

# Méthodes Numériques I

## Algorithmique numérique

François Cuvelier

Laboratoire d'Analyse Géométrie et Applications  
Institut Galilée  
Université Paris XIII.

2 avril 2015

# Plan

## 1 Polynômes d'interpolation de Lagrange

## 2 Dérivation numérique

## 3 Intégration numérique

- Méthodes simplistes
- Méthodes de Newton-Cotes
- Méthodes composites
- Autres méthodes
- Intégrales multiples

## 4 Résolution de systèmes linéaires

- Vecteurs
- Matrices
- Factorisation LU
- Factorisation de Cholesky

# Définition

## Définition

Soient  $n \in \mathbb{N}^*$  et  $(x_i, y_i)_{i \in \llbracket 0, n \rrbracket}$  avec  $(x_i, y_i) \in \mathbb{R}^2$  et les  $x_i$  distincts deux à deux. Le **polynôme d'interpolation de Lagrange** associé aux  $n + 1$  points  $(x_i, y_i)_{i \in \llbracket 0, n \rrbracket}$ , noté  $\mathcal{P}_n$ , est donné par

$$\mathcal{P}_n(t) = \sum_{i=0}^n y_i L_i(t), \quad \forall t \in \mathbb{R} \quad (1)$$

avec

$$L_i(t) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{t - x_j}{x_i - x_j}, \quad \forall i \in \llbracket 0, n \rrbracket, \quad \forall t \in \mathbb{R}. \quad (2)$$

# Théorème

## Théorème

Le **polynôme d'interpolation de Lagrange**,  $\mathcal{P}_n$ , associé aux  $n + 1$  points  $(x_i, y_i)_{i \in \llbracket 0, n \rrbracket}$ , est l'unique polynôme de degré au plus  $n$ , vérifiant

$$\mathcal{P}_n(x_i) = y_i, \quad \forall i \in \llbracket 0, n \rrbracket. \quad (3)$$

## Exemple

A titre d'exemple, on représente, En figure 1, le polynôme d'interpolation de Lagrange associé à 7 points donnés.

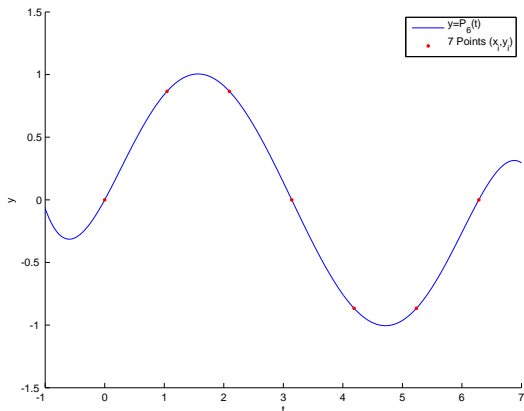


FIGURE : Polynôme d'interpolation de Lagrange avec 7 points donnés)

# Exercices

## Exercice

*Ecrire la fonction LAGRANGE permettant de calculer  $\mathcal{P}_n$  (polynôme d'interpolation de Lagrange associé aux  $n + 1$  points  $(x_i, y_i)_{i \in \llbracket 0, n \rrbracket}$ ) au point  $x \in \mathbb{R}$ .*

## Exercice

*Soit  $\mathcal{P}_n$  le polynôme d'interpolation de Lagrange associé aux  $n + 1$  points  $(x_i, y_i)_{i \in \llbracket 0, n \rrbracket}$  et  $\mathbf{X}$  un vecteur de  $\mathbb{R}^m$ .*

- 1 *Ecrire la fonction LAGRANGEVEC permettant de calculer le vecteur  $\mathbf{Y} \in \mathbb{R}^m$  tel que*

$$Y_i = \mathcal{P}_n(X_i), \quad \forall i \in \llbracket 1, m \rrbracket.$$

- 2 *Evaluer le coût arithmétique de la fonction LAGRANGEVEC en fonction de  $n$  et  $m$ .*

# Exercices

## Exercice

Soient  $n \in \mathbb{N}^*$  et  $(a, b) \in \mathbb{R}^2$  avec  $a < b$ . On note  $\mathbf{X} = (X_1, \dots, X_{n+1})$  la discrétisation régulière de l'intervalle  $[a, b]$  avec  $n + 1$  points et  $\mathbf{Y} \in \mathbb{R}^{n+1}$  tel que  $Y_i = f(X_i)$ ,  $\forall i \in \llbracket 1, n + 1 \rrbracket$ .

Écrire un programme permettant de représenter graphiquement  $f$  et  $\mathcal{P}_n$  (polynôme d'interpolation de Lagrange associé aux  $n + 1$  points  $(X_i, Y_i)_{i \in \llbracket 0, n \rrbracket}$ ) sur l'intervalle  $[a, b]$ .

On utilisera pour cela la fonction `PLOT` dont la syntaxe est `PLOT(x, y)` où  $x$  et  $y$  sont des vecteurs de  $\mathbb{R}^k$ . Cette fonction relie successivement les points  $(x(j), y(j))$ , pour  $j$  allant de 1 à  $k$ , par des segments.

# Erreur de l'interpolation

Soit une fonction  $f : [a, b] \longrightarrow \mathbb{R}$ . On suppose que les  $y_i$  sont donnés par

$$y_i = f(x_i), \quad \forall i \in \llbracket 0, n \rrbracket. \quad (4)$$

On cherche à évaluer l'erreur  $E_n(t) = f(t) - \mathcal{P}_n(t)$ .

# Erreur de l'interpolation

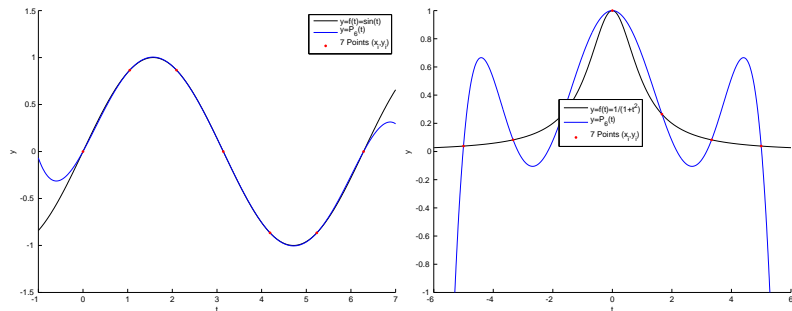


FIGURE : Erreurs d'interpolation avec  $n = 6$

# Erreur de l'interpolation

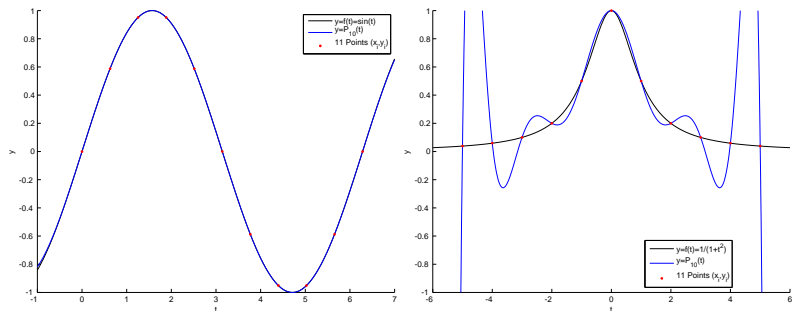


FIGURE : Erreurs d'interpolation avec  $n = 10$

# Erreur de l'interpolation

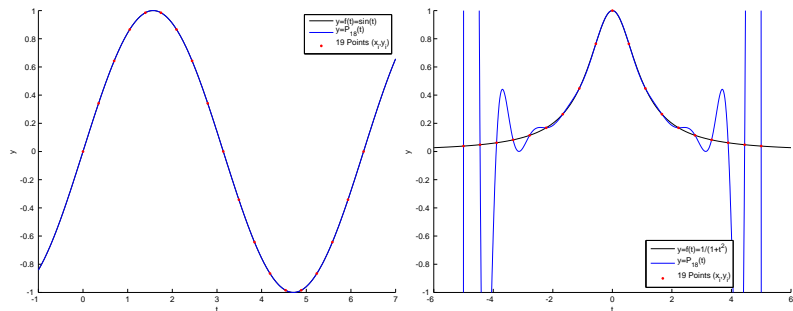


FIGURE : Erreurs d'interpolation avec  $n = 18$

# Erreur de l'interpolation

## Théorème (Cauchy, 1840)

Soit  $f : [a, b] \rightarrow \mathbb{R}$ , une fonction  $(n + 1)$ -fois différentiable et soit  $\mathcal{P}_n(t)$  le polynôme d'interpolation de degré  $n$  passant par

$(x_i, f(x_i)), \forall i \in \llbracket 0, n \rrbracket$ . Alors,  $\forall t \in [a, b]$ ,

$\exists \xi_t \in (\min_{i \in \llbracket 0, n \rrbracket} (x_i, t), \max_{i \in \llbracket 0, n \rrbracket} (x_i, t))$ ,

$$f(t) - \mathcal{P}_n(t) = \frac{f^{(n+1)}(\xi_t)}{(n+1)!} \prod_{i=0}^n (t - x_i) \quad (5)$$

# Points de Chebyshev

Pour minimiser l'erreur commise lors de l'interpolation d'une fonction  $f$  par un polynôme d'interpolation de Lagrange, on peut, pour un  $n$  donné, "jouer" sur le choix des points  $x_i$  :

Trouver  $(\bar{x}_i)_{i=0}^n$ ,  $x_i \in [a, b]$ , distincts deux à deux, tels que

$$\max_{t \in [a, b]} \prod_{i=0}^n |t - \bar{x}_i| \leq \max_{t \in [a, b]} \prod_{i=0}^n |t - x_i|, \quad \forall (x_i)_{i=0}^n, \quad x_i \in [a, b], \text{ distincts 2 à 2} \quad (6)$$

# Points de Chebyshev

## Théorème

*Les points réalisant (6) sont les points de Chebyshev donnés par*

$$\bar{x}_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(2i+1)\pi}{2n+2}\right), \quad \forall i \in \llbracket 0, n \rrbracket. \quad (7)$$

# Points de Chebyshev

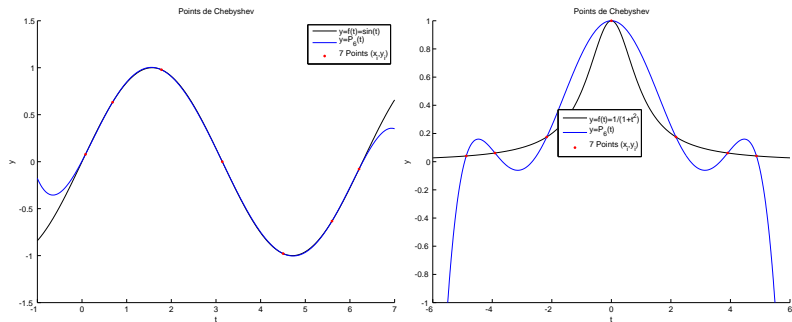


FIGURE : Erreurs d'interpolation avec  $n = 6$

# Points de Chebyshev

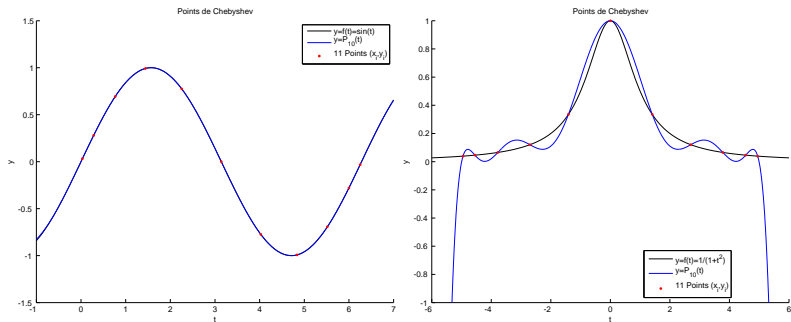


FIGURE : Erreurs d'interpolation avec  $n = 10$

# Points de Chebyshev

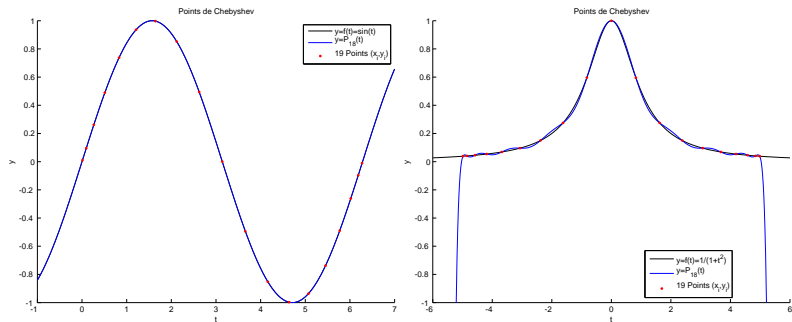


FIGURE : Erreurs d'interpolation avec  $n = 18$

# Plan

## 1 Polynômes d'interpolation de Lagrange

## 2 Dérivation numérique

## 3 Intégration numérique

- Méthodes simplistes
- Méthodes de Newton-Cotes
- Méthodes composites
- Autres méthodes
- Intégrales multiples

## 4 Résolution de systèmes linéaires

- Vecteurs
- Matrices
- Factorisation LU
- Factorisation de Cholesky

# Dérivée

On propose de chercher une approximation de la dérivée première de  $f$  en un point  $\bar{x} \in ]a, b[$ .

