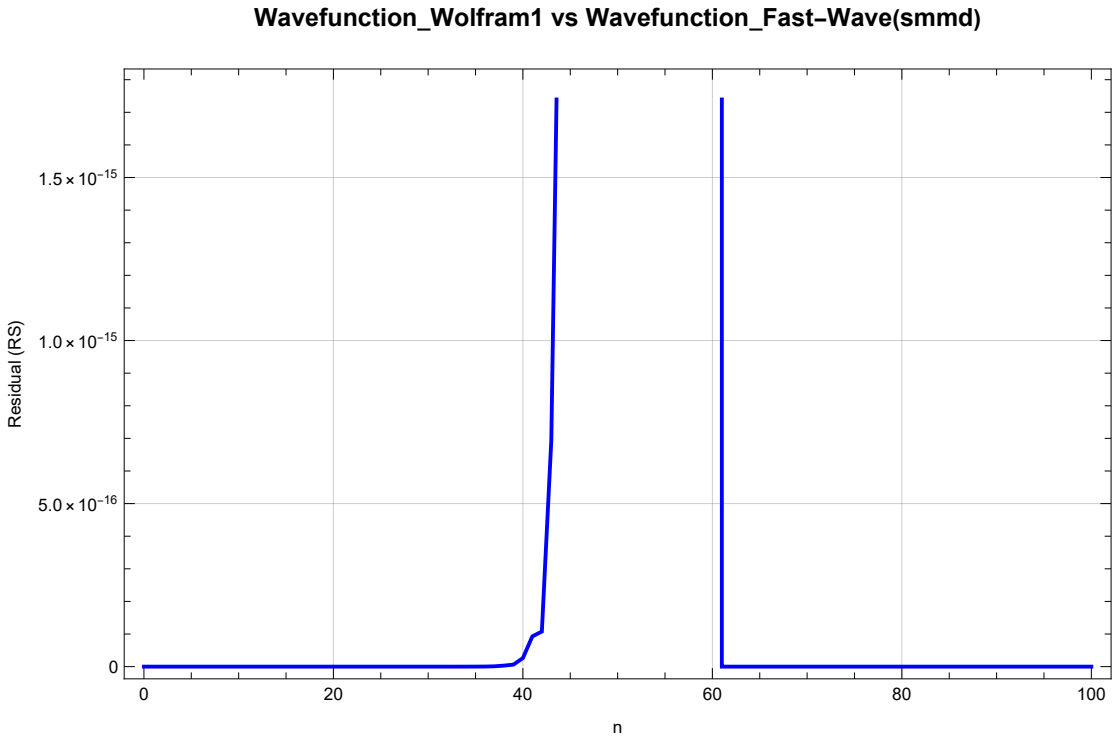
avg(RS) = $[1.0973586208608373206 \times 10^{-81}]$

★ Single-Mode and Multidimensional Function with X = [(-20) → 20; 100]

