

## 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]]];
  wavefunction[[n + 1]]];
```

## Wavefuntion\_Wolfram\_Mathematica\_3

```
In[12]:= 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 speed test

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

```

FastWaveSMODList = Normal[ExternalEvaluate["Python",
    normal execução externa
    "from fast_wave.wavefunction import wavefunction_smod;import timeit;N_max
    = 100;x = 20.0;[(timeit.timeit(lambda : wavefunction_smod(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[WavefunctionMathematical[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(smod)", Bold, 13],
    linha estilo negrito
    "\n", Style["x: 20.0, to each value of n \n", 12]}],
    estilo

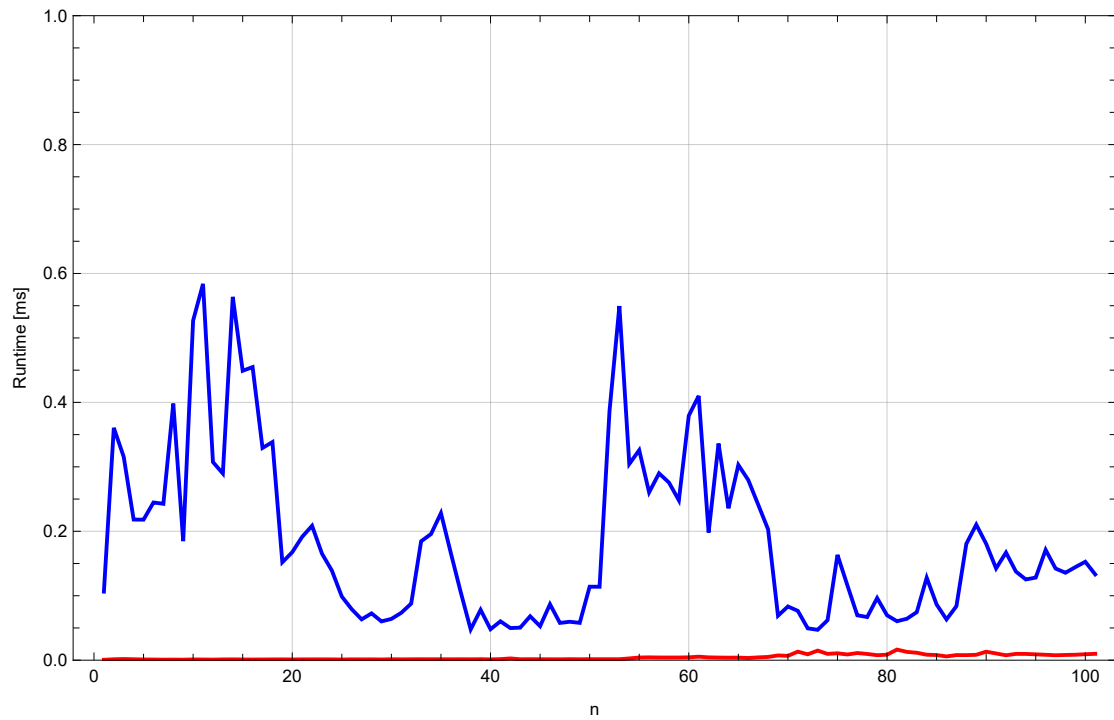
    GridLines → Automatic,
    grade de linhas automático
    PlotLegends →
    legenda do gráfico
    Placed[{" avg[Wavefunction_Wolfram1] = " <> ToString[SetPrecision[Mean[
    situado converte... define precisão média
    WavefunctionWolframSMODList], 20], TraditionalForm] <> " ms;" ,
    forma tradicional
    " avg[Fast-Wave(smod)] = " <> 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[ $\ast$ ]=**Wavefunction\_Wolfram1 vs Wavefunction\_Fast-Wave(smod)**

x: 20.0, to each value of n



— avg[Wavefunction\_Wolfram1] = 0.18305078940816441047 ms;

— avg[Fast-Wave(smod)] = 0.0044505618812169384091 ms

## ★ Single-Mode and Onedimensional speed test (less\_fast)

```

In[25]:= FastWaveSMODList =
  Normal[ExternalEvaluate["Python",
    "from fast_wave.wavefunction
    import wavefunction_smod;import timeit;N_max = 100;x =
    20.0;[(timeit.timeit(lambda : wavefunction_smod(n, x, more_fast=False)
    , number=10000)/10000)*1000 for n in range(N_max+1)];"]];

