Quick guide to *Mathematica*

To get help, place the cursor on the function and press F1
To run all commands in one cell, place the cursor there and press Shift + Enter or Enter on the numerical keyboard.
Output of many commands is suppressed by ; at the end of the command. If you want to see the output, delete ;
Argument of functions must be in square brackets [...], braces {...} are for arrays, ranges etc., use double brackets [[...]] to access elements of arrays
expr /. {*x* → *a*} means substitute *a* for *x* in expr, expr // function means apply function to expr, i.e. it is equivalent to function[expr].
D[f[x],x] = derivative of *f*[*x*] with respect to *x*.
To define a function of x one can use $f[x\_] := 1 + x^2$, notice the underscore and the colon, or # nad & like in $f := 1 + \#^2$ &
N[expr] evaluates expr with machine precision, if you use the decimal point in the expression it will be also evaluated with machine precision
Special characters can be inserted by pressing Esc ... Esc, e.g. Esc p Esc gives π
Clear[...] is used to unset any variables which could have been assigned previously.
Some other useful commands: Simplify, Expand, Factor; Integrate, Series, Sum

Clear all symbols from previous evaluations to avoid problems

```
Clear["Global`*"]
```

Jacobi diagonalization of real symmetric matrix

```
In[1]:= MyJacobiDiagonalization[A_, niter_] :=
          Module[{EVal, EVec, SumOffDiag, m, θ, t, s, c, τ, i, p, q, r, x, y, nskip},
           m = Length[A];
           EVal = A;
           EVec = IdentityMatrix[m];
           SumOffDiag = ConstantArray[0.0, niter + 1];
           SumOffDiag〚1〛 = Total[Abs[UpperTriangularize[EVal, 1]]^2, 2];
           nskip = 0;
           (* only upper triangular part of the matrix is modified *)
           Do[
            Do[
             Do[
              If[Abs[EVal〚p, q〛] < 10.0^(-16),
               nskip++,
               (* θ = cotg (2ϕ) = (c^2 - s^2)/2sc *)
               θ = (EVal〚q, q〛 - EVal〚p, p〛) / (2 * EVal〚p, q〛);
               (* t = -θ + Sqrt[θ^2 + 1]; *)
               t = 1/(Abs[θ] + Sqrt[θ^2 + 1]); If[θ < 0.0, t = -t];
```

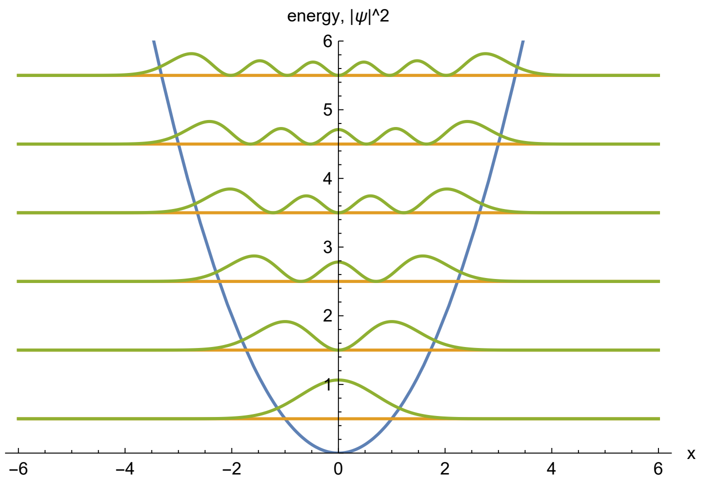
```
        c = 1 / Sqrt[1 + t^2]; s = t * c;
        τ = s / (1 + c);

        (* diagonal elements *)
        EVal⟦p, p⟧ = EVal⟦p, p⟧ - t * EVal⟦p, q⟧;
        EVal⟦q, q⟧ = EVal⟦q, q⟧ + t * EVal⟦p, q⟧;
        EVal⟦p, q⟧ = 0.0;
        Do[ (* elements in columns p and q up to the row p-1 *)
         x = EVal⟦r, p⟧; y = EVal⟦r, q⟧;
         EVal⟦r, p⟧ = x - s * (y + τ * x);
         EVal⟦r, q⟧ = y + s * (x - τ * y),
         {r, 1, p - 1}
        ];
        Do[ (* row p and column q between p and q *)
         x = EVal⟦p, r⟧; y = EVal⟦r, q⟧;
         EVal⟦p, r⟧ = x - s * (y + τ * x);
         EVal⟦r, q⟧ = y + s * (x - τ * y),
         {r, p + 1, q - 1}
        ];
        Do[ (* rows p and q from q *)
         x = EVal⟦p, r⟧; y = EVal⟦q, r⟧;
         EVal⟦p, r⟧ = x - s * (y + τ * x);
         EVal⟦q, r⟧ = y + s * (x - τ * y),
         {r, q + 1, m}
        ];
        (* eigenvectors *)
        Do[
         x = EVec⟦r, p⟧; y = EVec⟦r, q⟧;
         EVec⟦r, p⟧ = x - s * (y + τ * x);
         EVec⟦r, q⟧ = y + s * (x - τ * y),
         {r, 1, m}
        ]
       ],
       {q, p + 1, m}
      ],
      {p, 1, m - 1}
     ];
     SumOffDiag⟦i + 1⟧ = Total[Abs[UpperTriangularize[EVal, 1]]^2, 2];
     (* Print[MatrixForm[EVal]] *),
     {i, 1, niter}
    ];
    Print["Number of skipped elements in Jacobi diagonalization: ", nskip];
    Return[{Diagonal[EVal], EVec, SumOffDiag}]
   ];
```

