```
begin
  using Pkg
  Pkg.activate(Base.Filesystem.homedir() * "/.config/julia/projects/oft")
  using Plots
  using PlutoUI
  using TikzPictures
  using LaTeXStrings
  using SparseArrays
end
```

Dans ce chapitre, on se concentre sur la résolution d'une EDO qui a forme

$$-(a(t)u'(t))' - c(t)u(t) = f(t),\ t \in (0,1),\ \text{où}$$

les fonction $a, c, f : [0,1] \to \mathbb{R}$ sont connue et que l'on veut résoudre $u : [0,1] \to \mathbb{R}$ avec $u(0) = u(1) = 0$.

En plus, on suppose qu'il existe une constante $a_0 > 0$ telle que $a(t) > a_0$ et que $c(t) \geq 0$ pour tout $t \in [0,1]$.

```
md"""
Dans ce chapitre, on se concentre sur la résolution d'une EDO qui a forme

```math
-(a(t)u'(t))' - c(t)u(t) = f(t),\; t \in (0, 1),\; \text{où}
```

les fonction $a, c, f: [0,1] \to \mathbb{R}$ sont connue et que l'on veut
résoudre $u: [0,1] \to \mathbb{R}$ avec $u(0) = u(1) = 0\,.$

En plus, on suppose qu'il existe une constante $\,a_0 > 0\,$ telle que $\,a(t) > a_0\,$ et que $\,c(t) \ge 0\,$
pour tout $\,t \in [0,1]\,.$

"""
```

(?) How to type the two diff a's in $LaTeX$?

Différence Finie

Nous avons les approximations suivantes

$$u'(t) = \frac{u(t) - u(t-h)}{h} + o(h)\,,$$
$$u''(t) = \frac{u(t-h) - 2u(t) + u(t+h)}{h^2} + o(h^2)\,.$$

Et on introduit quelques notations.

- D'abord, on va diviser l'intervalle $[0,1]$ en $M-1$ morceaux, comme dans la figure ci-dessous. La longueur de chaque morceau est égale à $h = \frac{1}{M-1}$.

- Puis on pose
  - $t_j = jh$ où $j = 0, 1, 2, \ldots, M-1$
  - $u_j = u(t_j)$
  - $f_j = f(t_j)$, etc.

CHALLENGE 23.1.

Déduisez une équation la plus simple possible que vous arrivez à la rendre, à partir de

$$M = 6,\ a(t) = 1,\ c(t) = 0\,.$$

(?) How to convert the whole markdown string to some, say, red, color?

Solution 23.1.

L'équation se simplifie comme

$$-u''(t) = f(t) \ \text{ sur } \ (0,1)\,.$$

Si l'on substitue l'approximation par différences finies aux dérivées ci-dessus, on aura

$$-\frac{u(t-h) - 2u(t) + u(t+h)}{h^2} \approx f(t) \ \text{ sur } \ (0,1)\,.$$

En réécrivant cette équation pour chacun entre $t_1, t_2, t_3, \ldots, t_{M-2}$, on obtient

$$\begin{aligned} -(u_0 - 2u_1 + u_2) &= h^2 f_1 \\ -(u_1 - 2u_2 + u_3) &= h^2 f_2 \\ &\cdots \\ -(u_{M-1} - 2u_{M-2} + u_{M-1}) &= h^2 f_{M-2}\,. \end{aligned}$$

En écriture matricielle, cela donne

$$-\underbrace{\begin{pmatrix} 1 & -2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & & & \ddots & & & & \vdots \\ 0 & & \cdots & & 0 & 1 & -2 & 1 \end{pmatrix}}_{\text{de taille } (M-2,\, M)} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{M-1} \end{pmatrix} = h^2 \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{M-2} \end{pmatrix}.$$

Comme nous savons

$$\begin{aligned} u_0 &= u(0) = 0 \\ u_{M-1} &= u(1) = 0\,, \end{aligned}$$

le produit matrice-vecteur ci-dessus peut encore se simplifier comme

$$\underbrace{\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & 0 & \cdots & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & \cdots & & 0 & -1 & 2 & -1 \\ 0 & \cdots & & 0 & 0 & -1 & 2 \end{pmatrix}}_{\text{de taille } (M-2,\, M-2)} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{M-2} \end{pmatrix} = h^2 \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{M-2} \end{pmatrix}.$$

On remarque que cela a forme $Au = g$, où A est une matrice **tridiagonale**.

