The One Dimensional Heat Equation

```
> restart; with(plots):
Warning, the name changecoords has been redefined
```

The one dimensional heat conduction

If u(x,t) is the temperature at position x and time t the one dimensional **heat equation** is given by:

```
> heat := diff(u(x,t),t)=diff(u(x,t),x,x);
```

$$heat := \frac{\partial}{\partial t} u(x,t) = \frac{\partial^2}{\partial x^2} u(x,t) \quad \textbf{(1.1)}$$

In the lectures, we have found the particular solution

```
> u_part := (x,t) -> exp( -(k*Pi)^2 *t) * sin(k*Pi*x);
```

$$u\_part := (x,t) \rightarrow e^{-k^2 \pi^2 t} \sin(k \pi x) \quad \textbf{(1.2)}$$

If k is an integer,

```
> k := 1;
```

$$k := 1 \quad \textbf{(1.3)}$$

we can see that u_part satifies the heat equation

```
> subs(u(x,t)=u_part(x,t),heat);
> simplify(%);
```

$$\frac{\partial}{\partial t}\left(e^{-\pi^2 t}\sin(\pi x)\right) = \frac{\partial^2}{\partial x^2}\left(e^{-\pi^2 t}\sin(\pi x)\right) \quad \textbf{(1.4)}$$

$$-\pi^2 e^{-\pi^2 t}\sin(\pi x) = -\pi^2 e^{-\pi^2 t}\sin(\pi x)$$

and the following boundary conditions:

```
> u_0 := u_part(0,t); u_1 := u_part(1,t);
```

$$u\_0 := 0 \quad \textbf{(1.5)}$$

$$u\_1 := 0$$

and initial conditions:

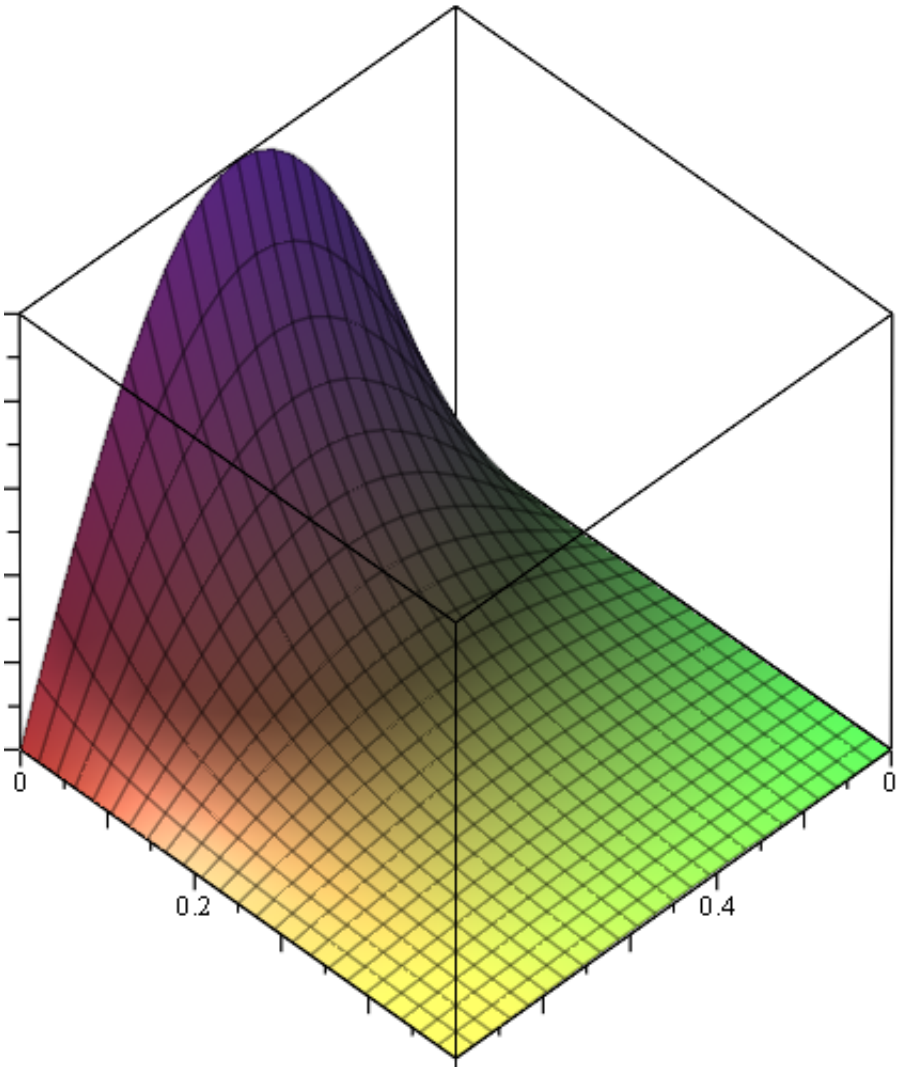
```
> u_part(x,0);
> u_init := unapply( eval(u_part(x,0)), x);
```

