Corrigé du TD du 9 février 2006

1 Euler

1)

```
Euler:=proc(t0,x0,T,f,g,h)
   local n,t,x,L,i;
      n:=floor(T/h);
      t:=t0;
      x:=x0;
      L:=[[t0,x0]];
      for i to n do
         x:=x+h*f(t,x);
         t:=t+h;
         L:=[op(L),[t,x]];
      end do;
      plot([L,g],t0..t0+T);
end proc;
```

Remarques:
_Dans la boucle `for`, faire attention à réactualiser `x` avant `t`. Si on faisait `t:=t+h;` puis `x:=x+h*f(t,x);`, on exécuterait l'algorithme $x_{n+1} = x_n + f(t_{n+1}, x_n)$...
_J'ai ajouté ici un argument supplémentaire: une fonction g à une variable (le temps) qui est la vraie solution de l'équation. A la fin, on peut donc comparer vraie solution et approximation sur un même graphique...

On exécute ensuite les commandes de test en faisant attention au fait que f est une fonction de deux variables (ex.: `Euler(0,1,1,(t,x)->2*x,t->exp(2*t),0.5);`)...

2)

```
erreurEuler:=proc(t0,x0,T,f,g,h)
   local n,t,x,L,i;
      n:=floor(T/h);
      t:=t0;
      x:=x0;
      L:=[];
      for i to n do
         x:=x+h*f(t,x);
         t:=t+h;
         L:=[op(L),[t,log10(abs(x-g(t)))]];
      end do;
      plot(L);
```

```
end proc;
```

La commande `erreurEuler(0,1,1,(t,x)->2*x,t->exp(2*t),0.001);` donne la figure 1. Bien sûr, la précision est de moins en moins bonne à mesure que l'algorithme progresse.

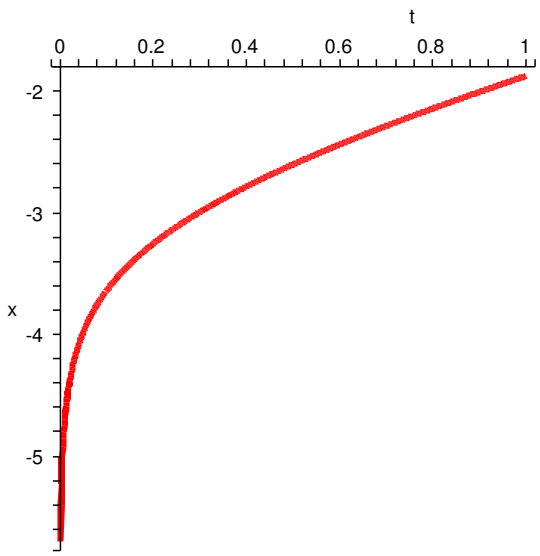

Figure 1:

2 Point-Milieu

```
PM:=proc(t0,x0,T,f,g,h)
   local n,t,x,L,i;
      n:=floor(T/h);
      t:=t0;
      x:=x0;
      L:=[[t0,x0]];
      for i to n do
         x:=x+h*f(t+h/2,x+h/2*f(t,x));
         t:=t+h;
         L:=[op(L),[t,x]];
      end do;
      plot([L,g],t0..t0+T);
end proc;
```

Comparez `Euler(0,1,1,(t,x)->2*x,t->exp(2*t),0.1);` et `PM(0,1,1,(t,x)->2*x,t->exp(2*t),0.1);`. On voit que cette méthode est plus efficace. Ici, au lieu d'approximer $x'(t)$ par $f(t_n, x_n)$ sur $[t_n, t_{n+1}]$ (approximation à gauche de l'intervalle), on l'approxime par une approximation de la

valeur de x' au milieu de l'intervalle $(f(t_n + h/2, x_n + h/2 * f(t_n, x_n)))$, qui est plus conforme à la valeur moyenne de x' sur $[t_n, t_{n+1}]$.

On peut faire un programme de tracé de l'erreur sur le modèle de `erreurEuler` et on obtient le graphe suivant:

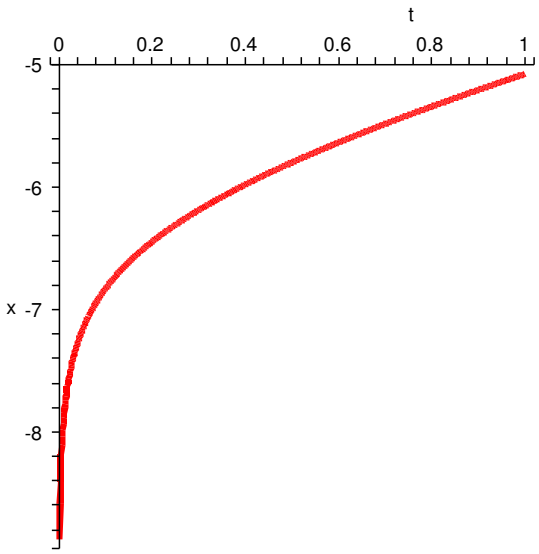

Figure 2:

3 Runge-Kutta

Ici, on fait une approximation encore plus fine de la valeur moyenne de $x'(t)$ sur $[t_n, t_{n+1}]$. C'est la plus efficace parmi les trois présentées ici.

```
   RK:=proc(t0,x0,T,f,g,h)
   local n,t,x,x1,x2,x3,L,i;
      n:=floor(T/h);
      t:=t0;
      x:=x0;
      L:=[[t0,x0]];
      for i to n do
         x1:=x+h/2*f(t,x);
         x2:=x+h/2*f(t+h/2,x1);
         x3:=x+h*f(t+h/2,x2);
         x:=x+h/6*(f(t,x)+2*f(t+h/2,x1)+2*f(t+h/2,x2)+f(t+h,x3));
         t:=t+h;
         L:=[op(L),[t,x]];
      end do;
      plot([L,g],t0..t0+T);
end proc;
```

Ici, il faut prendre soin d'augmenter la précision de Maple (prendre `Digits:=20;` par exemple) car l'algorithme colle bien à la solution.

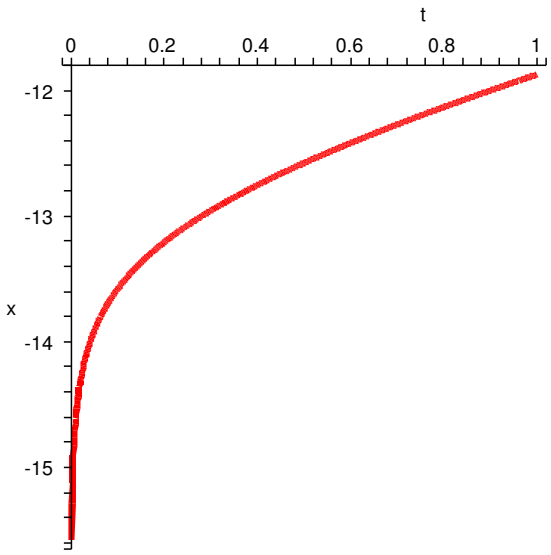

Figure 3: