Marcel Délèze
Edition 2017

§ 1.2 Résolution numérique d'une équation différentielle ordinaire du premier ordre

Interprétation géométrique et résolution graphique

Dans le § 1.1, nous avons considéré une équation différentielle avec condition initiale de la forme

$$\begin{cases} v' = f(t, v) \\ v(t_0) = v_0 \end{cases}$$

Que faire lorsque la solution *v* ne nous est pas donnée ?

Pour distinguer la solution que nous allons construire de la solution donnée dans le § 1.1, utilisons un autre symbole et écrivons le système sous la forme

$$\begin{cases} y' = f(t, y) \\ y(t_0) = y_0 \end{cases}$$

où $y(t)$ est la fonction cherchée.
A toute équation différentielle correspond un **champ de directions** associé à la fonction *f*. Pour l'exemple du § 1.1, voici la situation:

```
p = 4.; r = 1.; m0 = 8.; q = 2.; Clear[f, t, y]
```
efface

$$f[t\_, y\_] := \frac{p - r\,y^2}{m0 - q\,t};\ f[t, y]$$

$$\frac{4. - 1.\,y^2}{8. - 2.\,t}$$

```
tmin = 0.; tmax = 0.95 m0/q; ymin = 0.; ymax = 3.;
```

A tout point du plan, on peut associer le nombre réel

$$(t, y) \mapsto f(t, y)$$

```
pentes = Table[NumberForm[{t, y, f[t, y]}, 2],
```
table apparence numérique

```
  {t, tmin, tmax, (tmax - tmin)/18.}, {y, ymin, ymax, (ymax - ymin)/13.}]
```

```
{{{0., 0., 0.5}, {0., 0.23, 0.49}, {0., 0.46, 0.47}, {0., 0.69, 0.44}, {0., 0.92, 0.39},
  {0., 1.2, 0.33}, {0., 1.4, 0.26}, {0., 1.6, 0.17}, {0., 1.8, 0.074}, {0., 2.1, -0.039},
  {0., 2.3, -0.17}, {0., 2.5, -0.31}, {0., 2.8, -0.46}, {0., 3., -0.63}},
 {{0.21, 0., 0.53}, {0.21, 0.23, 0.52}, {0.21, 0.46, 0.5}, {0.21, 0.69, 0.46},
  {0.21, 0.92, 0.42}, {0.21, 1.2, 0.35}, {0.21, 1.4, 0.27}, {0.21, 1.6, 0.18},
  {0.21, 1.8, 0.078}, {0.21, 2.1, -0.041}, {0.21, 2.3, -0.17},
  {0.21, 2.5, -0.32}, {0.21, 2.8, -0.48}, {0.21, 3., -0.66}},
 {{0.42, 0., 0.56}, {0.42, 0.23, 0.55}, {0.42, 0.46, 0.53}, {0.42, 0.69, 0.49},
  {0.42, 0.92, 0.44}, {0.42, 1.2, 0.37}, {0.42, 1.4, 0.29}, {0.42, 1.6, 0.19},
  {0.42, 1.8, 0.083}, {0.42, 2.1, -0.044}, {0.42, 2.3, -0.19},
  {0.42, 2.5, -0.34}, {0.42, 2.8, -0.51}, {0.42, 3., -0.7}},
 {{0.63, 0., 0.59}, {0.63, 0.23, 0.59}, {0.63, 0.46, 0.56}, {0.63, 0.69, 0.52},
  {0.63, 0.92, 0.47}, {0.63, 1.2, 0.4}, {0.63, 1.4, 0.31}, {0.63, 1.6, 0.21},
  {0.63, 1.8, 0.088}, {0.63, 2.1, -0.047}, {0.63, 2.3, -0.2},
```

```
 {0.63, 2.5, -0.36}, {0.63, 2.8, -0.54}, {0.63, 3., -0.74}},
{{0.84, 0., 0.63}, {0.84, 0.23, 0.63}, {0.84, 0.46, 0.6}, {0.84, 0.69, 0.56},
 {0.84, 0.92, 0.5}, {0.84, 1.2, 0.42}, {0.84, 1.4, 0.33}, {0.84, 1.6, 0.22},
 {0.84, 1.8, 0.094}, {0.84, 2.1, -0.05}, {0.84, 2.3, -0.21},
 {0.84, 2.5, -0.39}, {0.84, 2.8, -0.58}, {0.84, 3., -0.79}},
{{1.1, 0., 0.68}, {1.1, 0.23, 0.67}, {1.1, 0.46, 0.64}, {1.1, 0.69, 0.6},
 {1.1, 0.92, 0.53}, {1.1, 1.2, 0.45}, {1.1, 1.4, 0.35}, {1.1, 1.6, 0.24},
 {1.1, 1.8, 0.1}, {1.1, 2.1, -0.053}, {1.1, 2.3, -0.23}, {1.1, 2.5, -0.41},
 {1.1, 2.8, -0.62}, {1.1, 3., -0.85}}, {{1.3, 0., 0.73}, {1.3, 0.23, 0.72},
 {1.3, 0.46, 0.69}, {1.3, 0.69, 0.64}, {1.3, 0.92, 0.58}, {1.3, 1.2, 0.49},
 {1.3, 1.4, 0.38}, {1.3, 1.6, 0.25}, {1.3, 1.8, 0.11}, {1.3, 2.1, -0.057},