$$\sin(\pi x) \quad \textbf{(1.6)}$$

$$u\_init := x \rightarrow \sin(\pi x)$$

And here's a 3D-plot of the solution

```
> plot3d(u_part,0..1,0..0.5,axes=boxed);
```



Numerical solution

Now, we will use the explicite time stepping scheme derived in the lectures to approximately compute a solution.

First, we define the number of unknowns, the mesh size, and the size and number of the time steps:

```
> n := 20; h := 1/n;
```

$$n := 20 \tag{2.1}$$

$$h := \frac{1}{20}$$

```
> tau := 1/800; steps := 100;
```

$$\tau := \frac{1}{800} \tag{2.2}$$

$$steps := 100$$

```
> v := array(0..n,0..steps);
```

$$v := array(0..20, 0..100, [\ ]) \qquad \textbf{(2.3)}$$

Then, we initialize our numerical solution:

```
> for j from 1 to n-1 do
     v[j,0] := u_init(j*h)
  end do:
```

and set the boundary conditions:

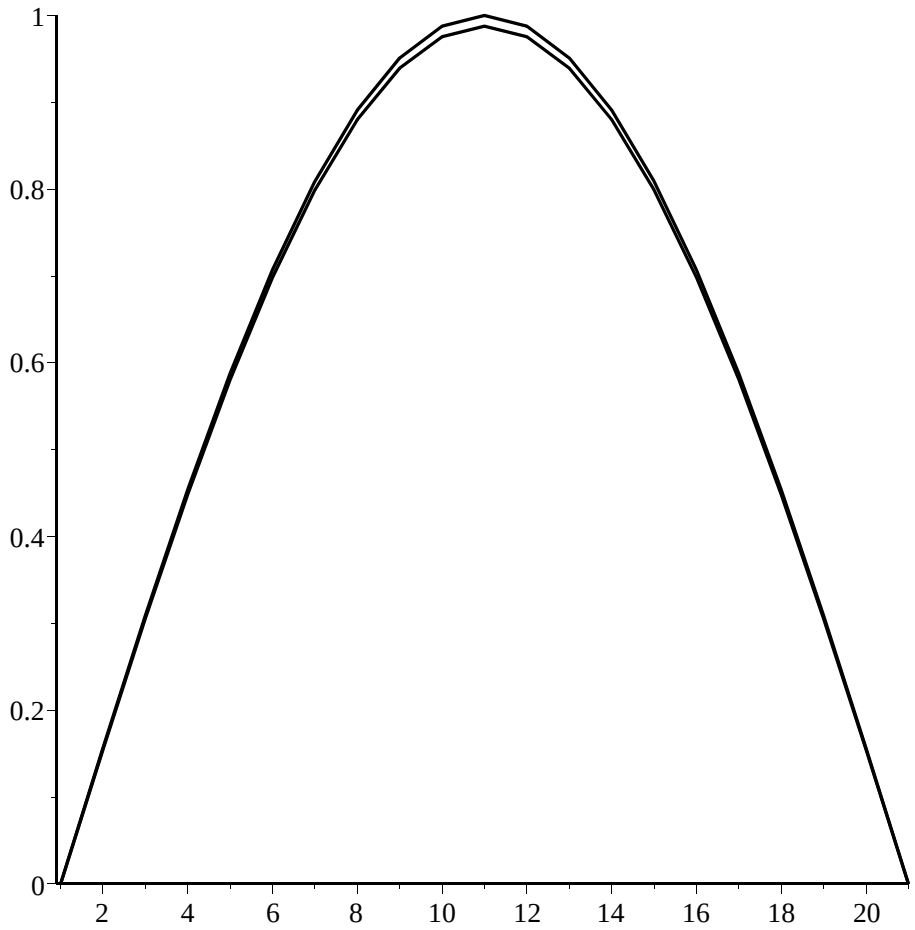
```
> for m from 0 to steps do
     v[0,m] := 0;
     v[n,m] := 0;
  end do:
```