## Définition

Une fonction  $f$  définie sur un intervalle  $[a, b]$  est dérivable en un point  $\bar{x} \in ]a, b[$  si la limite suivante existe et est finie

$$f'(\bar{x}) = \lim_{h \rightarrow 0} \frac{1}{h} (f(\bar{x} + h) - f(\bar{x})) \quad (8)$$

# Premières approximations

différence finie progressive :

$$f'(\bar{x}) \approx \frac{f(\bar{x} + h) - f(\bar{x})}{h} \quad (9)$$

différence finie rétrograde :

$$f'(\bar{x}) \approx \frac{f(\bar{x}) - f(\bar{x} - h)}{h} \quad (10)$$

# Estimation d'erreur

## Théorème (Taylor-Lagrange)

*On suppose que  $f \in \mathcal{C}^{n+1}$  sur  $I$ . Alors, pour tout  $h \in \mathbb{R}$  tel que  $\bar{x} + h$  appartienne à  $I$ , il existe  $\theta_h \in ]0, 1[$  tel que l'on ait*

$$f(\bar{x} + h) = \sum_{k=0}^n \frac{h^k}{k!} f^{(k)}(\bar{x}) + \frac{h^{n+1}}{(n+1)!} f^{(n+1)}(\bar{x} + \theta_h h) \quad (11)$$

# Estimation d'erreur

Soit  $h > 0$ . Si  $f \in \mathcal{C}^2(]a, b[)$ , alors  $\exists \xi_+ \in ]\bar{x}, \bar{x} + h[$ ,  $\exists \xi_- \in ]\bar{x} - h, \bar{x}[$ , tel que

$$\frac{f(\bar{x} + h) - f(\bar{x})}{h} = f'(\bar{x}) + \frac{h}{2}f^{(2)}(\xi_+) \quad (12)$$

$$\frac{f(\bar{x}) - f(\bar{x} - h)}{h} = f'(\bar{x}) - \frac{h}{2}f^{(2)}(\xi_-) \quad (13)$$

# Estimation d'erreur

Soit  $h > 0$ . Si  $f \in \mathcal{C}^2(]a, b[)$ , alors  $\exists \xi_+ \in ]\bar{x}, \bar{x} + h[$ ,  $\exists \xi_- \in ]\bar{x} - h, \bar{x}[$ , tel que

$$\frac{f(\bar{x} + h) - f(\bar{x})}{h} = f'(\bar{x}) + \frac{h^1}{2} f^{(2)}(\xi_+) \quad (12)$$

$$\frac{f(\bar{x}) - f(\bar{x} - h)}{h} = f'(\bar{x}) - \frac{h^1}{2} f^{(2)}(\xi_-) \quad (13)$$

- Ces formules sont des approximations d'ordre **1** de  $f'(\bar{x})$  par rapport à  $h$ .

# Estimation d'erreur

Soit  $h > 0$ . Si  $f \in \mathcal{C}^2(]a, b[)$ , alors  $\exists \xi_+ \in ]\bar{x}, \bar{x} + h[$ ,  $\exists \xi_- \in ]\bar{x} - h, \bar{x}[$ , tel que

$$\frac{f(\bar{x} + h) - f(\bar{x})}{h} = f'(\bar{x}) + \frac{h^1}{2} f^{(2)}(\xi_+) \quad (12)$$

$$\frac{f(\bar{x}) - f(\bar{x} - h)}{h} = f'(\bar{x}) - \frac{h^1}{2} f^{(2)}(\xi_-) \quad (13)$$

- Ces formules sont des approximations d'ordre **1** de  $f'(\bar{x})$  par rapport à  $h$ .
- On peut aussi obtenir ces formules en dérivant les polynômes d'interpolation associés aux points  $\{\bar{x}, \bar{x} + h\}$  et  $\{\bar{x} - h, \bar{x}\}$ .

# Exercice 1

## Exercice

On note  $x_i = a + ih$ ,  $i \in \llbracket 0, n \rrbracket$ , une discrétisation régulière de l'intervalle  $[a, b]$ . Soit une fonction  $f : [a, b] \rightarrow \mathbb{R}$  suffisamment régulière. On suppose que les  $y_i$  sont donnés par

$$y_i = f(x_i), \quad \forall i \in \llbracket 0, n \rrbracket. \quad (14)$$

Écrire une fonction `DERIVE1` permettant de calculer des approximations d'ordre 1 de  $f'(x_i)$  pour  $i \in \llbracket 0, n \rrbracket$ .

## Ordre 2

Pour obtenir une formule d'approximation d'ordre 2 de  $f'(\bar{x})$ , on suppose  $f \in \mathcal{C}^3(]a, b[)$  et on peut alors développer les formules de Taylor de  $f(\bar{x} + h)$  et  $f(\bar{x} - h)$  jusqu'au troisième ordre.

## Ordre 2

Pour obtenir une formule d'approximation d'ordre 2 de  $f'(\bar{x})$ , on suppose  $f \in \mathcal{C}^3(]a, b[)$  et on peut alors développer les formules de Taylor de  $f(\bar{x} + h)$  et  $f(\bar{x} - h)$  jusqu'au troisième ordre.

On obtient alors

$$f'(\bar{x}) = \frac{f(\bar{x} + h) - f(\bar{x} - h)}{2h} + \mathcal{O}(h^2). \quad (15)$$

## Ordre 2

Pour obtenir une formule d'approximation d'ordre 2 de  $f'(\bar{x})$ , on suppose  $f \in \mathcal{C}^3(]a, b[)$  et on peut alors développer les formules de Taylor de  $f(\bar{x} + h)$  et  $f(\bar{x} - h)$  jusqu'au troisième ordre.

On obtient alors

$$f'(\bar{x}) = \frac{f(\bar{x} + h) - f(\bar{x} - h)}{2h} + \mathcal{O}(h^2). \quad (15)$$

- Cette approximation est la formule des **différences finies centrées**.

## Ordre 2

Pour obtenir une formule d'approximation d'ordre 2 de  $f'(\bar{x})$ , on suppose  $f \in \mathcal{C}^3(]a, b[)$  et on peut alors développer les formules de Taylor de  $f(\bar{x} + h)$  et  $f(\bar{x} - h)$  jusqu'au troisième ordre.

On obtient alors

$$f'(\bar{x}) = \frac{f(\bar{x} + h) - f(\bar{x} - h)}{2h} + \mathcal{O}(h^2). \quad (15)$$

- Cette approximation est la formule des **différences finies centrées**.
- Cette approximation est d'ordre **2**.

## Exercice 2

### Exercice

On note  $x_i = a + ih$ ,  $i \in \llbracket 0, n \rrbracket$ , une discrétisation régulière de l'intervalle  $[a, b]$ . Soit une fonction  $f : [a, b] \rightarrow \mathbb{R}$  suffisamment régulière. On suppose que les  $y_i$  sont donnés par

$$y_i = f(x_i), \quad \forall i \in \llbracket 0, n \rrbracket. \quad (16)$$

Écrire une fonction `DERIVE2` permettant de calculer des approximations d'ordre 2 de  $f'(x_i)$  pour  $i \in \llbracket 0, n \rrbracket$ .

# Dérivée seconde

Si  $f \in \mathcal{C}^4(]a, b[)$ , on peut alors développer les formules de Taylor de  $f(\bar{x} + h)$  et  $f(\bar{x} - h)$  jusqu'au quatrième ordre.

## Dérivée seconde

Si  $f \in \mathcal{C}^4(]a, b[)$ , on peut alors développer les formules de Taylor de  $f(\bar{x} + h)$  et  $f(\bar{x} - h)$  jusqu'au quatrième ordre.

On obtient alors

$$f^{(2)}(\bar{x}) = \frac{f(\bar{x} + h) - 2f(\bar{x}) + f(\bar{x} - h)}{h^2} + \mathcal{O}(h^2). \quad (17)$$

## Dérivée seconde

Si  $f \in \mathcal{C}^4(]a, b[)$ , on peut alors développer les formules de Taylor de  $f(\bar{x} + h)$  et  $f(\bar{x} - h)$  jusqu'au quatrième ordre.

On obtient alors

$$f^{(2)}(\bar{x}) = \frac{f(\bar{x} + h) - 2f(\bar{x}) + f(\bar{x} - h)}{h^2} + \mathcal{O}(h^2). \quad (17)$$

Cette approximation est d'ordre **2**.

## Exercice 3

### Exercice

On note  $x_i = a + ih$ ,  $i \in \llbracket 0, n \rrbracket$ , une discrétisation régulière de l'intervalle  $[a, b]$ . Soit une fonction  $f : [a, b] \rightarrow \mathbb{R}$  suffisamment régulière. On suppose que les  $y_i$  sont donnés par

$$y_i = f(x_i), \quad \forall i \in \llbracket 0, n \rrbracket. \quad (18)$$

Ecrire une fonction `DERIVESECONDE2` permettant de calculer des approximations d'ordre 2 de  $f^{(2)}(x_i)$  pour  $i \in \llbracket 0, n \rrbracket$ .

# Plan

1 Polynômes d'interpolation de Lagrange

2 Dérivation numérique

3 **Intégration numérique**

- Méthodes simplistes
- Méthodes de Newton-Cotes
- Méthodes composites
- Autres méthodes
- Intégrales multiples

4 Résolution de systèmes linéaires

- Vecteurs
- Matrices
- Factorisation LU
- Factorisation de Cholesky

# Intégration

Soit  $f$  une fonction définie et intégrable sur un intervalle  $[a, b]$  donné.  
On propose de chercher une approximation de

$$I = \int_a^b f(t) dt$$

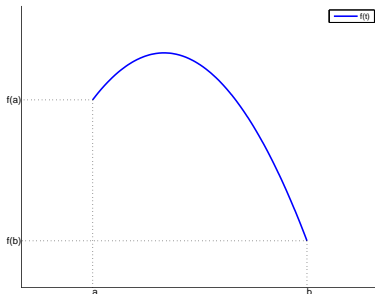


FIGURE : Représentation de la fonction  $f$

# Intégration

Soit  $f$  une fonction définie et intégrable sur un intervalle  $[a, b]$  donné.  
On propose de chercher une approximation de

$$I = \int_a^b f(t) dt$$

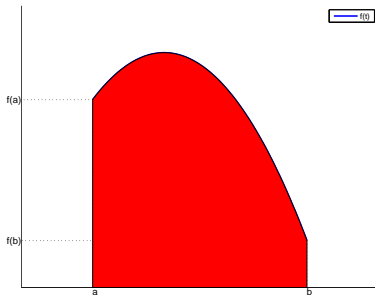


FIGURE : Représentation de  $\int_a^b f(t) dt$

# Méthodes simplistes

On approche  $f$  par le polynôme constant  $P(t) = f(a)$ .

