Equation de diffusion (chaleur) 1D instationnaire

==========

Dr. Laïd MESSAOUDI

Département de Mécanique

Université HL Batna

==========

Master: Energétique

Matière: Méthodes Numériques Appliquées

==========

Problématique:
Détermination de la température $T(x, t)$ à travers la longueur d'une barre de très faible section dont les extrémités sont maintenues à des températures constantes (conditions de Dirichlet). Pour celà, nous allons utiliser toute la puissance de Maple à résoudre ce problème *analytiquement* par la méthode de séparation de variables (MSV) et *numériquement* par deux méthodes. Une solution directe par des commandes spécifiques à Maple et une solution discrétisée par programmation classique. Dans cette dernière, les deux cas de schémas de discrétisation *Explicite* et *Implicite* sont traités.

Le problème mathématique est défini comme suit:

$$\frac{\partial}{\partial t} T(x, t) = \frac{\partial^2}{\partial x^2} T(x, t)$$

avec les conditions aux limites et la condition initiale:

$$T(0, t) = 0,$$
$$T(1, t) = 0,$$
$$T(x, 0) = 1$$

Solution analytique:

```
> Restart:
```

```
> Eq := diff(T(X, τ), τ) = diff(diff(T(X, τ), X), X);
```

$$\frac{\partial}{\partial \tau} T(X, \tau) = \frac{\partial}{\partial X}\left(\frac{\partial}{\partial X} T(X, \tau)\right)$$

```
> Eq1 := subs(T(X, τ) = f(X)·g(τ), Eq);
```

$$\frac{\partial}{\partial \tau}\left(f(X)\, g(\tau)\right) = \frac{\partial}{\partial X}\left(\frac{\partial}{\partial X}\left(f(X)\, g(\tau)\right)\right)$$

```
> expand(Eq1);
```

$$f(X)\left(\frac{d}{d\tau} g(\tau)\right) = \left(\frac{d}{dX}\left(\frac{d}{dX} f(X)\right)\right) g(\tau)$$

```
> Eq2 := (%)/(f(X)·g(τ));
```

$$\frac{\frac{d}{d\tau} g(\tau)}{g(\tau)} = \frac{\frac{d}{dX}\left(\frac{d}{dX} f(X)\right)}{f(X)}$$

```
> Eq3 := rhs(Eq2) = -λ^2;
```

$$\frac{\frac{d}{dX}\left(\frac{d}{dX} f(X)\right)}{f(X)} = -\lambda^2$$

```
> Eq3 := Eq3·f(X);
```

$$\frac{d}{dX}\left(\frac{d}{dX} f(X)\right) = -f(X)\,\lambda^2$$

```
> Eq3 := lhs(%) − rhs(%) = 0;
```

$$\frac{d}{dX}\left(\frac{d}{dX} f(X)\right) + f(X)\,\lambda^2 = 0$$

Résolution de l'équation différentielle pour *X*:

```
> dsolve(Eq3);
```

$$f(X) = \_C1 \sin(\lambda X) + \_C2 \cos(\lambda X)$$

Appliquons les conditions aux limites pour déterminer les constantes:

```
> p := eval(subs( X = 0, rhs(%) ));
```

$$\_C2$$

```
> q := eval(subs( X = 1, rhs(%%) ));
```

$$\_C1 \sin(\lambda) + \_C2 \cos(\lambda)$$

```
> solve( {p = 0, q = 0}, [_C1, _C2]);
```

$$[[\_C1 = 0, \_C2 = 0]]$$

On trouve alors la solution triviale qui est à écarter.

```
> _C2 := 0; q;
```

$$0$$

$$\_C1 \sin(\lambda)$$

Résolution de l'équation pour λ:

> $solve(q = 0, \{\lambda\});$

$$\{\lambda = 0\}$$

Revenons donc à l'étape juste après la commande: $dsolve(Eq3)$.

> $f(X) := rhs(\%);$

$$\_C1 \sin(\lambda X) + \_C2 \cos(\lambda X)$$

> $f[n](X) := subs(\{\lambda = n^*\pi, \_C2 = 0\}, f(X));$

$$\_C1 \sin(n\pi X)$$

> $Eq4 := lhs(Eq2) = -\lambda^2;$

$$\frac{\frac{d}{d\tau} g(\tau)}{g(\tau)} = -\lambda^2$$

> $Eq4 := Eq4 \cdot g(\tau);$

$$\frac{d}{d\tau} g(\tau) = -g(\tau)\, \lambda^2$$

> $Eq4 := lhs(\%) - rhs(\%) = 0;$

$$\frac{d}{d\tau} g(\tau) + g(\tau)\, \lambda^2 = 0$$

Résolution de l'équation différentielle:

> $dsolve(Eq4);$

$$g(\tau) = \_C1\, e^{-\lambda^2 \tau}$$

> $dsolve(\{Eq4, g(0) = 1\});$

$$g(\tau) = e^{-\lambda^2 \tau}$$

> $g(\tau) := rhs(\%);$

$$e^{-\lambda^2 \tau}$$

> $g[n](\tau) := subs(\lambda = n^*\pi, g(\tau));$

$$e^{-n^2 \pi^2 \tau}$$

> $T[n](X, \tau) := f[n](X) \cdot g[n](\tau);$

$$\_C1 \sin(n\pi X)\, e^{-n^2 \pi^2 \tau}$$

Appliquons le théorème de superposition:

> $T(X, \tau) := Sum(T[n](X, \tau), n = 1..\infty);$

$$\sum_{n=1}^{\infty} \_C1 \sin(n\pi X)\, e^{-n^2 \pi^2 \tau}$$

Condition Initiale:

> $eval(subs(\tau = 0, T(X, \tau) = 1));$

$$\sum_{n=1}^{\infty} \_C1 \sin(n\pi X) = 1$$

> $\_C1 := 2 \cdot Int(\sin(n \cdot \pi \cdot x), x = 0..1) = 2 \cdot int(\sin(n \cdot \pi \cdot x), x = 0..1);$

$$2\left(\int_0^1 \sin(n\pi x)\,dx\right) = -\frac{2\,(-1+\cos(n\pi))}{n\pi}$$

```
> _C1 := subs( {cos(n*π) = (-1)^n}, _C1);
```

$$2\left(\int_0^1 \sin(n\pi x)\,dx\right) = -\frac{2\,(-1+(-1)^n)}{n\pi}$$

n doit être impaire, sinon $_C1 = 0$

```
> _C1 := 4/(n·π);
```

$$\frac{4}{n\pi}$$

```
> T(X, τ) := eval(T(X, τ));
```

$$\sum_{n=1}^{\infty} \frac{4\sin(n\pi X)\,e^{-n^2\pi^2\tau}}{n\pi}$$

```
> T(X, τ) := eval(subs(n = 2·k+1, T(X, τ)));
```

$$\sum_{2k+1=1}^{\infty} \frac{4\sin((2k+1)\pi X)\,e^{-(2k+1)^2\pi^2\tau}}{(2k+1)\pi}$$

```
> T(X, τ) := sum(4*sin((2*k+1)*π*X)*exp(-(2*k+1)^2*π^2*τ)/((2*k+1)*π), k = 0..infinity);
```