Au cas particulier où $M = 6$, on a

$$\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \frac{1}{25} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix}.$$

```
• Enter cell code...
```

CHALLENGE 23.2.

Le fichier `finitediff1.m` peut se télécharger du site web de l'auteur. Le code Matlab dedans est **sans faute**, même s'il est un peu difficile à confirmer sa justesse, surtout pour ceux qui n'utilisent pas souvent Matlab/Octave. (Ces notes sont rédigées en 2021 où les langues les plus populaires pour faire de la Scientific Computing, ce sont Python et Julia.)

Pour faciliter sa lecture, j'ai ajouté des commentaires dans `finitediff1.m`.

De plus, ce code est d'autant plus difficile parce que, même si l'auteur ne le dit pas, on traite l'EDO dans sa complétude, non pas comme dans CHALLENGE 23.1 où on simplifie en prenant $M = 6, a(t) = 1, c(t) = 0$.

Mais, en fait, ce n'est pas si difficile que ça. L'équation en sa plus grande généralité donne cela en différences finies :

$$-a'_j \frac{u_j - u_{j-1}}{h} - a_j \frac{u_{j-1} - 2u_j + u_{j+1}}{h^2} + c_j u_j = f_j\,, \quad \forall\, j = 1, 2, \ldots, M-2$$

$$\Longleftrightarrow$$

$$\left(\frac{a'_j}{h} - \frac{a_j}{h^2}\right) u_{j-1} + \left(-\frac{a'_j}{h} + 2\frac{a_j}{h^2} + c_j\right) u_j + \left(-\frac{a_j}{h^2}\right) u_{j+1} = f_j\,, \quad \forall\, j = 1, 2, \ldots, M-2$$

$$\Longleftrightarrow$$

$$\begin{pmatrix}
-\frac{a'_1}{h} + 2\frac{a_1}{h^2} + c_1 & -\frac{a_1}{h^2} & \bullet & \bullet & & \\
\frac{a'_2}{h} - \frac{a_2}{h^2} & -\frac{a'_2}{h} + 2\frac{a_2}{h^2} + c_2 & -\frac{a_2}{h^2} & \bullet & & \\
\bullet & \frac{a'_3}{h} - \frac{a_3}{h^2} & -\frac{a'_3}{h} + 2\frac{a_3}{h^2} + c_3 & -\frac{a_3}{h^2} & & \\
 & & & \ddots & & \\
 & & & & \frac{a'_{M-3}}{h} - \frac{a_{M-3}}{h^2} & -\frac{a'_{M-3}}{h} + \\
 & & & & \bullet & \frac{a'_M}{h}
\end{pmatrix}$$

Rmq. J'utilise le `\bullet` (i.e. ●) pour représenter 0 dans la grande matrice ci-dessus.

(?) Allez voir comment la fonction `spdiags` est implémentée en Matlab/Octave. C'est un peu curieux que tous les trois diagonales (`input args`) doivent avoir les mêmes longueurs (padded by zeros for sub/sup-diagonl).

(?) Pourquoi parfois `$(HTML("<br>"))` ne fonctionne pas? Que faire?

`finitediff1.jl`

Ensuite, on va coder `finitediff1.m` en Julia.

En Julia, il n'y a pas de `linspace`. Par contre, on a le remplacement suivant.

```
0.0:0.2:1.0
```

```
let
  M = 6
  0:1/(M-1):1
end
```

```
xyz = 123
```

```
xyz = zyx = 123
```

123

```
zyx
```

- **spdigm (sparse diagonal matrix)**

```
Enter cell code...
```

finitediff1 (generic function with 1 method)

```
function finitediff1(M::Number, a::Function, c::Function, f::Function)
  """
  `a, c, f` are functions whose input is a vector and output is a vector.
  Accurately speaking, `a` returns two vectors, the first one `a` itself,
  the second one its derivative.


  TODO:
  01. Use @view

  """
  t = 0:1/(M-1):1  # This cuts [0, 1] into `M-1` pieces
  h = t[2]         # same as h = 1 / (M-1) but save the work to recompute
  #h2inv = 1 / h^2
  #n = M - 2
  tmesh = t[2:end-1]
  a0_and_a1 = a(tmesh)  # a0: 0th derivative, a1: 1st derivative
  a0 = @view a0_and_a1[1:end, 1]
  a1 = @view a0_and_a1[1:end, 2]
  a1_over_h = a1 ./ h
  a0_over_h² = a0 ./ h^2
  c0 = c.(tmesh)
  #g = f0 = f(tmesh)
  g = f.(tmesh)
  diag = -a1_over_h + 2 .* a0_over_h² + c0
  ldiag = (a1_over_h - a0_over_h²)[2:end]
  udiag = - @view a0_over_h²[1:end-1]
  A = spdiagm(-1 => ldiag, 0 => diag, 1 => udiag)
  # A * ucomp = g
  ucomp = A \ g
end
```

