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[
             define precisão
              2 x (wavefunction[[i]] / Sqrt[2 (i)]) - Sqrt[(i - 1) / i] wavefunction[[i - 1]], prec]];
                                      raiz quadrada  raiz quadrada
           wavefunction];
```

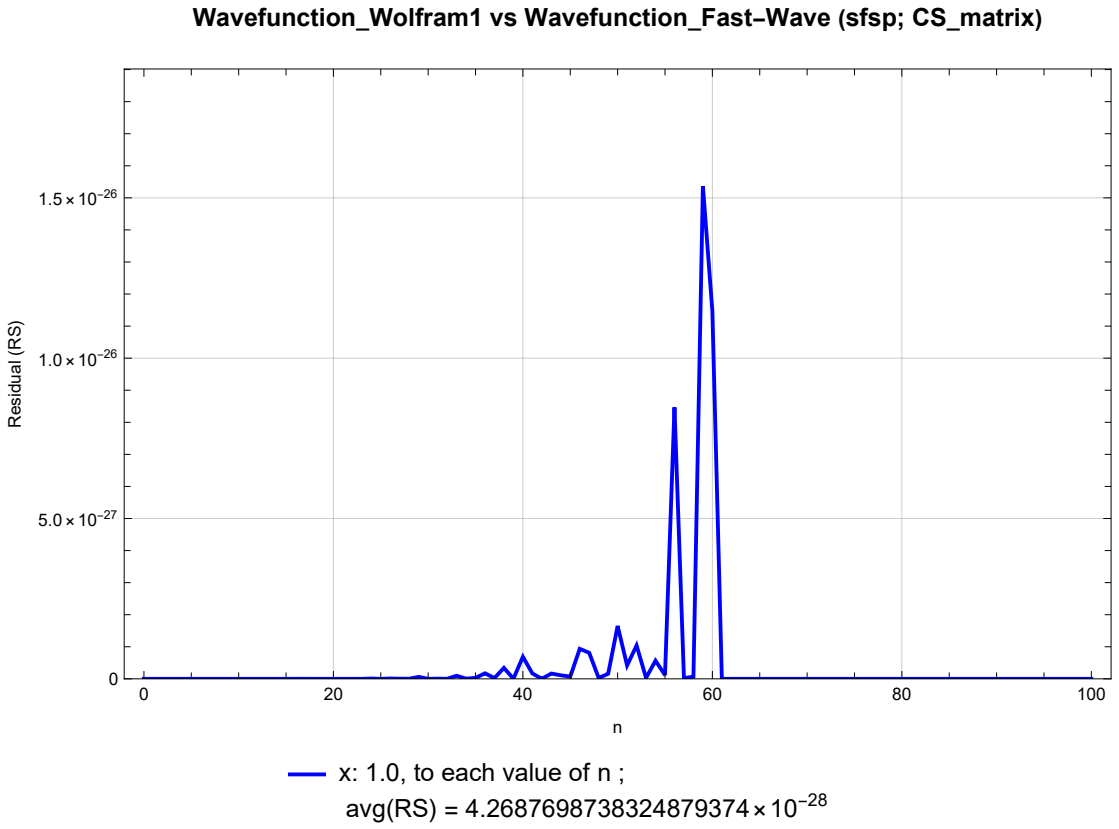
Tests

★ Single Fock and Single Position Function with x = 1.0 to the Normalized Hermite Coefficients Matrix

```
FastWaveSFSpN100x1 = SetPrecision[Normal[ExternalEvaluate["Python",
                     define precisão   normal   execução externa
      "import fast_wave.wavefunction_numba as wn;import numpy as np;N_max =
        100;x1 = 1.0;np.array([wn.psi_n_single_fock_single_position(n,x1)
        for n in range(N_max+1)]);"]], prec];

(*WolframSFSpN100x1 = WavefunctionMathematica3[Nmax,x1,prec];*)
WolframSFSpN100x1 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSFSpN100x1, WavefunctionMathematica1[i - 1, x1, prec]];]
 adiciona a

Residual = (WolframSFSpN100x1 - FastWaveSFSpN100x1)^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 → Style[
 etiqueta de gr··  estilo
   "Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave (sfsp; CS_matrix)\n", 13, Bold],
                                                                          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
 PlotRange → {{Automatic, Automatic}, {0, 1.2 * 10^(-25.8)}}
 intervalo do gráf···  automático  automático
]
```

