In[1]:=
```
Clear["Global`*"];
```

Problem

Solve numerically the differential equation (in atomic units $\hbar = 1, m_e = 1$)

$$i \frac{\partial \psi(x,t)}{\partial t} = -\frac{1}{2\mu} \frac{\partial^2 \psi(x,t)}{\partial x^2} \tag{1}$$

with the following initial condition

$$\psi(x,0) = \frac{1}{(2\pi\sigma^2)^{1/4}} e^{-(x-x_0)^2/(4\sigma^2) + i p x} \tag{2}$$

and Dirichlet boundary conditions

$$\begin{aligned} u(-\infty, t) &= 0 \\ u(+\infty, t) &= 0 \end{aligned} \tag{3}$$

Exact solution

Initial normalized Gaussian packet :

In[2]:=
```
psi0[x_, x0_, σ_, p_] = 1 / (2 π σ^2) ^ (1 / 4) Exp[(- (x - x0) ^2 / (4 σ^2) + ⅈ p x)];
Assuming[σ > 0, Integrate[psi0[x, x0, σ, p] * psi0[x, x0, σ, -p], {x, -Infinity, Infinity}]]
```

Out[3]= 1

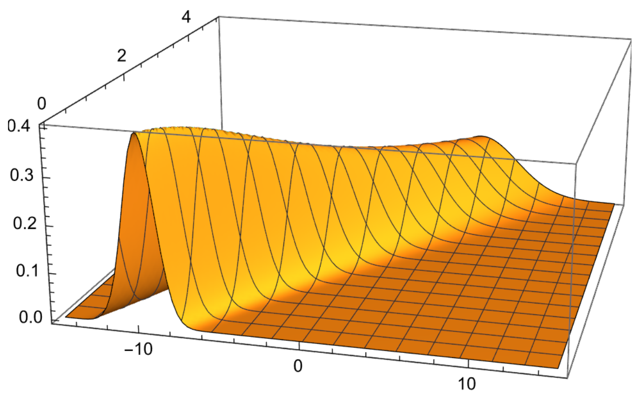
Exact solution:

In[4]:=
```
psiExact[x_, t_, x0_, σ_, p_, μ_] = ⅇ^((-(x-x0)^2 μ-2 ⅈ p^2 t σ^2+p (-2 t x0+4 ⅈ x μ σ^2))/(2 ⅈ t+4 μ σ^2)) (2/π)^(1/4) Sqrt[σ] / (Sqrt[(ⅈ t)/μ + 2 σ^2]);
```

Parameters:

In[5]:=
```
pini = 3; mass = 1; xini = -10; σini = 1;
xmin = -15; xmax = 15; tmax = 5;
Plot3D[Abs[psiExact[x, t, xini, σini, pini, mass]^2],
 {x, xmin, xmax}, {t, 0, tmax}, PlotRange → All, PlotPoints → 100]
```

Out[7]=



Numerical solution in the FEM-DVR basis

Gauss-Lobatto quadrature points and weights will be used for integration over one element

In[8]:=
```
getGaussLobattoPointsAndWeights[n_, a_, b_] :=
 Module[{x, w, p},
  (* roots of the derivative of the (n-1)st Legendre polynomial are inner points
      of the Gauss-Lobatto quadrature on [-1,1]*)
  p[z_] = LegendreP[n - 1, z];
  If[n == 2,
   x = {-1.0, 1.0},
   x = N[Flatten[{-1.0, Sort[Re[z /. N[Solve[D[p[z], z] == 0, z], 20]]], 1.0}]]
  ];
  (* to get weights we need values of this polynomial *)
  w = 2.0 Flatten[{1.0, Table[1.0 / (N[p[x[[i]]], 20])^2, {i, 2, n - 1}], 1.0}] / (n (n - 1));
  (* shifting and scaling to the interval [a,b]*)
  x = (b - a) x / 2 + (b + a) / 2;
  w = (b - a) w / 2;
  Return[{x, w}]
 ]
getGaussLobattoPointsAndWeights[5, 0, 1]
```

Out[9]=
```
{{0., 0.172673164646, 0.5, 0.827326835354, 1.},
 {0.05, 0.272222222222, 0.355555555556, 0.272222222222, 0.05}}
```

Full grid and weights

In[10]:=
```
getFEMPointsAndWeights[nGL_, endPoints_] :=
 Module[{nEl, nPoints, xGL, wGL, x, w},
  nEl = Length[endPoints] - 1;
  nPoints = nEl * (nGL - 1) + 1;
  (* Print["Number of all points/basis functions is ", nPoints]; *)
  x = ConstantArray[0.0, nPoints];
  w = ConstantArray[0.0, nPoints];
  Do[
   {xGL, wGL} = getGaussLobattoPointsAndWeights[nGL, endPoints[[i]], endPoints[[i + 1]]];
   x[[(i - 1) * (nGL - 1) + 1 ;; i * (nGL - 1) + 1]] = xGL;
   (* weights at points which are common to two elements are added up *)
   w[[(i - 1) * (nGL - 1) + 1 ;; i * (nGL - 1) + 1]] += wGL,
   {i, 1, nEl}
  ];
  Return[{x, w}]
 ]
getFEMPointsAndWeights[4, {0, 1, 3, 6}]
```