[6.12323e-17, 1.0]

```
(t -> [cos(t), sin(t)])(π/2)
```

[[1.0, 0.0], [6.12323e-17, 1.0], [-1.0, 1.22465e-16], [-1.83697e-16, -1.0], [1.0, -

```
(t -> [cos(t), sin(t)]).(0:π/2:2π)
```

Array{Array{Float64,1},1}

```
typeof((t -> [cos(t), sin(t)]).(0:π/2:2π))
```

```
Function
```

```
Function
```

```
[1×2 Array{Float64,2}:, 1×2 Array{Float64,2}:, 1×2 Array{Float64,2}:, 1×2 Array{Float64
  1.0  0.0               6.12323e-17  1.0       -1.0  1.22465e-16       -1.83697e-16  -1
```

```
(t -> [cos(t) sin(t)]).(0:π/2:2π)
```

```
Array{Array{Float64,2},1}
```

```
typeof((t -> [cos(t) sin(t)]).(0:π/2:2π))
```

```
5×2 Array{Float64,2}:
  1.0           0.0
  6.12323e-17   1.0
 -1.0           1.22465e-16
 -1.83697e-16  -1.0
  1.0          -2.44929e-16
```

```
(t -> [cos.(t) sin.(t)])(0:π/2:2π)  # This is what we are looking for! for 'a'
```

```
1×2 Array{Float64,2}:
 1.0  0.0
```

```
(t -> [cos.(t) sin.(t)])(0)  # double-check it works on a single number
```

```
6.123233995736766e-17
```

```
let
  a0, a1 = (t -> [cos.(t) sin.(t)])(0:π/2:2π)
  a1
end
```

(?) Is there a way in Julia to do what the above cell tries to do?

```
(2, 3)
```

```
size([1 2 3; 4 5 6])
```

```
(1×2 Array{Float64,2}:, 2×6 Array{Float64,2}:        )
  1.0  0.0               1.0  1.0  1.0  0.0  0.0  0.0
                         1.0  1.0  1.0  0.0  0.0  0.0
```

```
let
  function a(t)
    [ones(size(t)) zeros(size(t))]
  end
  a(3), a([1 2 3; 4 5 6])
end
```

Prenons un exemple quelconque pour tester notre implémentation de `finitediff1`. Disons,

$$u(t) = \sin t$$
$$a(t) = 1$$
$$c(t) = 0$$

ce qui entraîne

$$f(t) = -\frac{d^2u}{dt^2} = \sin t\,.$$

```
[0.0304766,  0.0530064,  0.0599595,  0.0443269]
```

```
let
  function a(t)
    [ones(size(t)) zeros(size(t))]
  end
  M = 6
  c(t) = 0
  f(t) = sin(t)
  # Posons u(t) = sin(t) ⇒  f(t) = -sin(t).
  # Tous les deux lignes suivantes marchent.
  finitediff1(M, a, t -> 0, t -> sin(t))
  #finitediff1(M, a, c, f)
end
```

```
[0.198669,  0.389418,  0.564642,  0.717356]
```

```
# Vérifions la solutions ci-dessus
[sin(t) for t = 1/5:1/5:4/5]
```

```
0.2:0.2:0.8
```

```
1/5:1/5:4/5
```

(?) Vous vous rendez compte d'où votre **"bug"** est?

(R) Votre solution u ne satisfait pas les boundary conditions. En effet,
$u(t) = \sin t \implies u(0) = 0$ et $u(1) = 0.8414709848078965$

Changeons la en

$$u(t) = \sin(\pi t)\,.$$

Ceci implique

$$f(t) = \pi^2 \sin(\pi t)\,.$$

```
[0.60751,  0.982972,  0.982972,  0.60751]
```

```
let
  function a(t)
    [ones(size(t)) zeros(size(t))]
  end
```

```
  M = 6
  finitediff1(M, a, t -> 0, t -> π^2 * sin(π*t))
end
```

```
[0.587785, 0.951057, 0.951057, 0.587785]
```

```
# Vérifions la solutions ci-dessus
[sin(π*t) for t = 1/5:1/5:4/5]
```