Global Variables

In[1]:=
```
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[10]:=
```
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[11]:=
```
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[12]:=
```
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];
```

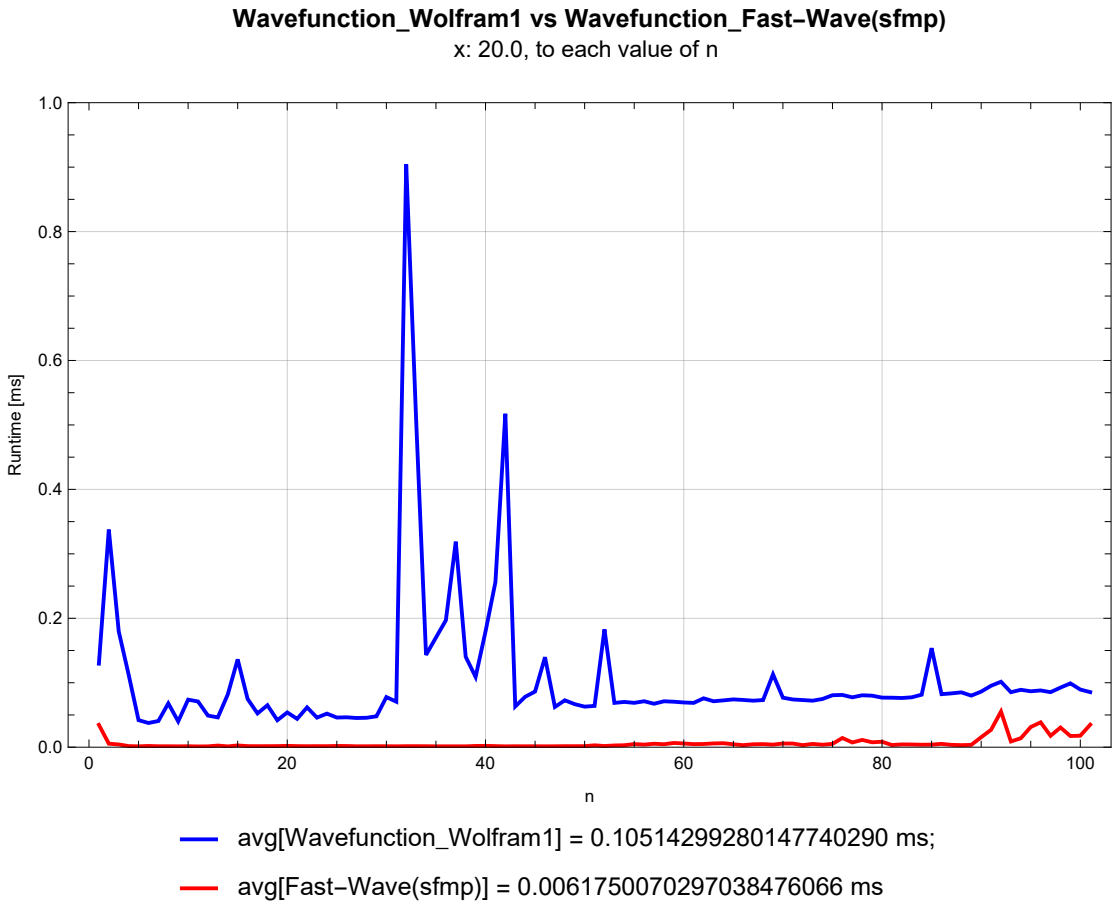
Tests

★ Single-Mode and Onedimensional speed test

In[25]:=
```
(*To use timeit.repeat in the Python is a option too*)

FastWaveSMODList = 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)];"]];

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

ListLinePlot[
gráfico de linha de uma lista de valores
 {WavefunctionWolframSMODList, FastWaveSMODList},
 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, 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
        WavefunctionWolframSMODList], 20], TraditionalForm] <> " ms;",
                                                forma tradicional
    " avg[Fast-Wave(sfmp)] = " <> ToString[
                                  converte em cadeia de caracteres
      SetPrecision[Mean[FastWaveSMODList], 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[28]=