Out[11]=
```
{{0., 0.27639320225, 0.72360679775, 1., 1.5527864045, 2.4472135955, 3., 3.82917960675,
  5.17082039325, 6.}, {0.0833333333333, 0.416666666667, 0.416666666667,
  0.25, 0.833333333333, 0.833333333333, 0.416666666667, 1.25, 1.25, 0.25}}
```

Derivatives of the Lagrange polynomials at GL points on [-1,1] - result is a matrix nGL x nGL of D[l_i(x), $x = x_k$]

In[12]:=
```
derivativesLagPol[nGL_] :=
  Module[{xGL, wGL, dLP, hlp},
   dLP = ConstantArray[0.0, {nGL, nGL}];
   {xGL, wGL} = getGaussLobattoPointsAndWeights[nGL, -1.0, 1.0];
   Do[
    (* Diagonal terms *)
    dLP[[i, i]] = 0.0;
    Do[
     If[i ≠ s, dLP[[i, i]] = dLP[[i, i]] + 1.0 / (xGL[[i]] - xGL[[s]])],
     {s, 1, nGL}
    ];
    (* Off-diagonal terms *)
    Do[
     hlp = 1.0;
     Do[
      If[(j ≠ i) && (j ≠ k), hlp = hlp * (xGL[[k]] - xGL[[j]]) / (xGL[[i]] - xGL[[j]])],
      {j, 1, nGL}
     ];
     dLP[[i, k]] = hlp / (xGL[[i]] - xGL[[k]]);
     dLP[[k, i]] = 1.0 / (hlp * (xGL[[k]] - xGL[[i]])),
     {k, i + 1, nGL}
    ],
    {i, 1, nGL}
   ];
   Return[dLP];
  ];
derivativesLagPol[4]
```

Out[13]=

$$\{\{-3., -0.809016994375, 0.309016994375, -0.5\}, \{4.04508497187, -3.33066907388\times10^{-16}, -1.11803398875, 1.54508497187\}, \{-1.54508497187, 1.11803398875, 2.22044604925\times10^{-16}, -4.04508497187\}, \{0.5, -0.309016994375, 0.809016994375, 3.\}\}$$

Construction of the stiffness matrix (ϕ_i’, ϕ_j’)

In[14]:=
```
constructStiffnessMatrix[nGL_, endPoints_] :=
 Module[{nEl, nPoints, xFEM, wFEM, xGL, wGL, dLP, dBF, k1, ii, jj, oldCorner, A},
  nEl = Length[endPoints] - 1;
  nPoints = nEl * (nGL - 1) + 1;
  (* get weights for all points *)
  {xFEM, wFEM} = getFEMPointsAndWeights[nGL, endPoints];
  (* calculate derivatives of the
   Lagrange interpolating polynomials at GL points on [-1,1] *)
  dLP = derivativesLagPol[nGL];
  (* build the stiffness matrix *)
  A = ConstantArray[0.0, {nPoints, nPoints}];
  oldCorner = 0.0;
  Do[
   {xGL, wGL} = getGaussLobattoPointsAndWeights[nGL, endPoints〚k〛, endPoints〚k + 1〛];
   (* dilatation of derivatives of LP to be
     the derivatives of the basis functions on the k-th element *)
   dBF = 2.0 * dLP / (endPoints〚k + 1〛 - endPoints〚k〛) ;
   k1 = (k - 1) * (nGL - 1) + 1; (* index of the first point of the k-th element in x *)
   Do[
    (* normalization factor of basis functions *)
    dBF〚i, All〛 = dBF〚i, All〛 / Sqrt[wFEM〚k1 + i - 1〛],
    {i, 1, nGL}
   ];
   Do[
    ii = k1 + i - 1; (* current row in the A matrix *)
    Do[
     jj = k1 + j - 1; (* current column in the A matrix *)
     A〚ii, jj〛 = Sum[wGL〚s〛 * dBF〚i, s〛 * dBF〚j, s〛, {s, 1, nGL}];
     A〚jj, ii〛 = A〚ii, jj〛,
     {j, i, nGL}
    ],
    {i, 1, nGL}
   ];
   A〚k1, k1〛 += oldCorner;
   oldCorner = A〚k1 + nGL - 1, k1 + nGL - 1〛,
   {k, 1, nEl}
  ];
  Return[A]
 ]
constructStiffnessMatrix[2, {0, 1, 2, 3, 4}] // MatrixForm
```

Out[15]//MatrixForm=

2.	-1.41421356237	0.	0.	0.
-1.41421356237	2.	-1.	0.	0.
0.	-1.	2.	-1.	0.
0.	0.	-1.	2.	-1.41421356237
0.	0.	0.	-1.41421356237	2.