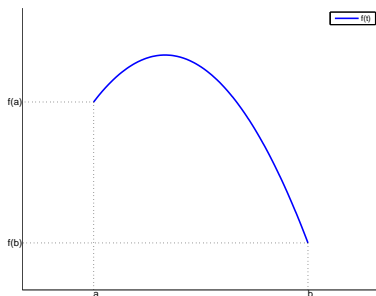


FIGURE : Représentation de la fonction  $f$

# Méthodes simplistes

On approche  $f$  par le polynôme constant  $P(t) = f(a)$ .

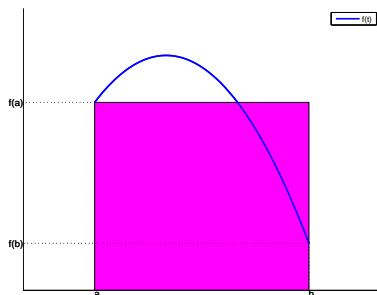


FIGURE : Représentation de  $\int_a^b P(t) dt$

# Méthodes simplistes

On approche  $f$  par le polynôme constant  $P(t) = f(a)$ .

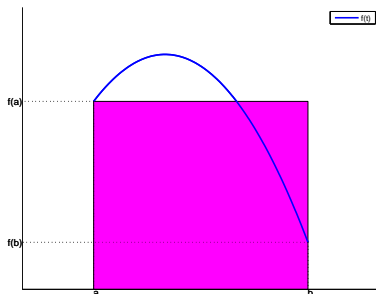


FIGURE : Représentation de  $\int_a^b P(t) dt$

$$\int_a^b f(t) dt \approx (b - a)f(a), \text{ formule du rectangle (à gauche)}$$

# Méthodes simplistes

On approche  $f$  par le polynôme constant  $P(t) = f(b)$ .

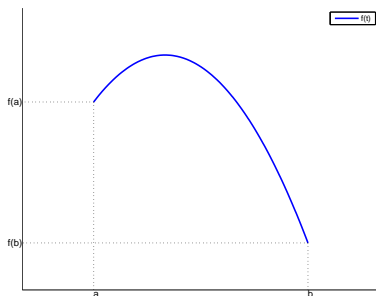


FIGURE : Représentation de la fonction  $f$

# Méthodes simplistes

On approche  $f$  par le polynôme constant  $P(t) = f(b)$ .

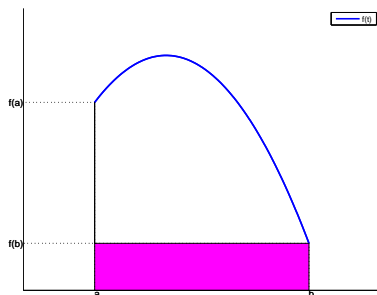


FIGURE : Représentation de  $\int_a^b P(t) dt$

# Méthodes simplistes

On approche  $f$  par le polynôme constant  $P(t) = f(b)$ .

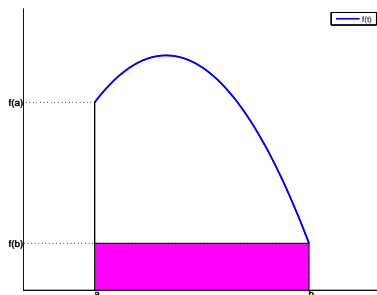


FIGURE : Représentation de  $\int_a^b P(t) dt$

$$\int_a^b f(t) dt \approx (b-a)f(b), \text{ formule du rectangle (à droite)}$$

# Méthodes simplistes

On approche  $f$  par le polynôme constant  $P(t) = f(c)$ , avec  $c = (a + b)/2$ .

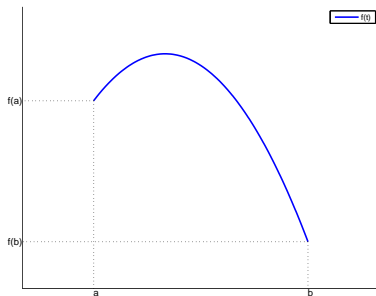


FIGURE : Représentation de la fonction  $f$

# Méthodes simplistes

On approche  $f$  par le polynôme constant  $P(t) = f(c)$ , avec  $c = (a + b)/2$ .

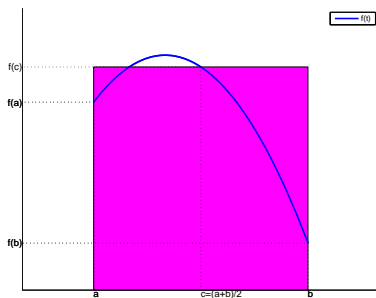


FIGURE : Représentation de  $\int_a^b P(t) dt$

# Méthodes simplistes

On approche  $f$  par le polynôme constant  $P(t) = f(c)$ , avec  $c = (a + b)/2$ .

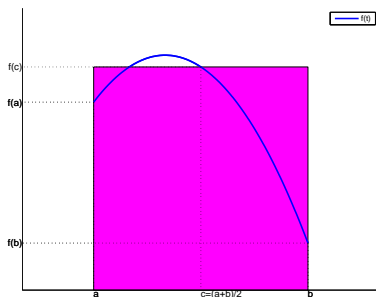


FIGURE : Représentation de  $\int_a^b P(t) dt$

$$\int_a^b f(t) dx \approx (b - a) f\left(\frac{a + b}{2}\right), \text{ formule du point milieu}$$

# Méthodes de Newton-Cotes

- 1 Discrétisation régulière de  $[a, b]$  :  $\forall i \in \llbracket 0, n \rrbracket, t_i = a + ih$  avec  $h = \frac{b-a}{n}$ .

# Méthodes de Newton-Cotes

- 1 Discrétisation régulière de  $[a, b]$  :  $\forall i \in \llbracket 0, n \rrbracket$ ,  $t_i = a + ih$  avec  $h = \frac{b-a}{n}$ .
- 2 On approche  $f$  par le polynôme d'interpolation de Lagrange  $\mathcal{P}_n$  de degré  $n$  tel que

$$\mathcal{P}_n(t_i) = f(t_i), \quad \forall i \in \llbracket 0, n \rrbracket.$$

# Méthodes de Newton-Cotes

- 1 Discrétisation régulière de  $[a, b]$  :  $\forall i \in \llbracket 0, n \rrbracket$ ,  $t_i = a + ih$  avec  $h = \frac{b-a}{n}$ .
- 2 On approche  $f$  par le polynôme d'interpolation de Lagrange  $\mathcal{P}_n$  de degré  $n$  tel que

$$\mathcal{P}_n(t_i) = f(t_i), \quad \forall i \in \llbracket 0, n \rrbracket.$$

$$\mathcal{P}_n(t) = \sum_{i=0}^n L_i(t) f(t_i)$$

# Méthodes de Newton-Cotes

- 1 Discrétisation régulière de  $[a, b]$  :  $\forall i \in \llbracket 0, n \rrbracket$ ,  $t_i = a + ih$  avec  $h = \frac{b-a}{n}$ .
- 2 On approche  $f$  par le polynôme d'interpolation de Lagrange  $\mathcal{P}_n$  de degré  $n$  tel que

$$\mathcal{P}_n(t_i) = f(t_i), \quad \forall i \in \llbracket 0, n \rrbracket.$$

$$\mathcal{P}_n(t) = \sum_{i=0}^n L_i(t) f(t_i)$$

- 3 On a alors

$$\int_a^b f(t) dt \approx \int_a^b \mathcal{P}_n(t) dt = \sum_{i=0}^n f(t_i) \int_a^b L_i(t) dt.$$

# Méthodes de Newton-Cotes

- 1 Discrétisation régulière de  $[a, b]$  :  $\forall i \in \llbracket 0, n \rrbracket$ ,  $t_i = a + ih$  avec  $h = \frac{b-a}{n}$ .
- 2 On approche  $f$  par le polynôme d'interpolation de Lagrange  $\mathcal{P}_n$  de degré  $n$  tel que

$$\mathcal{P}_n(t_i) = f(t_i), \quad \forall i \in \llbracket 0, n \rrbracket.$$

$$\mathcal{P}_n(t) = \sum_{i=0}^n L_i(t) f(t_i)$$

- 3 On a alors

$$\int_a^b f(t) dt \approx \int_a^b \mathcal{P}_n(t) dt = \sum_{i=0}^n f(t_i) \int_a^b L_i(t) dt.$$

Les formules de Newton-Cotes génériques :

$$\int_a^b f(t) dt \approx \sum_{i=0}^n \alpha_i f(t_i)$$

# Méthodes de Newton-Cotes

$$\int_a^b f(t)dt \approx \sum_{i=0}^n \alpha_i f(t_i) \text{ avec } \alpha_i = \int_a^b L_i(t)dt$$

En posant  $\alpha_i = hAw_i$ , on a

$n$	$A$	$w_0$	$w_1$	$w_2$	$w_3$	$w_4$	nom	ordre
1	1/2	1	1				trapèzes	1
2	1/3	1	4	1			Simpson	3
3	3/8	1	3	3	1		Simpson (3/8)	3
4	2/45	7	32	12	32	7	Villarceau	5

$$\text{Simpson : } \int_a^b f(t)dt \approx$$

# Méthodes de Newton-Cotes

$$\int_a^b f(t)dt \approx \sum_{i=0}^n \alpha_i f(t_i) \text{ avec } \alpha_i = \int_a^b L_i(t)dt$$

En posant  $\alpha_i = hAw_i$ , on a

$n$	$A$	$w_0$	$w_1$	$w_2$	$w_3$	$w_4$	nom	ordre
1	1/2	1	1				trapèzes	1
2	1/3	1	4	1			Simpson	3
3	3/8	1	3	3	1		Simpson (3/8)	3
4	2/45	7	32	12	32	7	Villarceau	5

$$\text{Simpson : } \int_a^b f(t)dt \approx \frac{b-a}{6} \left( f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right)$$

# Méthodes de Newton-Cotes

## Définition

On dit qu'une formule d'intégration (ou formule de quadrature) est d'ordre  $n$  si elle est exacte pour les polynômes de degré inférieur ou égal à  $n$ .

## Théorème

*Les formules de Newton-Cotes à  $n + 1$  points sont d'ordre  $n$  si  $n$  est impair et d'ordre  $n + 1$  sinon.*

# Méthodes de Newton-Cotes

## Définition

On dit qu'une formule d'intégration (ou formule de quadrature) est d'ordre  $n$  si elle est exacte pour les polynômes de degré inférieur ou égal à  $n$ .

## Théorème

*Les formules de Newton-Cotes à  $n + 1$  points sont d'ordre  $n$  si  $n$  est impair et d'ordre  $n + 1$  sinon.*

## Attention

Du au phénomène de Runge, ces formules ne sont pas "fiables" pour des ordres élevés.