```
Functionality Test Passed: True
```

Out[•]=

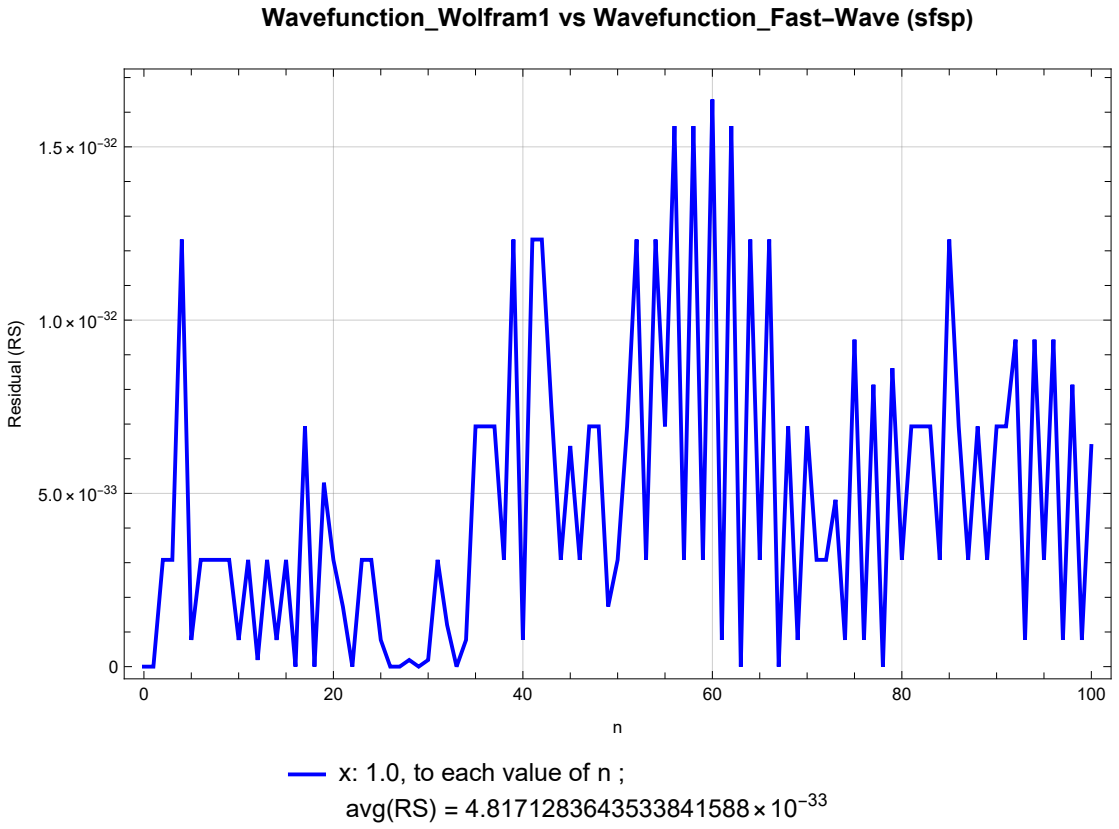

Wavefunction_Wolfram1 vs Wavefunction_Fast–Wave (sfsp; CS_matrix)
Residual (RS)
1.5 × 10^−26
1.0 × 10^−26
5.0 × 10^−27
0
0
20
40
60
80
100
n
x: 1.0, to each value of n ;
avg(RS) = 4.2687698738324879374 × 10^−28

★ Single Fock and Single Position Function with x = 1.0

```
FastWaveSFSpN100x1 = SetPrecision[Normal[ExternalEvaluate["Python",
                     define precisão  normal  execução externa
      "import fast_wave.wavefunction_numba as wn;import numpy as np;N_max = 100;x1 =
        1.0;np.array([wn.psi_n_single_fock_single_position(n,x1,CS_matrix=False)
                                                                    falso
         for n in range(N_max+1)]);"]], prec];

(*WolframSFSpN100x1 = WavefunctionMathematica3[Nmax,x1,prec];*)
WolframSFSpN100x1 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSFSpN100x1, WavefunctionMathematica1[i - 1, x1, prec]];]
 adiciona a

Residual = (WolframSFSpN100x1 - FastWaveSFSpN100x1)^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 (sfsp)\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
```

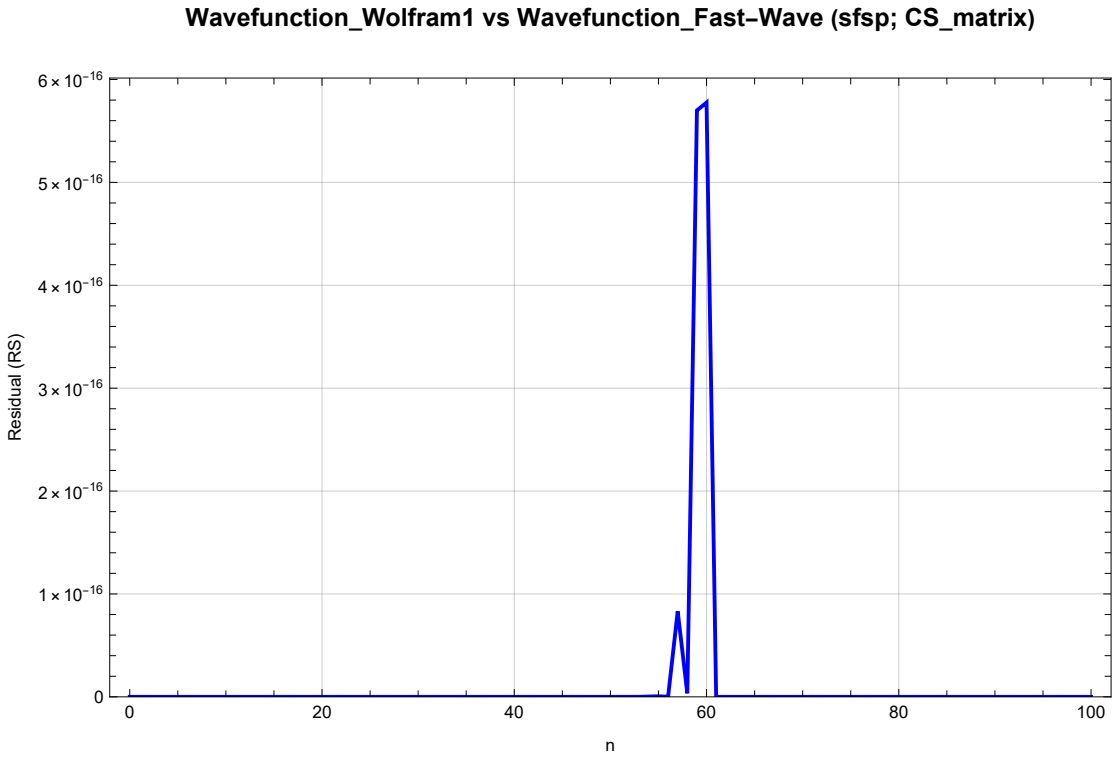
Out[•]=


Wavefunction_Wolfram1 vs Wavefunction_Fast–Wave (sfsp)
Residual (RS)
1.5 × 10^-32
1.0 × 10^-32
5.0 × 10^-33
0
0
20
40
60
80
100
n
x: 1.0, to each value of n ;
avg(RS) = 4.8171283643533841588 × 10^-33

★ Single Fock and Single Position Function with x = 10.0 to the Normalized Hermite Coefficients Matrix

