

```
In[2]:= Clear["Global`*"];
```

Problem

Solve numerically the differential equation

$$\frac{\partial u(x, t)}{\partial t} = \frac{\partial u(x, t)}{\partial x} \quad (1)$$

with the following initial condition

$$u(x, 0) = e^{-x^2} \quad (2)$$

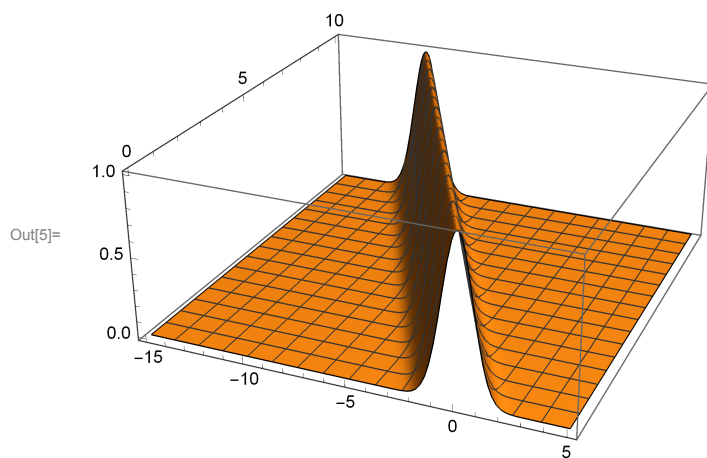
and Dirichlet boundary conditions

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

Exact solution

```
In[3]:= u0[x_, t_] = Exp[-(t + x)^2];  
sol = DSolve[{D[u[x, t], t] == D[u[x, t], x], u[x, 0] == u0[x, 0]}, u, {x, t}]  
Plot3D[u[x, t] /. sol, {x, -15, 5}, {t, 0, 10}, PlotRange -> All, PlotPoints -> 100]
```

```
Out[4]= {{u -> Function[{x, t}, e^{-(t+x)^2}]}}
```



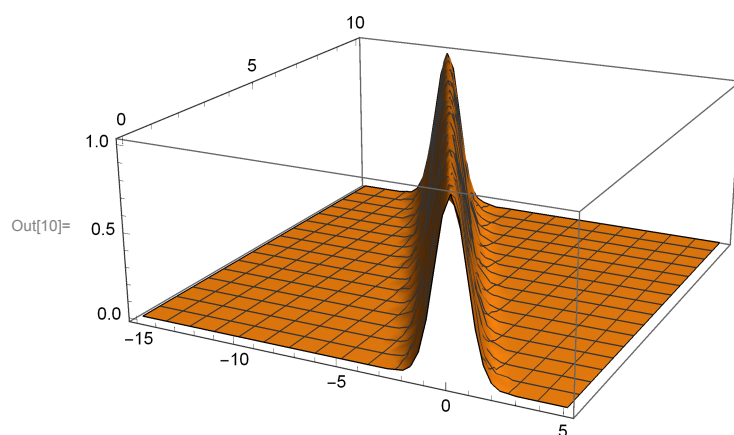
```
In[6]:=
```

Numerical solution using built-in function

```
In[7]:= {xmin, xmax} = {-15, 5};
{tmin, tmax} = {0, 10};
numsol =
  NDSolve[{D[u[x, t], t] == D[u[x, t], x], u[x, 0] == u0[x, 0], u[xmin, t] == u0[xmin, 0],
    u[xmax, t] == u0[xmax, 0]}, u, {x, xmin, xmax}, {t, tmin, tmax}]
Plot3D[Evaluate[u[x, t] /. numsol], {x, xmin, xmax}, {t, tmin, tmax}, PlotRange -> All]
```

Out[9]= $\left\{ \left\{ u \rightarrow \text{InterpolatingFunction} \left[\left\{ \left\{ \begin{array}{c} \text{+} \end{array} \right\} \right\} \right\} \right\} \right\}$