# Méthodes composites

Les méthodes composites sont basées sur la relation de Chasles.  
Soit  $(x_k)_{k \in \llbracket 0, n \rrbracket}$  une discrétisation régulière de l'intervalle  $[a, b]$  :  
 $x_k = a + kh$  avec  $h = (b - a)/n$ . On a alors

$$\int_a^b f(x) dx = \sum_{k=1}^n \int_{x_{k-1}}^{x_k} f(x) dx.$$

# Méthodes composites des points milieux

$$\int_a^b f(x) dx = \sum_{k=1}^n \int_{x_{k-1}}^{x_k} f(x) dx.$$

On note  $m_k = \frac{x_{k-1} + x_k}{2}$ .

$$\int_{x_{k-1}}^{x_k} f(x) dx \approx hf(m_k)$$

# Méthodes composites des points milieux

$$\int_a^b f(x) dx = \sum_{k=1}^n \int_{x_{k-1}}^{x_k} f(x) dx.$$

On note  $m_k = \frac{x_{k-1} + x_k}{2}$ .

$$\int_{x_{k-1}}^{x_k} f(x) dx \approx hf(m_k)$$

## Théorème

$$\int_a^b f(x) dx = h \sum_{k=1}^n f(m_k) + \mathcal{O}(h^2).$$

# Méthodes composites des points milieux

$$\int_a^b f(x) dx = \sum_{k=1}^n \int_{x_{k-1}}^{x_k} f(x) dx.$$

On note  $m_k = \frac{x_{k-1} + x_k}{2}$ .

$$\int_{x_{k-1}}^{x_k} f(x) dx \approx hf(m_k)$$

## Théorème

$$\int_a^b f(x) dx = h \sum_{k=1}^n f(m_k) + \mathcal{O}(h^2).$$

Erreur d'ordre **2** (par rapport à  $h$ .)

# Méthodes composites des points milieux : Exercice

## Exercice

*Soit  $f$  une fonction définie sur l'intervalle  $[a, b]$ . Ecrire la fonction QUADPM permettant de calculer une approximation de l'intégrale de  $f$  sur  $[a, b]$  par la méthode composite des points milieux.*

# Méthodes composites des trapèzes

$$\int_a^b f(x) dx = \sum_{k=1}^n \int_{x_{k-1}}^{x_k} f(x) dx.$$

$$\int_{x_{k-1}}^{x_k} f(x) dx \approx h \left( \frac{f(x_{k-1}) + f(x_k)}{2} \right)$$

# Méthodes composites des trapèzes

$$\int_a^b f(x) dx = \sum_{k=1}^n \int_{x_{k-1}}^{x_k} f(x) dx.$$

$$\int_{x_{k-1}}^{x_k} f(x) dx \approx h \left( \frac{f(x_{k-1}) + f(x_k)}{2} \right)$$

## Théorème

$$\int_a^b f(x) dx = h \sum_{k=1}^n \left( \frac{f(x_{k-1}) + f(x_k)}{2} \right) + \mathcal{O}(h^2).$$

# Méthodes composites des trapèzes

$$\int_a^b f(x) dx = \sum_{k=1}^n \int_{x_{k-1}}^{x_k} f(x) dx.$$

$$\int_{x_{k-1}}^{x_k} f(x) dx \approx h \left( \frac{f(x_{k-1}) + f(x_k)}{2} \right)$$

## Théorème

$$\int_a^b f(x) dx = h \sum_{k=1}^n \left( \frac{f(x_{k-1}) + f(x_k)}{2} \right) + \mathcal{O}(h^2).$$

Erreur d'ordre **2** (par rapport à  $h$ .)

# Méthodes composites des trapèzes : Exercice

## Exercice

*Soit  $f$  une fonction définie sur l'intervalle  $[a, b]$ . Ecrire la fonction QUADTRAPEZE permettant de calculer une approximation de l'intégrale de  $f$  sur  $[a, b]$  par la méthode composite des trapèzes.*

# Méthodes composites de Simpson

$$\int_a^b f(x) dx = \sum_{k=1}^n \int_{x_{k-1}}^{x_k} f(x) dx.$$

On note  $m_k = \frac{x_{k-1} + x_k}{2}$ .

$$\int_{x_{k-1}}^{x_k} f(x) dx \approx \frac{h}{6} (f(x_{k-1}) + 4f(m_k) + f(x_k))$$

# Méthodes composites de Simpson

$$\int_a^b f(x) dx = \sum_{k=1}^n \int_{x_{k-1}}^{x_k} f(x) dx.$$

On note  $m_k = \frac{x_{k-1} + x_k}{2}$ .

$$\int_{x_{k-1}}^{x_k} f(x) dx \approx \frac{h}{6} (f(x_{k-1}) + 4f(m_k) + f(x_k))$$

## Théorème

$$\int_a^b f(x) dx = \frac{h}{6} \sum_{k=1}^n (f(x_{k-1}) + 4f(m_k) + f(x_k)) + \mathcal{O}(h^4).$$

# Méthodes composites de Simpson

$$\int_a^b f(x) dx = \sum_{k=1}^n \int_{x_{k-1}}^{x_k} f(x) dx.$$

On note  $m_k = \frac{x_{k-1} + x_k}{2}$ .

$$\int_{x_{k-1}}^{x_k} f(x) dx \approx \frac{h}{6} (f(x_{k-1}) + 4f(m_k) + f(x_k))$$

## Théorème

$$\int_a^b f(x) dx = \frac{h}{6} \sum_{k=1}^n (f(x_{k-1}) + 4f(m_k) + f(x_k)) + \mathcal{O}(h^4).$$

Erreur d'ordre 4 (par rapport à  $h$ .)

# Méthodes composites de Simpson : Exercice

## Exercice

*Soit  $f$  une fonction définie sur l'intervalle  $[a, b]$ . Ecrire la fonction QUADSIMPSON permettant de calculer une approximation de l'intégrale de  $f$  sur  $[a, b]$  par la méthode composite de Simpson.*

# Ordres (numériques) des méthodes composites

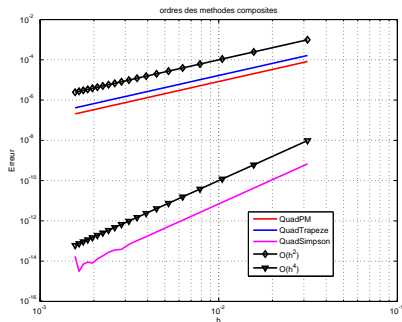


FIGURE : Ordre de l'erreur des méthodes composites

## Exercice

*Ecrire un programme Matlab permettant d'obtenir cette figure sachant qu'ici  $f(x) = \sin(x)$ ,  $a = 0$  et  $b = \pi$ .*

# Autres méthodes

Il existe un grand nombre de méthodes d'intégration numérique :

- Méthode de Gauss-Legendre
- Méthode de Gauss-Tchebychev
- Méthode de Gauss-Laguerre
- Méthode de Gauss-Hermitte
- Méthode de Gauss-Lobatto
- Méthode de Romberg...

# Intégrales multiples

On veut approcher, en utilisant la formule de Simpson, l'intégrale

$$I = \int_a^b \int_c^d f(x, y) dy dx$$

$$g(x) = \int_c^d f(x, y) dy \approx \tilde{g}(x) = \frac{d-c}{6} \left( f(x, c) + 4f\left(x, \frac{c+d}{2}\right) + f(x, d) \right).$$

On a

$$\begin{aligned} I = \int_a^b g(x) dx &\approx \frac{b-a}{6} \left( g(a) + 4g\left(\frac{a+b}{2}\right) + g(b) \right) \\ &\approx \frac{b-a}{6} \left( \tilde{g}(a) + 4\tilde{g}\left(\frac{a+b}{2}\right) + \tilde{g}(b) \right) \end{aligned}$$

# Intégrales multiples : formule de Simpson 2D

On pose  $\alpha = \frac{a+b}{2}$  et  $\beta = \frac{c+d}{2}$

$$I \approx \frac{b-a}{6} \frac{d-c}{6} \begin{pmatrix} f(a, c) + 4f(a, \beta) + f(a, d) \\ +4(f(\alpha, c) + 4f(\alpha, \beta) + f(\alpha, d)) \\ +f(b, c) + 4f(b, \beta) + f(b, d) \end{pmatrix}$$

# Intégrales multiples : méthodes composites

- 1 Discrétisation régulière de  $[a, b]$  :  $\forall k \in \llbracket 0, n \rrbracket$ ,  $x_k = a + kh_x$  avec  $h_x = \frac{b-a}{n}$ .
- 2 Discrétisation régulière de  $[c, d]$  :  $\forall l \in \llbracket 0, m \rrbracket$ ,  $y_l = c + lh_y$  avec  $h_y = \frac{d-c}{m}$ .

# Intégrales multiples : méthodes composites

- 1 Discrétisation régulière de  $[a, b]$  :  $\forall k \in \llbracket 0, n \rrbracket$ ,  $x_k = a + kh_x$  avec  $h_x = \frac{b-a}{n}$ .
- 2 Discrétisation régulière de  $[c, d]$  :  $\forall l \in \llbracket 0, m \rrbracket$ ,  $y_l = c + lh_y$  avec  $h_y = \frac{d-c}{m}$ .
- 3 Relation de Chasles :

$$\int_a^b \int_c^d f(x, y) dy dx = \sum_{k=1}^n \sum_{l=1}^m \int_{x_{k-1}}^{x_k} \int_{y_{l-1}}^{y_l} f(x, y) dy dx.$$

# Intégrales multiples : méthodes composites

- 1 Discrétisation régulière de  $[a, b]$  :  $\forall k \in \llbracket 0, n \rrbracket$ ,  $x_k = a + kh_x$  avec  $h_x = \frac{b-a}{n}$ .
- 2 Discrétisation régulière de  $[c, d]$  :  $\forall l \in \llbracket 0, m \rrbracket$ ,  $y_l = c + lh_y$  avec  $h_y = \frac{d-c}{m}$ .
- 3 Relation de Chasles :

$$\int_a^b \int_c^d f(x, y) dy dx = \sum_{k=1}^n \sum_{l=1}^m \int_{x_{k-1}}^{x_k} \int_{y_{l-1}}^{y_l} f(x, y) dy dx.$$

- 4 Formule de Simpson 2D :

$$\begin{aligned} & \int_a^b \int_c^d f(x, y) dy dx \\ &= \\ & \frac{h_x h_y}{36} \sum_{k=1}^n \sum_{l=1}^m \left( \begin{array}{l} f(x_{k-1}, y_{l-1}) + 4f(x_{k-1}, \beta_l) + f(x_{k-1}, y_l) \\ + 4(f(\alpha_k, y_{l-1}) + 4f(\alpha_k, \beta_l) + f(\alpha_k, y_l)) \\ + f(x_k, y_{l-1}) + 4f(x_k, \beta_l) + f(x_k, y_l) \end{array} \right) \end{aligned}$$

avec  $\alpha_k = \frac{x_{k-1} + x_k}{2}$  et  $\beta_l = \frac{y_{l-1} + y_l}{2}$ .

# Plan

## 1 Polynômes d'interpolation de Lagrange

## 2 Dérivation numérique

## 3 Intégration numérique

- Méthodes simplistes
- Méthodes de Newton-Cotes
- Méthodes composites
- Autres méthodes
- Intégrales multiples

## 4 Résolution de systèmes linéaires

- Vecteurs
- Matrices
- Factorisation LU
- Factorisation de Cholesky

# Introduction

A faire

# Vecteurs : $\mathbf{z} = \alpha\mathbf{x} + \mathbf{y}$

## Exercice

Soient  $\mathbf{x}$  et  $\mathbf{y}$  deux vecteurs de  $\mathbb{R}^n$  et  $\alpha \in \mathbb{R}$ . Ecrire la fonction `VECAXPY` permettant de calculer  $\mathbf{z} = \alpha\mathbf{x} + \mathbf{y}$ .

# Vecteurs : $\mathbf{z} = \alpha\mathbf{x} + \mathbf{y}$

## Exercice

Soient  $\mathbf{x}$  et  $\mathbf{y}$  deux vecteurs de  $\mathbb{R}^n$  et  $\alpha \in \mathbb{R}$ . Ecrire la fonction `VECAXPY` permettant de calculer  $\mathbf{z} = \alpha\mathbf{x} + \mathbf{y}$ .