```
FastWaveSFSpN100x2 = SetPrecision[Normal[ExternalEvaluate["Python",
                     define precisão  normal  execução externa
      "import fast_wave.wavefunction_numba as wn;import numpy as np;N_max = 100;x2
        = 10.0;np.array([wn.psi_n_single_fock_single_position(n,x2)
        for n in range(N_max+1)]);"]], prec];

(*WolframSFSpN100x2 = WavefunctionMathematica3[Nmax,x2,prec];*)

WolframSFSpN100x2 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSFSpN100x2, WavefunctionMathematica1[i - 1, x2, prec]];]
 adiciona a

Residual = (WolframSFSpN100x2 - FastWaveSFSpN100x2)^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 → Style[
 etiqueta de gr··  estilo
   "Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave (sfsp; CS_matrix)\n", 13, Bold],
                                                                          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
 PlotRange → {{Automatic, Automatic}, {0, 1.2 * 10^(-15.3)}}
 intervalo do gráf···  automático  automático
]
Functionality Test Passed: True
```

Out[•]=



x: 10.0, to each value of n ;

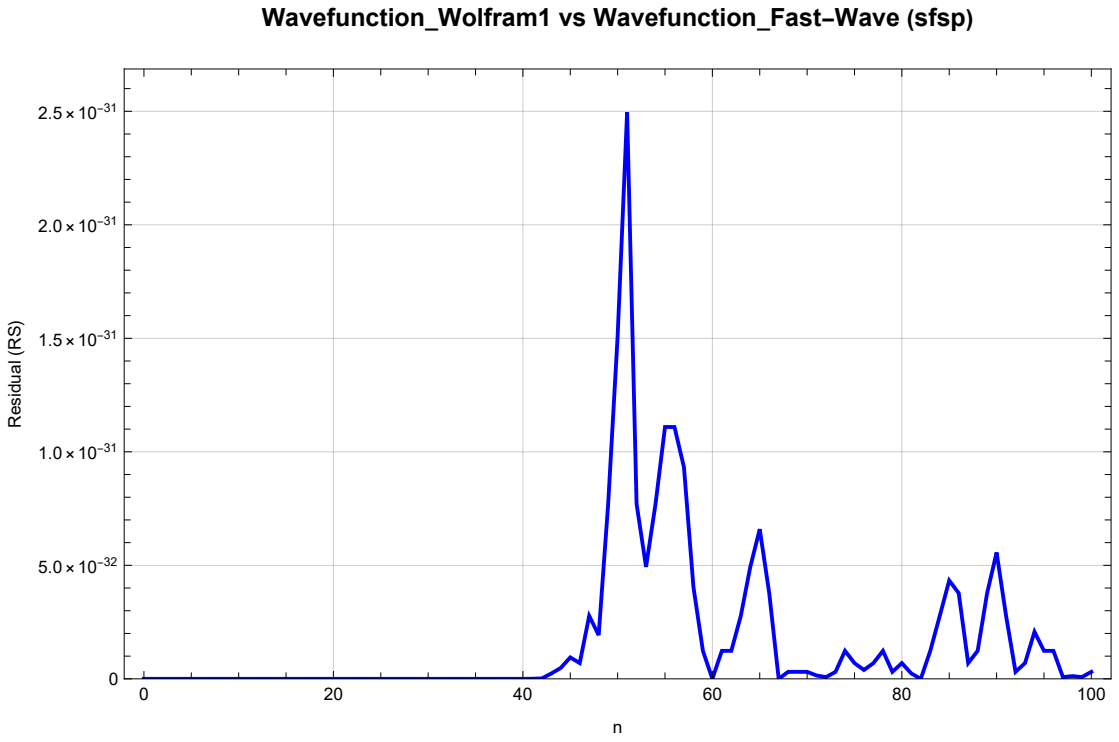
avg(RS) = $[1.2227826509448020424 \times 10^{-17}]$

★ Single Fock and Single Position Function with x = 10.0

```
FastWaveSFSpN100x2 = SetPrecision[Normal[ExternalEvaluate["Python",
                     define precisão  normal  execução externa
      "import fast_wave.wavefunction_numba as wn;import numpy as np;N_max = 100;x2 =
        10.0;np.array([wn.psi_n_single_fock_single_position(n,x2,CS_matrix=False
                                                                          falso
        ) for n in range(N_max+1)]);"]], prec];

(*WolframSFSpN100x2 = WavefunctionMathematica3[Nmax,x2,prec];*)

WolframSFSpN100x2 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSFSpN100x2, WavefunctionMathematica1[i - 1, x2, prec]];]
 adiciona a

Residual = (WolframSFSpN100x2 - FastWaveSFSpN100x2)^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 (sfsp)\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
 PlotRange → {{Automatic, Automatic}, {0, 1.2 * 10^(-30.65)}}
 intervalo do gráf···  automático  automático
]
```