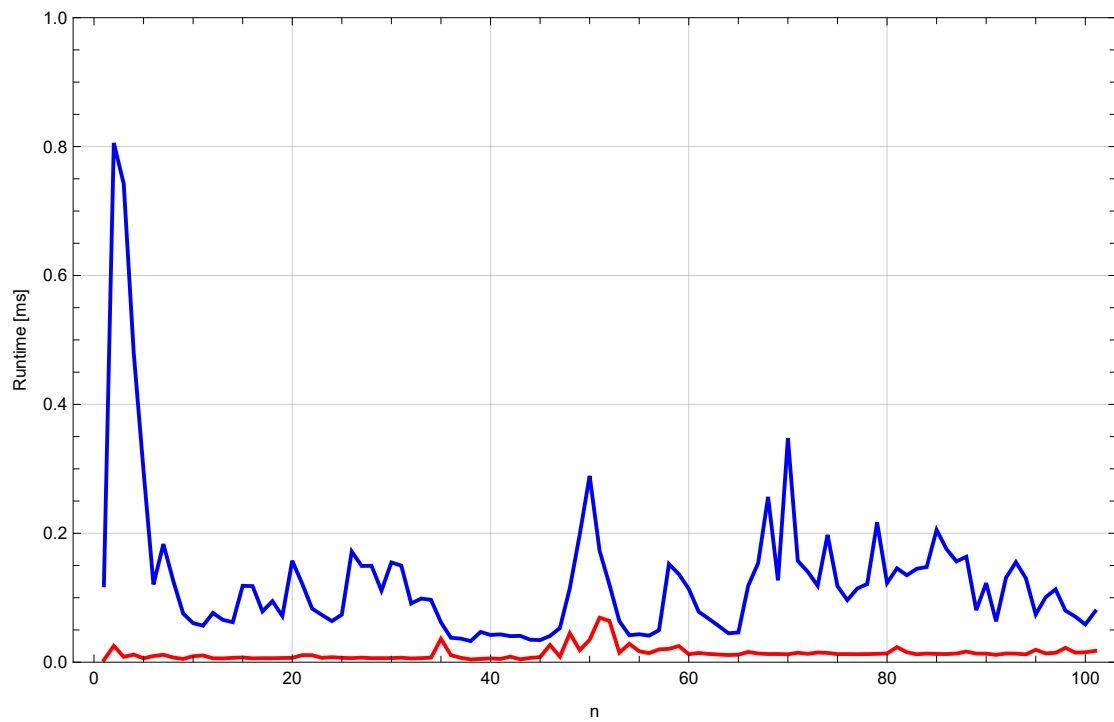
WavefunctionWolframSMODList = {};
For[i = 1, i ≤ (Nmax + 1), i++, AppendTo[WavefunctionWolframSMODList,
  RepeatedTiming[WavefunctionMathematica1[i - 1, x, prec]]][1] * 1000];

ListLinePlot[
  {WavefunctionWolframSMODList, FastWaveSMODList},
  PlotStyle → {Blue, Red},
  Frame → True,
  FrameLabel → {"n", "Runtime [ms]"},
  PlotLabel →
  Row[{Style["Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave(smod; less_fast)",
    Bold, 13], "\n", Style["x: 20.0, to each value of n \n", 12]}],
  GridLines → Automatic,
  Placed[{" avg[Wavefunction_Wolfram1] = " <> ToString[SetPrecision[Mean[
    WavefunctionWolframSMODList], 20], TraditionalForm] <> " ms;" ,
    " avg[Fast-Wave(smod; less_fast)] = " <> ToString[
    SetPrecision[Mean[FastWaveSMODList], 20], TraditionalForm] <> " ms"}, Below],
  ImageSize → Large ,
  PlotRange → {{Automatic, Automatic}, {0, 1.0}}
]
Functionality Test Passed: True

```

Out[28]=

**Wavefunction\_Wolfram1 vs Wavefunction\_Fast-Wave(smod; less\_fast)**  
x: 20.0, to each value of n



— avg[Wavefunction\_Wolfram1] = 0.12689482548779781879 ms;

— avg[Fast-Wave(smod; less\_fast)] = 0.013502909405886588667 ms

## ★ Single-Mode and Multidimensional speed test