## Correction

On pose  $\mathbf{z} = \alpha\mathbf{x} + \mathbf{y}$ .  $\mathbf{z}$  est un vecteur de  $\mathbb{R}^n$  et

$$z_i = \alpha x_i + y_i, \quad \forall i \in \llbracket 1, n \rrbracket.$$



# Vecteurs : $\mathbf{z} = \alpha\mathbf{x} + \mathbf{y}$

---

**Algorithme 1** Fonction VECAXPY retournant  $\mathbf{z} = \alpha\mathbf{x} + \mathbf{y}$  avec  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$   
et  $\alpha \in \mathbb{R}$

---

**Données :**  $\mathbf{x}, \mathbf{y}$  : deux vecteurs de  $\mathbb{R}^n$   
 $\alpha$  : un réel.

**Résultat :**  $\mathbf{z}$  : vecteur de  $\mathbb{R}^n$ .

- 1: **Fonction**  $\mathbf{z} \leftarrow \text{VECAXPY}(\alpha, \mathbf{x}, \mathbf{y})$
  - 2:     **Pour**  $i \leftarrow 1$  à  $n$  **faire**
  - 3:          $z(i) \leftarrow \alpha * x(i) + y(i)$
  - 4:     **Fin Pour**
  - 5: **Fin Fonction**
-

# Vecteurs

## Exercice

Soient  $\mathbf{x}$  et  $\mathbf{y}$  deux vecteurs de  $\mathbb{R}^n$ .

- 1 Ecrire la fonction `VECDOT` permettant de calculer le produit scalaire entre les vecteurs  $\mathbf{x}$  et  $\mathbf{y}$  :

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i. \quad (19)$$

- 2 Ecrire la fonction `VECNORM1` permettant de calculer

$$\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|, \quad \forall \mathbf{x} \in \mathbb{R}^n. \quad (20)$$

- 3 Ecrire la fonction `VECNORM2` permettant de calculer

$$\|\mathbf{x}\|_2 = \left( \sum_{i=1}^n |x_i|^2 \right)^{1/2}, \quad \forall \mathbf{x} \in \mathbb{R}^n. \quad (21)$$

# Correction : fonction VEC DOT

---

**Algorithme 2** Fonction VEC DOT permettant de calculer le produit scalaire des vecteurs  $\mathbf{x}$  et  $\mathbf{y}$  où  $\mathbf{x} \in \mathbb{R}^n$  et  $\mathbf{y} \in \mathbb{R}^n$ .

---

**Données :**

$\mathbf{x}$  : vecteur de  $\mathbb{R}^n$ ,

$\mathbf{y}$  : vecteur de  $\mathbb{R}^n$ .

**Résultat :**  $s$  : le réel tel que  $s = \langle \mathbf{x}, \mathbf{y} \rangle$ .

1: **Fonction**  $s \leftarrow \text{VECDOT}(\mathbf{x}, \mathbf{y})$

2:      $s \leftarrow 0$

3:     **Pour**  $i \leftarrow 1$  à  $n$  **faire**

4:          $s \leftarrow s + x(i) * y(i)$

5:     **Fin Pour**

6: **Fin Fonction**

---

# Correction : fonction VECNORM1

---

**Algorithme 3** Fonction VECNORM1 permettant de retourner  $\|\mathbf{x}\|_1$  avec  $\mathbf{x} \in \mathbb{R}^n$

---

**Données :**

$\mathbf{x}$  : vecteur de  $\mathbb{R}^n$ .

**Résultat :**

$s$  : le réel tel que  $s = \|\mathbf{x}\|_1$ .

- 1: **Fonction**  $s \leftarrow \text{VECNORM1}(\mathbf{x})$
- 2:      $s \leftarrow 0$
- 3:     **Pour**  $i = 1$  à  $n$  **faire**
- 4:          $s \leftarrow s + \text{ABS}(x(i))$
- 5:     **Fin Pour**
- 6: **Fin Fonction**

# Correction : fonction VECNORM2 (version 1)

---

**Algorithme 4** Fonction VECNORM2 permettant de retourner  $\|\mathbf{x}\|_2$  avec  $\mathbf{x} \in \mathbb{R}^n$

---

**Données :**

$\mathbf{x}$  : vecteur de  $\mathbb{R}^n$ .

**Résultat :**

$s$  : le réel tel que  $s = \|\mathbf{x}\|_2$ .

```
1: Fonction  $s \leftarrow \text{VECNORM2}(\mathbf{x})$ 
2:    $s \leftarrow 0$ 
3:   Pour  $i \leftarrow 1$  à  $n$  faire
4:      $s \leftarrow s + x(i) * x(i)$ 
5:   Fin Pour
6:    $s \leftarrow \text{SQRT}(s)$ 
7: Fin Fonction
```

## Correction : fonction VECNORM2 (version 2)

---

**Algorithme 5** Fonction VECNORM2 permettant de retourner  $\|\mathbf{x}\|_2$  avec  $\mathbf{x} \in \mathbb{R}^n$  (utilise la fonction VECDOT).

---

**Données :**

$\mathbf{x}$  : vecteur de  $\mathbb{R}^n$ .

**Résultat :**

$s$  : le réel tel que  $s = \|\mathbf{x}\|_2$ .

- 1: **Fonction**  $s \leftarrow \text{VECNORM2}(\mathbf{x})$
  - 2:  $s \leftarrow \text{SQRT}(\text{VECDOT}(\mathbf{x}, \mathbf{x}))$
  - 3: **Fin Fonction**
-

Matrices :  $\mathbb{Z} = \alpha\mathbb{X} + \mathbb{Y}$

## Exercice

Soient  $\mathbb{X}$  et  $\mathbb{Y}$  deux matrices de  $\mathcal{M}_{m,n}(\mathbb{R})$  et  $\alpha \in \mathbb{R}$ . Ecrire la fonction `MATXPY` permettant de retourner  $\mathbb{Z} = \alpha\mathbb{X} + \mathbb{Y}$ .

# Matrices : $\mathbb{Z} = \alpha\mathbb{X} + \mathbb{Y}$

## Exercice

Soient  $\mathbb{X}$  et  $\mathbb{Y}$  deux matrices de  $\mathcal{M}_{m,n}(\mathbb{R})$  et  $\alpha \in \mathbb{R}$ . Ecrire la fonction MATAXPY permettant de retourner  $\mathbb{Z} = \alpha\mathbb{X} + \mathbb{Y}$ .

**Correction** On a  $\mathbb{Z} \in \mathcal{M}_{m,n}(\mathbb{R})$  et

$$\forall i \in \llbracket 1, m \rrbracket, \forall j \in \llbracket 1, n \rrbracket, Z_{i,j} = \alpha X_{i,j} + Y_{i,j}.$$



# Matrices : $\mathbb{Z} = \alpha\mathbb{X} + \mathbb{Y}$

---

**Algorithme 6** Fonction MATAXPY permettant de retourner  $\mathbb{Z} = \alpha\mathbb{X} + \mathbb{Y}$  avec  $\mathbb{X}$  et  $\mathbb{Y}$  dans  $\mathcal{M}_{m,n}(\mathbb{R})$ , et  $\alpha \in \mathbb{R}$

---

**Données :**

- $\alpha$  : un réel,
- $\mathbb{X}$  : matrice de  $\mathcal{M}_{m,n}(\mathbb{R})$ ,
- $\mathbb{Y}$  : matrice de  $\mathcal{M}_{m,n}(\mathbb{R})$ .

**Résultat :**

- $\mathbb{Z}$  : matrice de  $\mathcal{M}_{m,n}(\mathbb{R})$  tel que  $\mathbb{Z} = \alpha\mathbb{X} + \mathbb{Y}$ .

- 1: **Fonction**  $\mathbb{Z} \leftarrow \text{MATAXPY}(\alpha, \mathbb{X}, \mathbb{Y})$
- 2:     **Pour**  $i \leftarrow 1$  à  $m$  **faire**
- 3:         **Pour**  $j \leftarrow 1$  à  $n$  **faire**
- 4:              $Z(i,j) \leftarrow \alpha * X(i,j) + Y(i,j)$
- 5:         **Fin Pour**
- 6:     **Fin Pour**
- 7: **Fin Fonction**

# Matrices : produit matrice-vecteur

## Exercice

*Ecrire la fonction MATMULT permettant de retourner le produit d'une matrice par un vecteur.*

# Matrices : produit matrice-vecteur

## Exercice

*Ecrire la fonction MATMULT permettant de retourner le produit d'une matrice par un vecteur.*

**Correction** On rappelle que le produit d'une matrice  $\mathbb{A} \in \mathcal{M}_{m,n}(\mathbb{R})$  par un vecteur  $\mathbf{x} \in \mathbb{R}^n$  est un vecteur de  $\mathbb{R}^m$ . On le note  $\mathbf{y}$  et on a

$$y_i = \sum_{j=1}^n A_{i,j} x_j, \quad \forall i \in \llbracket 1, m \rrbracket,$$

# Matrices : produit matrice-vecteur

## Algorithme 7- $\mathcal{R}_0$

---

1: Calcul de  $\mathbf{y} \leftarrow \mathbb{A}\mathbf{x}$ .

# Matrices : produit matrice-vecteur

Algorithme 7-  $\mathcal{R}_0$

Algorithme 7-  $\mathcal{R}_1$

1: Calcul de  $\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}$ .

1: **Pour**  $i \leftarrow 1$  à  $m$  **faire**

2:     Calcul de  $y_i \leftarrow \sum_{j=1}^n A_{i,j}x_j$

3: **Fin Pour**

# Matrices : produit matrice-vecteur

## Algorithme 7- $\mathcal{R}_1$

---

1: **Pour**  $i \leftarrow 1$  à  $m$  **faire**

2: Calcul de  $y_i \leftarrow \sum_{j=1}^n A_{i,j}x_j$

3: **Fin Pour**

# Matrices : produit matrice-vecteur

## Algorithme 7- $\mathcal{R}_1$

1: **Pour**  $i \leftarrow 1$  à  $m$  **faire**

2: Calcul de  $y_i \leftarrow \sum_{j=1}^n A_{i,j} x_j$

3: **Fin Pour**

## Algorithme 7- $\mathcal{R}_2$

1: **Pour**  $i \leftarrow 1$  à  $m$  **faire**

2:  $S \leftarrow 0$

3: **Pour**  $j \leftarrow 1$  à  $n$  **faire**

4:  $S \leftarrow S + A(i,j) * x(j)$

5: **Fin Pour**

6:  $y(i) \leftarrow S$

7: **Fin Pour**

# Matrices : produit matrice-vecteur

---

## Algorithme 7 Fonction MATMULT

---

### Données :

$\mathbb{A}$  : matrice de  $\mathcal{M}_{m,n}(\mathbb{R})$ ,

$\mathbf{x}$  : vecteur de  $\mathbb{R}^n$ .

### Résultat :

$\mathbf{y}$  : vecteur de  $\mathbb{R}^m$  tel que  $\mathbf{y} = \mathbb{A}\mathbf{x}$ .

```
1: Fonction  $\mathbf{x} \leftarrow \text{MATMULT}(\mathbb{A}, \mathbf{x})$ 
2:   Pour  $i \leftarrow 1$  à  $m$  faire
3:      $S \leftarrow 0$ 
4:     Pour  $j \leftarrow 1$  à  $n$  faire
5:        $S \leftarrow S + A(i, j) * x(j)$ 
6:     Fin Pour
7:      $y(i) \leftarrow S$ 
8:   Fin Pour
9: Fin Fonction
```

# Matrices : produit matrice-matrice

## Exercice

*Ecrire la fonction `MATMATMULT` permettant de retourner le produit de deux matrices.*

# Matrices : produit matrice-matrice

## Exercice

*Ecrire la fonction MATMATMULT permettant de retourner le produit de deux matrices.*

**Correction** Soient  $X \in \mathcal{M}_{m,n}(\mathbb{R})$  et  $Y \in \mathcal{M}_{n,p}(\mathbb{R})$ . On note  $Z \in \mathcal{M}_{m,p}(\mathbb{R})$  la matrice produit i.e.  $Z = XY$ .