 {1.3, 2.3, -0.24}, {1.3, 2.5, -0.45}, {1.3, 2.8, -0.67}, {1.3, 3., -0.91}},
{{1.5, 0., 0.79}, {1.5, 0.23, 0.78}, {1.5, 0.46, 0.75}, {1.5, 0.69, 0.7},
 {1.5, 0.92, 0.62}, {1.5, 1.2, 0.53}, {1.5, 1.4, 0.41}, {1.5, 1.6, 0.28},
 {1.5, 1.8, 0.12}, {1.5, 2.1, -0.062}, {1.5, 2.3, -0.26},
 {1.5, 2.5, -0.48}, {1.5, 2.8, -0.73}, {1.5, 3., -0.99}},
{{1.7, 0., 0.87}, {1.7, 0.23, 0.85}, {1.7, 0.46, 0.82}, {1.7, 0.69, 0.76},
 {1.7, 0.92, 0.68}, {1.7, 1.2, 0.58}, {1.7, 1.4, 0.45}, {1.7, 1.6, 0.3},
 {1.7, 1.8, 0.13}, {1.7, 2.1, -0.068}, {1.7, 2.3, -0.29}, {1.7, 2.5, -0.53},
 {1.7, 2.8, -0.79}, {1.7, 3., -1.1}}, {{1.9, 0., 0.95}, {1.9, 0.23, 0.94},
 {1.9, 0.46, 0.9}, {1.9, 0.69, 0.84}, {1.9, 0.92, 0.75}, {1.9, 1.2, 0.64},
 {1.9, 1.4, 0.5}, {1.9, 1.6, 0.33}, {1.9, 1.8, 0.14}, {1.9, 2.1, -0.075},
 {1.9, 2.3, -0.32}, {1.9, 2.5, -0.58}, {1.9, 2.8, -0.87}, {1.9, 3., -1.2}},
{{2.1, 0., 1.1}, {2.1, 0.23, 1.}, {2.1, 0.46, 1.}, {2.1, 0.69, 0.93},
 {2.1, 0.92, 0.83}, {2.1, 1.2, 0.71}, {2.1, 1.4, 0.55}, {2.1, 1.6, 0.37},
 {2.1, 1.8, 0.16}, {2.1, 2.1, -0.083}, {2.1, 2.3, -0.35}, {2.1, 2.5, -0.65},
 {2.1, 2.8, -0.97}, {2.1, 3., -1.3}}, {{2.3, 0., 1.2}, {2.3, 0.23, 1.2},
 {2.3, 0.46, 1.1}, {2.3, 0.69, 1.}, {2.3, 0.92, 0.94}, {2.3, 1.2, 0.8},
 {2.3, 1.4, 0.62}, {2.3, 1.6, 0.41}, {2.3, 1.8, 0.18}, {2.3, 2.1, -0.093},
 {2.3, 2.3, -0.39}, {2.3, 2.5, -0.73}, {2.3, 2.8, -1.1}, {2.3, 3., -1.5}},
{{2.5, 0., 1.4}, {2.5, 0.23, 1.3}, {2.5, 0.46, 1.3}, {2.5, 0.69, 1.2}, {2.5, 0.92, 1.1},
 {2.5, 1.2, 0.91}, {2.5, 1.4, 0.71}, {2.5, 1.6, 0.47}, {2.5, 1.8, 0.2}, {2.5, 2.1, -0.11},
 {2.5, 2.3, -0.45}, {2.5, 2.5, -0.83}, {2.5, 2.8, -1.3}, {2.5, 3., -1.7}},
{{2.7, 0., 1.6}, {2.7, 0.23, 1.6}, {2.7, 0.46, 1.5}, {2.7, 0.69, 1.4}, {2.7, 0.92, 1.3},
 {2.7, 1.2, 1.1}, {2.7, 1.4, 0.83}, {2.7, 1.6, 0.55}, {2.7, 1.8, 0.24}, {2.7, 2.1, -0.12},
 {2.7, 2.3, -0.53}, {2.7, 2.5, -0.97}, {2.7, 2.8, -1.5}, {2.7, 3., -2.}},
{{3., 0., 1.9}, {3., 0.23, 1.9}, {3., 0.46, 1.8}, {3., 0.69, 1.7}, {3., 0.92, 1.5},
 {3., 1.2, 1.3}, {3., 1.4, 1.}, {3., 1.6, 0.67}, {3., 1.8, 0.28}, {3., 2.1, -0.15},
 {3., 2.3, -0.63}, {3., 2.5, -1.2}, {3., 2.8, -1.8}, {3., 3., -2.4}},
{{3.2, 0., 2.4}, {3.2, 0.23, 2.4}, {3.2, 0.46, 2.3}, {3.2, 0.69, 2.1}, {3.2, 0.92, 1.9},
 {3.2, 1.2, 1.6}, {3.2, 1.4, 1.2}, {3.2, 1.6, 0.83}, {3.2, 1.8, 0.36}, {3.2, 2.1, -0.19},
 {3.2, 2.3, -0.8}, {3.2, 2.5, -1.5}, {3.2, 2.8, -2.2}, {3.2, 3., -3.}},
{{3.4, 0., 3.2}, {3.4, 0.23, 3.2}, {3.4, 0.46, 3.}, {3.4, 0.69, 2.8}, {3.4, 0.92, 2.5},
 {3.4, 1.2, 2.1}, {3.4, 1.4, 1.7}, {3.4, 1.6, 1.1}, {3.4, 1.8, 0.48}, {3.4, 2.1, -0.25},
 {3.4, 2.3, -1.1}, {3.4, 2.5, -2.}, {3.4, 2.8, -2.9}, {3.4, 3., -4.}},
{{3.6, 0., 4.9}, {3.6, 0.23, 4.8}, {3.6, 0.46, 4.6}, {3.6, 0.69, 4.3}, {3.6, 0.92, 3.8},
 {3.6, 1.2, 3.2}, {3.6, 1.4, 2.5}, {3.6, 1.6, 1.7}, {3.6, 1.8, 0.72}, {3.6, 2.1, -0.38},
 {3.6, 2.3, -1.6}, {3.6, 2.5, -3.}, {3.6, 2.8, -4.5}, {3.6, 3., -6.1}},
{{3.8, 0., 10.}, {3.8, 0.23, 9.9}, {3.8, 0.46, 9.5}, {3.8, 0.69, 8.8}, {3.8, 0.92, 7.9},
 {3.8, 1.2, 6.7}, {3.8, 1.4, 5.2}, {3.8, 1.6, 3.5}, {3.8, 1.8, 1.5}, {3.8, 2.1, -0.78},
 {3.8, 2.3, -3.3}, {3.8, 2.5, -6.1}, {3.8, 2.8, -9.2}, {3.8, 3., -12.}}}
```