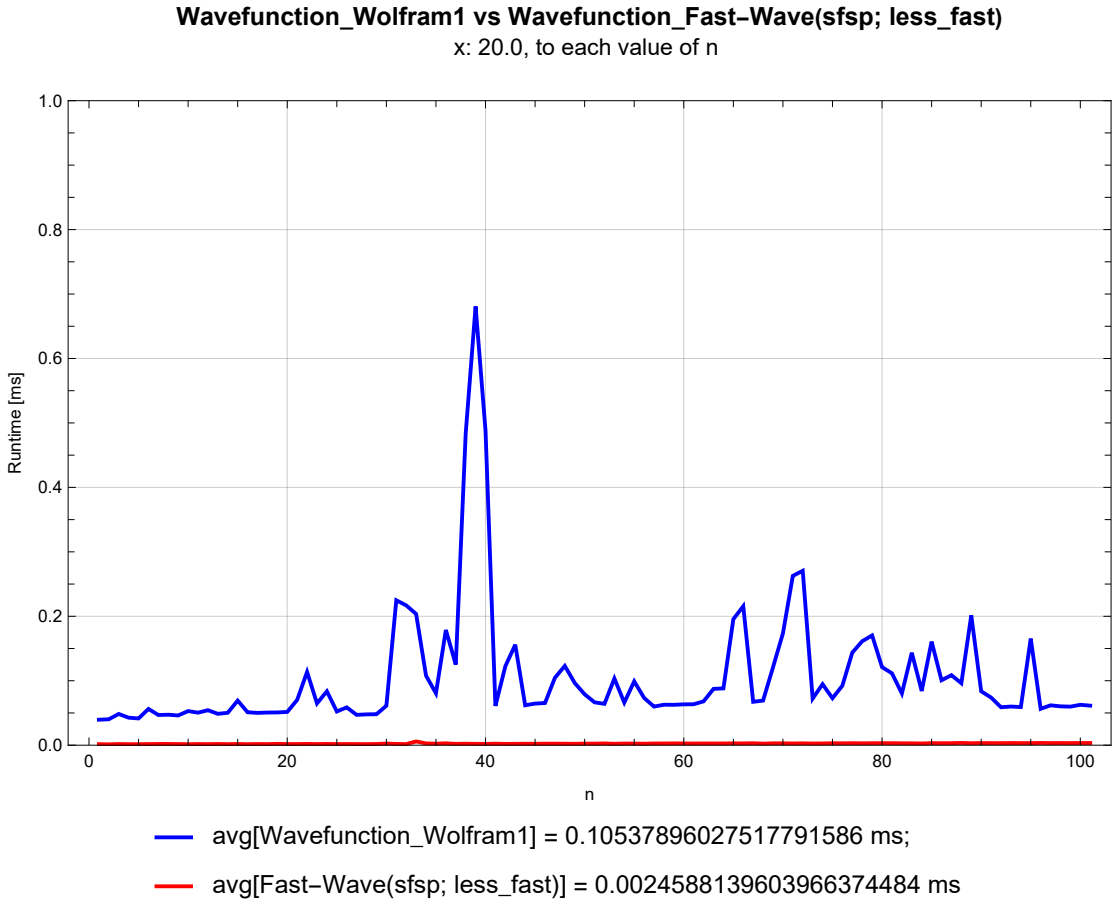

Wavefunction_Wolfram1 vs Wavefunction_Fast−Wave(sfmp)
x: 20.0, to each value of n
Runtime [ms]
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
n
avg[Wavefunction_Wolfram1] = 0.10514299280147740290 ms;
avg[Fast−Wave(sfmp)] = 0.0061750070297038476066 ms

★ Single-Mode and Onedimensional speed test (less_fast)