On rappelle que l'on a

$$Z_{i,j} = \sum_{k=1}^n X_{i,k} Y_{k,j}, \quad \forall (i,j) \in \llbracket 1, m \rrbracket \times \llbracket 1, p \rrbracket.$$

# Matrices : produit matrice-matrice

## Algorithme 8- $\mathcal{R}_0$

---

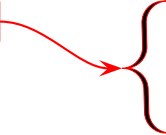
1: Calcul de  $\mathbb{Z} = \mathbb{X}\mathbb{Y}$ .

# Matrices : produit matrice-matrice

Algorithme 8-  $\mathcal{R}_0$

Algorithme 8-  $\mathcal{R}_1$

1: Calcul de  $\mathbb{Z} = \mathbb{X}\mathbb{Y}$ .



1: **Pour**  $i \leftarrow 1$  à  $m$  **faire**  
2:     Calcul ligne  $i$  matrice  $\mathbb{Z}$   
3: **Fin Pour**

# Matrices : produit matrice-matrice

## Algorithme 8- $\mathcal{R}_1$

---

- 1: **Pour**  $i \leftarrow 1$  à  $m$  **faire**
- 2: Calcul ligne  $i$  matrice  $\mathbb{Z}$
- 3: **Fin Pour**

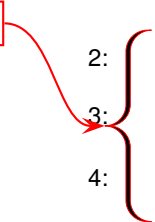
# Matrices : produit matrice-matrice

## Algorithme 8- $\mathcal{R}_1$

- 1: **Pour**  $i \leftarrow 1$  à  $m$  **faire**
- 2: Calcul ligne  $i$  matrice  $\mathbb{Z}$
- 3: **Fin Pour**

## Algorithme 8- $\mathcal{R}_2$

- 1: **Pour**  $i \leftarrow 1$  à  $m$  **faire**
- 2: **Pour**  $j \leftarrow 1$  à  $p$  **faire**
- 3: 
$$Z_{i,j} \leftarrow \sum_{k=1}^n X_{i,k} Y_{k,j}$$
- 4: **Fin Pour**
- 5: **Fin Pour**



# Matrices : produit matrice-matrice

## Algorithme 8- $\mathcal{R}_2$

---

- 1: **Pour**  $i \leftarrow 1$  à  $m$  **faire**
- 2:     **Pour**  $j \leftarrow 1$  à  $p$  **faire**
- 3:         
$$Z_{i,j} \leftarrow \sum_{k=1}^n X_{i,k} Y_{k,j}$$
- 4:     **Fin Pour**
- 5: **Fin Pour**

# Matrices : produit matrice-matrice

## Algorithme 8- $\mathcal{R}_2$

- 1: **Pour**  $i \leftarrow 1$  à  $m$  faire
- 2:   **Pour**  $j \leftarrow 1$  à  $p$  faire
- 3:      $Z_{i,j} \leftarrow \sum_{k=1}^n X_{i,k} Y_{k,j}$
- 4:   **Fin Pour**
- 5: **Fin Pour**

## Algorithme 8- $\mathcal{R}_3$

- 1: **Pour**  $i \leftarrow 1$  à  $m$  faire
- 2:   **Pour**  $j \leftarrow 1$  à  $p$  faire
- 3:      $S \leftarrow 0$
- 4:     **Pour**  $k \leftarrow 1$  à  $n$  faire
- 5:        $S \leftarrow S + X_{i,k} Y_{k,j}$
- 6:     **Fin Pour**
- 7:      $Z_{i,j} \leftarrow S$
- 8:   **Fin Pour**
- 9: **Fin Pour**

# Matrices : produit matrice-matrice

---

## Algorithme 8 Fonction MATMATMULT

---

### Données :

$\mathbb{X}$  : matrice de  $\mathcal{M}_{m,n}(\mathbb{R})$ ,

$\mathbb{Y}$  : matrice de  $\mathcal{M}_{n,p}(\mathbb{R})$ .

### Résultat :

$\mathbb{Z}$  : matrice de  $\mathcal{M}_{m,p}(\mathbb{R})$  telle que  $\mathbb{Z} = \mathbb{X}\mathbb{Y}$ .

```
1: Fonction  $\mathbb{Z} \leftarrow \text{MATMATMULT}(\mathbb{X}, \mathbb{Y})$ 
2:   Pour  $i \leftarrow 1$  à  $m$  faire
3:     Pour  $j \leftarrow 1$  à  $p$  faire
4:        $S \leftarrow 0$ 
5:       Pour  $k \leftarrow 1$  à  $n$  faire
6:          $S \leftarrow S + X(i, k) * Y(k, j)$ 
7:       Fin Pour
8:        $Z(i, j) \leftarrow S$ 
9:     Fin Pour
10:  Fin Pour
11: Fin Fonction
```

# Factorisation LU

## Théorème

*Soit  $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$  une matrice dont les sous-matrices principales sont inversibles alors il existe une unique matrice  $\mathbb{L} \in \mathcal{M}_n(\mathbb{R})$  triangulaire inférieure à diagonale unité et une unique matrice  $\mathbb{U} \in \mathcal{M}_n(\mathbb{R})$  triangulaire supérieure telles que*

$$\mathbb{A} = \mathbb{L}\mathbb{U}.$$

# Factorisation LU

Trouver  $\mathbf{x} \in \mathbb{R}^n$  tel que

$$\mathbf{Ax} = \mathbf{b}. \quad (22)$$

est équivalent à

Trouver  $\mathbf{y} \in \mathbb{R}^n$  solution de

$$\mathbf{Ly} = \mathbf{b} \quad (23)$$

puis  $\mathbf{x} \in \mathbb{R}^n$  solution de

$$\mathbf{Ux} = \mathbf{y}. \quad (24)$$

# Factorisation LU

---

**Algorithme 9** Fonction RSLFACTLU pour résoudre  $\mathbb{A}\mathbf{x} = \mathbf{b}$

---

**Données :**  $\mathbb{A}$  : matrice de  $\mathcal{M}_n(\mathbb{R})$  dont les sous-matrices principales sont inversibles définie positive,  
 $\mathbf{b}$  : vecteur de  $\mathbb{R}^n$ .

**Résultat :**  $\mathbf{x}$  : vecteur de  $\mathbb{R}^n$ .

- 1: **Fonction**  $\mathbf{x} \leftarrow \text{RSLFACTLU}(\mathbb{A}, \mathbf{b})$
  - 2:  $[\mathbb{L}, \mathbb{U}] \leftarrow \text{FACTLU}(\mathbb{A})$  ▷ Factorisation LU
  - 3:  $\mathbf{y} \leftarrow \text{RESTRINF}(\mathbb{L}, \mathbf{b})$  ▷ Résolution du système  $\mathbb{L}\mathbf{y} = \mathbf{b}$
  - 4:  $\mathbf{x} \leftarrow \text{RESTRISUP}(\mathbb{U}, \mathbf{y})$  ▷ Résolution du système  $\mathbb{U}\mathbf{x} = \mathbf{y}$
  - 5: **Fin Fonction**
-

# Résolution système triangulaire inférieur

Soit  $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$  triangulaire inférieure inversible et  $\mathbf{b} \in \mathbb{R}^n$  donnés.

$$\mathbb{A}\mathbf{x} = \mathbf{b} \Leftrightarrow \begin{pmatrix} A_{1,1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ A_{n,1} & \dots & \dots & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\forall i \in \llbracket 1, n \rrbracket, \quad b_i =$$

# Résolution système triangulaire inférieur

Soit  $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$  triangulaire inférieure inversible et  $\mathbf{b} \in \mathbb{R}^n$  donnés.

$$\mathbb{A}\mathbf{x} = \mathbf{b} \Leftrightarrow \begin{pmatrix} A_{1,1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ A_{n,1} & \dots & \dots & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\forall i \in \llbracket 1, n \rrbracket, \quad b_i = \sum_{j=1}^n A_{i,j} x_j =$$

# Résolution système triangulaire inférieur

Soit  $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$  triangulaire inférieure inversible et  $\mathbf{b} \in \mathbb{R}^n$  donnés.

$$\mathbb{A}\mathbf{x} = \mathbf{b} \Leftrightarrow \begin{pmatrix} A_{1,1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ A_{n,1} & \dots & \dots & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\forall i \in \llbracket 1, n \rrbracket, \quad b_i = \sum_{j=1}^n A_{i,j}x_j = \sum_{j=1}^i A_{i,j}x_j$$

# Résolution système triangulaire inférieur

Soit  $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$  triangulaire inférieure inversible et  $\mathbf{b} \in \mathbb{R}^n$  donnés.

$$\mathbb{A}\mathbf{x} = \mathbf{b} \Leftrightarrow \begin{pmatrix} A_{1,1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ A_{n,1} & \dots & \dots & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\forall i \in \llbracket 1, n \rrbracket, \quad b_i = \sum_{j=1}^n A_{i,j}x_j = \sum_{j=1}^i A_{i,j}x_j$$

$$\forall i \in \llbracket 1, n \rrbracket, \quad b_i = \sum_{j=1}^{i-1} A_{i,j}x_j + A_{i,i}x_i \quad (25)$$

# Résolution système triangulaire inférieur

## Remarque

La matrice  $\mathbb{A}$  est triangulaire inférieure inversible donc  $A_{i,i} \neq 0$ ,  $\forall i \in \llbracket 1, n \rrbracket$ .

La formule

$$\forall i \in \llbracket 1, n \rrbracket, \quad b_i = \sum_{j=1}^{i-1} A_{i,j}x_j + A_{i,i}x_i \quad (25)$$

peut donc s'écrire

$$\forall i \in \llbracket 1, n \rrbracket, \quad x_i = \frac{1}{A_{i,i}} \left( b_i - \sum_{j=1}^{i-1} A_{i,j}x_j \right) \quad (26)$$

## Remarque

On peut alors calculer successivement  $x_1, x_2, \dots, x_n$ .

# Résolution système triangulaire inférieur

## Algorithme 10- $\mathcal{R}_0$

---

1:

Résoudre  $\mathbb{A}\mathbf{x} = \mathbf{b}$   
en calculant  
successivement  
 $x_1, x_2, \dots, x_n$ .

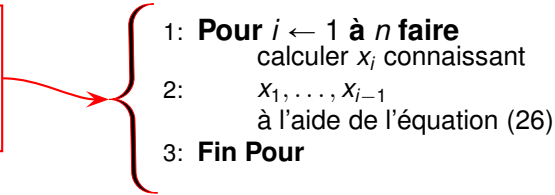
# Résolution système triangulaire inférieur

Algorithme 10-  $\mathcal{R}_0$

Algorithme 10-  $\mathcal{R}_1$

1:

Résoudre  $\mathbb{A}\mathbf{x} = \mathbf{b}$   
en calculant  
successivement  
 $x_1, x_2, \dots, x_n$ .

