Global Variables

In[21]:=
```
prec = 100;

Nmax = 100;

x = 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[15]:=
```
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];
   wavefunction = Table[SetPrecision[0, prec], {n + 1}, {Length[x]}];
   wavefunction[[1]] = SetPrecision[Pi^(-1 / 4) Exp[-(x^2) / 2], prec];
   For[i = 1, i ≤ n, i++,
    wavefunction[[i + 1]] =
     SetPrecision[2 x (wavefunction[[i]] / Sqrt[2 (i)]) -
        Sqrt[(i - 1) / i] wavefunction[[i - 1]], prec]];
   wavefunction];
```

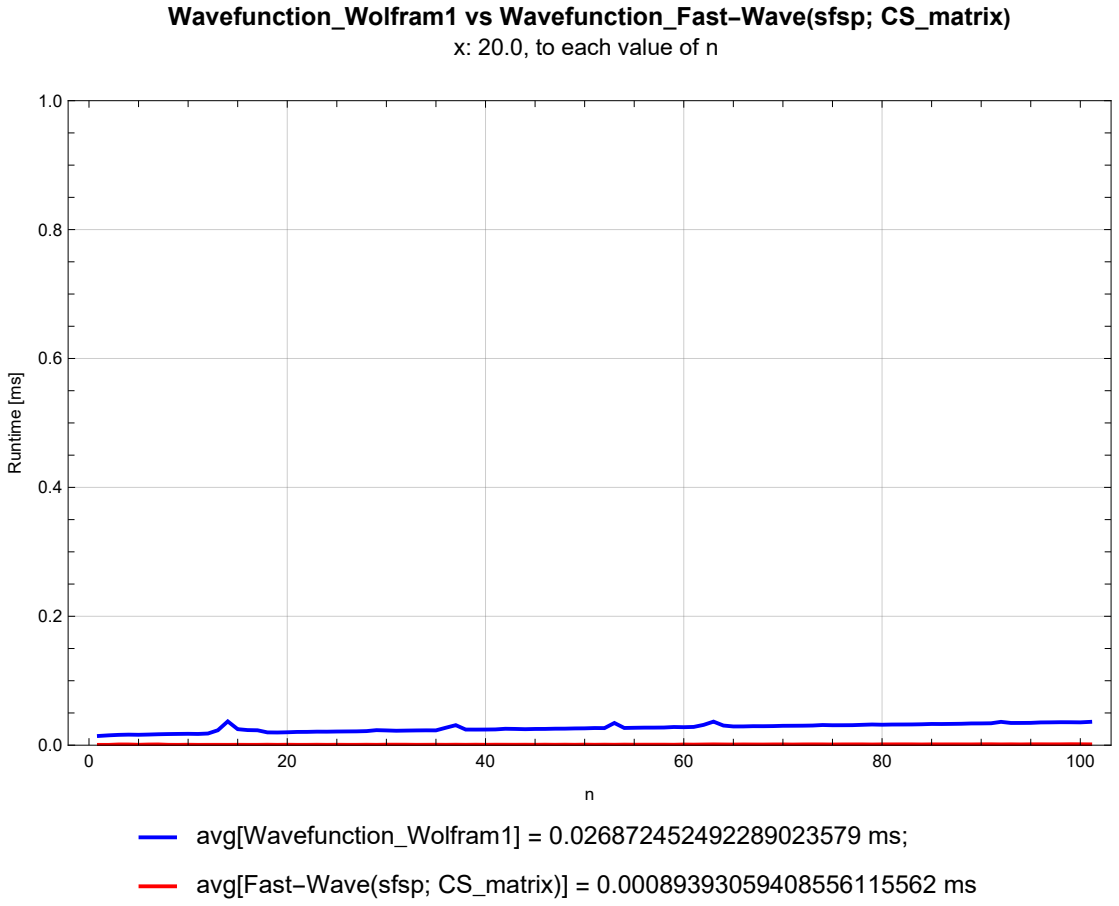
Tests

★ Single Fock and Single Position speed test with the Normalized Coefficients Matrix

In[30]:=
```
(*To use timeit.repeat in the Python is a option too*)

FastWaveSFSPList = Normal[ExternalEvaluate["Python",
                   normal  execução externa
    "import fast_wave.wavefunction_numba as wn;import timeit;N_max = 100;x =
      20.0;[(timeit.timeit(lambda : wn.psi_n_single_fock_single_position(n,
      x) , number=10000)/10000)*1000 for n in range(N_max+1)];"]];