Chacun de ces nombres représente la pente que doit avoir la fonction $y(t)$ au point (t, y)

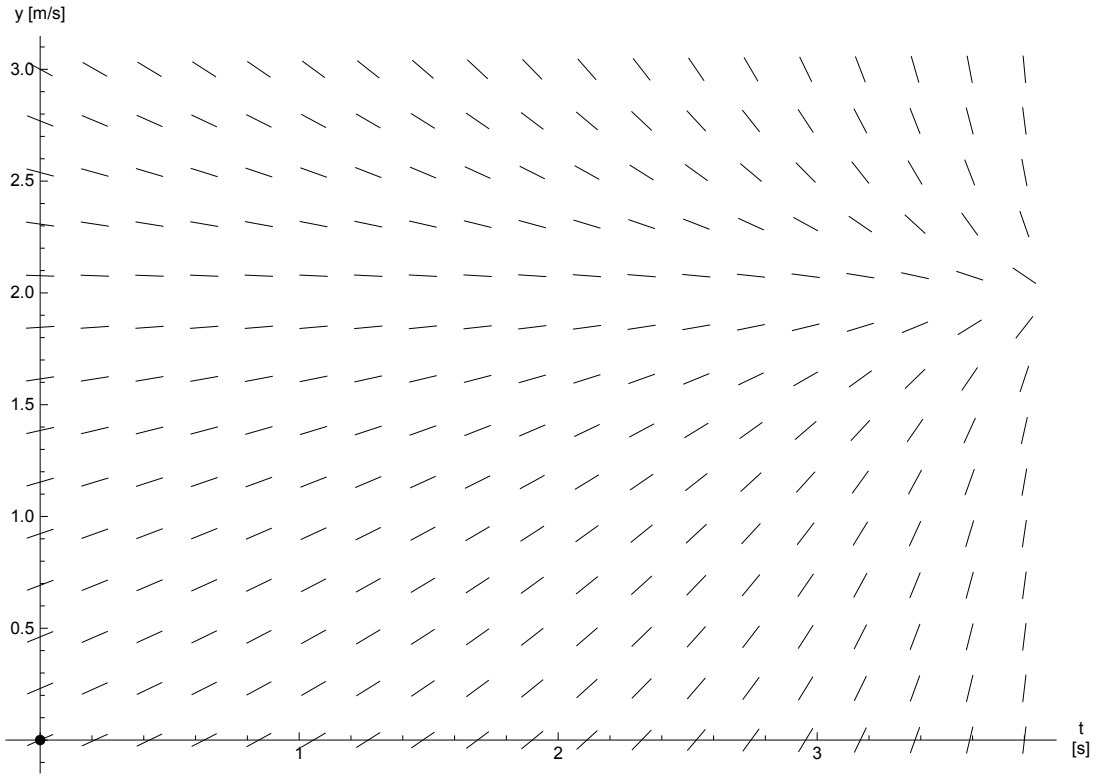
```
y' = f (t, y)
```

On représente la pente en chacun de ces points par un segment. Les vecteurs directeurs de ces