- 
- 1: **Pour**  $i \leftarrow 1$  à  $n$  **faire**  
calculer  $x_i$  connaissant
  - 2:  $x_1, \dots, x_{i-1}$   
à l'aide de l'équation (26)
  - 3: **Fin Pour**

# Résolution système triangulaire inférieur

## Algorithme 10- $\mathcal{R}_1$

---

1: **Pour**  $i \leftarrow 1$  à  $n$  **faire**

Calculer  $x_i$  connaissant

2:  $x_1, \dots, x_{i-1}$   
à l'aide de l'équation (26)

3: **Fin Pour**

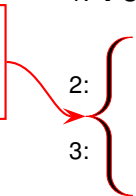
# Résolution système triangulaire inférieur

Algorithme 10-  $\mathcal{R}_1$

- 1: **Pour**  $i \leftarrow 1$  à  $n$  **faire**  
Calculer  $x_i$  connaissant
- 2:  $x_1, \dots, x_{i-1}$   
à l'aide de l'équation (26)
- 3: **Fin Pour**

Algorithme 10-  $\mathcal{R}_2$

- 1: **Pour**  $i \leftarrow 1$  à  $n$  **faire**
- 2:  $S \leftarrow \sum_{j=1}^{i-1} A_{i,j} x_j$
- 3:  $x_i \leftarrow (b_i - S) / A_{i,i}$
- 4: **Fin Pour**



# Résolution système triangulaire inférieur

## Algorithme 10- $\mathcal{R}_2$

---

1: **Pour**  $i \leftarrow 1$  à  $n$  **faire**

2: 
$$S \leftarrow \sum_{j=1}^{i-1} A_{i,j}x_j$$

3:  $x_i \leftarrow (b_i - S)/A_{i,i}$

4: **Fin Pour**

# Résolution système triangulaire inférieur

Algorithme 10-  $\mathcal{R}_2$

1: **Pour**  $i \leftarrow 1$  à  $n$  **faire**

2:  $S \leftarrow \sum_{j=1}^{i-1} A_{i,j}x_j$

3:  $x_i \leftarrow (b_i - S)/A_{i,i}$

4: **Fin Pour**

Algorithme 10-  $\mathcal{R}_3$

1: **Pour**  $i \leftarrow 1$  à  $n$  **faire**

2:  $S \leftarrow 0$

3: **Pour**  $j \leftarrow 1$  à  $i - 1$  **faire**

4:  $S \leftarrow S + A(i,j) * x(j)$

5: **Fin Pour**

6:  $x_i \leftarrow (b_i - S)/A_{i,i}$

7: **Fin Pour**

# Résolution système triangulaire inférieur

---

## Algorithme 10 Fonction RESTRIINF

---

**Données :**  $\mathbb{A}$  : matrice de  $\mathcal{M}_n(\mathbb{R})$  triangulaire inférieure inversible.  
 $\mathbf{b}$  : vecteur de  $\mathbb{R}^n$ .

**Résultat :**  $\mathbf{x}$  : vecteur de  $\mathbb{R}^n$ .

```
1: Fonction  $\mathbf{x} \leftarrow \text{RESTRIINF}(\mathbb{A}, \mathbf{b})$ 
2:   Pour  $i \leftarrow 1$  à  $n$  faire
3:      $S \leftarrow 0$ 
4:     Pour  $j \leftarrow 1$  à  $i - 1$  faire
5:        $S \leftarrow S + A(i, j) * x(j)$ 
6:     Fin Pour
7:      $x(i) \leftarrow (b(i) - S) / A(i, i)$ 
8:   Fin Pour
9:   return  $\mathbf{x}$ 
10: Fin Fonction
```

# Résolution système triangulaire supérieur

Soit  $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$  triangulère supérieure inversible et  $\mathbf{b} \in \mathbb{R}^n$  donnés.

$$\mathbb{A}\mathbf{x} = \mathbf{b} \Leftrightarrow \begin{pmatrix} A_{1,1} & \dots & \dots & A_{1,n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\forall i \in \llbracket 1, n \rrbracket, b_i =$$

# Résolution système triangulaire supérieur

Soit  $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$  triangulère supérieure inversible et  $\mathbf{b} \in \mathbb{R}^n$  donnés.

$$\mathbb{A}\mathbf{x} = \mathbf{b} \Leftrightarrow \begin{pmatrix} A_{1,1} & \dots & \dots & A_{1,n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\forall i \in \llbracket 1, n \rrbracket, \quad b_i = \sum_{j=1}^n A_{i,j} x_j =$$

# Résolution système triangulaire supérieur

Soit  $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$  triangulère supérieure inversible et  $\mathbf{b} \in \mathbb{R}^n$  donnés.

$$\mathbb{A}\mathbf{x} = \mathbf{b} \Leftrightarrow \begin{pmatrix} A_{1,1} & \dots & \dots & A_{1,n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\forall i \in \llbracket 1, n \rrbracket, \quad b_i = \sum_{j=1}^n A_{i,j}x_j = \sum_{j=i}^n A_{i,j}x_j$$

# Résolution système triangulaire supérieur

Soit  $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$  triangulère supérieure inversible et  $\mathbf{b} \in \mathbb{R}^n$  donnés.

$$\mathbb{A}\mathbf{x} = \mathbf{b} \Leftrightarrow \begin{pmatrix} A_{1,1} & \dots & \dots & A_{1,n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\forall i \in \llbracket 1, n \rrbracket, \quad b_i = \sum_{j=1}^n A_{i,j}x_j = \sum_{j=i}^n A_{i,j}x_j$$

$$\forall i \in \llbracket 1, n \rrbracket, \quad b_i = A_{i,i}x_i + \sum_{j=i+1}^n A_{i,j}x_j \quad (27)$$

# Résolution système triangulaire supérieur

## Remarque

La matrice  $\mathbb{A}$  est triangulaire supérieure inversible donc  $A_{i,i} \neq 0$ ,  $\forall i \in \llbracket 1, n \rrbracket$ .

La formule

$$\forall i \in \llbracket 1, n \rrbracket, \quad \forall i \in \llbracket 1, n \rrbracket, \quad b_i = A_{i,i}x_i + \sum_{j=i+1}^n A_{i,j}x_j \quad (27)$$

peut donc s'écrire

$$\forall i \in \llbracket 1, n \rrbracket, \quad x_i = \frac{1}{A_{i,i}} \left( b_i - \sum_{j=i+1}^n A_{i,j}x_j \right) \quad (28)$$

## Remarque

On peut alors calculer successivement  $x_n, x_{n-1}, \dots, x_1$ .

# Résolution système triangulaire supérieur

## Algorithme 11- $\mathcal{R}_0$

---

1:

Résoudre  $\mathbb{A}\mathbf{x} = \mathbf{b}$   
en calculant  
successivement  
 $x_n, x_{n-1}, \dots, x_1$ .

# Résolution système triangulaire supérieur

Algorithme 11-  $\mathcal{R}_0$

Algorithme 11-  $\mathcal{R}_1$

1:

Résoudre  $\mathbb{A}\mathbf{x} = \mathbf{b}$   
en calculant  
successivement  
 $x_n, x_{n-1}, \dots, x_1$ .

- 1: **Pour**  $i \leftarrow n$  à 1 **faire** (**pas**  $-1$ )  
calculer  $x_i$  connaissant
- 2:  $x_{i+1}, \dots, x_n$   
à l'aide de l'équation (28)
- 3: **Fin Pour**

# Résolution système triangulaire supérieur

## Algorithme 11- $\mathcal{R}_1$

---

1: **Pour**  $i \leftarrow n$  à 1 **faire** (pas  $-1$ )

Calculer  $x_i$  connaissant

2:  $x_{i+1}, \dots, x_n$   
à l'aide de l'équation (28)

3: **Fin Pour**

# Résolution système triangulaire supérieur

Algorithme 11-  $\mathcal{R}_1$

1: **Pour**  $i \leftarrow n$  à 1 **faire** (pas  $-1$ )

2: Calculer  $x_i$  connaissant  
 $x_{i+1}, \dots, x_n$   
à l'aide de l'équation (28)

3: **Fin Pour**

Algorithme 11-  $\mathcal{R}_2$

1: **Pour**  $i \leftarrow n$  à 1 **faire** (pas  $-1$ )

2:  $S \leftarrow \sum_{j=i+1}^n A_{i,j}x_j$

3:  $x_i \leftarrow (b_i - S)/A_{i,i}$

4: **Fin Pour**

# Résolution système triangulaire supérieur

## Algorithme 11- $\mathcal{R}_2$

---

1: **Pour**  $i \leftarrow n$  à 1 **faire** (pas  $-1$ )

2: 
$$S \leftarrow \sum_{j=i+1}^n A_{i,j}x_j$$

3:  $x_i \leftarrow (b_i - S)/A_{i,i}$

4: **Fin Pour**

# Résolution système triangulaire supérieur

Algorithme 11-  $\mathcal{R}_2$

1: **Pour**  $i \leftarrow n$  à 1 **faire** (pas -1)

2:  $S \leftarrow \sum_{j=i+1}^n A_{i,j}x_j$

3:  $x_i \leftarrow (b_i - S)/A_{i,i}$

4: **Fin Pour**

Algorithme 11-  $\mathcal{R}_3$

1: **Pour**  $i \leftarrow n$  à 1 **faire** (pas -1)

2:  $S \leftarrow 0$

3: **Pour**  $j \leftarrow i + 1$  à  $n$  **faire**

4:  $S \leftarrow S + A(i,j) * x(j)$

5: **Fin Pour**

6:  $x_i \leftarrow (b_i - S)/A_{i,i}$

7: **Fin Pour**

# Résolution système triangulaire supérieur

---

## Algorithme 11 Fonction RESTRISUP

---

**Données :**  $\mathbb{A}$  : matrice de  $\mathcal{M}_n(\mathbb{R})$  triangulaire supérieur inversible,  
 $\mathbf{b}$  : vecteur de  $\mathbb{R}^n$ .

**Résultat :**  $\mathbf{x}$  : vecteur de  $\mathbb{R}^n$ .

- 1: **Fonction**  $\mathbf{x} \leftarrow \text{RESTRISUP}(\mathbb{A}, \mathbf{b})$
- 2:     **Pour**  $i \leftarrow n$  à 1 **faire** (pas de  $-1$ )
- 3:          $S \leftarrow 0$
- 4:         **Pour**  $j \leftarrow i + 1$  à  $n$  **faire**
- 5:              $S \leftarrow S + A(i, j) * x(j)$
- 6:         **Fin Pour**
- 7:          $x(i) \leftarrow (b(i) - S) / A(i, i)$
- 8:     **Fin Pour**
- 9: **Fin Fonction**

# Factorisation LU

$$\mathbb{A} = \mathbb{L}\mathbb{U} \quad (29)$$

c'est à dire

$$\begin{pmatrix} \mathbb{A}_{1,1} & \dots & \mathbb{A}_{1,n} \\ \vdots & \ddots & \dots \\ \mathbb{A}_{n,1} & \dots & \mathbb{A}_{n,n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \mathbb{L}_{2,1} & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ \mathbb{L}_{n,1} & \dots & \mathbb{L}_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} \mathbb{U}_{1,1} & \dots & \dots & \mathbb{U}_{n,1} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \mathbb{U}_{n,n} \end{pmatrix}.$$

# Factorisation LU

## Remarque

*Par récurrence, en supposant connues*

- *les  $i - 1$  premières lignes de  $\mathbb{U}$*
- *les  $i - 1$  premières colonnes de  $\mathbb{L}$ ,*

*on peut déterminer*

- *la  $i$ -ème ligne de  $\mathbb{U}$*
- *puis la  $i$ -ème colonne de  $\mathbb{L}$*

*par les formules valables **pour  $i$  allant de 1 à  $n$***

# Factorisation LU

## Remarque

Par récurrence, en supposant connues

- les  $i - 1$  premières lignes de  $\mathbb{U}$
- les  $i - 1$  premières colonnes de  $\mathbb{L}$ ,

on peut déterminer

- la  $i$ -ème ligne de  $\mathbb{U}$
- puis la  $i$ -ème colonne de  $\mathbb{L}$

par les formules valables **pour  $i$  allant de 1 à  $n$**

$$\mathbb{U}_{i,j} = \begin{cases} \mathbb{A}_{i,j} - \sum_{k=1}^{i-1} \mathbb{L}_{i,k} \mathbb{U}_{k,j}, & \forall j \in \llbracket i, n \rrbracket. \\ 0, & \forall j \in \llbracket 1, i-1 \rrbracket. \end{cases} \quad (30)$$