WavefunctionWolframSFSPList = {};
For[i = 1, i ≤ (Nmax + 1), i++, AppendTo[WavefunctionWolframSFSPList,
para cada                          adiciona a
   RepeatedTiming[WavefunctionMathematica1[i - 1, x, prec]][[1]] * 1000];]
   cronometra repetidamente

ListLinePlot[
gráfico de linha de uma lista de valores
 {WavefunctionWolframSFSPList, FastWaveSFSPList},
 PlotStyle → {Blue, Red},
 estilo do gráfico  azul   vermelho
 Frame → True,
 quadro   verdadeiro
 FrameLabel → {"n", "Runtime [ms]"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Row[{Style["Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave(sfsp; CS_matrix)",
   linha  estilo
      Bold, 13], "\n", Style["x: 20.0, to each value of n \n", 12]}],
       negrito          estilo
 GridLines → Automatic,
 grade de linhas  automático
 PlotLegends →
 legenda do gráfico
  Placed[{" avg[Wavefunction_Wolfram1] = " <> ToString[SetPrecision[Mean[
   situado                                  converte···  define precisão  média
         WavefunctionWolframSFSPList], 20], TraditionalForm] <> " ms;",
                                              forma tradicional
    " avg[Fast-Wave(sfsp; CS_matrix)] = " <> ToString[
                                              converte em cadeia de caracteres
      SetPrecision[Mean[FastWaveSFSPList], 20], TraditionalForm] <> " ms"}, Below],
       define precisão  média                  forma tradicional              abaixo
 ImageSize → Large ,
 tamanho da ···  grande
 PlotRange → {{Automatic, Automatic}, {0, 1.0}}
 intervalo do gráf···  automático  automático
]
```

```
Functionality Test Passed: True
```

Out[33]=

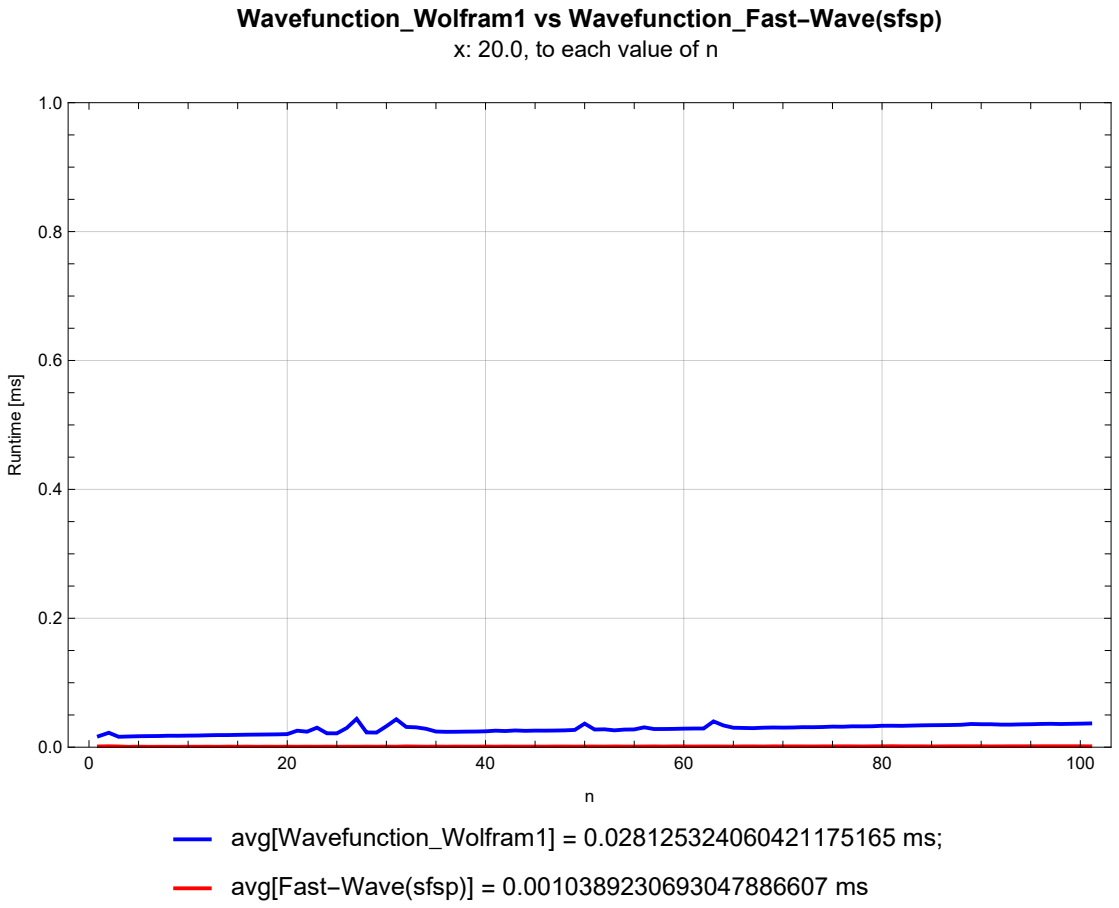

Wavefunction_Wolfram1 vs Wavefunction_Fast−Wave(sfsp; CS_matrix)
x: 20.0, to each value of n
Runtime [ms]
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
n
avg[Wavefunction_Wolfram1] = 0.026872452492289023579 ms;
avg[Fast−Wave(sfsp; CS_matrix)] = 0.00089393059408556115562 ms

★ Single Fock and Single Position speed test

In[34]:=
```
FastWaveSFSPList =
  Normal[ExternalEvaluate["Python",          "import fast_wave.wavefunction_numba
   normal  execução externa
      as wn;import timeit;N_max = 100;x = 20.0;[(timeit.timeit(lambda
      : wn.psi_n_single_fock_single_position(n, x, CS_matrix=False)
                                                                  falso
      , number=10000)/10000)*1000 for n in range(N_max+1)];"]];