Application to quantum 1D harmonic oscillator

Exact energies and eigenfunctions of the 1D harmonic oscillator

In[2]:=
```
n = 6;
a = 6;
μ = 1; ω = 1;
```
$$V[x\_] = \frac{1}{2}\,\mu\,\omega^2\,x^2;$$
$$En[i\_] = \omega\left(i + \frac{1}{2}\right);$$
$$\psi[i\_, x\_] = \frac{1}{\sqrt{2^i\, i!}}\,\sqrt[4]{\frac{\mu\,\omega}{\pi}}\;Exp\left[-\frac{\mu\,\omega\,x^2}{2}\right]\,HermiteH[i, x];$$
```
Plot[{V[x], Table[En[i], {i, 0, n - 1}], Table[Abs[ψ[i, x]]^2 + En[i], {i, 0, n - 1}]},
 {x, -a, a}, PlotRange → {0, 6}, AxesLabel → {"x", "energy, |ψ|^2"}]
```

Out[8]=



To find eigenenergies and eigenfunctions of the harmonic oscillator numerically we can use finite differences. If we are interested in a few eigenfunctions with the lowest energies we can safely set boundary conditions

$$\psi(a) = \psi(-a) = 0 \qquad (1)$$

for a certain sufficiently large *a.*

Construction of the Hamiltonian tridiagonal matrix if we approximate the second derivatives by the simplest center difference:

```
In[9]:= n = 40; (* number of points where ψ will be determined (without boundary points) *)
        h = 2 a / (n + 1); (* step in space *)
        X = Range[- a + h, a - h, h];
        H = ConstantArray[0.0, {n, n}];
        (* diagonal elements *)
        Do[
          H⟦i, i⟧ = 1.0 / (μ * h * h) + V[X⟦i⟧],
          {i, 1, n}
         ];
        (* off-diagonal elements *)
        Do[
          H⟦i, i + 1⟧ = -1.0 / (2 * μ * h * h);
          H⟦i + 1, i⟧ = H⟦i, i + 1⟧,
          {i, 1, n - 1}
         ];
        H⟦1 ;; 6, 1 ;; 6⟧ // MatrixForm
```

```
Out[15]//MatrixForm=
 ( 27.960345198   -5.83680555556       0.              0.              0.              0.
  -5.83680555556  26.3327425805   -5.83680555556       0.              0.              0.
       0.        -5.83680555556   24.7908032586   -5.83680555556       0.              0.
       0.              0.        -5.83680555556   23.3345272325   -5.83680555556       0.
       0.              0.              0.        -5.83680555556   21.9639145019   -5.83680555
       0.              0.              0.              0.        -5.83680555556   20.67896506
```