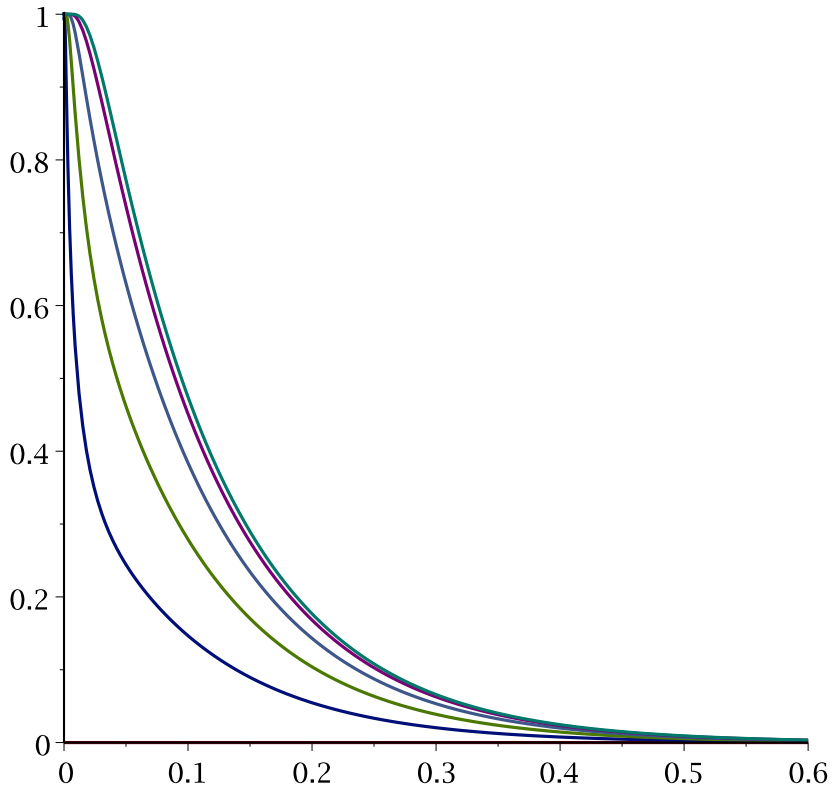
$$(X, \tau) \to \sum_{k=0}^{\infty} \frac{4\sin((2k+1)\pi X)\,e^{-(2k+1)^2\pi^2\tau}}{(2k+1)\pi}$$

```
> restart:
> T(X, τ) := sum(4*sin((2*k+1)*π*X)*exp(-(2*k+1)^2*π^2*τ)/((2*k+1)*π), k = 0..100);
```

$$(X, \tau) \to \sum_{k=0}^{100} \frac{4\sin((2k+1)\pi X)\,e^{-(2k+1)^2\pi^2\tau}}{(2k+1)\pi}$$

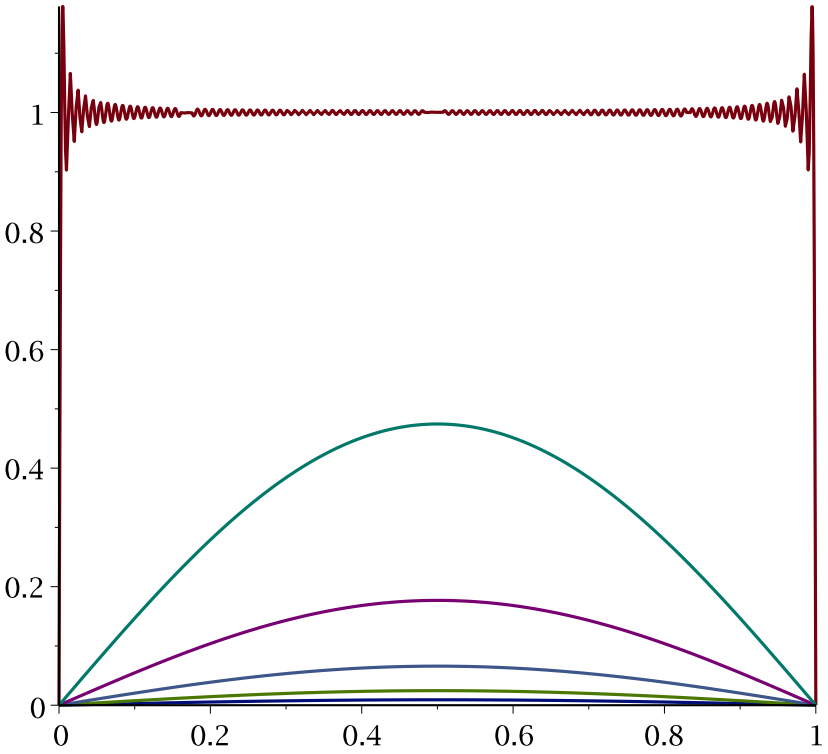
```
> plot( {T(0.1, τ), T(0.2, τ), T(0.3, τ), T(0.4, τ), T(0.5, τ), T(1, τ)}, τ = 0..0.6, title
      = Distribution de la température en fonction du temps, labels = [ `τ`, `T` ]);
```