WavefunctionWolframSFSPList = {};
For[i = 1, i ≤ (Nmax + 1), i++, AppendTo[WavefunctionWolframSFSPList,
 para cada                      adiciona a
   RepeatedTiming[WavefunctionMathematica1[i - 1, x, prec]][[1]] * 1000];]
    cronometra repetidamente

ListLinePlot[
 gráfico de linha de uma lista de valores
  {WavefunctionWolframSFSPList, FastWaveSFSPList},
  PlotStyle → {Blue, Red},
  estilo do gráfico  azul  vermelho
  Frame → True,
  quadro  verdadeiro
  FrameLabel → {"n", "Runtime [ms]"},
  legenda do quadro
  PlotLabel →
  etiqueta de gráfico
   Row[{Style["Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave(sfsp)", Bold, 13],
    linha  estilo                                                      negrito
     "\n", Style["x: 20.0, to each value of n \n", 12]}],
           estilo
  GridLines → Automatic,
  grade de linhas  automático
  PlotLegends →
  legenda do gráfico
   Placed[{" avg[Wavefunction_Wolfram1] = " <> ToString[SetPrecision[Mean[
   situado                                  converte···  define precisão  média
         WavefunctionWolframSFSPList], 20], TraditionalForm] <> " ms;",
                                              forma tradicional
     " avg[Fast-Wave(sfsp)] = " <> ToString[
                                   converte em cadeia de caracteres
       SetPrecision[Mean[FastWaveSFSPList], 20], TraditionalForm] <> " ms"}, Below],
        define precisão  média                  forma tradicional          abaixo
  ImageSize → Large ,
  tamanho da ···  grande
  PlotRange → {{Automatic, Automatic}, {0, 1.0}}
  intervalo do gráf···  automático  automático
]
```

```
Functionality Test Passed: True
```

Out[37]=

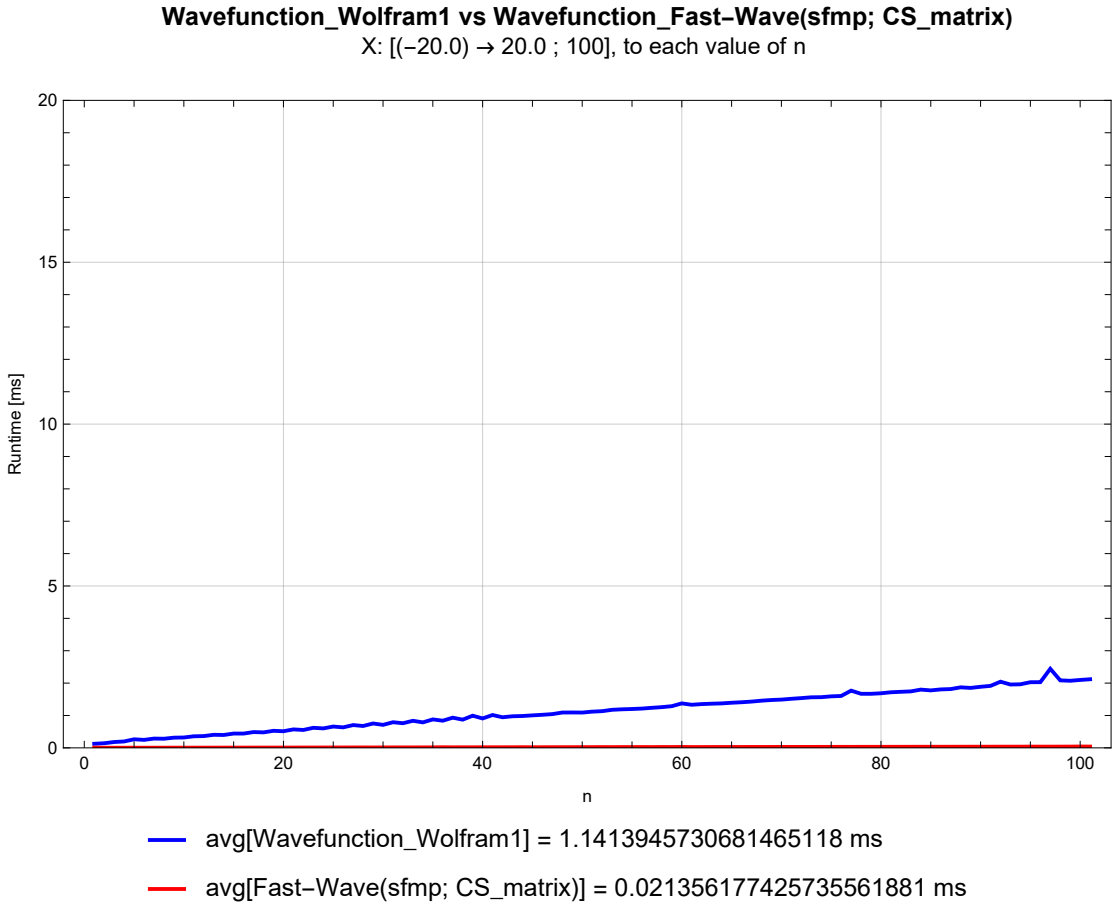

Wavefunction_Wolfram1 vs Wavefunction_Fast−Wave(sfsp)
x: 20.0, to each value of n
Runtime [ms]
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
n
avg[Wavefunction_Wolfram1] = 0.028125324060421175165 ms;
avg[Fast−Wave(sfsp)] = 0.0010389230693047886607 ms

★ Single Fock and Multiple Position speed test with the Normalized Coefficients Matrix

In[38]:=
```
FastWaveSFMPList =
  Normal[ExternalEvaluate["Python",          "import fast_wave.wavefunction_numba
  normal  execução externa
      as wn;import timeit;import numpy as np;N_max = 100;xmax =
      20.0;xmin=-20.0;xsize=100;X=np.linspace(xmin,xmax,xsize);[(timeit.timeit(
      lambda : wn.psi_n_single_fock_multiple_position(n,
      X) , number=10000)/10000)*1000 for n in range(N_max+1)];"]];

WavefunctionWolframSFMPList = {};
For[i = 1, i ≤ (Nmax + 1), i++, AppendTo[WavefunctionWolframSFMPList,
para cada                       adiciona a
   RepeatedTiming[WavefunctionMathematica1[i - 1, Xvector, prec]][[1]] * 1000];]
   cronometra repetidamente
ListLinePlot[
gráfico de linha de uma lista de valores
 {WavefunctionWolframSFMPList, FastWaveSFMPList},
 PlotStyle → {Blue, Red},
 estilo do gráfico  azul  vermelho
 Frame → True,
 quadro  verdadeiro
 FrameLabel → {"n", "Runtime [ms]"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Row[{Style["Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave(sfmp; CS_matrix)", Bold,
  linha  estilo                                                                  negrito
     13], "\n", Style["X: [(-20.0) → 20.0 ; 100], to each value of n \n", 12]}],
                 estilo
 GridLines → Automatic,
 grade de linhas  automático
 PlotLegends → Placed[{" avg[Wavefunction_Wolfram1] = " <> ToString[
 legenda do gráfic  situado                                converte em cadeia de caracteres
      SetPrecision[Mean[WavefunctionWolframSFMPList], 20], TraditionalForm] <> " ms",
      define precisão  média                               forma tradicional
    " avg[Fast-Wave(sfmp; CS_matrix)] = " <> ToString[
                                          converte em cadeia de caracteres
      SetPrecision[Mean[FastWaveSFMPList], 20], TraditionalForm] <> " ms"}, Below],
      define precisão  média                    forma tradicional          abaixo
 ImageSize → Large , PlotRange → {{Automatic, Automatic}, {0, 20.0}}
 tamanho da ···  grande  intervalo do gráf···  automático  automático
]
```