In[29]:=
```
FastWaveSMODList =
  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, more_fast=False)
                                                                  falso
      , number=10000)/10000)*1000 for n in range(N_max+1)];"]];

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

ListLinePlot[
gráfico de linha de uma lista de valores
 {WavefunctionWolframSMODList, FastWaveSMODList},
 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; less_fast)",
  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
        WavefunctionWolframSMODList], 20], TraditionalForm] <> " ms;",
                                              forma tradicional
    " avg[Fast-Wave(sfsp; less_fast)] = " <> ToString[
                                              converte em cadeia de caracteres
      SetPrecision[Mean[FastWaveSMODList], 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[32]=

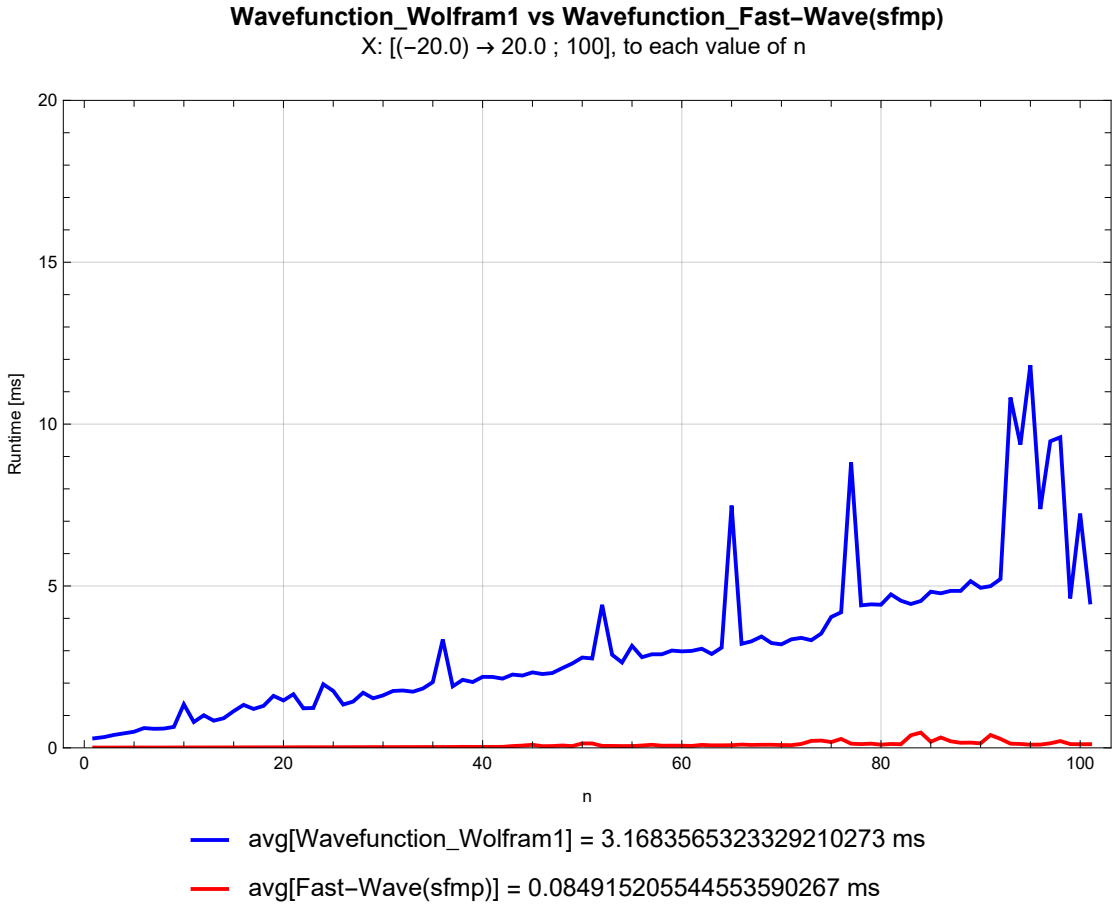

Wavefunction_Wolfram1 vs Wavefunction_Fast−Wave(sfsp; less_fast)
x: 20.0, to each value of n
Runtime [ms]
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
n
avg[Wavefunction_Wolfram1] = 0.10537896027517791586 ms;
avg[Fast−Wave(sfsp; less_fast)] = 0.0024588139603966374484 ms

★ Single-Mode and Multidimensional speed test

In[41]:=
```
FastWaveSMMDList =
  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)];"]];

WavefunctionWolframSMMDList = {};
For[i = 1, i ≤ (Nmax + 1), i++, AppendTo[WavefunctionWolframSMMDList,
para cada                       adiciona a
   RepeatedTiming[WavefunctionMathematica1[i - 1, Xvector, prec]][[1]] * 1000];]
   cronometra repetidamente
ListLinePlot[
gráfico de linha de uma lista de valores
 {WavefunctionWolframSMMDList, FastWaveSMMDList},
 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 gráficc  situado                              converte em cadeia de caracteres
      SetPrecision[Mean[WavefunctionWolframSMMDList], 20], TraditionalForm] <> " ms",
      define precisão  média                               forma tradicional
    " avg[Fast-Wave(sfmp)] = " <> ToString[
                                  converte em cadeia de caracteres
      SetPrecision[Mean[FastWaveSMMDList], 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[44]=