Distribution de la température en fonction du temps

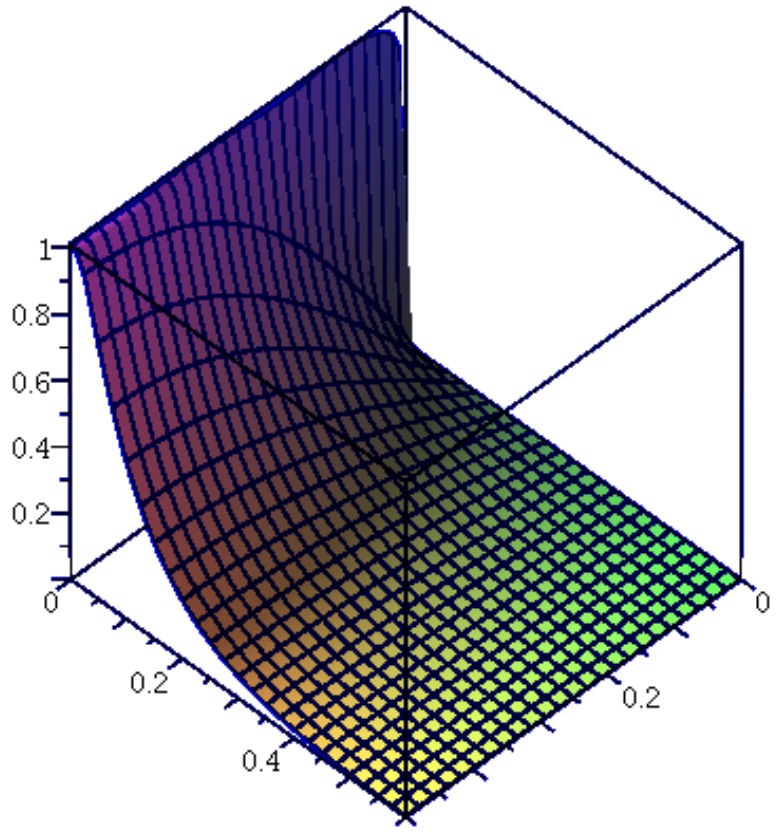



```
> plot({T(X, 0), T(X, 0.1), T(X, 0.2), T(X, 0.3), T(X, 0.4), T(X, 0.5)}, X = 0..1, title
      = "Distribution de la température en fonction de la distance", labels
      = [`X`, `T`]);
```

Distribution de la température en fonction de la distance




```
> plot3d( T(X, τ), X = 0..0.5, τ = 0..0.6, axes = boxed);
```



Solution directe:

```
> Restart:with(plots):
```

```
> Eq := ∂/∂t T(x, t) = ∂/∂x ( ∂/∂x T(x, t) );
```

$$Eq := \frac{\partial}{\partial t} T(x, t) = \frac{\partial^2}{\partial x^2} T(x, t)$$

```
> CLI:={T(x,0) = 1, T(0,t) = 0, T(1,t) = 0};
```

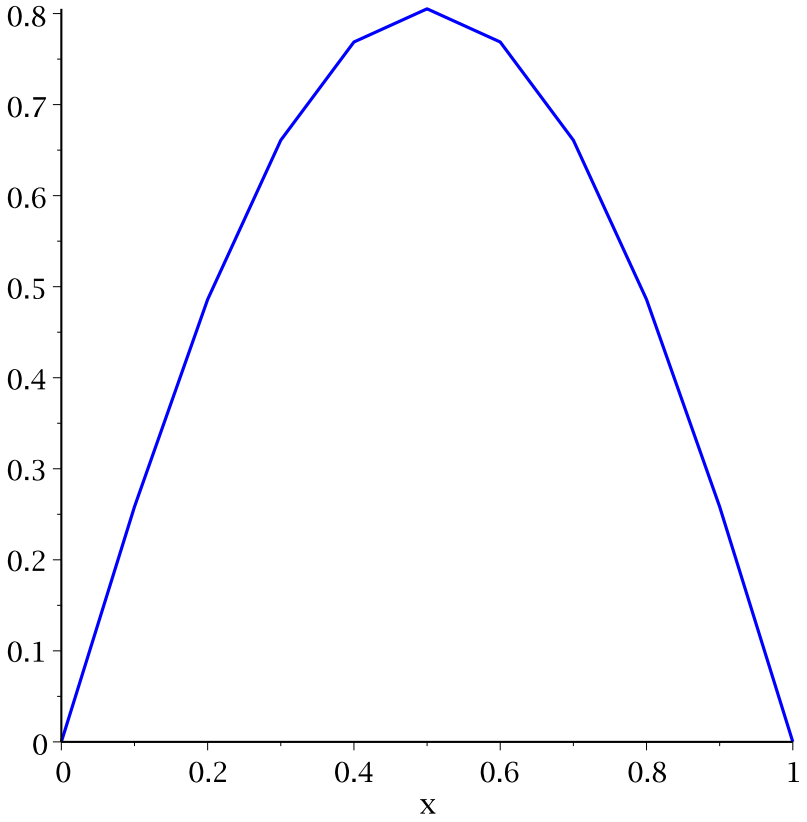
$$CLI := \{T(0, t) = 0, T(1, t) = 0, T(x, 0) = 1\}$$

Résolution et création d'un module de solutions comprenant les fonctions: plot, plot3d, animate, value, ...etc.

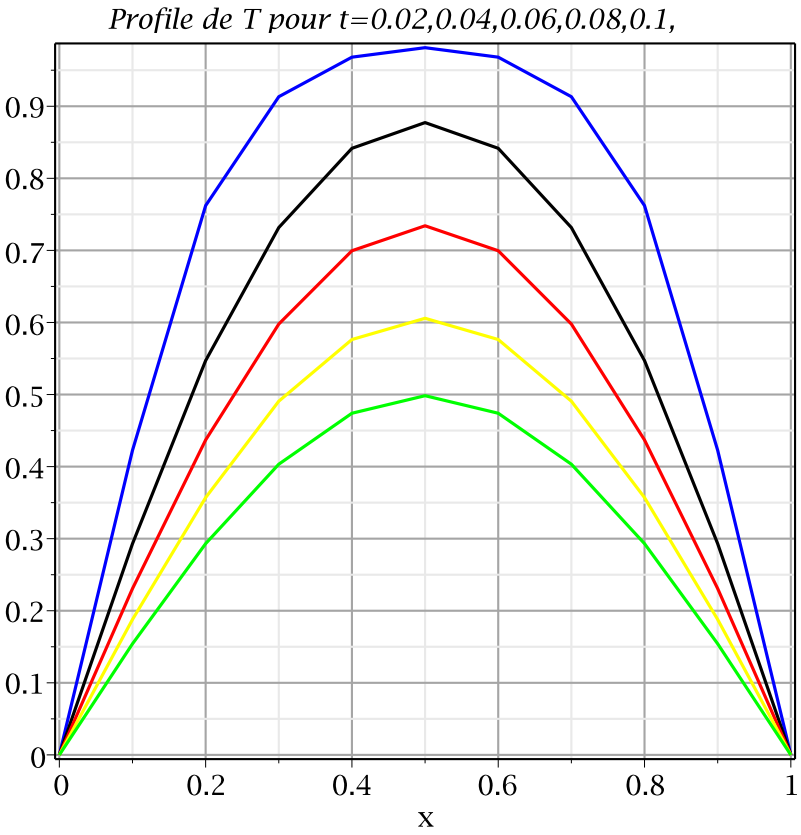
```
> Sol := pdsolve( Eq, CLI, numeric, timestep=0.01, spacestep=0.1)
  ;
```

$$Sol := \textbf{module}(\,)\ \textbf{export}\ plot, plot3d, animate, value, settings;\ \ldots\ \textbf{end module}$$

```
> Sol:- plot( T(x, t) , x = 0..1, t = 0.05, color = blue) ;
```