In[•]:=
```
FastWaveSMMdN100X = SetPrecision[
                    define precisão
   Normal[ExternalEvaluate["Python",          "from fast_wave.wavefunction
   normal  execução externa
       import wavefunction_smmd;import numpy as np;N_max = 100;xmax =
       20.0;xmin=-20.0;xsize=100;X=np.linspace(xmin,xmax,xsize);np.array([
       wavefunction_smmd(n,X) for n in range(N_max+1)]);"]],
   prec];

(*WolframSMMdN100X = WavefunctionMathematica3[Nmax,Xvector,prec];*)
WolframSMMdN100X = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSMMdN100X, WavefunctionMathematica1[i - 1, Xvector, prec]];]
 adiciona a


ResidualMatrix = (WolframSMMdN100X - FastWaveSMMdN100X)^2;

Residual = Mean /@ ResidualMatrix;
           média

ListLinePlot[
gráfico de linha de uma lista de valores
 Transpose[{Nvector, Residual}],
 transposição
 PlotStyle → {Thick, Blue},
 estilo do gráfico  espesso  azul
 Frame → True,
 quadro  verdadeiro
 FrameLabel → {"n", "Residual (RS)"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Style["Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave(smmd)\n", 13, Bold],
  estilo                                                          negrito
 GridLines → Automatic,
 grade de linhas  automático
 PlotLegends → Placed[{" X: [(-20)→20;100], to each value of n ; \n avg(RS) = [" <>
 legenda do gráfico  situado
       ToString[SetPrecision[Mean[Residual], 20],
       converte···  define precisão  média
        TraditionalForm]} <> "]", Below],
        forma tradicional              abaixo
 ImageSize → Large ]
 tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[•]=



X: [(−20)→20;100], to each value of n ;

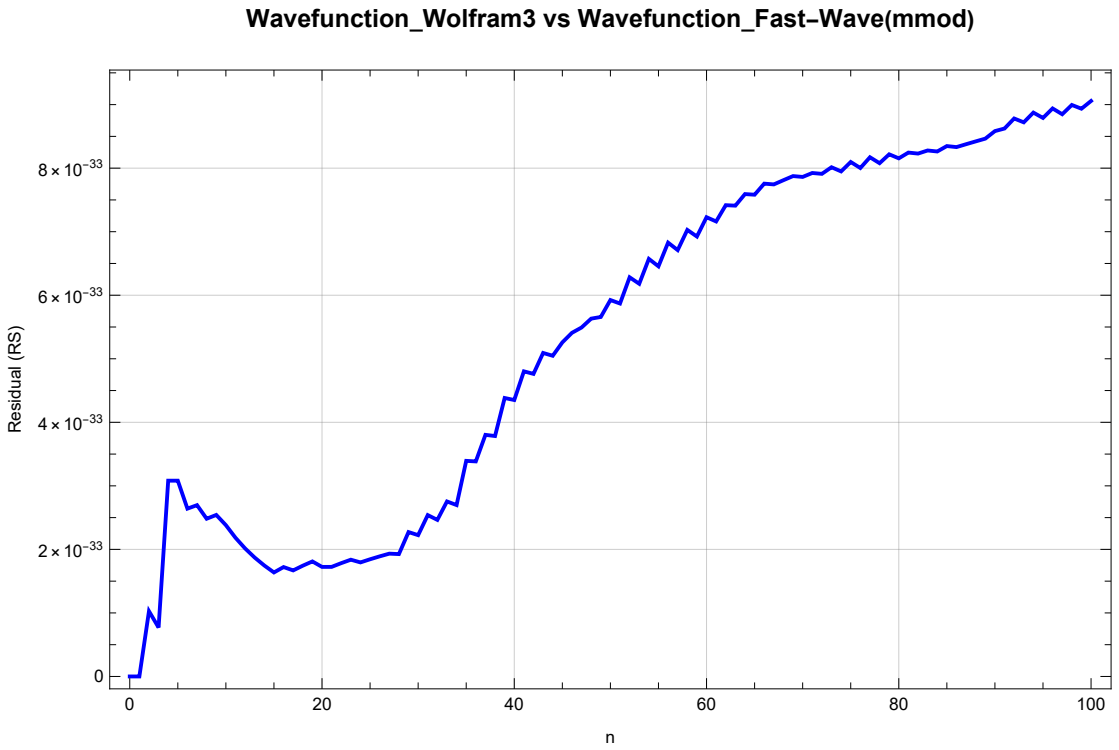
avg(RS) = [$1.2257737831923982400 \times 10^{-9}$]

★ Multi-Mode and Onedimensional Function with x = 1.0

In[ ]:=
```
FastWaveMMOdN100x1 = SetPrecision[
                     define precisão
   Normal[ExternalEvaluate["Python",          "from fast_wave.wavefunction
   normal  execução externa
        import wavefunction_mmod;import numpy as np;N_max = 100;x1
        =1.0;[wavefunction_mmod(n,x1) for n in range(N_max+1)];"]], prec];

WavefunctionMMOdN100x1 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WavefunctionMMOdN100x1 , WavefunctionMathematica3[i - 1, x1, prec]];]
 adiciona a
ResidualList = (WavefunctionMMOdN100x1 - FastWaveMMOdN100x1)^2;

Residual = Mean /@ ResidualList;
           média

ListLinePlot[
gráfico de linha de uma lista de valores
 Transpose[{Nvector, Residual}],
 transposição
 PlotStyle → {Thick, Blue},
 estilo do gráfico  espessc azul
 Frame → True,
 quadro  verdadeiro
 FrameLabel → {"n", "Residual (RS)"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Style["Wavefunction_Wolfram3 vs Wavefunction_Fast-Wave(mmod)\n", 13, Bold],
  estilo                                                                negrito
 GridLines → Automatic,
 grade de linhas automático
 PlotLegends → Placed[{" x: 1.0, to each value of n ; \n avg(RS) = [" <>
 legenda do gráficc situado
      ToString[SetPrecision[Mean[Residual], 20],
      converte···  define precisão  média
       TraditionalForm]} <> "]", Below],
       forma tradicional              abaixo
 ImageSize → Large ]
 tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[•]=



x: 1.0, to each value of n ;