```
Functionality Test Passed: True
```

Out[41]=

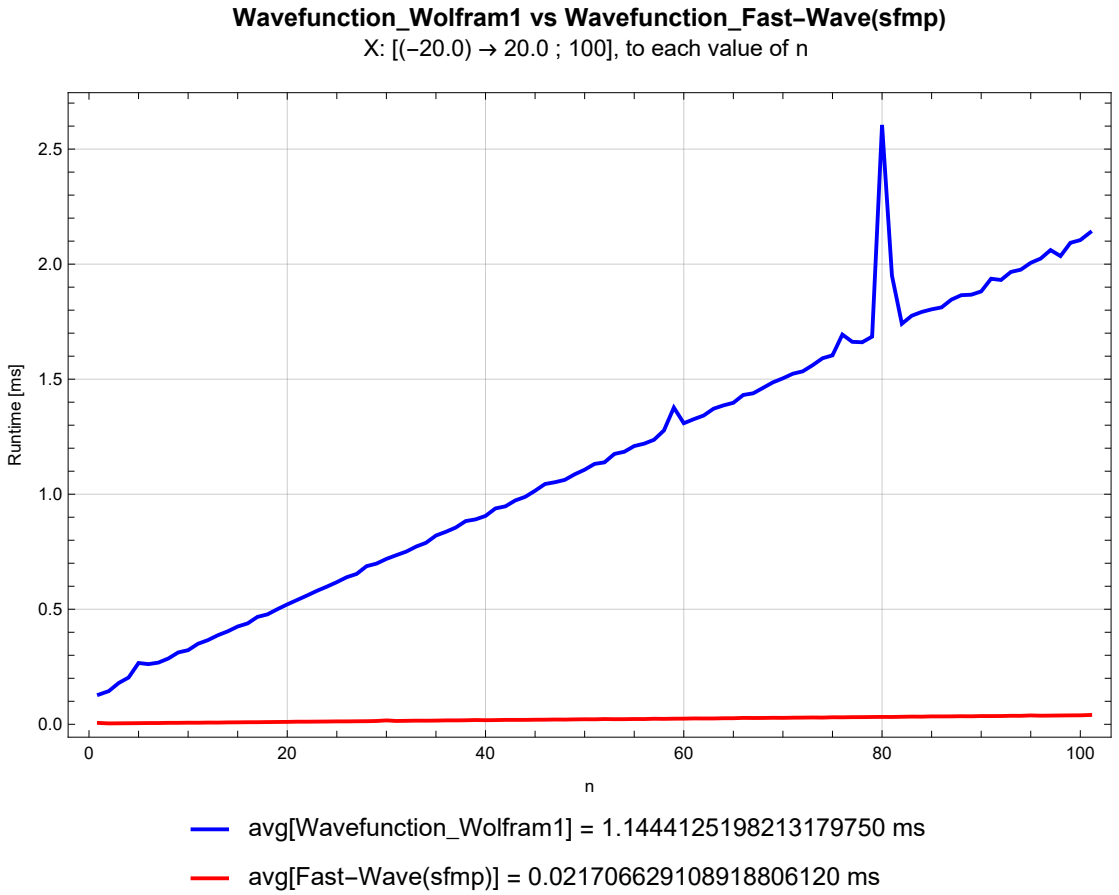

Wavefunction_Wolfram1 vs Wavefunction_Fast−Wave(sfmp; CS_matrix)
X: [(−20.0) → 20.0 ; 100], to each value of n
Runtime [ms]
n
0
5
10
15
20
0
20
40
60
80
100
avg[Wavefunction_Wolfram1] = 1.1413945730681465118 ms
avg[Fast−Wave(sfmp; CS_matrix)] = 0.021356177425735561881 ms

★ Single Fock and Multiple Position speed test

In[42]:=
```
FastWaveSFMPList =
  Normal[ExternalEvaluate["Python",          "import fast_wave.wavefunction_numba
  normal  execução externa
      as wn;import timeit;import numpy as np;N_max = 100;xmax =
      20.0;xmin=-20.0;xsize=100;X=np.linspace(xmin,xmax,xsize);[(timeit.timeit(
      lambda : wn.psi_n_single_fock_multiple_position(n,
      X, CS_matrix = False) , number=10000)/10000)*1000
                  falso
      for n in range(N_max+1)];"]];

WavefunctionWolframSFMPList = {};
For[i = 1, i ≤ (Nmax + 1), i++, AppendTo[WavefunctionWolframSFMPList,
para cada                       adiciona a
   RepeatedTiming[WavefunctionMathematica1[i - 1, Xvector, prec]][[1]] * 1000];]
   cronometra repetidamente