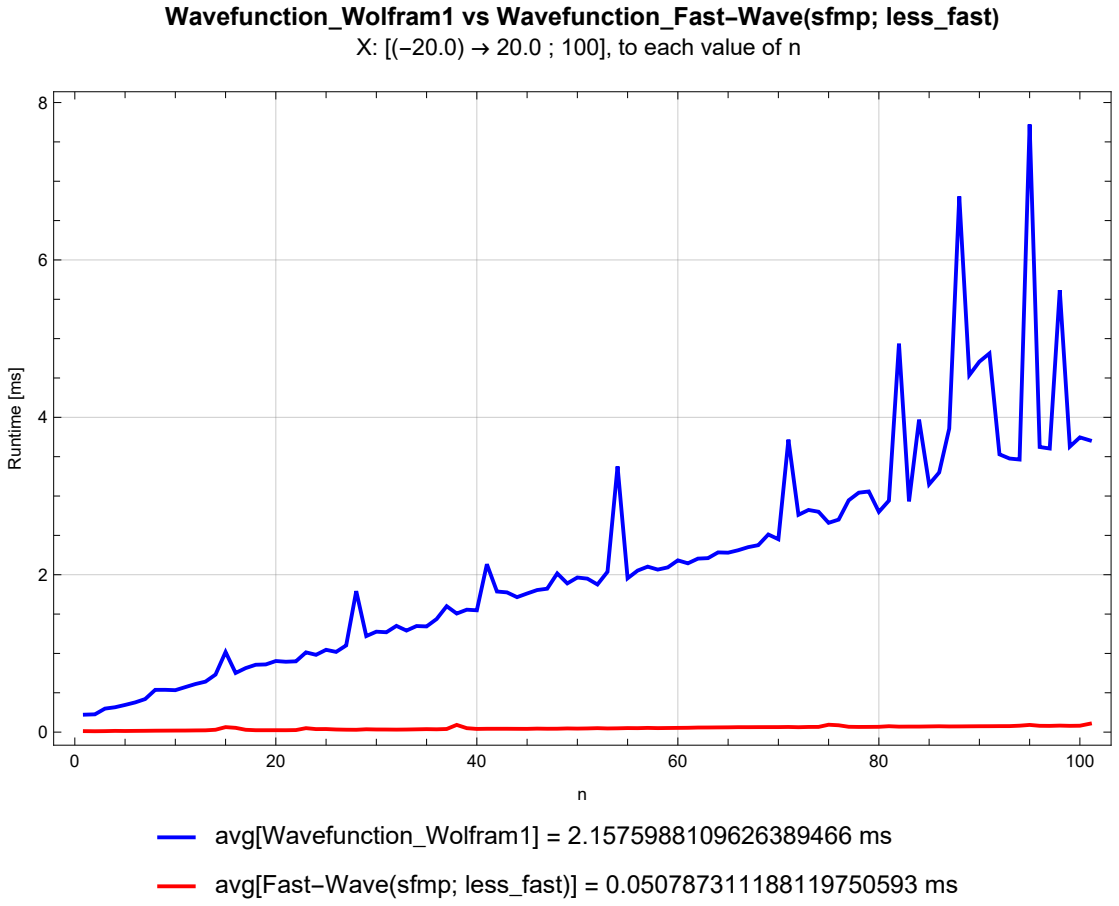

Wavefunction_Wolfram1 vs Wavefunction_Fast−Wave(sfmp)
X: [(−20.0) → 20.0 ; 100], to each value of n
Runtime [ms]
n
0
5
10
15
20
0
20
40
60
80
100
avg[Wavefunction_Wolfram1] = 3.1683565323329210273 ms
avg[Fast−Wave(sfmp)] = 0.084915205544553590267 ms

★ Single-Mode and Multidimensional speed test (less_fast)