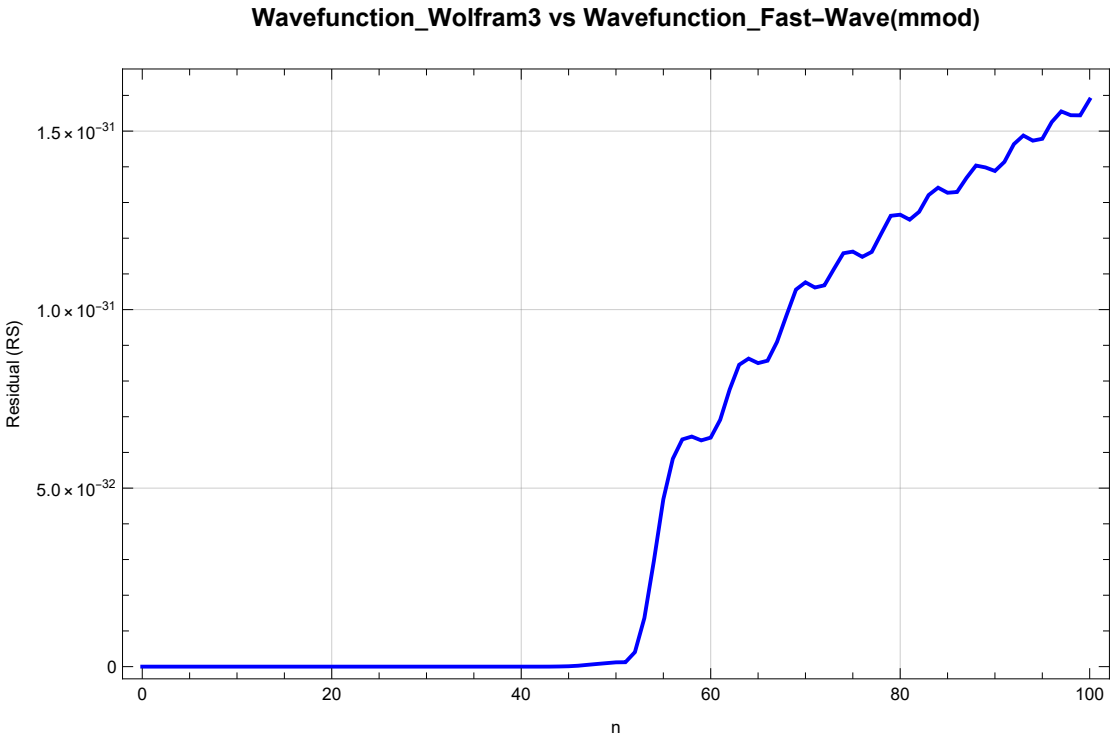
avg(RS) = [5.3425573728452427011 × 10^-33]