ListLinePlot[
gráfico de linha de uma lista de valores
 {WavefunctionWolframSFMPList, FastWaveSFMPList},
 PlotStyle → {Blue, Red},
 estilo do gráfico  azul   vermelho
 Frame → True,
 quadro   verdadeiro
 FrameLabel → {"n", "Runtime [ms]"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Row[{Style["Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave(sfmp)", Bold, 13],
  linha  estilo                                                          negrito
    "\n", Style["X: [(-20.0) → 20.0 ; 100], to each value of n \n", 12]}],
          estilo
 GridLines → Automatic,
 grade de linhas  automático
 PlotLegends → Placed[{" avg[Wavefunction_Wolfram1] = " <> ToString[
 legenda do gráfic  situado                              converte em cadeia de caracteres
      SetPrecision[Mean[WavefunctionWolframSFMPList], 20], TraditionalForm] <> " ms",
      define precisão  média                              forma tradicional
    " avg[Fast-Wave(sfmp)] = " <> ToString[
                                  converte em cadeia de caracteres
      SetPrecision[Mean[FastWaveSFMPList], 20], TraditionalForm] <> " ms"}, Below],
      define precisão  média                    forma tradicional          abaixo
 ImageSize → Large ]
 tamanho da ···  grande
```

```
Functionality Test Passed: True
```

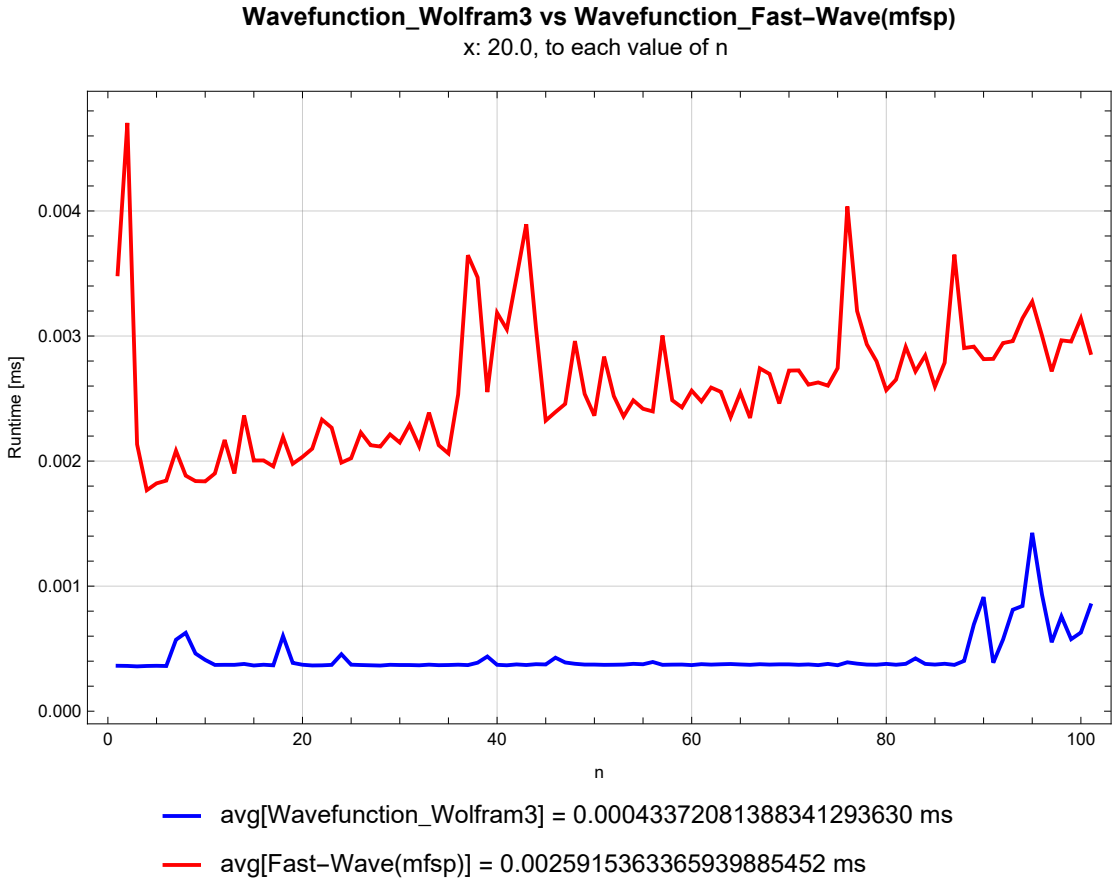
Out[45]=


Wavefunction_Wolfram1 vs Wavefunction_Fast−Wave(sfmp)
X: [(−20.0) → 20.0 ; 100], to each value of n
Runtime [ms]
n
0.0
0.5
1.0
1.5
2.0
2.5
0
20
40
60
80
100
avg[Wavefunction_Wolfram1] = 1.1444125198213179750 ms
avg[Fast−Wave(sfmp)] = 0.021706629108918806120 ms

★ Multiple Fock and Single Position speed test

In[46]:=
```
FastWaveMFSPList = Normal[ExternalEvaluate["Python",
                   normal  execução externa
    "import fast_wave.wavefunction_numba as wn;import timeit;N_max = 100;x =
      20.0;[(timeit.timeit(lambda : wn.psi_n_multiple_fock_single_position(n,
      x) , number=10000)/10000)*1000 for n in range(N_max+1)];"]];