```
Functionality Test Passed: True
```

Out[•]=



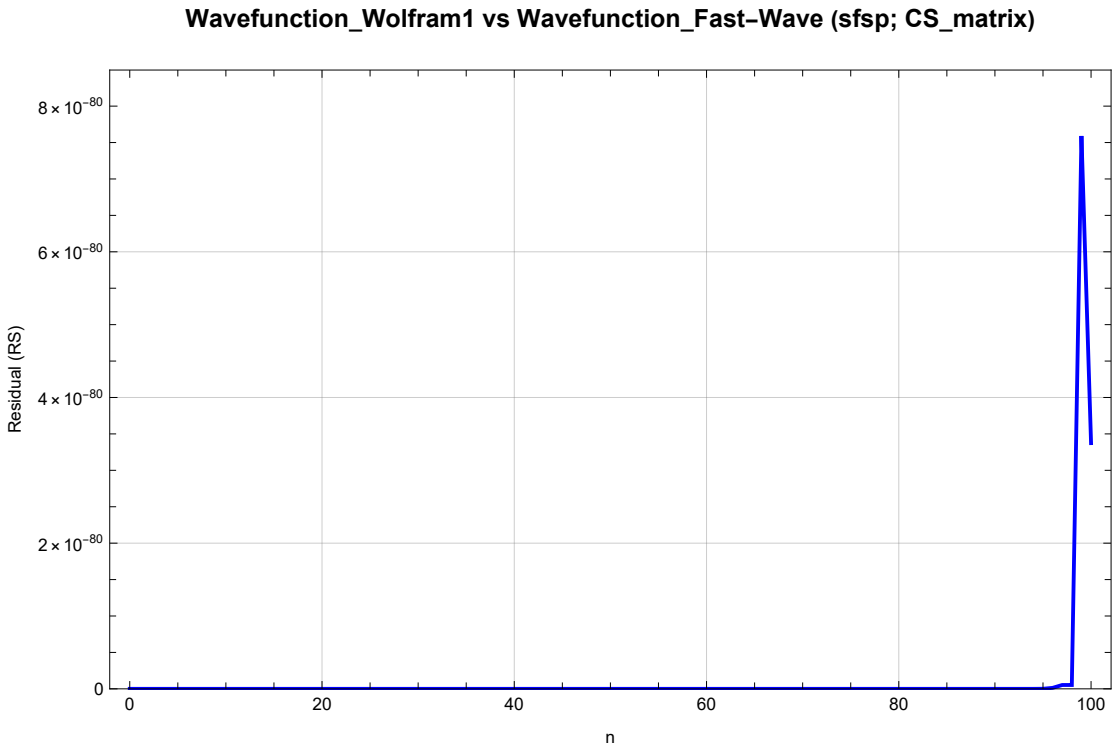
x: 10.0, to each value of n ;

avg(RS) = $[1.7001583826933030519 \times 10^{-32}]$

★ Single Fock and Single Position Function with x = 20.0 to the Normalized Hermite Coefficients Matrix

```
FastWaveSFSpN100x3 = SetPrecision[Normal[ExternalEvaluate["Python",
                     define precisão  normal  execução externa
      "import fast_wave.wavefunction_numba as wn;import numpy as np;N_max = 100;x3
        = 20.0;np.array([wn.psi_n_single_fock_single_position(n,x3)
        for n in range(N_max+1)]);"]], prec];

(*WolframSFSpN100x3 = WavefunctionMathematica3[Nmax,x3,prec];*)
WolframSFSpN100x3 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSFSpN100x3, WavefunctionMathematica1[i - 1, x3, prec]];]
 adiciona a

Residual = (WolframSFSpN100x3 - FastWaveSFSpN100x3)^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 → Style[
 etiqueta de gr··  estilo
   "Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave (sfsp; CS_matrix)\n", 13, Bold],
                                                                          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
 PlotRange → {{Automatic, Automatic}, {0, 1.2 * 10^(-79.15)}}
 intervalo do gráf···  automático  automático
]

Functionality Test Passed: True
```

Out[•]=



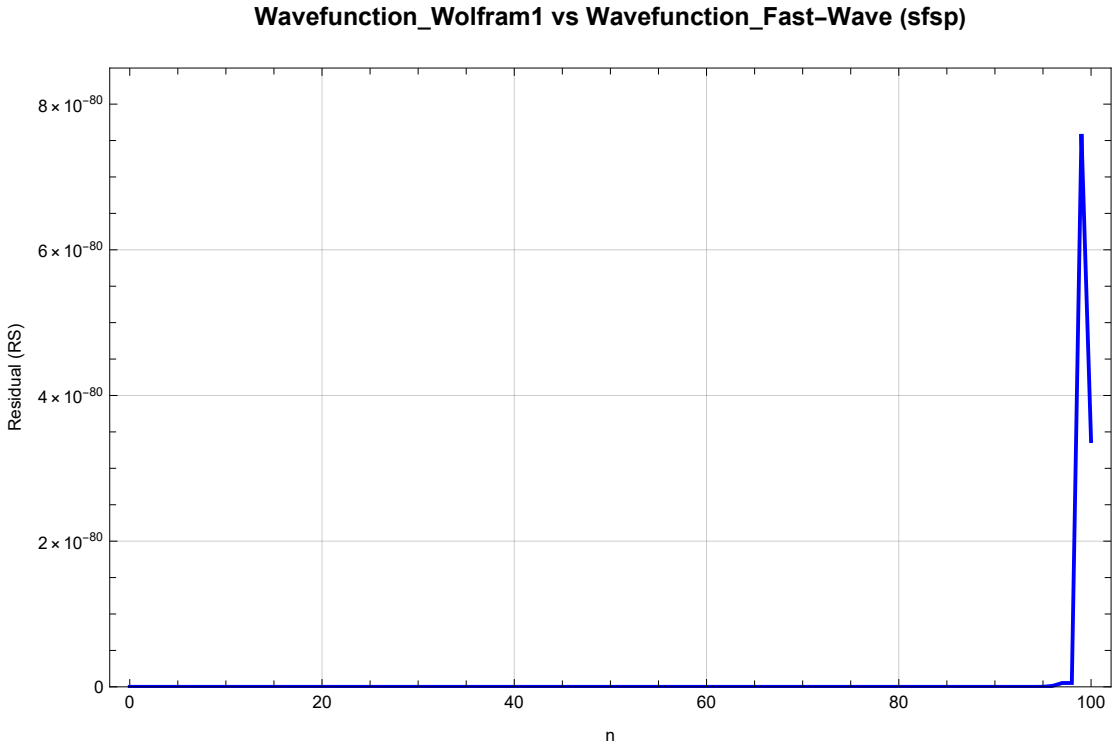
x: 20.0, to each value of n ;

avg(RS) = [1.0973586208608373206 × 10^-81]

★ Single Fock and Single Position Function with x = 20.0

```
FastWaveSFSpN100x3 = SetPrecision[Normal[ExternalEvaluate["Python",
                     define precisão  normal  execução externa
      "import fast_wave.wavefunction_numba as wn;import numpy as np;N_max = 100;x3
        = 20.0;np.array([wn.psi_n_single_fock_single_position(n,x3,
        CS_matrix= False) for n in range(N_max+1)]);"]], prec];
                    falso

(*WolframSFSpN100x3 = WavefunctionMathematica3[Nmax,x3,prec];*)
WolframSFSpN100x3 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSFSpN100x3, WavefunctionMathematica1[i - 1, x3, prec]];]
 adiciona a

Residual = (WolframSFSpN100x3 - FastWaveSFSpN100x3)^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 (sfsp)\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
 PlotRange → {{Automatic, Automatic}, {0, 1.2 * 10^(-79.15)}}
 intervalo do gráf···  automático  automático
]
Functionality Test Passed: True
```

Out[•]=



x: 20.0, to each value of n ;

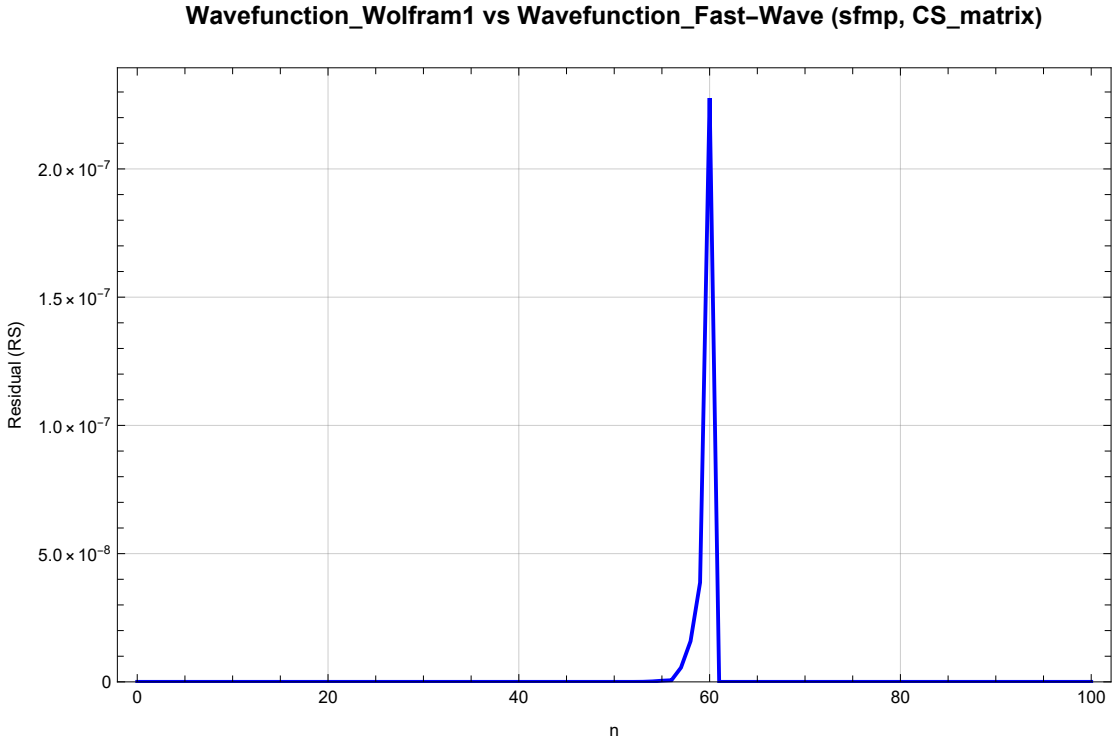
avg(RS) = [1.0973586208608373206 × 10^-81]

★ Single Fock and Multiple Position Function with X = [(-20) → 20; 100] to the Normalized Hermite Coefficients Matrix

```
FastWaveSFMpN100X = SetPrecision[
                    define precisão
   Normal[ExternalEvaluate["Python",          "import fast_wave.wavefunction_numba
   normal  execução externa
        as wn;import numpy as np;N_max = 100;xmax =
        20.0;xmin=-20.0;xsize=100;X=np.linspace(xmin,xmax,xsize);np.array([wn.
        psi_n_single_fock_multiple_position(n,X)
        for n in range(N_max+1)]);"]], prec];

(*WolframSFMpN100X = WavefunctionMathematica3[Nmax,Xvector,prec];*)
WolframSFMpN100X = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSFMpN100X, WavefunctionMathematica1[i - 1, Xvector, prec]];]
 adiciona a


ResidualMatrix = (WolframSFMpN100X - FastWaveSFMpN100X)^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 → Style[
 etiqueta de gr··  estilo
   "Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave (sfmp, CS_matrix)\n", 13, Bold],
                                                                          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
 PlotRange → {{Automatic, Automatic}, {0, 1.2 * 10^(-6.7)}}
 intervalo do gráf···  automático  automático
]
```

```
Functionality Test Passed: True
```

Out[•]=



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

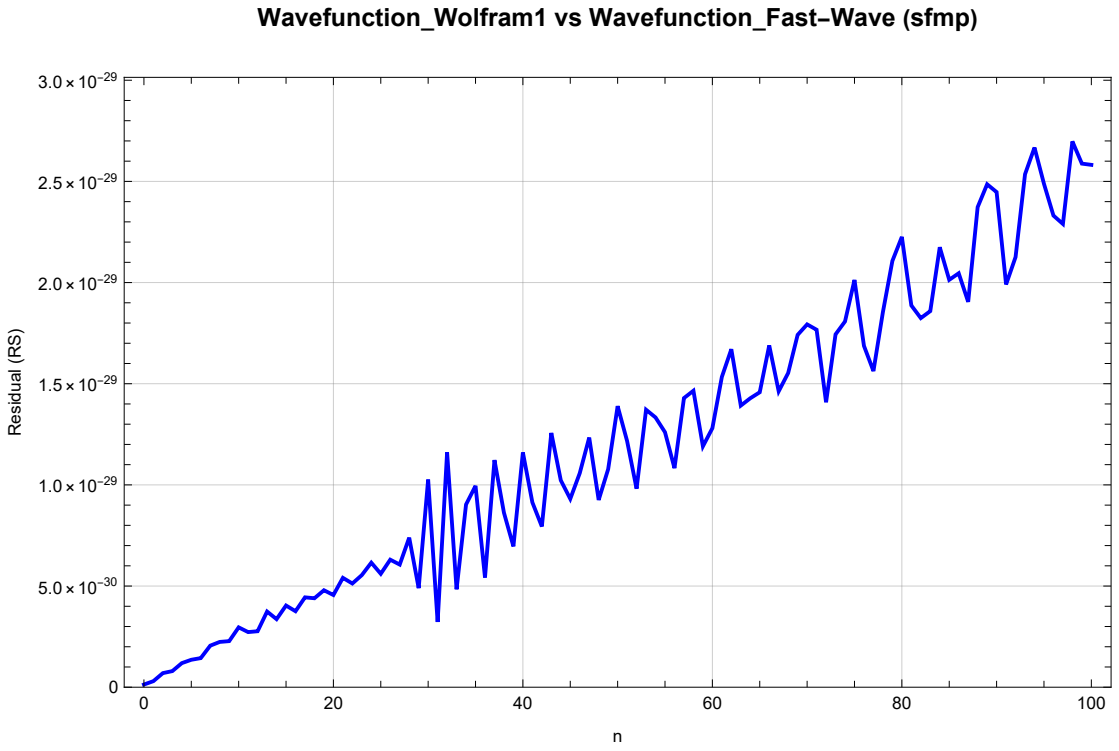
avg(RS) = [$2.8624974439968216501 \times 10^{-9}$]

★ Single Fock and Multiple Position Function with X = [(-20) → 20; 100]

```
FastWaveSFMpN100X = SetPrecision[
                    define precisão
   Normal[ExternalEvaluate["Python",          "import fast_wave.wavefunction_numba
   normal  execução externa
        as wn;import numpy as np;N_max = 100;xmax =
        20.0;xmin=-20.0;xsize=100;X=np.linspace(xmin,xmax,xsize);np.array([wn.
        psi_n_single_fock_multiple_position(n,X,CS_matrix=False)
                                                              falso
        for n in range(N_max+1)]);"]], prec];

(*WolframSFMpN100X = WavefunctionMathematica3[Nmax,Xvector,prec];*)
WolframSFMpN100X = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframSFMpN100X, WavefunctionMathematica1[i - 1, Xvector, prec]];]
 adiciona a


ResidualMatrix = (WolframSFMpN100X - FastWaveSFMpN100X)^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 (sfmp)\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
 PlotRange → {{Automatic, Automatic}, {0, 1.2 * 10^(-28.6)}}
 intervalo do gráf···  automático  automático
]
```

```
Functionality Test Passed: True
```

Out[•]=



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

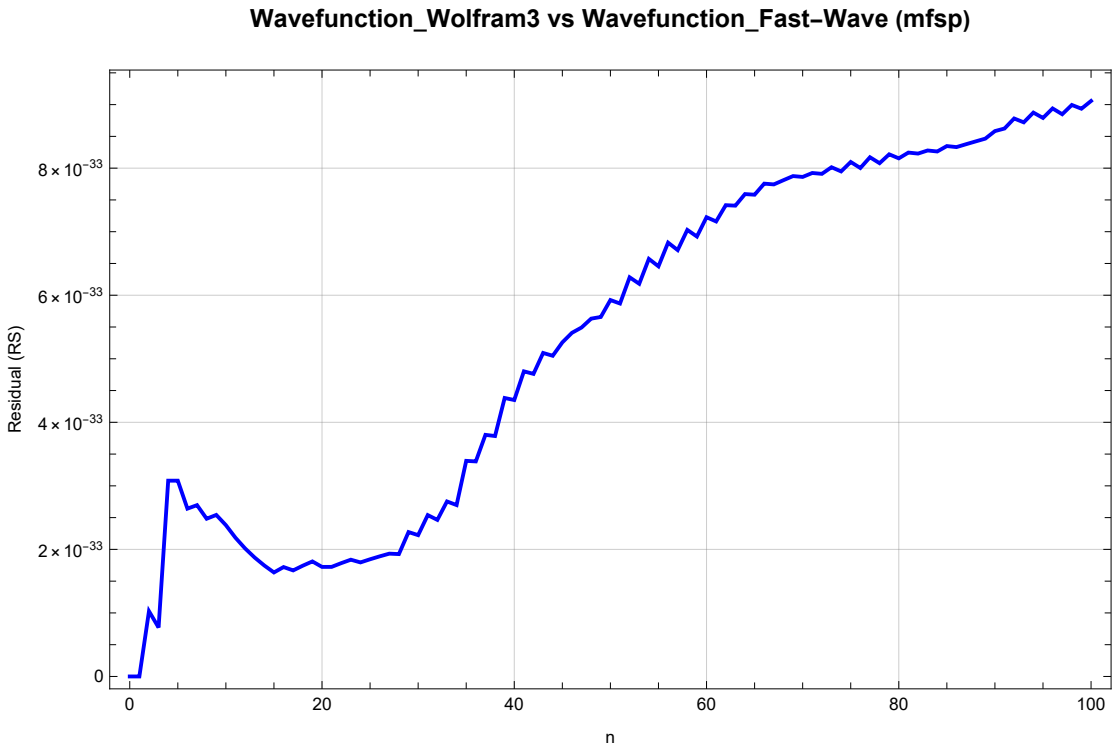
avg(RS) = [1.2113401822724537959 × 10^-29]

★ Multiple Fock and Single Position Function with x = 1.0

In[ ]:=
```
FastWaveMFSpN100x1 = SetPrecision[Normal[ExternalEvaluate["Python",
                     define precisão  normal  execução externa
      "import fast_wave.wavefunction_numba as wn;import numpy as np;N_max
        = 100;x1 =1.0;[wn.psi_n_multiple_fock_single_position(n,x1)
        for n in range(N_max+1)];"]], prec];