Diagonalization using the Jacobi method:

```
In[16]:= {Energies, Psi, Sums} = MyJacobiDiagonalization[H, 8];
         index = Ordering[Energies];
         Print["Energies:"]
         Energies⟦index⟧
         (* Print["Wave functions:"]
           Psi//MatrixForm *)
         Print["Sums of off-diagonal elements after each sweep:"]
         Sums
```

```
Number of skipped elements in Jacobi diagonalization: 940

Energies:
```

Out[19]=

```
{0.49730845377, 1.48648252848, 2.46467747195, 3.43170080965, 4.38734841773,
 5.33140332401, 6.26363439007, 7.18379509966, 8.09162346781, 8.98684655902,
 9.86919925674, 10.7384783895, 11.5946661552, 12.4381522225, 13.2700271148,
 14.0922957552, 14.9077507503, 15.7193275438, 16.5290938575, 17.337330403,
 18.1420987846, 18.9393276232, 19.7231522163, 20.4861402151, 21.2196325839,
 21.9090642187, 22.5606263332, 23.0322113437, 23.7706876205, 23.850757283,
 25.1438515406, 25.1456648201, 26.8794973069, 26.8795086732, 29.0514753462,
 29.0514753672, 31.8144207467, 31.8144207467, 35.6009863154, 35.6009863154}

Sums of off-diagonal elements after each sweep:
```

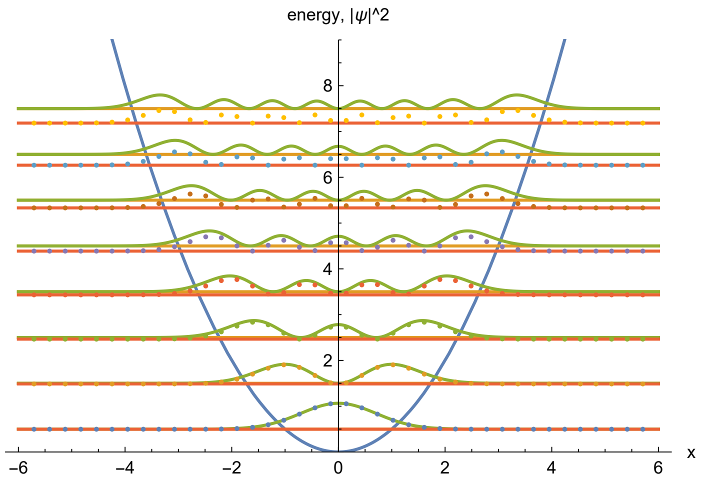
Out[21]=

$\{1328.66366464, 240.31518962, 43.0393955927, 3.54698329901, 0.113918379419, 0.000247264282749, 3.37236467922 \times 10^{-11}, 7.77303571323 \times 10^{-21}, 1.04432533797 \times 10^{-31}\}$

In[22]:=

```
m = 8;
exactPsi = Plot[{V[x], Table[En[i], {i, 0, m - 1}],
     Table[Abs[ψ[i, x]]^2 + En[i], {i, 0, m - 1}], Table[Energies[[index[[i]]]], {i, 1, m}]},
    {x, -a, a}, PlotRange → {0, m + 1}, AxesLabel → {"x", "energy, |ψ|^2"}];
numPsi = ListPlot[
   Table[Table[{X[[j]], Abs[Psi[[j, index[[i]]]] / Sqrt[h]]^2 + Energies[[index[[i]]]]}, {j, 1, n}],
    {i, 1, m}], PlotRange → {0, m + 1}];
Show[exactPsi, numPsi]
```

Out[25]=


energy, |ψ|^2
x
8
6
4
2
−6
−4
−2
0
2
4
6