Time evolution using the Crank-Nicolson implicit

Parameters of numerical solution:

In[54]:=
```
pini = 3.0; mass = 1.0; xini = -7.0; σini = 1.0;
{xmin, xmax} = {-20.0, 25.0};
{tmin, tmax} = {0.0, 3.0};
(* check that the interval is large enough *)
N[psiExact[xmin, tmin, xini, σini, pini, mass], 20]
N[psiExact[xmax, tmax, xini, σini, pini, mass], 20]
```

Out[57]=

$-2.69363333653\times10^{-19}+8.62071461867\times10^{-20}\,i$

Out[58]=

$-8.89351293386\times10^{-19}+4.57423463869\times10^{-19}\,i$

Set equidistant elements and calculate initial state and Hamiltonian matrix on the FEM-DVR grid:

In[152]:=
```
nGL = 15;
nEl = 40;
endPoints = Table[N[xmin + i * (xmax - xmin) / nEl], {i, 0, nEl}];
{xFEM, wFEM} = getFEMPointsAndWeights[nGL, endPoints];
Nb = Length[xFEM];
Print["Number of points/basis functions: " , Nb - 2];

(* coefficients of the initial wave packet ψ(x) in the FEM basis *)
psiini = N[psi0[xFEM, xini, σini, pini]] * Sqrt[wFEM];
(* stiffness matrix which is used to construct the Hamiltonian matrix *)
A = constructStiffnessMatrix[nGL, endPoints];
(* Hamiltonian matrix *)
H = A[[2 ;; Nb - 1, 2 ;; Nb - 1]] / (2.0 * mass);
```

```
Number of points/basis functions: 559
```

Time evolution:

In[236]:=
```
Print["Number of basis functions (points), number of time steps:"]
{nx, nt} = {Nb - 2, 61}
dt = N[(tmax - tmin) / (nt - 1)]
T = N[Range[tmin, tmax, dt]];

(* Initialization of the array with zeroes - Dirichlet's boundary conditions *)
psi = ConstantArray[0.0, {nx, nt}];
error = ConstantArray[0.0, nt];

(* Initial state *)
psi[[All, 1]] = psiini[[2 ;; Nb - 1]];
```

```
(* generelized Crank-Nicolson method -
 exp(-ⅈ H dt) approximated by a Pade [n/n] approximant *)
nCN = 5;
roots = N[x /. Solve[PadeApproximant[Exp[x], {x, 0, {nCN, nCN}}] == 0, x]];
Do[
  tmppsi = psi[[All, n]];
  Do[
   tmppsi = (IdentityMatrix[nx] + ⅈ * dt * H / roots[[i]]).tmppsi;
   tmppsi = LinearSolve[IdentityMatrix[nx] - ⅈ * dt * H / Conjugate[roots[[i]]], tmppsi],
   {i, 1, nCN}
  ];
  psi[[All, n + 1]] = tmppsi,
  {n, 1, nt - 1}
 ];

(* to get the functional values of the solution at
 grid points we have to multiply the coefficients by Sqrt[w] *)
Do[
  psi[[All, n]] = psi[[All, n]] / Sqrt[wFEM[[2 ;; Nb - 1]]],
  {n, 1, nt}
 ];

(* compare with the exact solution *)
Do[
  error[[n]] = 0.0;
  Do[
   error[[n]] =
    Max[error[[n]], Abs[psi[[j, n]] - psiExact[xFEM[[j + 1]], T[[n]], xini, σini, pini, mass]]],
   {j, 1, nx}
  ],
  {n, 1, nt}
 ];
ListLogPlot[{Table[{T[[n]], error[[n]]}, {n, 1, nt}]},
 PlotRange → All]
Print["Maximal error: ", Max[error]]

(* fancy plotting *)
Manipulate[
 n = Round[t / dt] + 1;
 ListLinePlot[Table[{xFEM[[j + 1]], Re[psi[[j, n]]]}, {j, 1, nx}],
  PlotRange → {-0.5, 1.5}],
 {t, tmin, tmax, dt}
]
```

Number of basis functions (points), number of time steps:

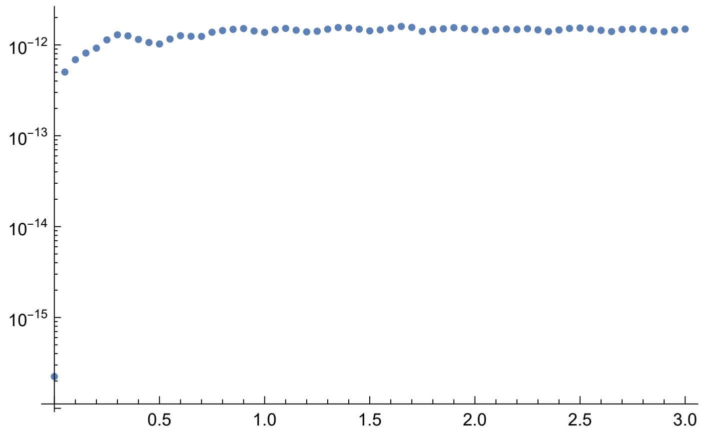
Out[237]=

{559, 61}

Out[238]=

0.05

Out[248]=



Maximal error: $1.5965479698 \times 10^{-12}$

Out[250]=