★ Multi-Mode and Onedimensional Function with x = 10.0

In[•]:=
```
FastWaveMMOdN100x10 = SetPrecision[
                      define precisão
   Normal[ExternalEvaluate["Python",          "from fast_wave.wavefunction
   normal  execução externa
        import wavefunction_mmod;import numpy as np;N_max = 100;x2
        =10.0;[wavefunction_mmod(n,x2) for n in range(N_max+1)];"]], prec];

WavefunctionMMOdN100x10 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WavefunctionMMOdN100x10 , WavefunctionMathematica3[i - 1, x2, prec]];]
 adiciona a
ResidualList = (WavefunctionMMOdN100x10 - FastWaveMMOdN100x10)^2;

Residual = Mean /@ ResidualList;
           média

ListLinePlot[
gráfico de linha de uma lista de valores
 Transpose[{Nvector, Residual}],
 transposição
 PlotStyle → {Thick, Blue},
 estilo do gráfico  espesso  azul
 Frame → True,
 quadro  verdadeiro
 FrameLabel → {"n", "Residual (RS)"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Style["Wavefunction_Wolfram3 vs Wavefunction_Fast-Wave(mmod)\n", 13, Bold],
  estilo                                                          negrito
 GridLines → Automatic,
 grade de linhas  automático
 PlotLegends → Placed[{" x: 10.0, to each value of n ; \n avg(RS) = [" <>
 legenda do gráfico  situado
       ToString[SetPrecision[Mean[Residual], 20],
       converte···  define precisão  média
        TraditionalForm]} <> "]", Below],
        forma tradicional            abaixo
 ImageSize → Large ]
 tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[•]=



x: 10.0, to each value of n ;

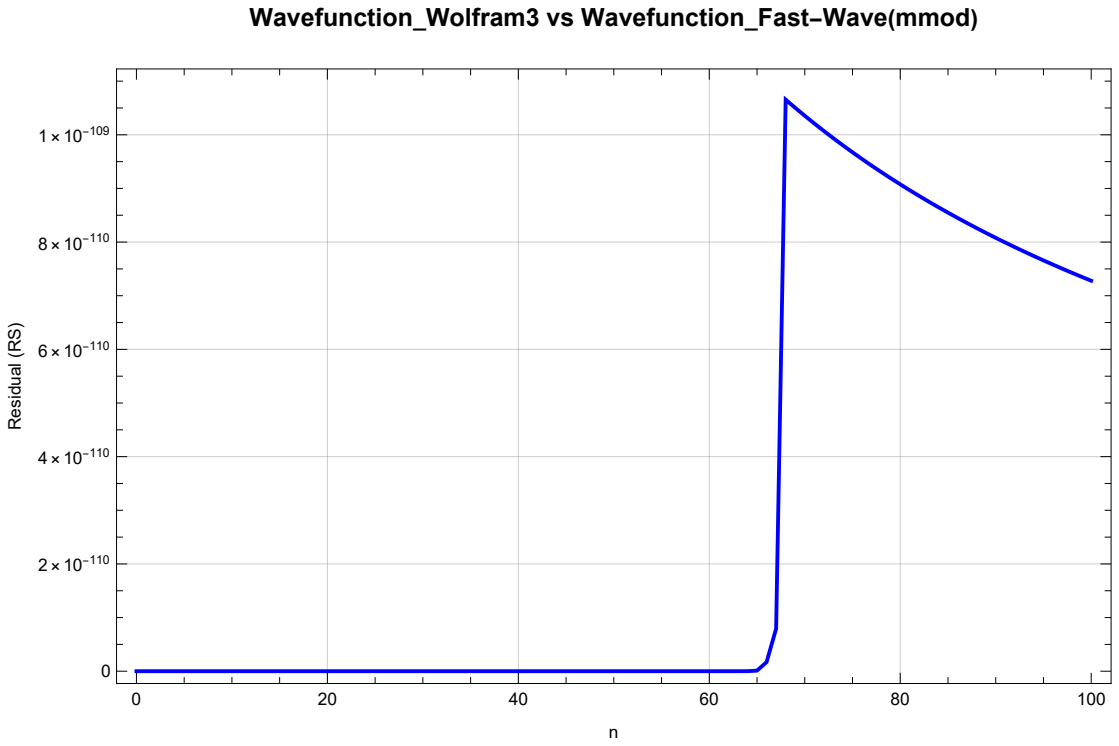
avg(RS) = [5.2606597608900719008 × 10^-32]

★ Multi-Mode and Onedimensional Function with x = 20.0

In[•]:=
```
FastWaveMMOdN100x20 = SetPrecision[
                      define precisão
   Normal[ExternalEvaluate["Python",          "from fast_wave.wavefunction
   normal  execução externa
       import wavefunction_mmod;import numpy as np;N_max = 100;x3
       =20.0;[wavefunction_mmod(n,x3) for n in range(N_max+1)];"]], prec];

WavefunctionMMOdN100x20 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WavefunctionMMOdN100x20 , WavefunctionMathematica3[i - 1, x3, prec]];]
 adiciona a
ResidualList = (WavefunctionMMOdN100x20 - FastWaveMMOdN100x20)^2;

Residual = Mean /@ ResidualList;
           média

ListLinePlot[
gráfico de linha de uma lista de valores
 Transpose[{Nvector, Residual}],
 transposição
 PlotStyle → {Thick, Blue},
 estilo do gráfico  espesso  azul
 Frame → True,
 quadro  verdadeiro
 FrameLabel → {"n", "Residual (RS)"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Style["Wavefunction_Wolfram3 vs Wavefunction_Fast-Wave(mmod)\n", 13, Bold],
  estilo                                                          negrito
 GridLines → Automatic,
 grade de linhas  automático
 PlotLegends → Placed[{" x: 20.0, to each value of n ; \n avg(RS) = [" <>
 legenda do gráfico  situado
       ToString[SetPrecision[Mean[Residual], 20],
       converte···  define precisão  média
        TraditionalForm]} <> "]", Below],
        forma tradicional            abaixo
 ImageSize → Large ]
 tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[•]=



x: 20.0, to each value of n ;