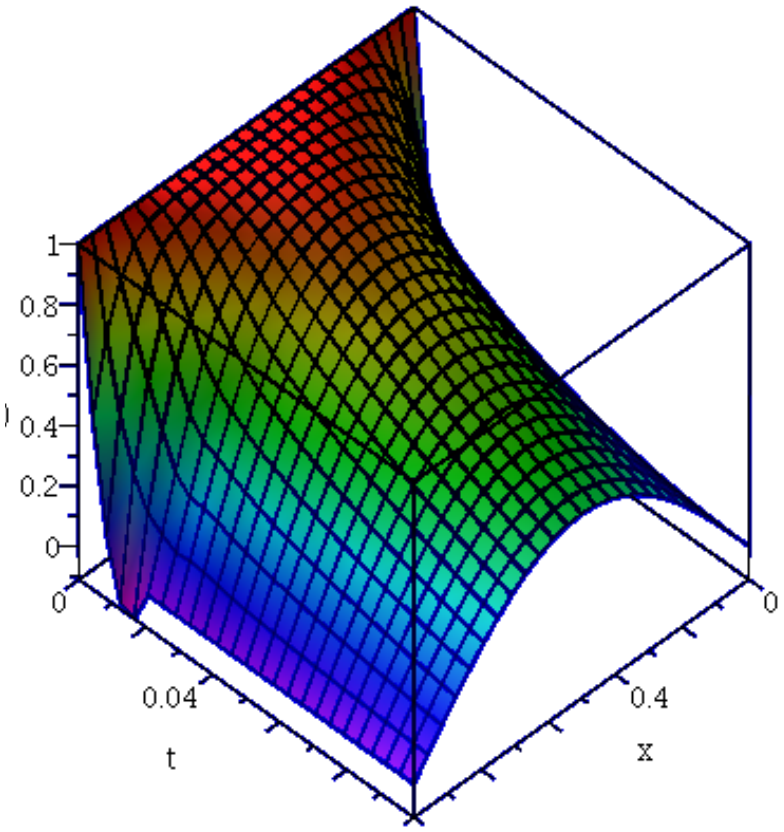

```
> g1 := Sol:- plot( t = 0.02, colour = blue) :
  g2 := Sol:- plot( t = 0.04, colour = black) :
  g3 := Sol:- plot( t = 0.06, colour = red) :
  g4 := Sol:- plot( t = 0.08, colour = yellow) :
  g5 := Sol:- plot( t = 0.1, colour = green) :
  plots[ display]( {g1, g2, g3, g4, g5}, gridlines = true, axes = boxed, title
      = `Profile de T pour t=0.02,0.04,0.06,0.08,0.1, `);
```



Cette commande permet de voir l'animation du refroidissement de la barre jusqu'à l'état stationnaire.

```
Sol :- animate( T(x,t) ,t=0..0.7, frames=50, labels=["x", "T(x,t)
"], labelfont=[TIMES,ROMAN,14]);
```

```
> Sol :- plot3d(T(x,t), t=0..0.1, shading=zhue, axes=boxed,
  labels=["x","t","T(x,t)"], labelfont=[TIMES,ROMAN,16]);
```



Pour avoir les valeurs numériques, on crée une procédure utilisant la commande value à partir du module de la solution.

```
> ValTemp := Sol :- value();
```

proc( ) ... end proc

La veleur de la temperature en $x = 0.1, t = 0.01$ est :

```
> ValTemp( 0.1, 0.01 );
```

$[x = 0.1,\ t = 0.01,\ T(x,\ t) = 0.7320441988950273]$

Solution discrétisée:

```
> Restart : with(plots) :
> Δx := 0.2 ; Δt := 0.005;
```

$$\Delta x := 0.2$$
$$\Delta t := 0.005$$

```
> λ := Δt/Δx^2;
```

$$\lambda := 0.1250000000$$

```
> i_max := 11;
```

$$i_{max} := 11$$

```
> n_max := 15;
```

$$n_{max} := 15$$

```
> α := 0; β := 0; σ := 1;
```

$$\alpha := 0$$
$$\beta := 0$$
$$\sigma := 1$$

Condition initiale:

```
> for i from 2 to i_max − 1 do T[i, 0] := σ end do;
```

$$T_{2,0} := 1$$
$$T_{3,0} := 1$$
$$T_{4,0} := 1$$
$$T_{5,0} := 1$$
$$T_{6,0} := 1$$
$$T_{7,0} := 1$$
$$T_{8,0} := 1$$
$$T_{9,0} := 1$$
$$T_{10,0} := 1$$

Condition limite gauche:

```
> for n from 0 to n_max do T[1, n] := α end do;
```

$$T_{1,0} := 0$$
$$T_{1,1} := 0$$
$$T_{1,2} := 0$$
$$T_{1,3} := 0$$
$$T_{1,4} := 0$$
$$T_{1,5} := 0$$
$$T_{1,6} := 0$$
$$T_{1,7} := 0$$
$$T_{1,8} := 0$$
$$T_{1,9} := 0$$
$$T_{1,10} := 0$$
$$T_{1,11} := 0$$
$$T_{1,12} := 0$$
$$T_{1,13} := 0$$
$$T_{1,14} := 0$$
$$T_{1,15} := 0$$

Condition limite droite:

```
> for n from 0 to n_max do T[i_max, n] := β end do;
```

$$T_{11,0} := 0$$
$$T_{11,1} := 0$$

$$T_{11,2} := 0$$
$$T_{11,3} := 0$$
$$T_{11,4} := 0$$
$$T_{11,5} := 0$$
$$T_{11,6} := 0$$
$$T_{11,7} := 0$$
$$T_{11,8} := 0$$
$$T_{11,9} := 0$$
$$T_{11,10} := 0$$
$$T_{11,11} := 0$$
$$T_{11,12} := 0$$
$$T_{11,13} := 0$$
$$T_{11,14} := 0$$
$$T_{11,15} := 0$$