To compute the numerical solution, we can then use the loop

```
> for j from 1 to n-1 do
     v[j,1] := v[j,0] + tau/h^2 * ( v[j-1,0] - 2*v[j,0] + v
  [j+1,0] )
  end do:
```

We may take a little time to plot the results:

```
> plot1 := listplot( [seq( v[j,0], j=0..n)] ):
  plot2 := listplot( [seq( v[j,1], j=0..n)] ):
  display(plot1,plot2);
```

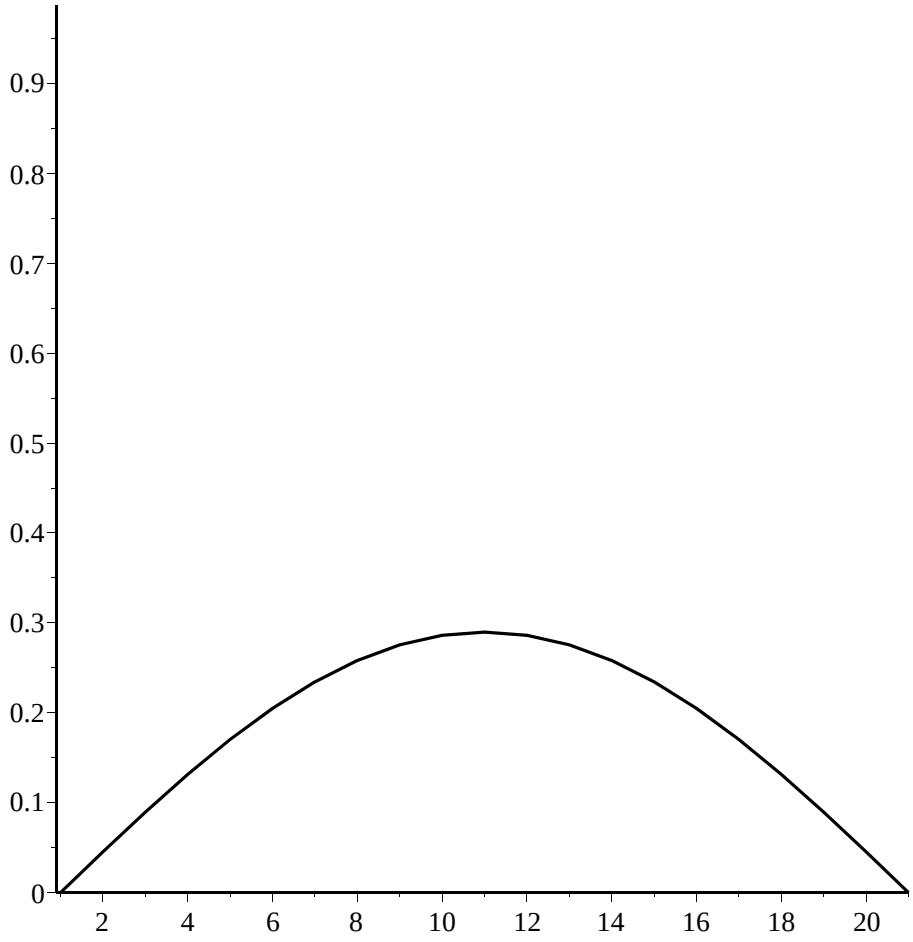


To compute the solution for all time steps, we use the following loop:

```
> for m from 0 to steps-1 do
     for j from 1 to n-1 do
        v[j,m+1] := v[j,m] + tau/h^2 * ( v[j-1,m] - 2*v[j,m] +
  v[j+1,m] )
     end do
  end do:
```