segments sont

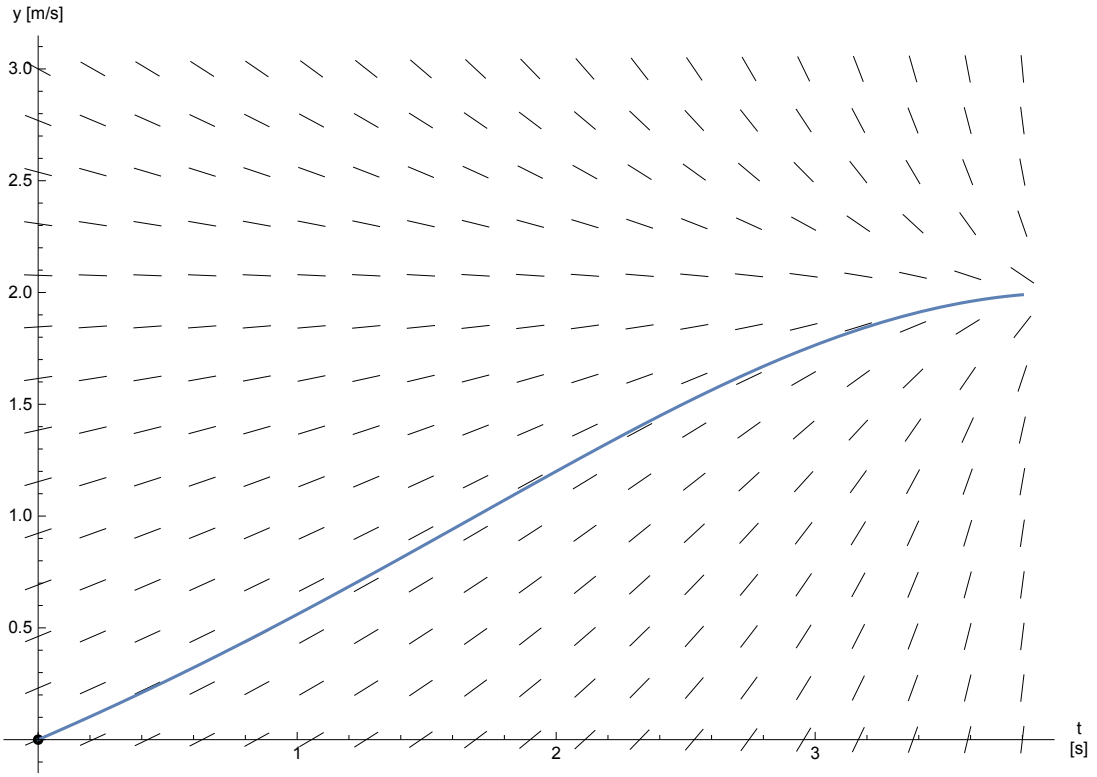
$$h\left(\frac{1}{f(t, y)}\right)$$

où *h* est un paramètre pour contrôler la longueur des segments. L'ensemble de ces segments représente un champ de directions (ou champ de pentes) que l'on peut dessiner.



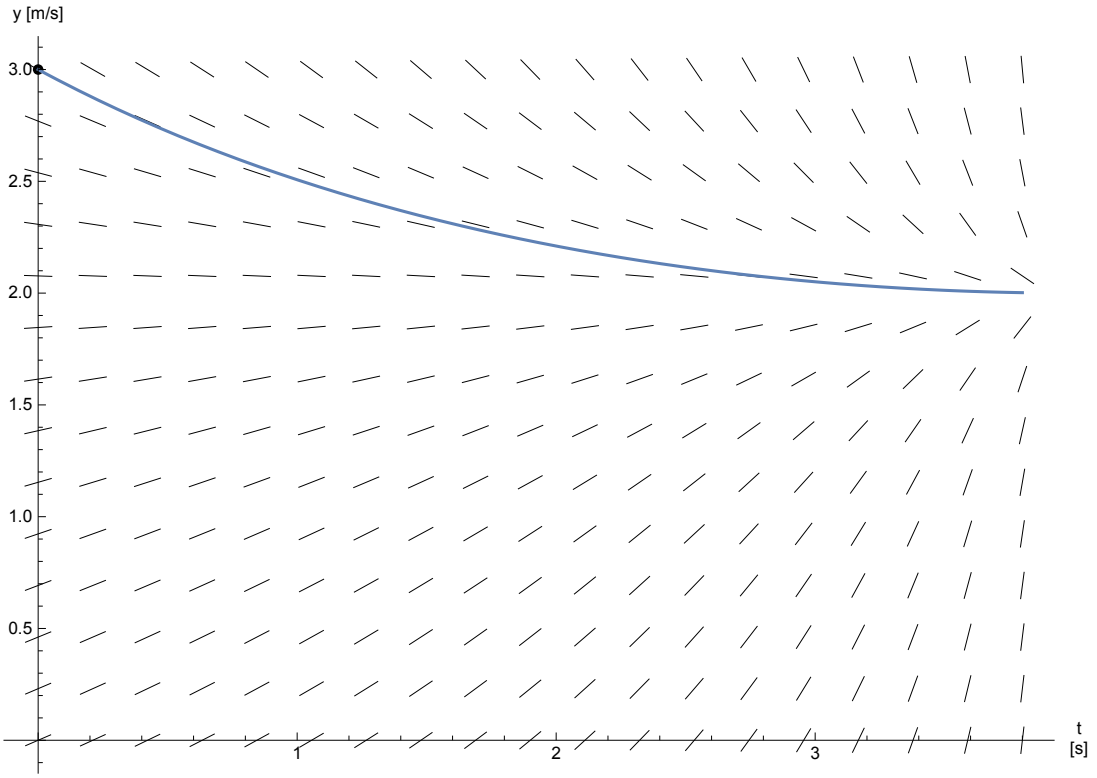
Le but est de construire une solution graphique de l'équation différentielle qui passe par le point (t_0, y_0). Il existe une et une seule courbe $t \mapsto y(t)$ qui passe par le point donné et suit le champ de pentes, c'est-à-dire l'équation différentielle avec condition initiale possède une et une seule solution (voir § 1.1)

```
t0 = 0; y0 = 0;
```



Pour montrer comment la solution dépend de la condition initiale, modifions-la.

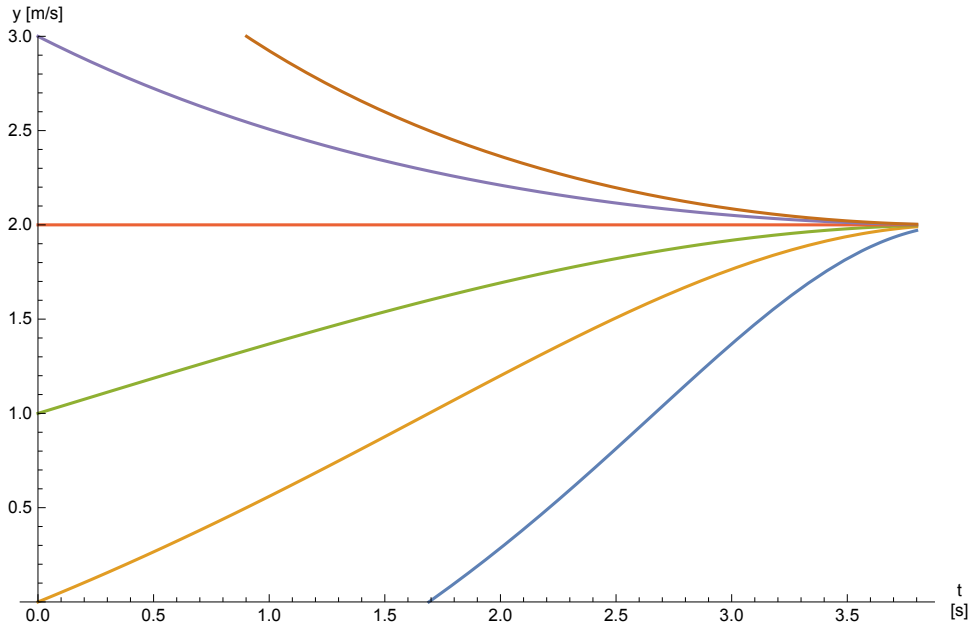
$$\begin{cases} y' = f(t, y) \\ y(0) = 3 \end{cases}$$



On appelle **solution générale** l'ensemble des solutions de l'équation différentielle sans condition

initiale

$$y' = f(t, y)$$



La solution générale est une famille de fonctions. Poser une condition initiale équivaut à choisir une fonction de cette famille.

Méthode numérique d'Euler

Dans la méthode graphique, pour tracer la courbe, on part du point donné par la condition initiale, puis on trace la courbe de proche en proche en suivant les directions données par le champ de pentes.

Leonhard Euler est né à Bâle en 1707. Il reçut un enseignement mathématique dans la famille Bernoulli. Il mourut à Saint-Petersburg en 1783. Dans la méthode d'Euler, en partant d'un point et de la pente en ce point, on progresse un petit bout en ligne droite. On obtient ainsi un nouveau point à partir duquel on recommence l'opération. Finalement, on obtient une fonction affine par morceaux qui est une approximation de la solution cherchée.

Le point de départ est $(t_0,\ y_0)$. En ce point, la pente est $y'(t_0) = f(t_0,\ y_0)$. On va effectuer "un petit pas" dans la direction $\begin{pmatrix} 1 \\ f(t_0,\ y_0) \end{pmatrix}$.

On s'intéresse à la situation au temps $t_1 = t_0 + h$ où h est un petit intervalle de temps donné. Effectuons le déplacement $h\begin{pmatrix} 1 \\ f(t_0,\ y_0) \end{pmatrix}$. La nouvelle position $(t_1,\ y_1)$ sera telle que

$$\begin{pmatrix} t_1 - t_0 \\ y_1 - y_0 \end{pmatrix} = h \begin{pmatrix} 1 \\ f(t_0, y_0) \end{pmatrix}$$

$$(t_1, y_1) = (t_0 + h, y_0 + h\, f(t_0, y_0))$$

En répétant l'opération (voir *Formulaires et tables*)

$$(t_2, y_2) = (t_1 + h, y_1 + h\, f(t_1, y_1))$$

$$\boxed{(t_{k+1}, y_{k+1}) = (t_k + h, y_k + h\, f(t_k, y_k))} \qquad k \in \mathbb{N}$$

```
n = 16;
h = (tmax - t0)/n;

euler[{t_, y_}] := {t + h, y + h f[t, y]}
```

Solution numérique d'Euler

```
solne = NestList[euler, {t0, y0}, n]
```

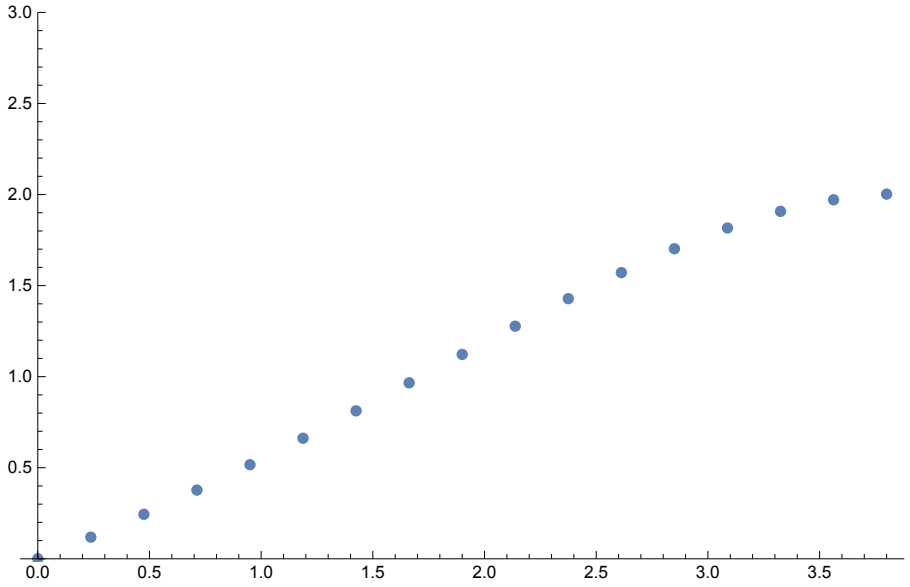
liste d'imbrication

```
{{0, 0.}, {0.2375, 0.11875}, {0.475, 0.244551}, {0.7125, 0.377288}, {0.95, 0.516633},
 {1.1875, 0.661978}, {1.425, 0.812365}, {1.6625, 0.966397}, {1.9, 1.12216},
 {2.1375, 1.27714}, {2.375, 1.42818}, {2.6125, 1.57143}, {2.85, 1.70243},
 {3.0875, 1.8162}, {3.325, 1.90748}, {3.5625, 1.97108}, {3.8, 2.00225}}
```

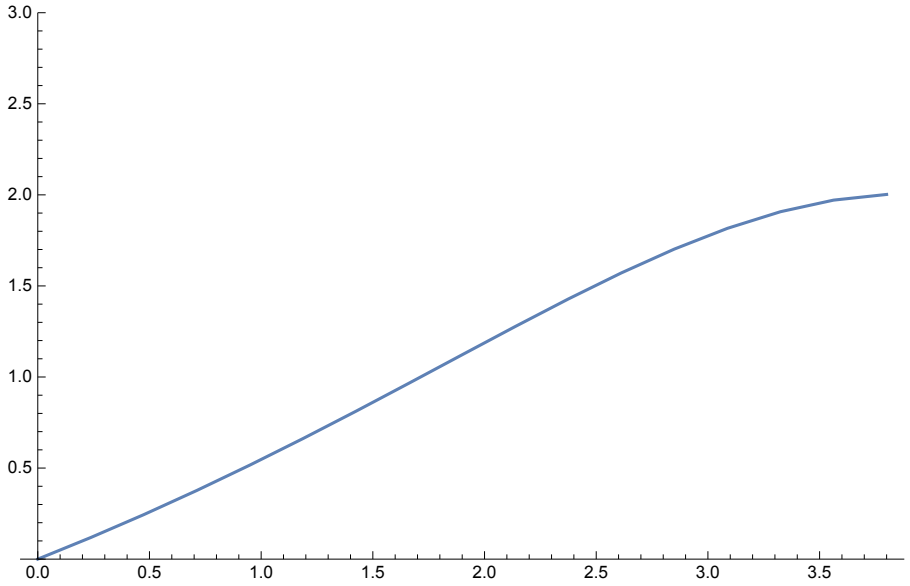
Graphique de l'approximation d'Euler (fonction affine par morceaux)

```
ListPlot[solne, PlotRange → {ymin, ymax}, ImageSize → {500, 300}]
```

tracé de liste | zone de tracé | taille d'image



```
ListLinePlot[solne, PlotRange → {ymin, ymax}, ImageSize → {500, 300}]
```
tracé de liste de ligne zone de tracé taille d'image

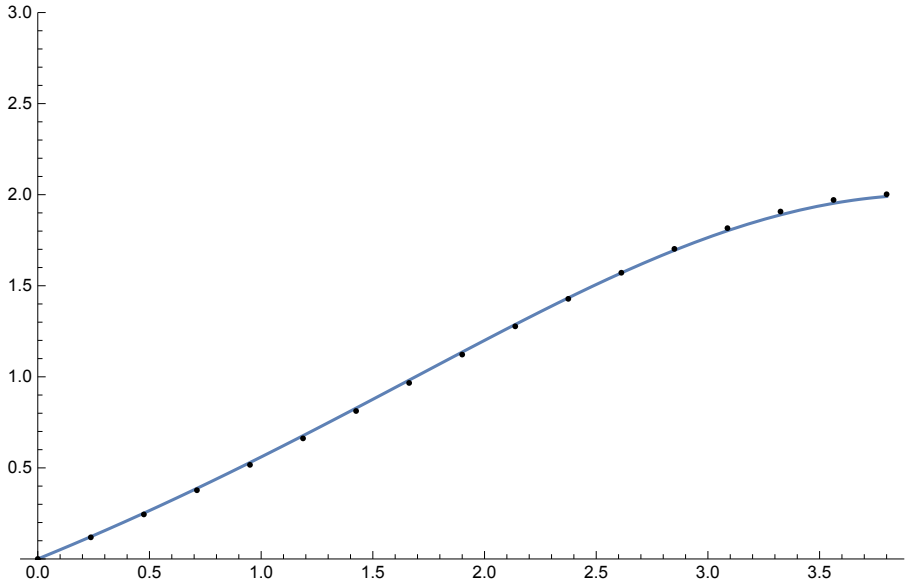


Comparons la solution exacte (§ 1.1) et la solution numérique approchée, graphiquement et numériquement

```
Clear[v, t];
```
efface

$$v[t\_] := 2\,\frac{-t^2+8\,t}{t^2-8\,t+32};$$

```
Plot[v[t], {t, tmin, tmax}, PlotRange → {ymin, ymax},
 ImageSize → {500, 300}, Epilog → Map[Point, solne]]
```
tracé de courbes zone de tracé taille d'image épilogue app· point



Liste des erreurs : (approximation d'Euler en t_i) – (valeur exacte en t_i)

```
{abscEuler, ordEuler} = Transpose[solne];
```
transposée

```
ordExact = Map[v, abscEuler];
           applique
```

```
ordErreur = ordEuler - ordExact
```

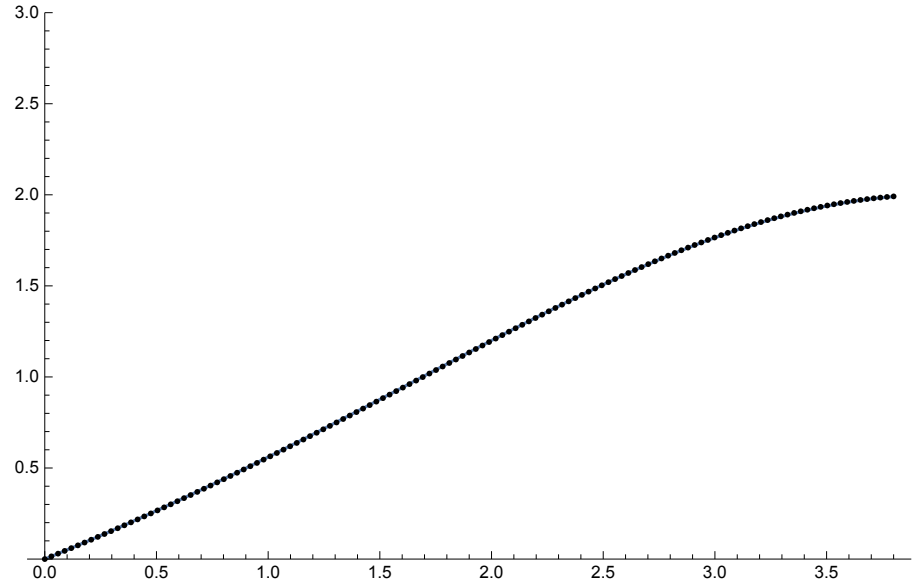
```
{0., -0.0035188, -0.00693885, -0.0100898, -0.0127616, -0.0147083,
 -0.0156612, -0.0153529, -0.0135575, -0.0101495, -0.00518024,
 0.00103241, 0.0078138, 0.0140631, 0.0182308, 0.0183661, 0.0122269}
```

Si on prend une plus petite valeur de *h*, l'erreur tend à diminuer

```
n = 128;
h = (tmax - t0)/n;
```

```
solne = NestList[euler, {t0, y0}, n];
        liste d'imbrication
```

```
Plot[v[t], {t, tmin, tmax}, PlotRange → {ymin, ymax},
tracé de courbes                zone de tracé
 ImageSize → {500, 300}, Epilog → Map[Point, solne]]
 taille d'image                 épilogue  app·  point
```



Liste des erreurs : (approximation d'Euler en t_i) – (valeur exacte en t_i)

```
{abscEuler, ordEuler} = Transpose[solne];
                        transposée
```

```
ordExact = Map[v, abscEuler];
           applique
```

```
ordErreur = ordEuler - ordExact
```

```
{0., -0.0000550827, -0.000110144, -0.000165152, -0.000220074, -0.000274876,
 -0.000329523, -0.000383979, -0.000438207, -0.000492168, -0.000545823, -0.000599131,
 -0.00065205, -0.000704537, -0.000756547, -0.000808035, -0.000858956, -0.00090926,
 -0.000958899, -0.00100782, -0.00105598, -0.00110332, -0.00114979, -0.00119534,
 -0.0012399, -0.00128343, -0.00132587, -0.00136716, -0.00140723, -0.00144604,
 -0.00148352, -0.00151961, -0.00155425, -0.00158738, -0.00161893, -0.00164885,
 -0.00167707, -0.00170353, -0.00172816, -0.00175091, -0.00177171, -0.0017905,
 -0.00180722, -0.00182181, -0.0018342, -0.00184435, -0.00185218, -0.00185765,
 -0.0018607, -0.00186128, -0.00185933, -0.00185481, -0.00184766, -0.00183785,
 -0.00182533, -0.00181006, -0.001792, -0.00177113, -0.00174741, -0.00172082,
 -0.00169134, -0.00165895, -0.00162364, -0.0015854, -0.00154423, -0.00150013,
 -0.00145311, -0.0014032, -0.0013504, -0.00129475, -0.00123628, -0.00117504,
 -0.00111107, -0.00104444, -0.000975203, -0.000903439, -0.000829227, -0.000752659,
 -0.000673835, -0.000592863, -0.00050986, -0.000424955, -0.000338285, -0.000249996,
 -0.000160245, -0.0000691998, 0.0000229629, 0.000116056, 0.000209881, 0.00030423,
 0.000398885, 0.000493617, 0.000588188, 0.000682348, 0.000775838, 0.00086839,
 0.000959725, 0.00104955, 0.00113758, 0.0012235, 0.001307, 0.00138775, 0.00146542,
 0.00153968, 0.00161018, 0.00167656, 0.00173848, 0.00179557, 0.00184747, 0.00189379,
 0.00193417, 0.00196823, 0.00199559, 0.00201587, 0.00202869, 0.00203367,
 0.00203043, 0.00201859, 0.00199778, 0.00196764, 0.00192778, 0.00187786,
 0.00181752, 0.00174642, 0.0016642, 0.00157056, 0.00146515, 0.00134768, 0.00121785}
```

Lorsque h tend vers 0 c'est-à-dire pour $n \to \infty$, la méthode numérique d'Euler converge vers la solution exacte.

Autres méthodes numériques

Il existe des méthodes numériques plus performantes que la méthode d'Euler, mais leur étude n'est pas abordée ici. Sachez simplement que *Mathematica* fait appel à des méthodes plus élaborées qui convergent plus vite vers la solution et donnent des réponses plus précises.

Calcul numérique avec Mathematica

Soit à résoudre l'équation différentielle ordinaire du premier ordre avec condition initiale

```
? NDSolve
```

```
NDSolve[eqns, y, {x, xmin, xmax}] trouve une solution numérique aux équations
  différentielles ordinaires eqns pour la fonction y avec la variable indépendante x
  de rang xmin à xmax. NDSolve[eqns, y, {x, xmin, xmax}, {t, tmin, tmax}] trouve une
  solution numérique aux équations aux dérivées partielles eqns. NDSolve[eqns, {y1,
  y2, ... }, {x, xmin, xmax}] trouve des solutions numériques pour les fonctions  yi.
```

```
soln = NDSolve[{y'[t] == f[t, y[t]], y[t0] == y0}, y, {t, tmin, tmax}]
```

résolveur numérique d'équations différentielles

```
{{y → InterpolatingFunction[ Domain: {{0., 3.8}}
                             Output: scalar ]}}
```

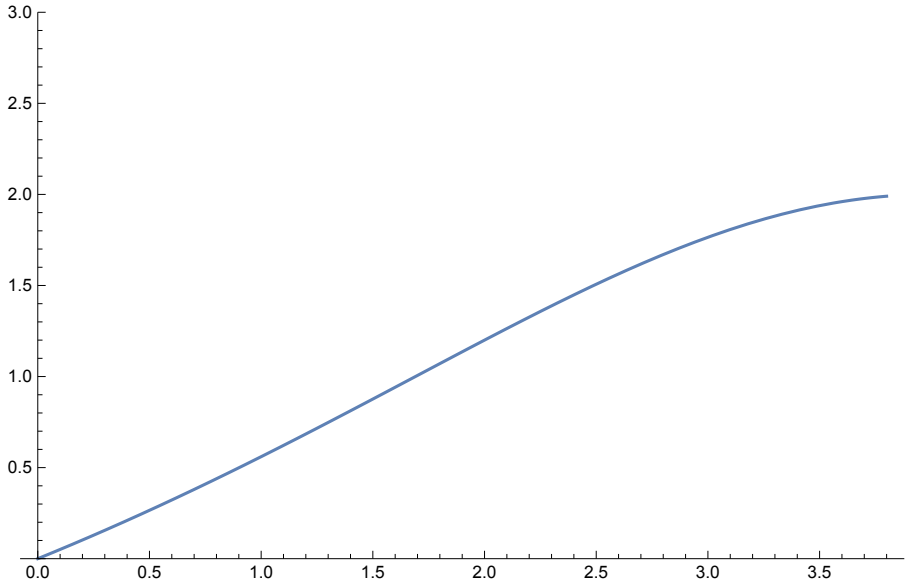
Dans un premier temps, la méthode numérique produit une liste finie d'approximations de la forme $\{t_i,\ y_i\}$. Les valeurs intermédiaires sont ensuite calculées par interpolation.

```
Clear[sn];
```
efface

```
sn = y /. soln[[1]]
```

InterpolatingFunction[Domain: {{0., 3.8}} Output: scalar]

```
Plot[sn[t], {t, tmin, tmax}, PlotRange → {ymin, ymax}, ImageSize → {500, 300}]
```
tracé de courbes | zone de tracé | taille d'image



Estimation numérique de l'erreur maximale:

```
Max[Table[Abs[sn[t] - v[t]], {t, tmin, tmax, (tmax - tmin)/100}]]
```
ma·· | table | valeur absolue

1.79301×10^{-7}

Travaux dirigés du § 1.2

1.2 - TD 1

On donne l'équation différentielle avec condition initiale

$$\begin{cases} y' = 1 + (y - t)^2 \\ y(0) = \frac{1}{2} \end{cases}$$

a) Résolution graphique

Dans le rectangle $[0;\ 1] \times [0;\ 2.5]$, calculez des valeurs numériques de la fonction

$$(t, y) \mapsto 1 + (y - t)^2$$

avec *Mathematica* puis, à la main, représentez le champ de pentes.

Tracez la solution $t \mapsto y(t)$ qui vérifie la condition initiale.

b) Méthode d'Euler sans ordinateur

Calculez une solution numérique approchée de l'équation différentielle par la méthode d'Euler sur l'intervalle $[0;\ 1]$, avec le pas $h = 0.1$

c) Méthode d'Euler avec ordinateur

Calculez une solution numérique approchée de l'équation différentielle par la méthode d'Euler sur l'intervalle [0; 1], avec le pas $h = 0.01$.

d) Résolution numérique avec *Mathematica*

Calculez une solution numérique approchée de l'équation différentielle sur l'intervalle [0; 1] avec la méthode numérique de *Mathematica*.

e) Vérification

Montrez que la fonction

$$y(t) = t + \frac{1}{2 - t}$$

est la solution de l'équation différentielle avec condition initiale.

f) Comparaison

Comparez les cinq solutions qui apparaissent sous les points *a*, *b*, *c*, *d* et *e*.

g) Calcul de l'erreur d'approximation

Pour les solutions *c* et *d*, calculez l'erreur d'approximation aux abscisses $t = 0,\ 0.05,\ 0.1,\ 0.15,\ ...,\ 0.95,\ 1$.

1.2 - TD 2

a) Méthode de Heun [avec *Mathematica*]

Pour résoudre les équations différentielles, une autre méthode numérique est décrite dans les *Formulaires et tables*: la méthode de **Heun**. Utilisez cette méthode pour résoudre sur l'intervalle [0, 2] l'équation suivante avec condition initiale

$$\begin{cases} y' = \frac{4t}{y+1} \\ y(0) = 1 \end{cases}$$

b) Méthode Runge-Kutta 4 [facultatif]

Pour résoudre les équations différentielles, une autre méthode numérique est décrite dans les *Formulaires et tables*: la méthode **Runge-Kutta 4**. Utilisez cette méthode pour résoudre l'équation différentielle précédente sur l'intervalle [0, 2].

1.2 - TD 3 [facultatif]

Champ de pentes [avec *Mathematica*]

Dessinez le champ de pentes pour résoudre graphiquement l'équation suivante sur l'intervalle [0, 2]

$$\begin{cases} y' = \frac{4t}{y+1} \\ y(0) = 1 \end{cases}$$

Indication: observez comment ont été réalisés les graphiques du support de cours:

https://www.deleze.name/marcel/sec2/applmaths/csud/eq-differentielles/1-2_EQ-DIFFERENTIELLES.nb

puis ouvrez la cellule qui précède le premier champ de pentes dessiné.

Liens

Vers les corrigés des exercices du § 1.2
https://www.deleze.name/marcel/sec2/applmaths/csud/corriges/eq-differentielles/1-2-equadiff-cor.pdf

Vers la page mère : Applications des mathématiques
https://www.deleze.name/marcel/sec2/applmaths/csud/index.html