Schéma Explicite:

```
> for n from 0 to n_max do
    for i from 2 to i_max - 1 do
        T[i, n+1] := λ·T[i-1, n] + (1 - 2·λ)·T[i, n] + λ·T[i+1, n]
    end do;
  end do;
```

Affichage de la solution:

```
> for i from 2 to i_max - 1 do T[i, n_max] end do;
```

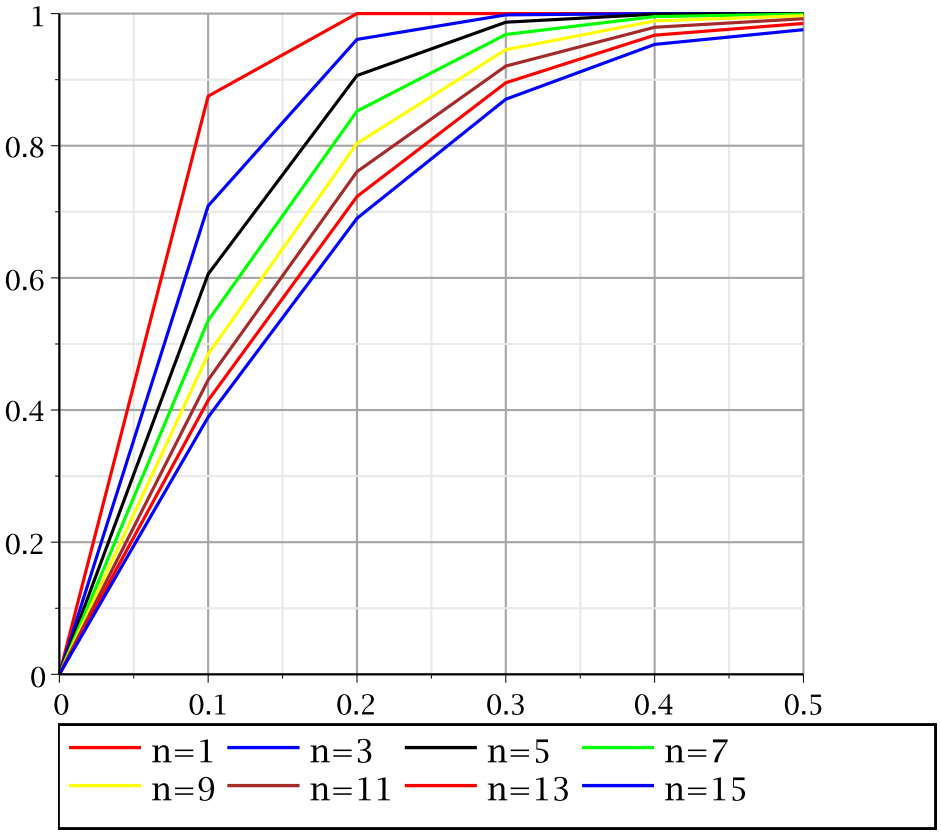
0.3892606696
0.6900617869
0.8704018496
0.9533285673
0.9756458209
0.9533285673
0.8704018496
0.6900617868
0.3892606696

Création des listes pour le tracé sur la première moitié du domaine:

```
> for n from 1 to n_max do
    liste[n] := [[0.0, T[1, n]], [0.1, T[2, n]], [0.2, T[3, n]], [0.3, T[4, n]], [0.4,
      T[5, n]], [0.5, T[6, n]]]:
  end do:
```

```
> multiple(listplot, [liste[1], color = red, legend = "n=1"], [liste[3], color
    = blue, legend = "n=3"], [liste[5], color = black, legend = "n=5"],
    [liste[7], color = green, legend = "n=7"], [liste[9], color = yellow, legend
    = "n=9"], [liste[11], color = brown, legend = "n=11"], [liste[13], color
    = red, legend = "n=13"], [liste[15], color = blue, legend = "n=15"],
```

gridlines = *true*);



Création améliorée des listes pour le tracé sur tout le domaine:

```
> for n from 1 to n_max do
    liste[n] := [α, seq(T[i, n], i = 2 .. i_max − 1), β]
  end do
```

$liste_1$:= [0, 0.8750000000, 1.000000000, 1.000000000, 1.000000000, 1.000000000, 1.000000000, 1.000000000, 1.000000000, 0.8750000000, 0]

$liste_2$:= [0, 0.7812500000, 0.9843750000, 1.000000000, 1.000000000, 1.000000000, 1.000000000, 1.000000000, 0.9843750000, 0.7812500000, 0]

$liste_3$:= [0, 0.7089843750, 0.9609375000, 0.9980468750, 1.000000000, 1.000000000, 1.000000000, 0.9980468750, 0.9609375000, 0.7089843750, 0]

$liste_4$:= [0, 0.6518554687, 0.9340820313, 0.9936523437, 0.9997558594, 1.000000000, 0.9997558594, 0.9936523437, 0.9340820313, 0.6518554687, 0]

$liste_5$:= [0, 0.6056518554, 0.9062500001, 0.9869689941, 0.9990234376, 0.9999389648, 0.9990234376, 0.9869689941, 0.9062500001, 0.6056518554, 0]

$liste_6 := [0, 0.5675201416, 0.8787651063, 0.9783859253, 0.9976310731, 0.9997100830, 0.9976310731, 0.9783859253, 0.8787651063, 0.5675201416, 0]$

$liste_7 := [0, 0.5354857445, 0.8523120881, 0.9683389664, 0.9954853059, 0.9991903304, 0.9954853059, 0.9683389664, 0.8523120881, 0.5354857445, 0]$

$liste_8 := [0, 0.5081533194, 0.8272121550, 0.9572288990, 0.9925551415, 0.9982640742, 0.9925551415, 0.9572288990, 0.8272121550, 0.5081533194, 0]$

$liste_9 := [0, 0.4845165090, 0.8035818935, 0.9453925863, 0.9888529778, 0.9968368410, 0.9888529778, 0.9453925863, 0.8035818935, 0.4845165090, 0]$

$liste_{10} := [0, 0.4638351185, 0.7814250570, 0.9330987986, 0.9844184118, 0.9948408752, 0.9844184118, 0.9330987986, 0.7814250570, 0.4638351185, 0]$

$liste_{11} := [0, 0.4455544710, 0.7606855324, 0.9205545326, 0.9793062680, 0.9922352594, 0.9793062680, 0.9205545326, 0.7606855324, 0.4455544710, 0]$

$liste_{12} := [0, 0.4292515448, 0.7412777748, 0.9079148745, 0.9735784250, 0.9890030116, 0.9735784250, 0.9079148744, 0.7412777748, 0.4292515448, 0]$

$liste_{13} := [0, 0.4145983804, 0.7231041335, 0.8952931809, 0.9672985545, 0.9851468649, 0.9672985545, 0.8952931808, 0.7231041335, 0.4145983804, 0]$

$liste_{14} := [0, 0.4013368020, 0.7060645452, 0.8827702217, 0.9605289216, 0.9806847873, 0.9605289216, 0.8827702216, 0.7060645452, 0.4013368020, 0]$

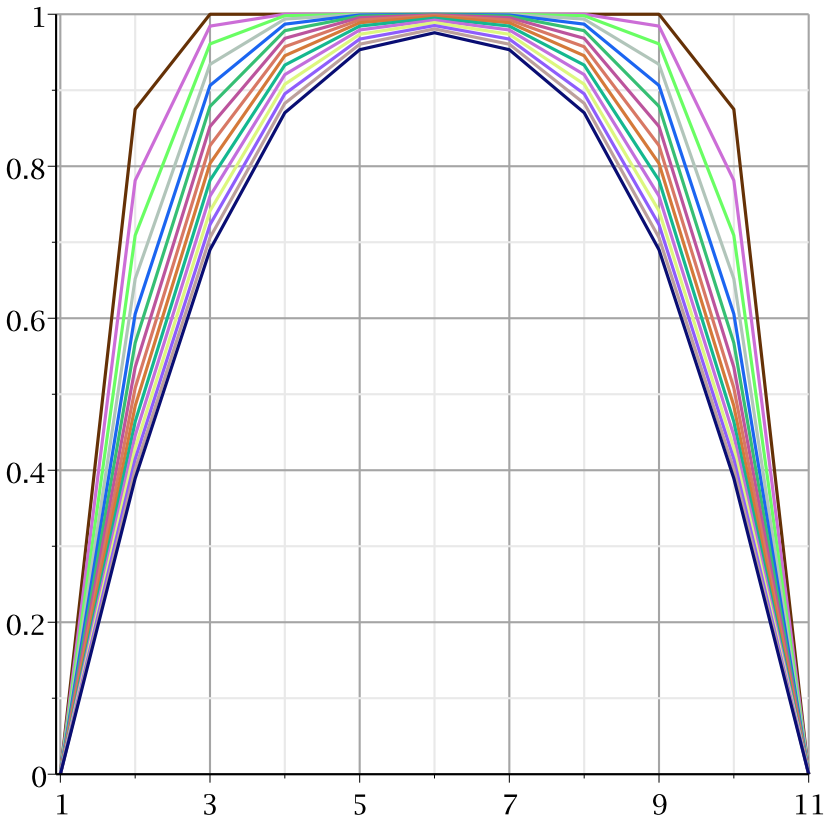
$liste_{15} := [0, 0.3892606696, 0.6900617869, 0.8704018496, 0.9533285673, 0.9756458209, 0.9533285673, 0.8704018496, 0.6900617868, 0.3892606696, 0]$

Enregistrement dans un fichier pour comparer par la suite avec la solution implicite.

```
> writedata( diffusionExp, liste[ n_max ])
```

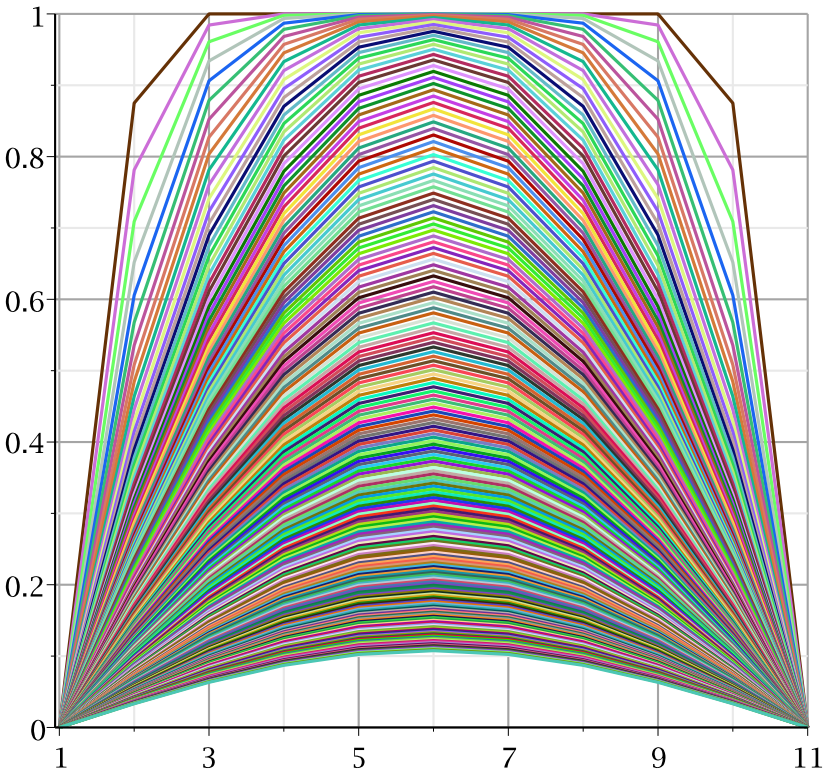
Une autre manière de tracer les courbes avec des couleurs aléatoires:

```
> multiple( listplot, seq( [ liste[ i], color = COLOR( RGB, rand( ) / 10^12, rand( ) / 10^12,
      rand( ) / 10^12 ) ], i = 1 .. n_max ), gridlines = true );
```

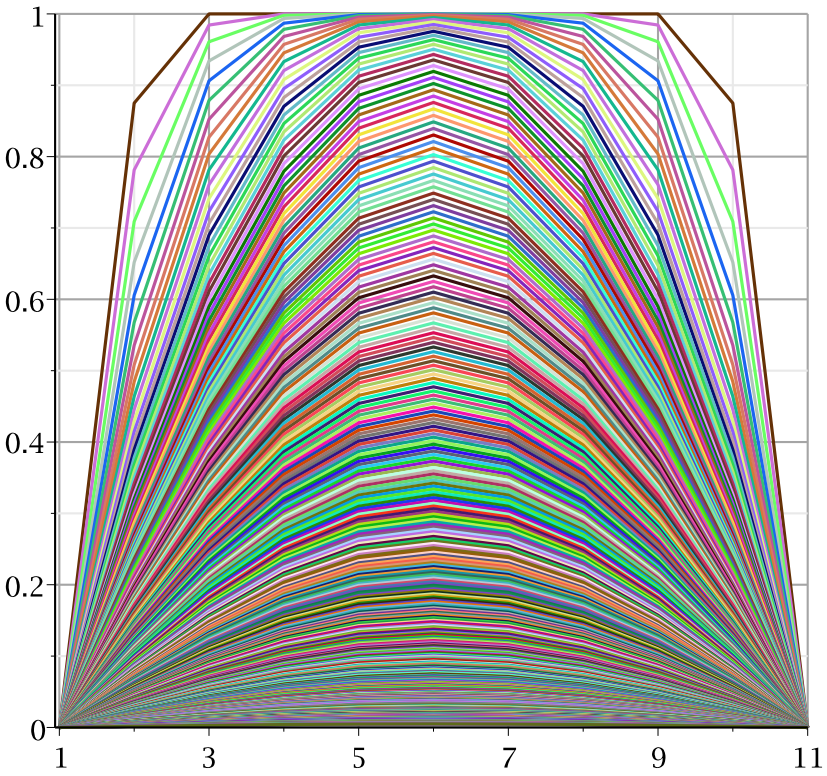


Nous remarquons à partir de ces courbes que l'état stationnaire n'est pas encore atteint. Il peut être déterminé en augmentant $n_{\max}$.

Distribution de la température pour n_{max}= 200



Distribution de la température pour $n_{max}= 450$

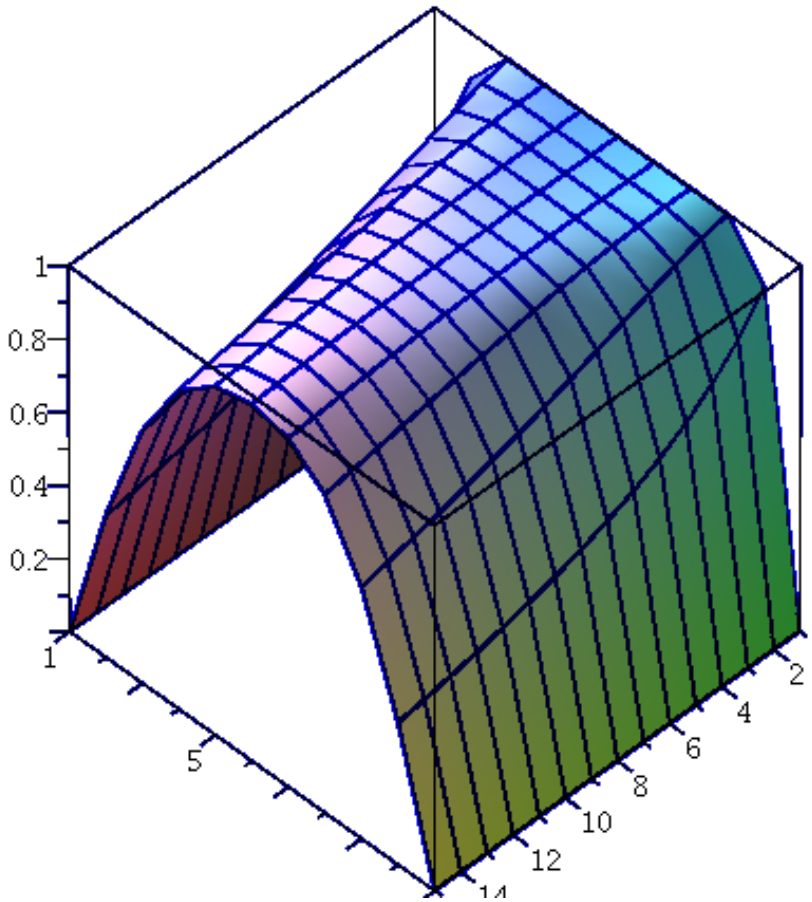


>

> Donc l'état stationnaire est atteint au voisinage de 500 pas de temps c'est à dire après environ 2.5 secondes (Théoriquement, d'après la première partie, ce temps est évalué à environ 6 s).

Tracé 3D:

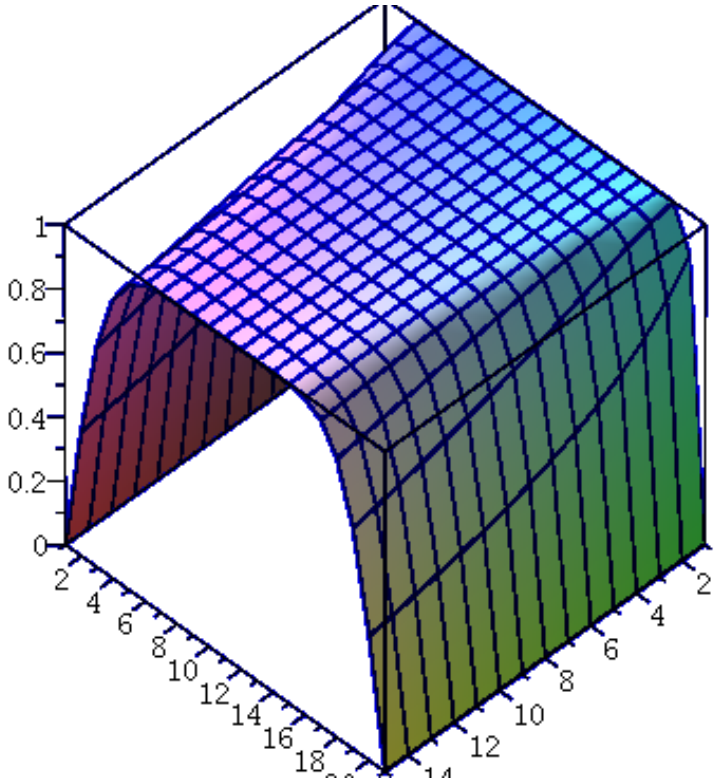
> $listplot3d([seq([seq(T[i,n], i=1..i_{max})], n=1..n_{max})]);$



>

En augmentant le nombre de noeuds à 21 puis à 51 nous aurons:

Nombre de noeuds = 21



Nombre de noeuds = 51

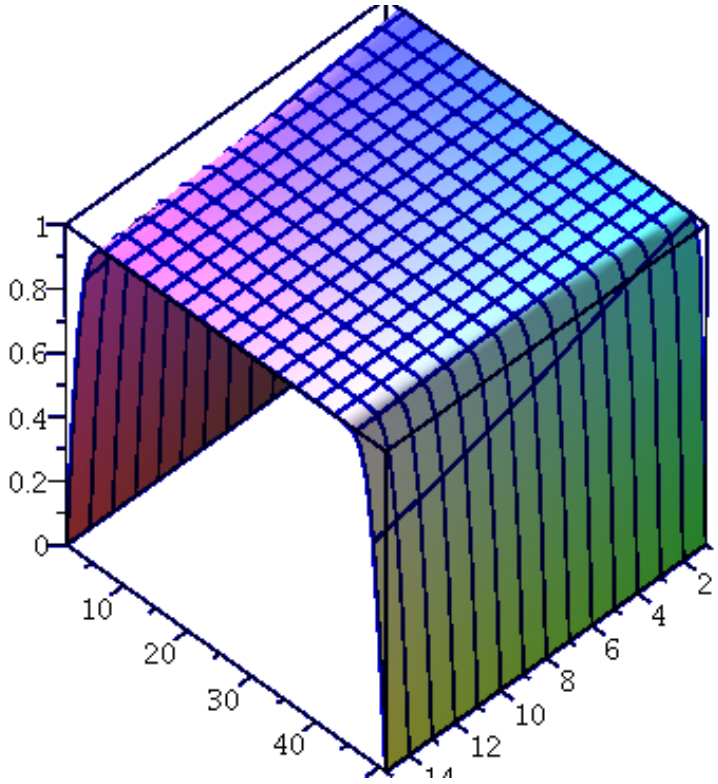


Schéma Implicite:

>

Boucle principale: à exécuter après avoir exécuter les données plus haut (CL et CI).

```
> for n from 0 to n_max do
   k := 1 :
    for i from 2 to i_max - 1 do
        Eq[k] := T[i, n] = -λ·T[i - 1, n + 1] + (1 + 2·λ)·T[i, n + 1] - λ·T[i + 1, n
      + 1] ;
         k := k + 1 :
     end do:
    for k from 1 to i_max - 2 do Eq[k] end do:
    Eqs := {seq(Eq[k], k = 1 .. i_max - 2)} :
    Temps := [seq(T[i, n + 1], i = 2 .. i_max - 1)] :
    SolTemp := solve(Eqs, Temps) :
   for i from 2 to i_max - 1 do T[i, n + 1] := rhs(SolTemp_{1, i - 1}) end do:
```

```
    end do:

> for n from 1 to n_max do
      liste[n] := [α, seq(T[i, n], i = 2 .. i_max − 1), β]
    end do
```

$liste_1 := [0, 0.8989794845, 0.9897948448, 0.9989689637, 0.9998947922, 0.9999789584, 0.9998947922, 0.9989689637, 0.9897948448, 0.8989794845, 0]$

$liste_2 := [0, 0.8164965719, 0.9731298431, 0.9964431002, 0.9995494494, 0.9998930566, 0.9995494494, 0.9964431002, 0.9731298431, 0.8164965719, 0]$

$liste_3 := [0, 0.7484551577, 0.9525790019, 0.9922961166, 0.9988373630, 0.9996819179, 0.9988373630, 0.9922961166, 0.9525790019, 0.7484551577, 0]$

$liste_4 := [0, 0.6917539090, 0.9298978282, 0.9865923577, 0.9976568159, 0.9992768975, 0.9976568159, 0.9865923577, 0.9298978282, 0.6917539090, 0]$

$liste_5 := [0, 0.6440302008, 0.9062707364, 0.9794945372, 0.9959357737, 0.9986086727, 0.9959357737, 0.9794945372, 0.9062707364, 0.6440302008, 0]$

$liste_6 := [0, 0.6034726172, 0.8824845659, 0.9712071504, 0.9936306406, 0.9976130663, 0.9936306406, 0.9712071504, 0.8824845659, 0.6034726172, 0]$

$liste_7 := [0, 0.5686831200, 0.8590502624, 0.9619429767, 0.9907223015, 0.9962349133, 0.9907223015, 0.9619429767, 0.8590502624, 0.5686831200, 0]$

$liste_8 := [0, 0.5385753134, 0.8362881741, 0.9519043281, 0.9872112929, 0.9944301892, 0.9872112929, 0.9519043281, 0.8362881741, 0.5385753134, 0]$

$liste_9 := [0, 0.5122990304, 0.8143877972, 0.9412735488, 0.9831130656, 0.9921667645, 0.9831130656, 0.9412735488, 0.8143877972, 0.5122990304, 0]$

$liste_{10} := [0, 0.4891841817, 0.7934495734, 0.9302091751, 0.9784537869, 0.9894241690, 0.9784537869, 0.9302091751, 0.7934495734, 0.4891841817, 0]$

$liste_{11} := [0, 0.4686987531, 0.7735140772, 0.9188454321, 0.9732668431, 0.9861927038, 0.9732668431, 0.9188454321, 0.7735140772, 0.4686987531, 0]$

$liste_{12} := [0, 0.4504172369, 0.7545823440, 0.9072935851, 0.9675900503, 0.9824721731, 0.9675900503, 0.9072935851, 0.7545823440, 0.4504172369, 0]$

$liste_{13} := [0, 0.4339967871, 0.7366299755, 0.8956442162, 0.9614635058, 0.9782704396, 0.9614635058, 0.8956442162, 0.7366299755, 0.4339967871, 0]$

$liste_{14} := [0, 0.4191591175, 0.7196168781, 0.8839698593, 0.9549279854,$

0.9736019488, 0.9549279854, 0.8839698593, 0.7196168781,
0.4191591175, 0]

$liste_{15}$:= [0, 0.4056766877, 0.7034939372, 0.8723276598, 0.9480237859,
0.9684863162, 0.9480237859, 0.8723276598, 0.7034939372,
0.4056766877, 0]

Nous pouvons tracer les même courbes que pour le cas explicite.

>

Enregistrement dans un fichier pour comparer par la solution explicite.

> *writedata*(*diffusionImp*, *liste*[n_{max}])

Lectures des résultats explicite et implicite à partir des fichiers:

> *SolExplicite* := *readdata*(*diffusionExp*)

SolExplicite := [0., 0.3892606696, 0.6900617869, 0.8704018496, 0.9533285673,
0.9756458209, 0.9533285673, 0.8704018496, 0.6900617868,
0.3892606696, 0.]

> *SolImplicite* := *readdata*(*diffusionImp*)

SolImplicite := [0., 0.4056766877, 0.7034939372, 0.8723276598,
0.9480237859, 0.9684863162, 0.9480237859, 0.8723276598,
0.7034939372, 0.4056766877, 0.]

Tracé des coubes comparatives:

> *multiple*(*listplot*, [*SolExplicite*, color = red], [*SolImplicite*, color = blue],
gridlines = *true*)

Comparaison entre les schémas Explicite (rouge) et Implicite (bleu).