WavefunctionWolframMFSPList = {};
For[i = 1, i ≤ (Nmax + 1), i++, AppendTo[WavefunctionWolframMFSPList,
 para cada                      adiciona a
   RepeatedTiming[WavefunctionMathematica3[i - 1, x, prec]][[1]] * 1000];]
    cronometra repetidamente


ListLinePlot[
 gráfico de linha de uma lista de valores
  {WavefunctionWolframMFSPList, FastWaveMFSPList},
  PlotStyle → {Blue, Red},
  estilo do gráfico  azul   vermelho
  Frame → True,
  quadro   verdadeiro
  FrameLabel → {"n", "Runtime [ms]"},
  legenda do quadro
  PlotLabel →
  etiqueta de gráfico
   Row[{Style["Wavefunction_Wolfram3 vs Wavefunction_Fast-Wave(mfsp)", Bold, 13],
    linha  estilo                                                      negrito
     "\n", Style["x: 20.0, to each value of n \n", 12]}],
           estilo
  GridLines → Automatic,
  grade de linhas  automático
  PlotLegends → Placed[{" avg[Wavefunction_Wolfram3] = " <> ToString[
  legenda do gráficc  situado                              converte em cadeia de caracteres
       SetPrecision[Mean[WavefunctionWolframMFSPList], 20], TraditionalForm] <> " ms",
        define precisão  média                              forma tradicional
     " avg[Fast-Wave(mfsp)] = " <> ToString[
                                   converte em cadeia de caracteres
       SetPrecision[Mean[FastWaveMFSPList], 20], TraditionalForm] <> " ms"}, Below],
        define precisão  média                    forma tradicional          abaixo
  ImageSize → Large ]
  tamanho da ···  grande
Functionality Test Passed: True
```

Out[49]=

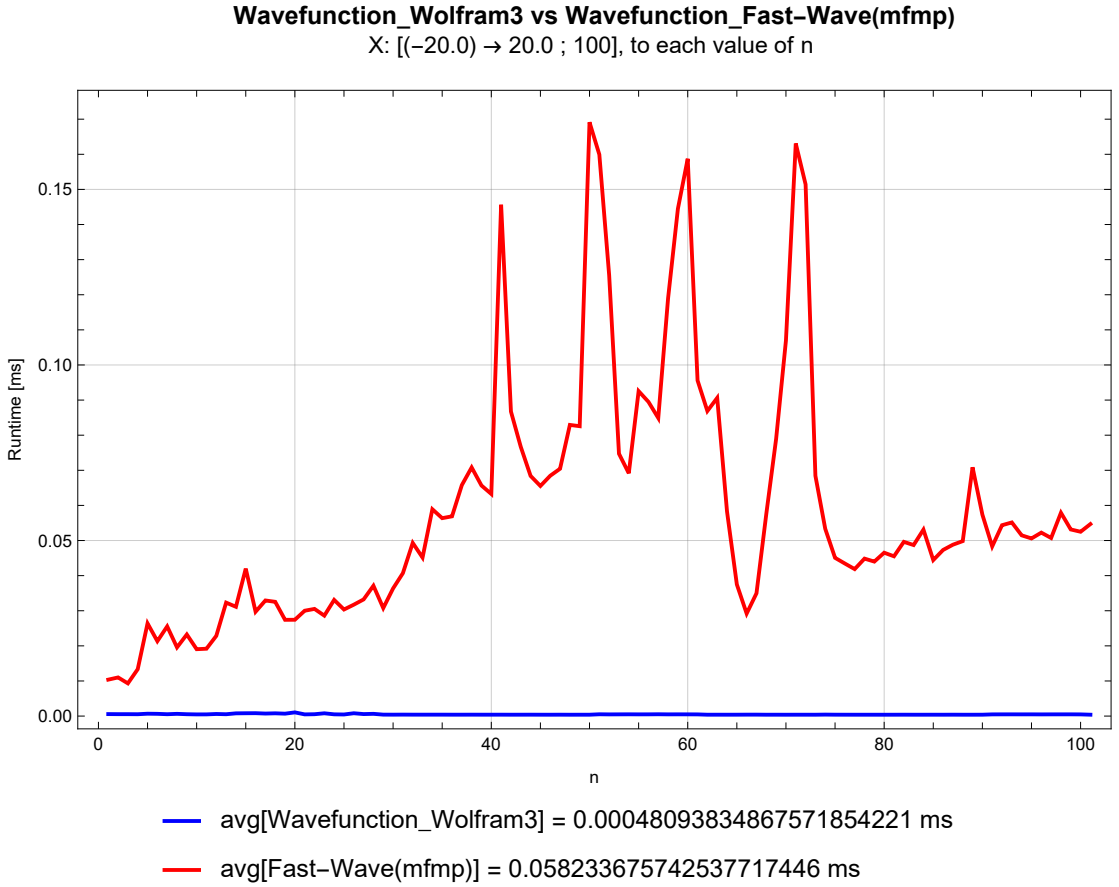

Wavefunction_Wolfram3 vs Wavefunction_Fast−Wave(mfsp)
x: 20.0, to each value of n
Runtime [ms]
0.004
0.003
0.002
0.001
0.000
0
20
40
60
80
100
n
avg[Wavefunction_Wolfram3] = 0.0004337208138834129363 0 ms
avg[Fast−Wave(mfsp)] = 0.0025915363365939885452 ms

★ Multiple Fock and Multiple Position speed test

In[50]:=
```
FastWaveMFMDList =
  Normal[ExternalEvaluate["Python",          "import fast_wave.wavefunction_numba as
  normal  execução externa
      wn; import timeit;import numpy as np;N_max = 100;xmax = 20.0;xmin =
      -20.0;xsize=100;X=np.linspace(xmin,xmax,xsize);[(timeit.timeit(lambda
      : wn.psi_n_multiple_fock_multiple_position(n, X) ,
      number=10000)/10000)*1000 for n in range(N_max+1)];"]];

WavefunctionWolframMFMDList = {};
For[i = 1, i ≤ (Nmax + 1), i++, AppendTo[WavefunctionWolframMFMDList,
para cada                       adiciona a
   RepeatedTiming[WavefunctionMathematica3[i - 1, Xvector, prec]][[1]] * 1000];]
   cronometra repetidamente

ListLinePlot[
gráfico de linha de uma lista de valores
 {WavefunctionWolframMFMDList, FastWaveMFMDList},
 PlotStyle → {Blue, Red},
 estilo do gráfico  azul  vermelho
 Frame → True,
 quadro  verdadeiro
 FrameLabel → {"n", "Runtime [ms]"},
 legenda do quadro
 PlotLabel →
 etiqueta de gráfico
  Row[{Style["Wavefunction_Wolfram3 vs Wavefunction_Fast-Wave(mfmp)", Bold, 13],
  linha  estilo                                                      negrito
    "\n", Style["X: [(-20.0) → 20.0 ; 100], to each value of n \n", 12]}],
          estilo
 GridLines → Automatic,
 grade de linhas  automático
 PlotLegends → Placed[{" avg[Wavefunction_Wolfram3] = " <> ToString[
 legenda do gráficc  situado                              converte em cadeia de caracteres
      SetPrecision[Mean[WavefunctionWolframMFMDList], 20], TraditionalForm] <> " ms",
      define precisão  média                               forma tradicional
    " avg[Fast-Wave(mfmp)] = " <> ToString[
                                  converte em cadeia de caracteres
      SetPrecision[Mean[FastWaveMFMDList], 20], TraditionalForm] <> " ms"}, Below],
      define precisão  média                   forma tradicional          abaixo
 ImageSize → Large ]
 tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[53]=


Wavefunction_Wolfram3 vs Wavefunction_Fast−Wave(mfmp)
X: [(−20.0) → 20.0 ; 100], to each value of n
Runtime [ms]
0.15
0.10
0.05
0.00
0
20
40
60
80
100
n
avg[Wavefunction_Wolfram3] = 0.0004809383486757185422 1 ms
avg[Fast−Wave(mfmp)] = 0.058233675742537717446 ms