In[45]:=
```
FastWaveSMMDList =
  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, more_fast = False) , number=10000)/10000)*1000
                  falso
      for n in range(N_max+1)];"]];

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


ListLinePlot[
gráfico de linha de uma lista de valores
 {WavefunctionWolframSMMDList, FastWaveSMMDList},
 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; less_fast)", 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[WavefunctionWolframSMMDList], 20], TraditionalForm] <> " ms",
      define precisão  média                              forma tradicional
    " avg[Fast-Wave(sfmp; less_fast)] = " <> ToString[
                                          converte em cadeia de caracteres
      SetPrecision[Mean[FastWaveSMMDList], 20], TraditionalForm] <> " ms"}, Below],
      define precisão  média                    forma tradicional          abaixo
 ImageSize → Large ]
 tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[48]=

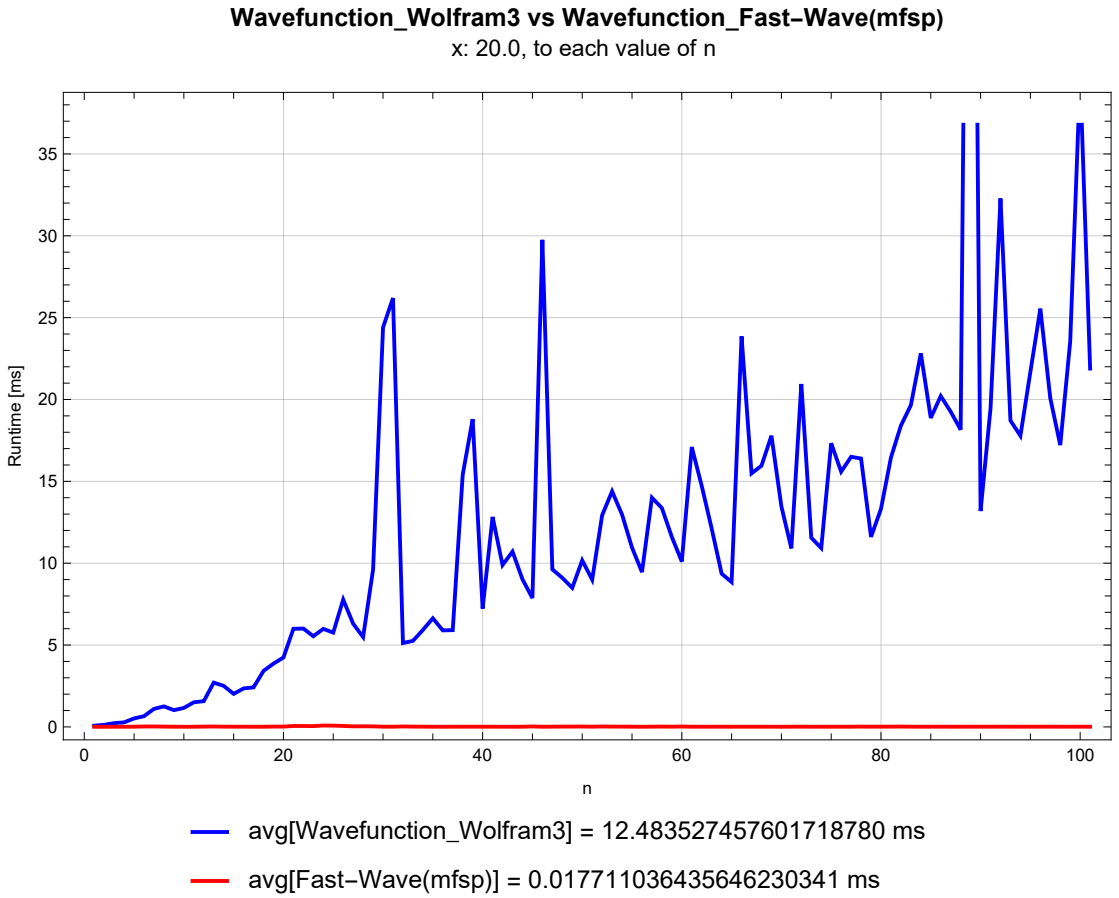

Wavefunction_Wolfram1 vs Wavefunction_Fast−Wave(sfmp; less_fast)
X: [(−20.0) → 20.0 ; 100], to each value of n
Runtime [ms]
n
0
2
4
6
8
0
20
40
60
80
100
avg[Wavefunction_Wolfram1] = 2.1575988109626389466 ms
avg[Fast−Wave(sfmp; less_fast)] = 0.050787311188119750593 ms

★ Multi-Mode and Onedimensional speed test

In[53]:=
```
FastWaveMMODList = 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)];"]];

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


ListLinePlot[
 gráfico de linha de uma lista de valores
  {WavefunctionWolframMMODList, FastWaveMMODList},
  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[WavefunctionWolframMMODList], 20], TraditionalForm] <> " ms",
        define precisão  média                              forma tradicional
     " avg[Fast-Wave(mfsp)] = " <> ToString[
                                   converte em cadeia de caracteres
       SetPrecision[Mean[FastWaveMMODList], 20], TraditionalForm] <> " ms"}, Below],
        define precisão  média                    forma tradicional          abaixo
  ImageSize → Large ]
  tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[56]=