```

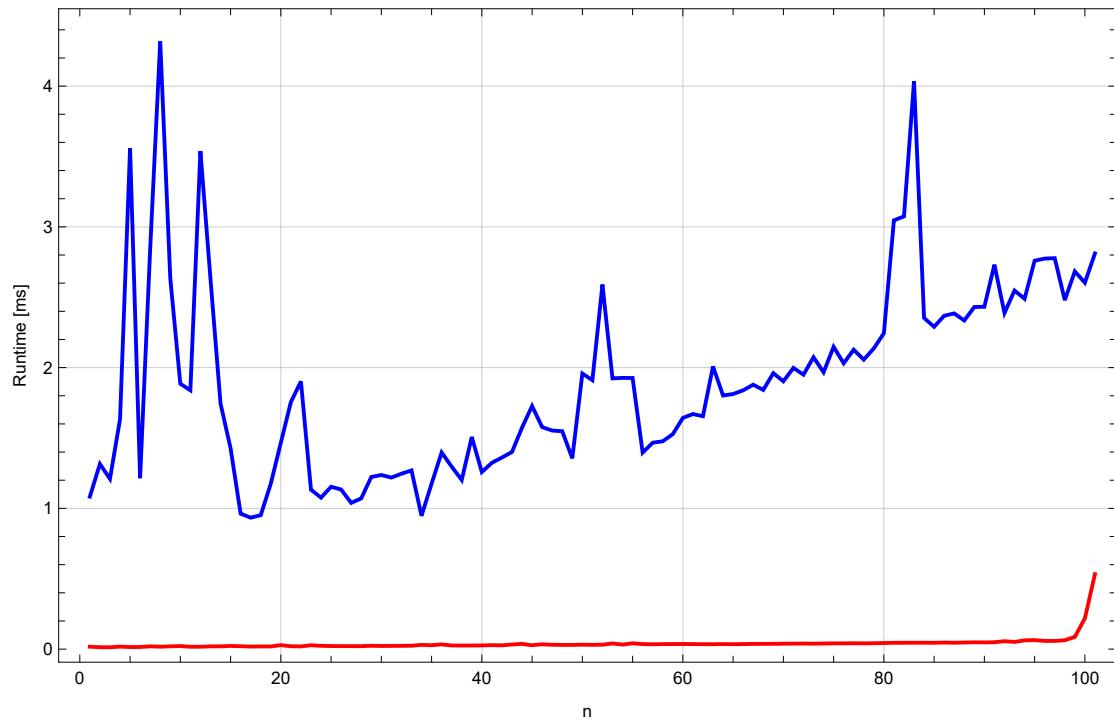
In[33]:= FastWaveSMMList =
  Normal[ExternalEvaluate["Python", "from fast_wave.wavefunction import
    wavefunction_smmd;import timeit;import numpy as np;N_max = 100;xmax =
    20.0;xmin=-20.0;xsize=100;X=np.linspace(xmin,xmax,xsize);[(timeit.timeit(
    lambda : wavefunction_smmd(n, X) ,
    number=10000)/10000)*1000 for n in range(N_max+1)];"]];

WavefunctionWolframSMMList = {};
For[i = 1, i ≤ (Nmax + 1), i++, AppendTo[WavefunctionWolframSMMList,
  RepeatedTiming[WavefunctionMathematical[i - 1, Xvector, prec]] [[1] * 1000];]

ListLinePlot[
  {WavefunctionWolframSMMList, FastWaveSMMList},
  PlotStyle → {Blue, Red},
  Frame → True,
  FrameLabel → {"n", "Runtime [ms]"},
  PlotLabel →
    Row[{Style["Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave(smmd)", Bold, 13],
      "\n", Style["X: [(-20.0) → 20.0 ; 100], to each value of n \n", 12]}],
  GridLines → Automatic,
  PlotLegends → Placed[{" avg[Wavefunction_Wolfram1] = " <> ToString[
    SetPrecision[Mean[WavefunctionWolframSMMList], 20], TraditionalForm] <> " ms",
    " avg[Fast-Wave(smmd)] = " <> ToString[
    SetPrecision[Mean[FastWaveSMMList], 20], TraditionalForm] <> " ms"}, Below],
  ImageSize → Large ]
Functionality Test Passed: True

```

Out[36]=

**Wavefunction\_Wolfram1 vs Wavefunction\_Fast-Wave(smmd)**X:  $[(-20.0) \rightarrow 20.0 ; 100]$ , to each value of n

— avg[Wavefunction\_Wolfram1] = 1.8976655988938737618 ms

— avg[Fast-Wave(smmd)] = 0.040078149207851548885 ms



## ★ Single-Mode and Multidimensional speed test (less\_fast)