WavefunctionMFSpN100x1 = {};
For[i = 1, i ≤ (Nmax + 1), i++,
 para cada
  AppendTo[WavefunctionMFSpN100x1 , WavefunctionMathematica3[i - 1, x1, prec]];]
  adiciona a
ResidualList = (WavefunctionMFSpN100x1 - FastWaveMFSpN100x1)^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 (mfsp)\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[•]=



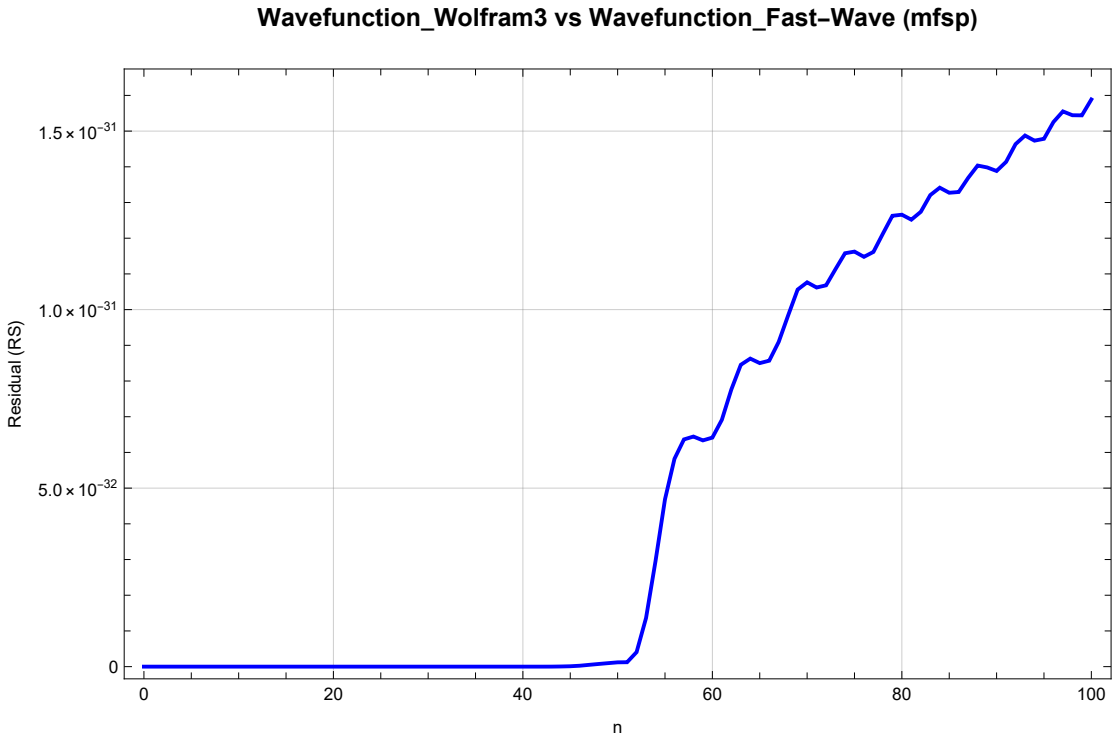
x: 1.0, to each value of n ;

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

★ Multiple Fock and Single Position Function with x = 10.0

```
In[•]:= FastWaveMFSpN100x10 = SetPrecision[Normal[ExternalEvaluate["Python",
                              define precisão  normal  execução externa
        "import fast_wave.wavefunction_numba as wn;import numpy as np;N_max
          = 100;x2 =10.0;[wn.psi_n_multiple_fock_single_position(n,x2)
          for n in range(N_max+1)];"]], prec];

      WavefunctionMFSpN100x10 = {};
      For[i = 1, i ≤ (Nmax + 1), i++,
     para cada
       AppendTo[WavefunctionMFSpN100x10 , WavefunctionMathematica3[i - 1, x2, prec]];]
       adiciona a
      ResidualList = (WavefunctionMFSpN100x10 - FastWaveMFSpN100x10)^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 (mfsp)\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[•]=



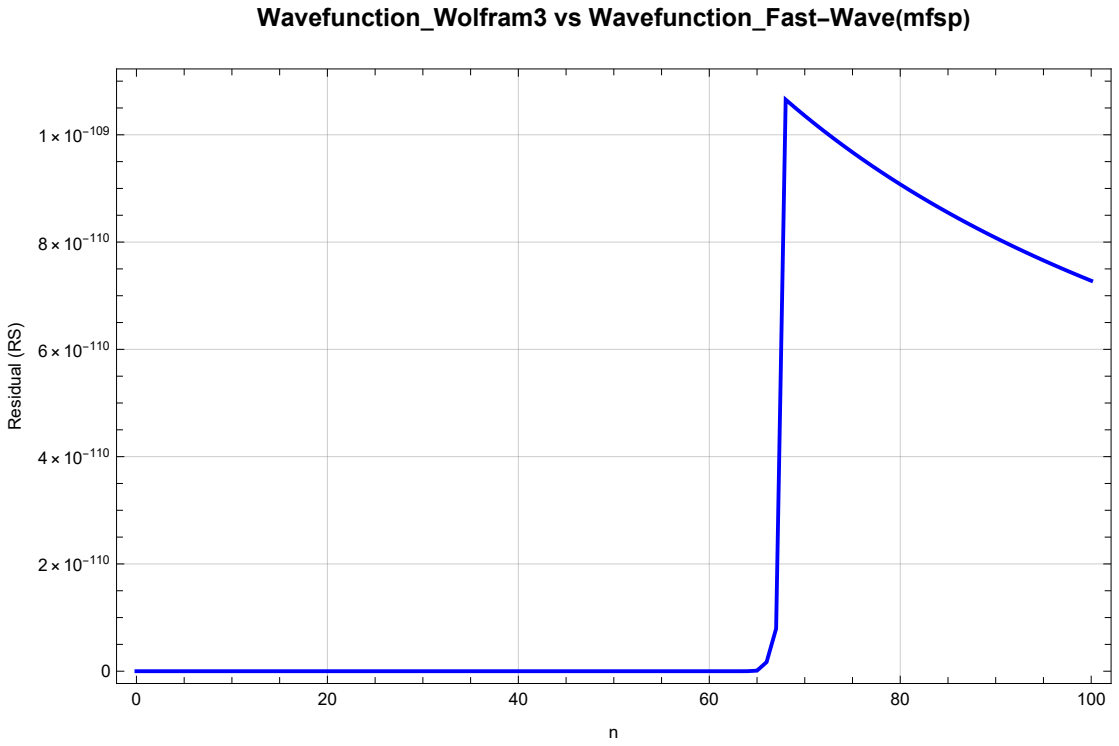
x: 10.0, to each value of n ;

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

★ Multiple Fock and Single Position Function with x = 20.0