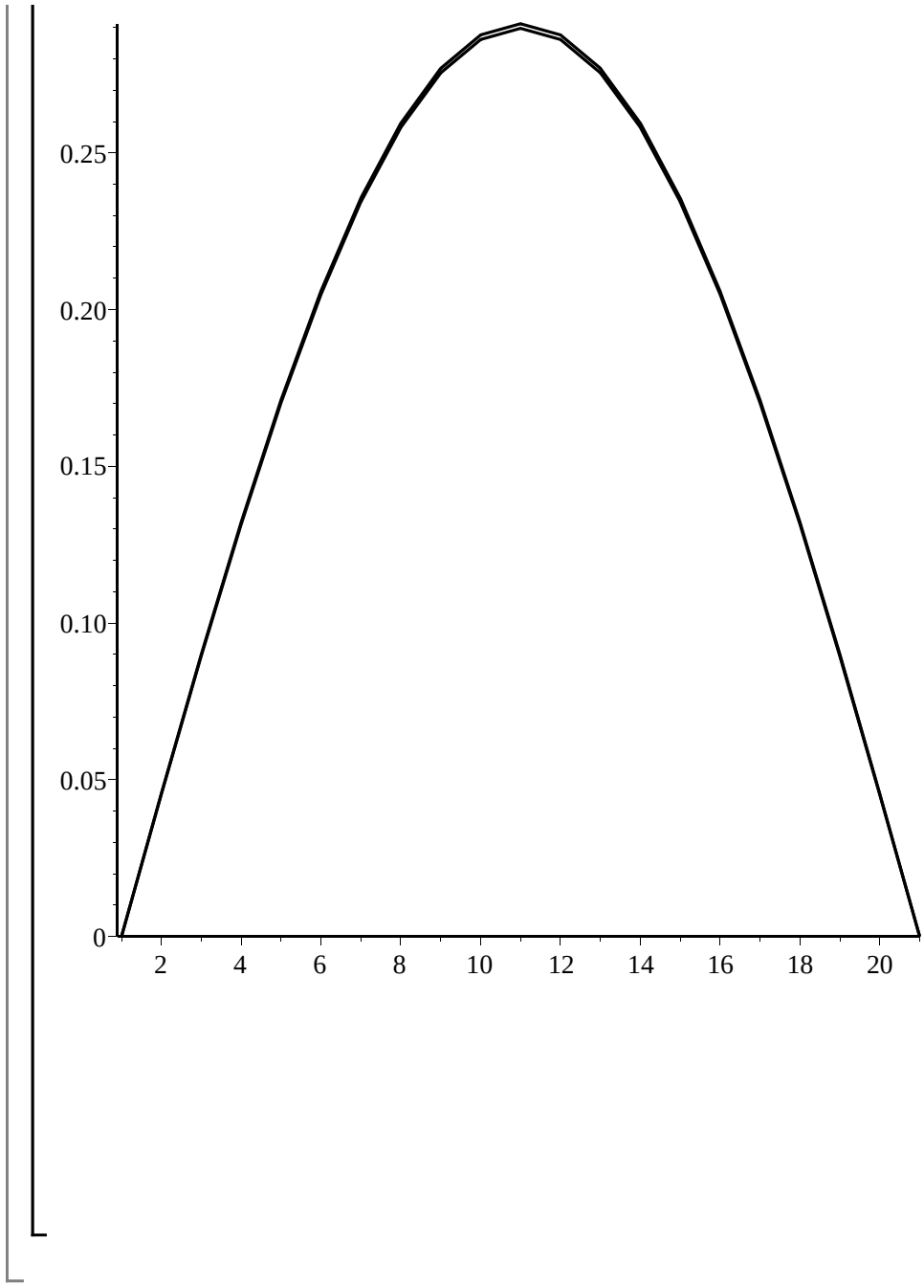
And plot the results again:

```
> resplot := [seq( listplot( [seq( v[j,m], j=0..n)] ), m=1..
  steps)]:
> display(resplot,insequence=true);
```



Of course, we would like to compare the numerical solution with the anayltic one. The following plot shows, that we can come pretty close ...

```
> exact_plot := listplot( [seq( u_part(j*h, steps*tau), j=0..
  n)] ):
> display(resplot[steps],exact_plot);
```


0.25
0.20
0.15
0.10
0.05
0
2
4
6
8
10
12
14
16
18
20