```

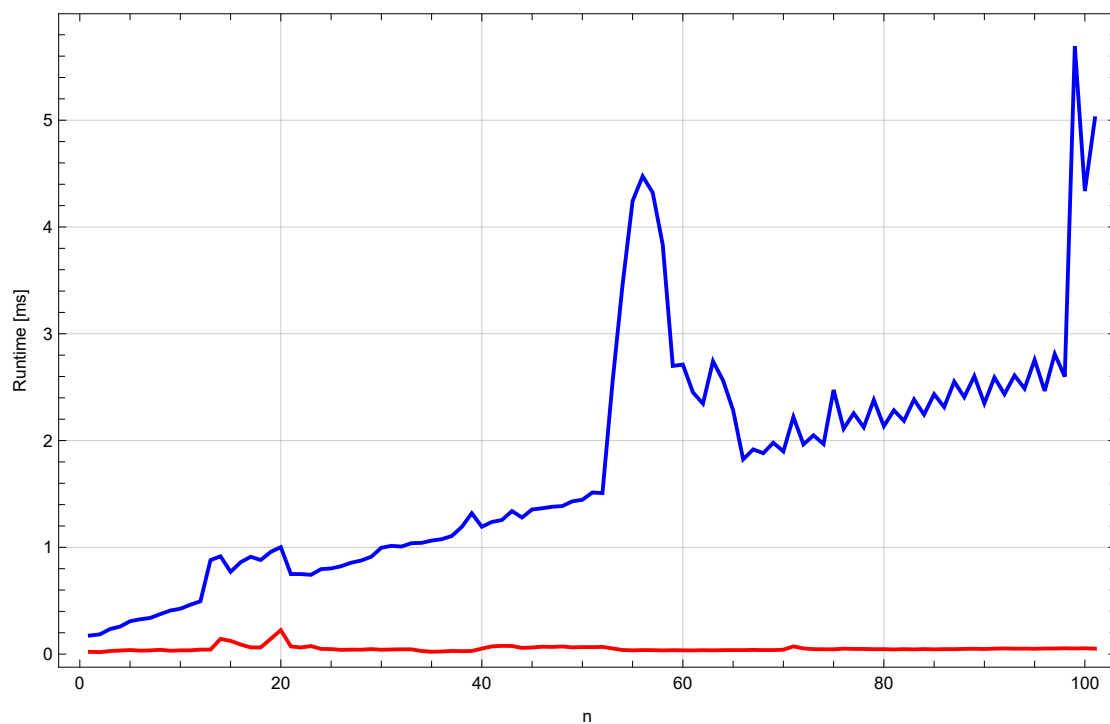
In[49]:= FastWaveSMMList =
  Normal[ExternalEvaluate["Python",
    "from fast_wave.wavefunction import
    wavefunction_smmd;import timeit;import numpy as np;N_max = 100;xmax =
    20.0;xmin=-20.0;xsize=100;X=np.linspace(xmin,xmax,xsize);[(timeit.timeit(
    lambda : wavefunction_smmd(n, X, more_fast = False)
    , number=10000)/10000)*1000 for n in range(N_max+1)];"]];

WavefunctionWolframSMMList = {};
For[i = 1, i ≤ (Nmax + 1), i++, AppendTo[WavefunctionWolframSMMList,
  RepeatedTiming[WavefunctionMathematical[i - 1, Xvector, prec]] [[1] * 1000];]