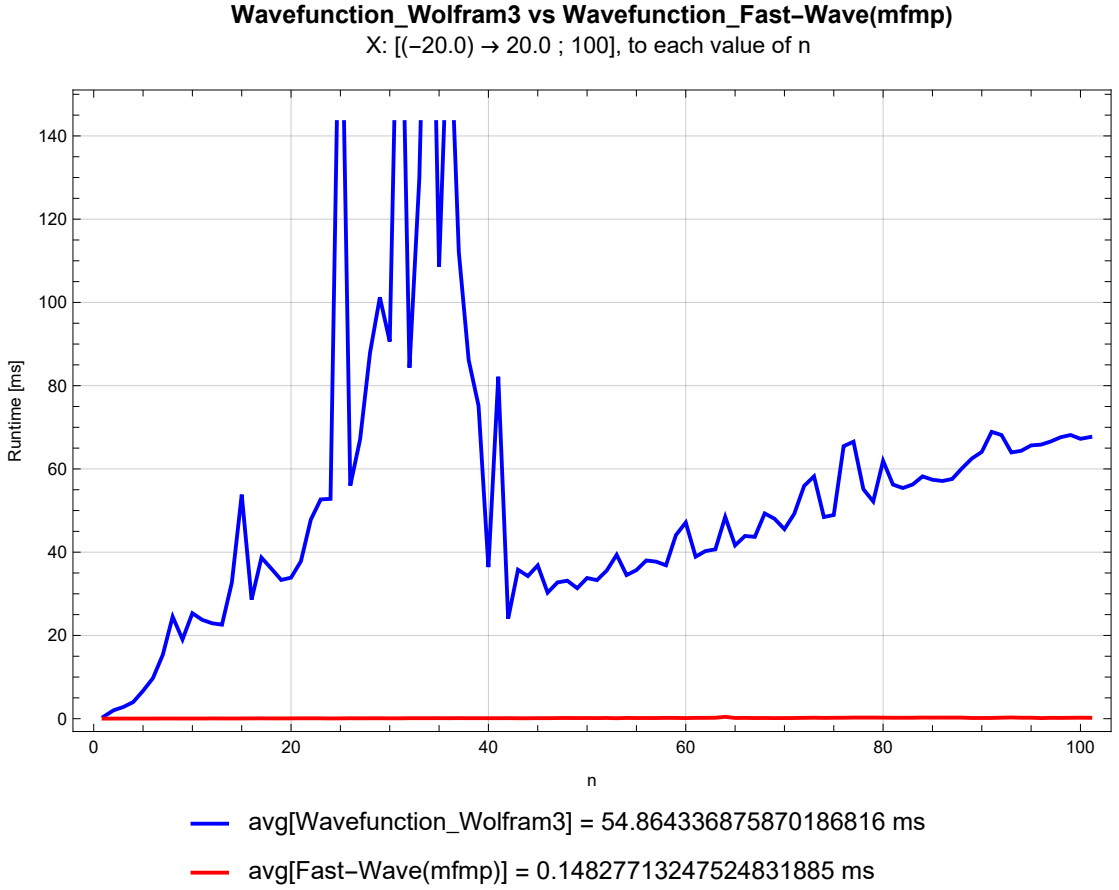

Wavefunction_Wolfram3 vs Wavefunction_Fast–Wave(mfsp)
x: 20.0, to each value of n
Runtime [ms]
n
0
5
10
15
20
25
30
35
0
20
40
60
80
100
avg[Wavefunction_Wolfram3] = 12.483527457601718780 ms
avg[Fast–Wave(mfsp)] = 0.017711036435646230341 ms

★ Multi-Mode and Multidimensional speed test

In[57]:=
```
FastWaveMMMDList =
  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)];"]];

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

ListLinePlot[
 gráfico de linha de uma lista de valores
  {WavefunctionWolframMMMDList, FastWaveMMMDList},
  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[WavefunctionWolframMMMDList], 20], TraditionalForm] <> " ms",
       define precisão  média                               forma tradicional
     " avg[Fast-Wave(mfmp)] = " <> ToString[
                                  converte em cadeia de caracteres
       SetPrecision[Mean[FastWaveMMMDList], 20], TraditionalForm] <> " ms"}, Below],
       define precisão  média                    forma tradicional          abaixo
  ImageSize → Large ]
  tamanho da ···  grande
```

```
Functionality Test Passed: True
```

Out[60]=


Wavefunction_Wolfram3 vs Wavefunction_Fast−Wave(mfmp)
X: [(−20.0) → 20.0 ; 100], to each value of n
Runtime [ms]
n
0
20
40
60
80
100
120
140
avg[Wavefunction_Wolfram3] = 54.864336875870186816 ms
avg[Fast−Wave(mfmp)] = 0.14827713247524831885 ms