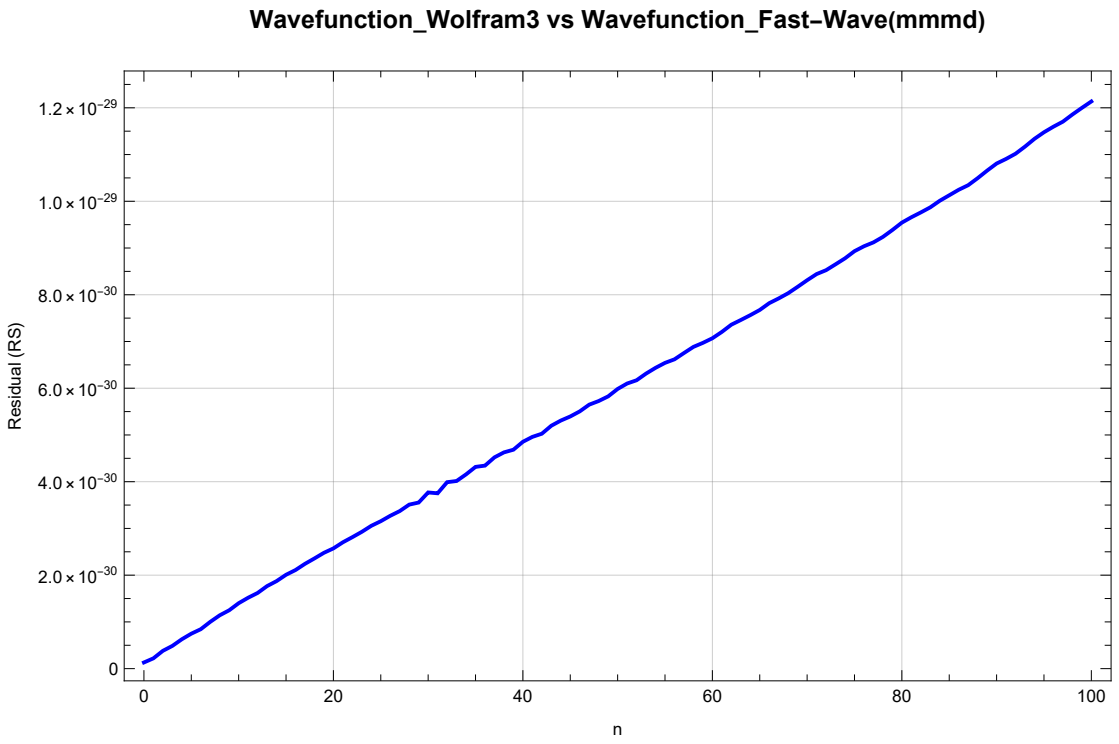
avg(RS) = [2.8719935011557144350 × 10^-110]

★ Multi-Mode and Multidimensional Function with X = [(-20) → 20; 100]

In[•]:=
```
FastWaveMMMdN100X = SetPrecision[
                    define precisão
   Normal[ExternalEvaluate["Python",          "from fast_wave.wavefunction
   normal  execução externa
       import wavefunction_mmmd;import numpy as np;N_max = 100;xmax =
       20.0;xmin=-20.0;xsize=100;X=np.linspace(xmin,xmax,xsize);[
       wavefunction_mmmd(n,X) for n in range(N_max+1)];"]],
   prec];

WolframSMMdN100X = {};

For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSMMdN100X, WavefunctionMathematica3[i - 1, Xvector, prec]];]
 adiciona a

ResidualMatrixList = (WolframSMMdN100X - FastWaveMMMdN100X)^2;

Residual = Mean /@ (Flatten /@ ResidualMatrixList);
           média     achatar

ListLinePlot[
gráfico de linha de uma lista de valores
 Transpose[{Nvector, Residual}],
 transposição
 PlotStyle → {Thick, Blue},
 estilo do gráfico  espessc  azul
 Frame → True,
 quadro   verdadeiro
 FrameLabel → {"n", "Residual (RS)"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Style["Wavefunction_Wolfram3 vs Wavefunction_Fast-Wave(mmmd)\n", 13, Bold],
  estilo                                                              negrito
 GridLines → Automatic,
 grade de linhas  automático
 PlotLegends → Placed[{" X: [(-20)→20;100], to each value of n ; \n avg(RS) = [" <>
 legenda do gráficc  situado
       ToString[SetPrecision[Mean[Residual], 20],
       converte···  define precisão  média
        TraditionalForm]} <> "]", Below],
        forma tradicional              abaixo
 ImageSize → Large ]
 tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[•]=



X: [(−20)→20;100], to each value of n ;

avg(RS) = [6.0291626306746868592 × 10^-30]