Domain: $\{-15., 5.\}, \{0., 10.\}$
Output: scalar



Numerical solution using basic explicit methods

```
In[273]:= (* grid initialization *)
{nx, nt} = {201, 101};
dx = (xmax - xmin) / (nx - 1);
dt = (tmax - tmin) / (nt - 1);
Print["λ = ", λ = N[dt / dx]]
X = N[Range[xmin, xmax, dx]];
T = N[Range[tmin, tmax, dt]];

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

(* Initial state *)
Do[v[[i, 1]] = N[u0[X[[i]], tmin]], {i, 2, nx - 1}];
Do[v[[i, 2]] = N[u0[X[[i]], tmin + dt]], {i, 2, nx - 1}];
method = 1;
Which[
  method == 1,
  Print["Euler explicit method - order 1, unstable"];
  Do[v[[j, n + 1]] = v[[j, n]] + λ / 2 (v[[j + 1, n]] - v[[j - 1, n]]),
    {n, 1, nt - 1}, {j, 2, nx - 1}],
```

```

method == 2,
Print["Upwind - order 1, stable  $\lambda \leq 1$ "];
Do[v[[j, n+1]] = v[[j, n]] +  $\lambda$  (v[[j+1, n]] - v[[j, n]]),
  {n, 1, nt-1}, {j, 2, nx-1}],
method == 3,
Print["Lax-Friedrichs method - order 1, stable  $\lambda \leq 1$ "];
Do[v[[j, n+1]] = 0.5 (v[[j+1, n]] + v[[j-1, n]]) +  $\lambda/2$  (v[[j+1, n]] - v[[j-1, n]]),
  {n, 1, nt-1}, {j, 2, nx-1}],
method == 4,
Print["Leap-frog method - order 2, stable  $\lambda < 1$ "];
Do[v[[j, n+1]] = v[[j, n-1]] +  $\lambda$  (v[[j+1, n]] - v[[j-1, n]]),
  {n, 2, nt-1}, {j, 2, nx-1}],
method == 5,
Print["Leap-frog method 4th order in space- order 2, stable  $\lambda < 0.728\dots$ "];
Do[v[[j, n+1]] = v[[j, n-1]] + 4 $\lambda$  (v[[j+1, n]] - v[[j-1, n]])/3 -
   $\lambda$  (v[[j+2, n]] - v[[j-2, n]])/6, {n, 2, nt-1}, {j, 3, nx-2}],
method == 6,
Print["Lax-Wendroff method - order 2, stable  $\lambda \leq 1$ "];
Do[v[[j, n+1]] = v[[j, n]] +  $\lambda/2$  (v[[j+1, n]] - v[[j-1, n]]) +
   $\lambda^2/2$  (v[[j+1, n]] - 2v[[j, n]] + v[[j-1, n]]), {n, 1, nt-1}, {j, 2, nx-1}]
];
Do[
  error[[n]] = 0.0;
  Do[
    error[[n]] = Max[error[[n]], Abs[v[[j, n]] - N[u0[X[[j]], T[[n]]]]],
    {j, 2, nx-1}
  ],
  {n, 1, nt-1}
]

(* fancy plotting *)
Manipulate[
  n = Round[t/dt] + 1;
  ListLinePlot[{Table[{X[[j]], u0[X[[j]], T[[n]]}], {j, 1, nx}},
    Table[{X[[j]], v[[j, n]]}], {j, 1, nx}],
  PlotRange -> {-0.5, 1.5}],
{t, tmin, tmax, dt}]

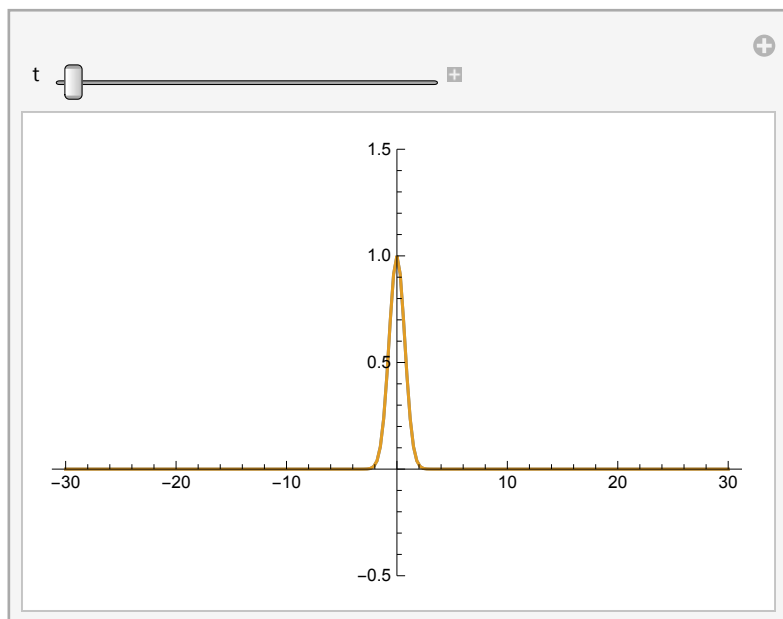
(* show maximal errors *)
ListLogPlot[{Table[{T[[n]], error[[n]]}], {n, 1, nt}},
  PlotRange -> All]
Print["Maximal error: ", Max[error]]

 $\lambda = 1.$ 

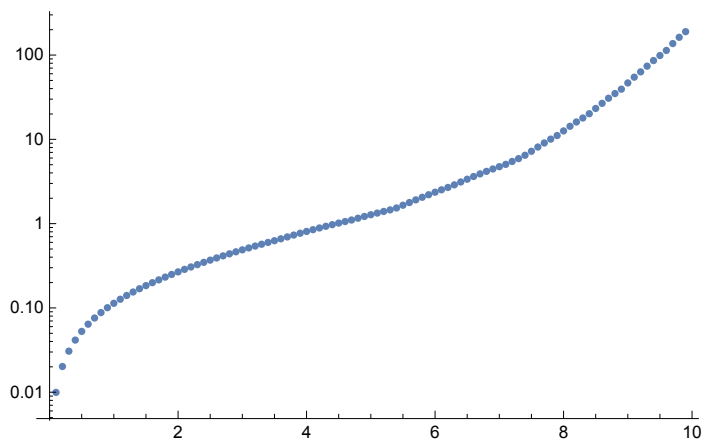
Euler explicit method - order 1, unstable

```

Out[286]=



Out[287]=



Maximal error: 189.4994658