ListLinePlot[
  {WavefunctionWolframSMMList, FastWaveSMMList},
  PlotStyle → {Blue, Red},
  Frame → True,
  FrameLabel → {"n", "Runtime [ms]"},
  PlotLabel →
  Row[{Style["Wavefunction_Wolfram1 vs Wavefunction_Fast-Wave(smmd; less_fast)", Bold,
    13], "\n", Style["X: [(-20.0) → 20.0 ; 100], to each value of n \n", 12]}],
  GridLines → Automatic,
  PlotLegends → Placed[{" avg[Wavefunction_Wolfram1] = " <> ToString[
    SetPrecision[Mean[WavefunctionWolframSMMList], 20], TraditionalForm] <> " ms",
    " avg[Fast-Wave(smmd; less_fast)] = " <> ToString[
    SetPrecision[Mean[FastWaveSMMList], 20], TraditionalForm] <> " ms"}, Below],
  ImageSize → Large ]
Functionality Test Passed: True

```

Out[52]=

**Wavefunction\_Wolfram1 vs Wavefunction\_Fast-Wave(smmd; less\_fast)**X:  $[(-20.0) \rightarrow 20.0 ; 100]$ , to each value of n

— avg[Wavefunction\_Wolfram1] = 1.7666022448851326221 ms

— avg[Fast-Wave(smmd; less\_fast)] = 0.051437425544533892097 ms

## ★ Multi-Mode and Onedimensional speed test

```

In[*]:= FastWaveMMODList = Normal[ExternalEvaluate["Python",
    [normal [execução externa]
    "from fast_wave.wavefunction import wavefunction_mmod;import timeit;N_max
      = 100;x = 20.0;[(timeit.timeit(lambda : wavefunction_mmod(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(mmod)", 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 gráfico [situado] [converte em cadeia de caracteres]
        SetPrecision[Mean[WavefunctionWolframMMODList], 20], TraditionalForm] <> " ms",
        [define precisão [média] [forma tradicional]
        " avg[Fast-Wave(mmod)] = " <> ToString[
        [converte em cadeia de caracteres]
        SetPrecision[Mean[FastWaveMMODList], 20], TraditionalForm] <> " ms"}, Below],
        [define precisão [média] [forma tradicional] [abaixo]

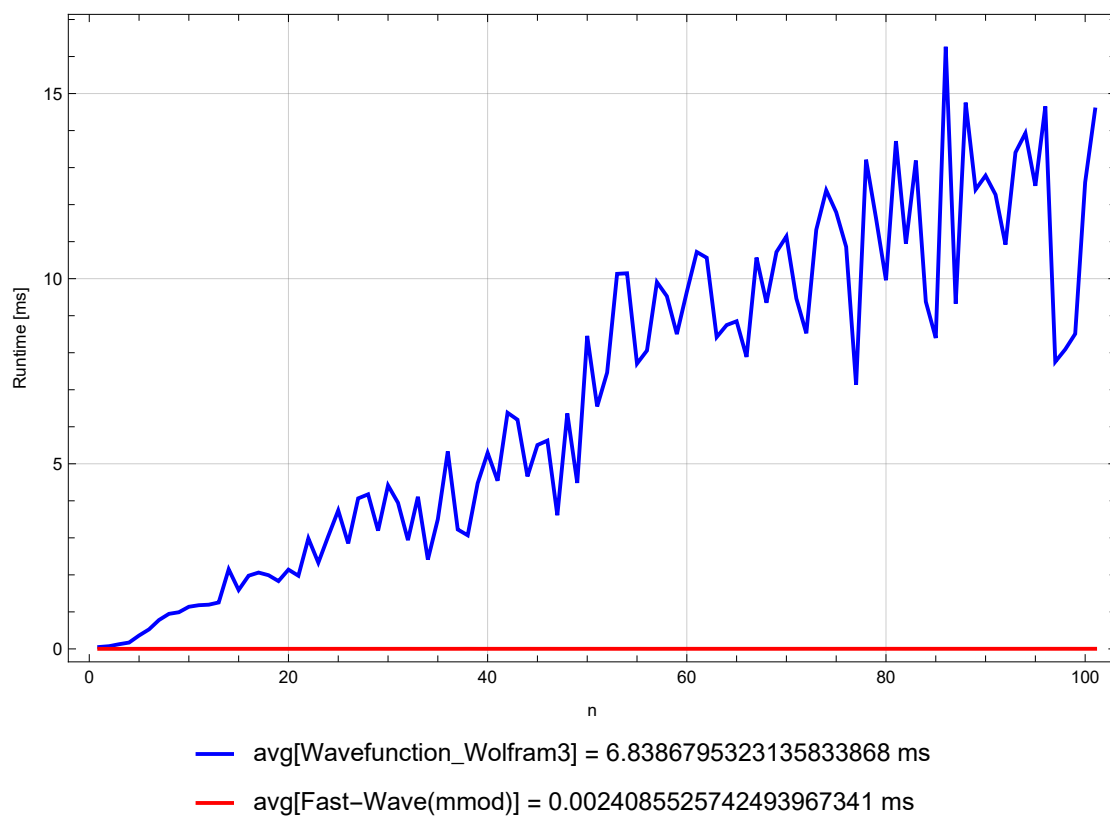
      ImageSize → Large ]
      [tamanho da ... [grande]