```
In[•]:= FastWaveMFSpN100x20 = SetPrecision[Normal[ExternalEvaluate["Python",
                              define precisão   normal   execução externa
        "import fast_wave.wavefunction_numba as wn;import numpy as np;N_max
          = 100;x3 =20.0;[wn.psi_n_multiple_fock_single_position(n,x3)
          for n in range(N_max+1)];"]], prec];

      WavefunctionMFSpN100x20 = {};
      For[i = 1, i ≤ (Nmax + 1), i++,
     para cada
       AppendTo[WavefunctionMFSpN100x20 , WavefunctionMathematica3[i - 1, x3, prec]];]
       adiciona a
      ResidualList = (WavefunctionMFSpN100x20 - FastWaveMFSpN100x20)^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(mfsp)\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 ;

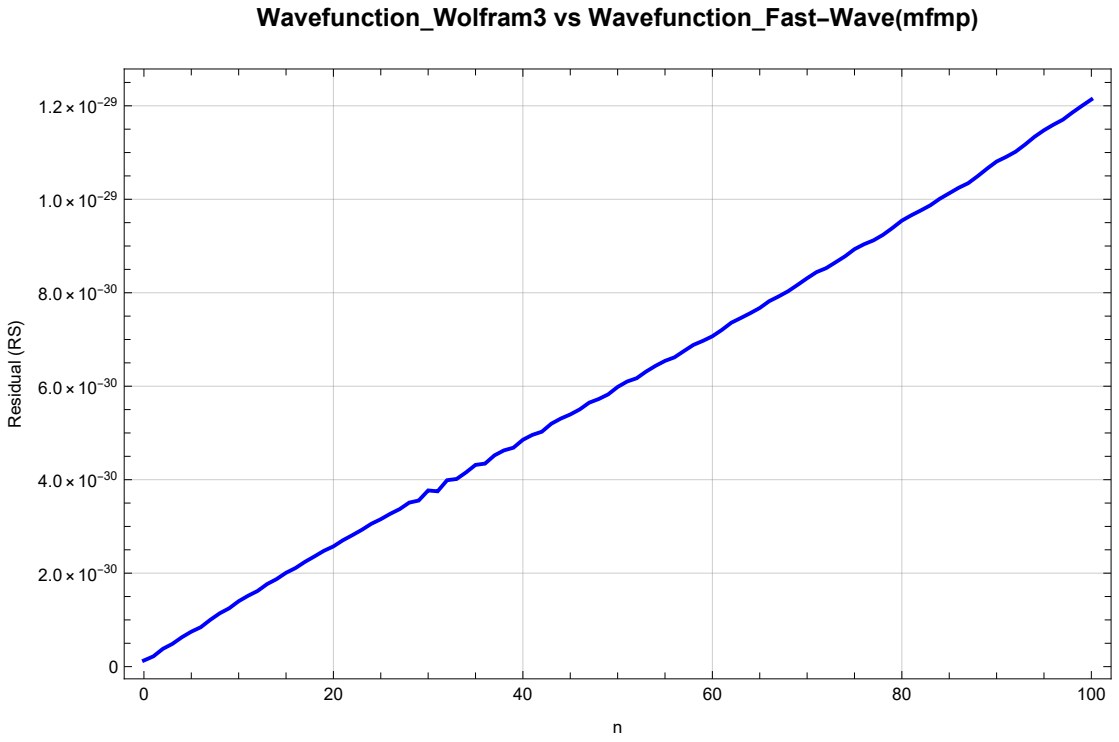
avg(RS) = [2.8719935011557144350 × 10^-110]

★ Multi Fock and Multiple Position Function with X = [(-20) → 20; 100]

In[•]:=
```
FastWaveMFMpN100X = SetPrecision[
                    define precisão
   Normal[ExternalEvaluate["Python",          "import fast_wave.wavefunction_numba
   normal  execução externa
        as wn;import numpy as np;N_max = 100;xmax =
        20.0;xmin=-20.0;xsize=100;X=np.linspace(xmin,xmax,xsize);[wn.psi_n
        _multiple_fock_multiple_position(n,X)
        for n in range(N_max+1)];"]], prec];

WolframMFMpN100X = {};
For[i = 1, i ≤ (Nmax + 1), i++,
para cada
 AppendTo[WolframMFMpN100X, WavefunctionMathematica3[i - 1, Xvector, prec]];]
 adiciona a

ResidualMatrixList = (WolframMFMpN100X - FastWaveMFMpN100X)^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  espesso azul
 Frame → True,
 quadro   verdadeiro
 FrameLabel → {"n", "Residual (RS)"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Style["Wavefunction_Wolfram3 vs Wavefunction_Fast-Wave(mfmp)\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]