Functionality Test Passed: True

```

Out[ ]=

**Wavefunction\_Wolfram3 vs Wavefunction\_Fast-Wave(mmod)**  
x: 20.0, to each value of n



## ★ Multi-Mode and Multidimensional speed test

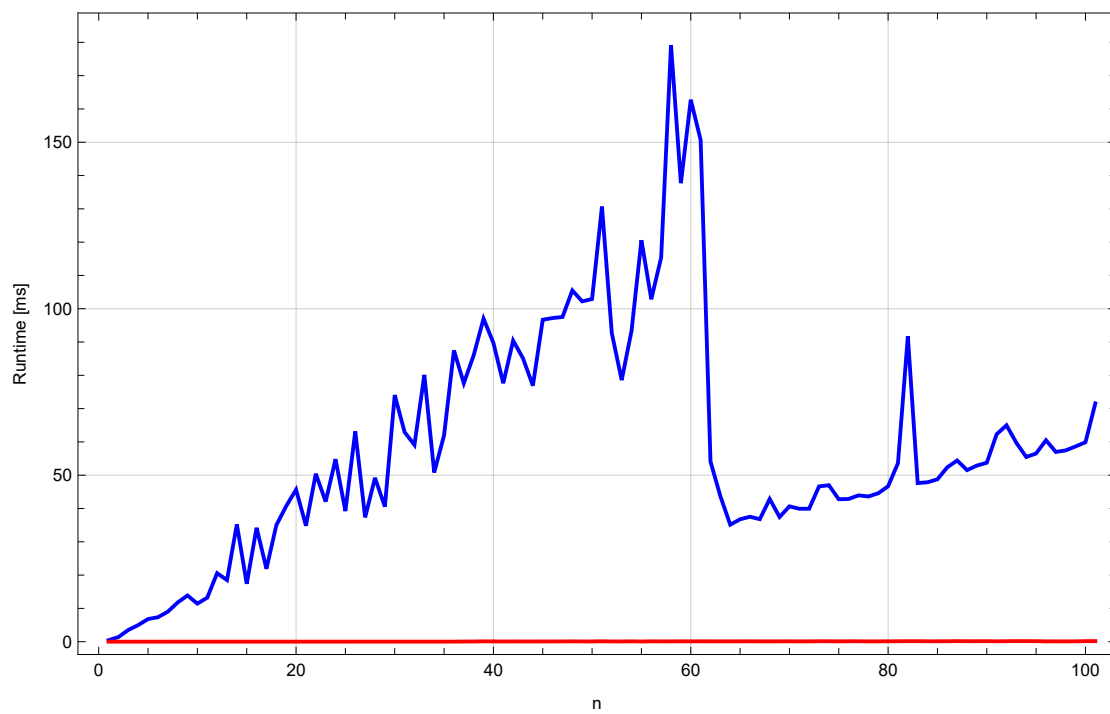
```

In[*]:= FastWaveMMMDList = Normal[ExternalEvaluate["Python",
    normal | execução externa
    "from fast_wave.wavefunction import wavefunction_mmmd;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
    : wavefunction_mmmd(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(mmmd)", 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 gráfico | situado | converte em cadeia de caracteres
        SetPrecision[Mean[WavefunctionWolframMMMDList], 20], TraditionalForm] <> " ms",
        define precisão | média | forma tradicional
        " avg[Fast-Wave(mmmd)] = " <> 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[ $\ast$ ]=**Wavefunction\_Wolfram3 vs Wavefunction\_Fast-Wave(mmmd)**X:  $[(-20.0) \rightarrow 20.0 ; 100]$ , to each value of n

— avg[Wavefunction\_Wolfram3] = 58.511392779238839523 ms

— avg[Fast-Wave(mmmd)] = 0.073597701089112330108 ms