# Factorisation LU

## Remarque

Par récurrence, en supposant connues

- les  $i - 1$  premières lignes de  $\mathbb{U}$
- les  $i - 1$  premières colonnes de  $\mathbb{L}$ ,

on peut déterminer

- la  $i$ -ème ligne de  $\mathbb{U}$
- puis la  $i$ -ème colonne de  $\mathbb{L}$

par les formules valables **pour  $i$  allant de 1 à  $n$**

$$\mathbb{U}_{i,j} = \begin{cases} \mathbb{A}_{i,j} - \sum_{k=1}^{i-1} \mathbb{L}_{i,k} \mathbb{U}_{k,j}, & \forall j \in \llbracket i, n \rrbracket. \\ 0, & \forall j \in \llbracket 1, i-1 \rrbracket. \end{cases} \quad (30)$$

**puis**

$$\mathbb{L}_{j,i} = \begin{cases} 0, & \forall j \in \llbracket 1, i-1 \rrbracket. \\ 1, & j = i \\ \frac{1}{\mathbb{U}_{i,i}} \left( \mathbb{A}_{j,i} - \sum_{k=1}^{i-1} \mathbb{L}_{j,k} \mathbb{U}_{k,i} \right), & \forall j \in \llbracket i+1, n \rrbracket, \end{cases} \quad (31)$$

# Factorisation LU

## Algorithme 12- $\mathcal{R}_0$

---

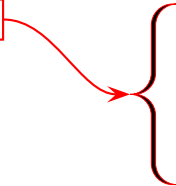
1: Calcul de  $\mathbb{L}$  et  $\mathbb{U}$

# Factorisation LU

Algorithme 12-  $\mathcal{R}_0$

1: Calcul de  $\mathbb{L}$  et  $\mathbb{U}$

Algorithme 12-  $\mathcal{R}_1$

- 
- 1: **Pour**  $i \leftarrow 1$  à  $n$  **faire**
  - 2:   Calcul ligne  $i$  de  $\mathbb{U}$ .
  - 3:   Calcul colonne  $i$  de  $\mathbb{L}$ .
  - 4: **Fin Pour**

# Factorisation LU

## Algorithme 12- $\mathcal{R}_1$

---

- 1: **Pour**  $i \leftarrow 1$  à  $n$  **faire**
- 2: Calcul ligne  $i$  de  $\mathbb{U}$ .
- 3: Calcul colonne  $i$  de  $\mathbb{L}$ .
- 4: **Fin Pour**

# Factorisation LU

## Algorithme 12- $\mathcal{R}_1$

- 1: **Pour**  $i \leftarrow 1$  à  $n$  faire
- 2: Calcul ligne  $i$  de  $\mathbb{U}$ .
- 3: Calcul colonne  $i$  de  $\mathbb{L}$ .
- 4: **Fin Pour**

## Algorithme 12- $\mathcal{R}_2$

- 1:  $\mathbb{U} \leftarrow 0$
- 2:  $\mathbb{L} \leftarrow 0$
- 3: **Pour**  $i \leftarrow 1$  à  $n$  faire
- 4: **Pour**  $j \leftarrow i$  à  $n$  faire
- 5: 
$$\mathbb{U}_{i,j} \leftarrow \mathbb{A}_{i,j} - \sum_{k=1}^{i-1} \mathbb{L}_{i,k} \mathbb{U}_{k,j}$$
- 6: **Fin Pour**
- 7:  $\mathbb{L}_{i,i} \leftarrow 1$
- 8: **Pour**  $j \leftarrow i + 1$  à  $n$  faire
- 9: 
$$\mathbb{L}_{j,i} \leftarrow \frac{1}{\mathbb{U}_{i,i}} \left( \mathbb{A}_{j,i} - \sum_{k=1}^{i-1} \mathbb{L}_{j,k} \mathbb{U}_{k,i} \right)$$
- 10: **Fin Pour**
- 11: **Fin Pour**

# Factorisation LU

## Algorithme 12- $\mathcal{R}_2$

1:  $\mathbb{U} \leftarrow 0$

2:  $\mathbb{L} \leftarrow 0$

3: **Pour**  $i \leftarrow 1$  à  $n$  faire

4:   **Pour**  $j \leftarrow i$  à  $n$  faire

5:     
$$\mathbb{U}_{i,j} \leftarrow \mathbb{A}_{i,j} - \sum_{k=1}^{i-1} \mathbb{L}_{i,k} \mathbb{U}_{k,j}$$

6:   **Fin Pour**

7:    $\mathbb{L}_{j,i} \leftarrow 1$

8:   **Pour**  $j \leftarrow i + 1$  à  $n$  faire

9:     
$$\mathbb{L}_{j,i} \leftarrow \frac{1}{\mathbb{U}_{i,i}} \left( \mathbb{A}_{j,i} - \sum_{k=1}^{i-1} \mathbb{L}_{j,k} \mathbb{U}_{k,i} \right)$$

10:   **Fin Pour**

11: **Fin Pour**

# Factorisation LU

## Algorithme 12- $\mathcal{R}_2$

```
1:  $\mathbb{U} \leftarrow 0$ 
2:  $\mathbb{L} \leftarrow 0$ 
3: Pour  $i \leftarrow 1$  à  $n$  faire
4:   Pour  $j \leftarrow i$  à  $n$  faire
5:     
$$\mathbb{U}_{i,j} \leftarrow A_{i,j} - \sum_{k=1}^{i-1} \mathbb{L}_{i,k} \mathbb{U}_{k,j}$$

6:   Fin Pour
7:    $\mathbb{L}_{i,i} \leftarrow 1$ 
8:   Pour  $j \leftarrow i + 1$  à  $n$  faire
9:     
$$\mathbb{L}_{j,i} \leftarrow \frac{1}{\mathbb{U}_{i,i}} \left( A_{j,i} - \sum_{k=1}^{i-1} \mathbb{L}_{j,k} \mathbb{U}_{k,i} \right)$$

10:  Fin Pour
11: Fin Pour
```

## Algorithme 12- $\mathcal{R}_3$

```
1:  $\mathbb{U} \leftarrow 0$ 
2:  $\mathbb{L} \leftarrow 0$ 
3: Pour  $i \leftarrow 1$  à  $n$  faire
4:   Pour  $j \leftarrow i$  à  $n$  faire
5:     
$$S_1 \leftarrow \sum_{k=1}^{i-1} \mathbb{L}_{i,k} \mathbb{U}_{k,j}$$

6:     
$$\mathbb{U}_{i,j} \leftarrow A_{i,j} - S_1$$

7:   Fin Pour
8:    $\mathbb{L}_{i,i} \leftarrow 1$ 
9:   Pour  $j \leftarrow i + 1$  à  $n$  faire
10:     
$$S_2 \leftarrow \sum_{k=1}^{i-1} \mathbb{L}_{j,k} \mathbb{U}_{k,i}$$

11:     
$$\mathbb{L}_{j,i} \leftarrow \frac{1}{\mathbb{U}_{i,i}} (A_{j,i} - S_2).$$

12:  Fin Pour
13: Fin Pour
```

# Factorisation LU

## Algorithme 12- $\mathcal{R}_3$

1:  $\mathbb{U} \leftarrow 0$

2:  $\mathbb{L} \leftarrow 0$

3: **Pour**  $i \leftarrow 1$  à  $n$  faire

4:   **Pour**  $j \leftarrow i$  à  $n$  faire

5:     
$$S_1 \leftarrow \sum_{k=1}^{i-1} L_{i,k} U_{k,j}$$

6:     
$$U_{i,j} \leftarrow A_{i,j} - S_1$$

7:     **Fin Pour**

8:     
$$L_{i,i} \leftarrow 1$$

9:     **Pour**  $j \leftarrow i + 1$  à  $n$  faire

10:     
$$S_2 \leftarrow \sum_{k=1}^{i-1} L_{j,k} U_{k,i}$$

11:     
$$L_{j,i} \leftarrow \frac{1}{U_{i,i}} (A_{j,i} - S_2).$$

12:     **Fin Pour**

13: **Fin Pour**

# Factorisation LU

## Algorithme 12- $\mathcal{R}_3$

```
1:  $\mathbb{U} \leftarrow 0$ 
2:  $\mathbb{L} \leftarrow 0$ 
3: Pour  $i \leftarrow 1$  à  $n$  faire
4:   Pour  $j \leftarrow i$  à  $n$  faire
5:      $S_1 \leftarrow \sum_{k=1}^{i-1} L_{i,k} U_{k,j}$ 
6:      $U_{i,j} \leftarrow A_{i,j} - S_1$ 
7:   Fin Pour
8:    $L_{i,i} \leftarrow 1$ 
9:   Pour  $j \leftarrow i + 1$  à  $n$  faire
10:     $S_2 \leftarrow \sum_{k=1}^{i-1} L_{j,k} U_{k,i}$ 
11:     $L_{j,i} \leftarrow \frac{1}{U_{i,i}} (A_{j,i} - S_2)$ 
12:   Fin Pour
13: Fin Pour
```

## Algorithme 12- $\mathcal{R}_4$

```
1:  $\mathbb{U} \leftarrow 0$ 
2:  $\mathbb{L} \leftarrow 0$ 
3: Pour  $i \leftarrow 1$  à  $n$  faire
4:   Pour  $j \leftarrow i$  à  $n$  faire
5:      $S_1 \leftarrow 0$ 
6:     Pour  $k \leftarrow 1$  à  $i - 1$  faire
7:        $S_1 \leftarrow S_1 + L_{i,k} * U_{k,j}$ 
8:     Fin Pour
9:      $U_{i,j} \leftarrow A_{i,j} - S_1$ 
10:   Fin Pour
11:    $L_{i,i} \leftarrow 1$ 
12:   Pour  $j \leftarrow i + 1$  à  $n$  faire
13:      $S_2 \leftarrow 0$ 
14:     Pour  $k \leftarrow 1$  à  $i - 1$  faire
15:        $S_2 \leftarrow S_2 + L_{j,k} * U_{k,i}$ 
16:     Fin Pour
17:      $L_{j,i} \leftarrow \frac{1}{U_{i,i}} (A_{j,i} - S_2)$ 
18:   Fin Pour
19: Fin Pour
```

# Factorisation LU

---

## Algorithme 12 Fonction FACTLU

---

**Données :**  $\mathbb{A}$  : matrice de  $\mathcal{M}_n(\mathbb{R})$  ..

**Résultat :**  $\mathbb{L}$  : matrice de  $\mathcal{M}_n(\mathbb{R})$  triangulaire inférieure  
avec  $\mathbb{L}_{i,i} = 1, \forall i \in \llbracket 1, n \rrbracket$

$\mathbb{U}$  : matrice de  $\mathcal{M}_n(\mathbb{R})$  triangulaire supérieure.

```
1: Fonction [ $\mathbb{L}, \mathbb{U}$ ]  $\leftarrow$  FACTLU( $\mathbb{A}$ )
2:    $\mathbb{U} \leftarrow 0$ 
3:    $\mathbb{L} \leftarrow 0$ 
4:   Pour  $i \leftarrow 1$  à  $n$  faire
5:     Pour  $j \leftarrow i$  à  $n$  faire
6:        $S_1 \leftarrow 0$ 
7:       Pour  $k \leftarrow 1$  à  $i - 1$  faire
8:          $S_1 \leftarrow S_1 + \mathbb{L}(i, k) * \mathbb{U}(k, j)$ 
9:       Fin Pour
10:       $\mathbb{U}(i, j) \leftarrow \mathbb{A}(i, j) - S_1$ 
11:     Fin Pour
12:      $\mathbb{L}(i, i) \leftarrow 1$ 
13:     Pour  $j \leftarrow i + 1$  à  $n$  faire
14:        $S_2 \leftarrow 0$ 
15:       Pour  $k \leftarrow 1$  à  $i - 1$  faire
16:          $S_2 \leftarrow S_2 + \mathbb{L}(j, k) * \mathbb{U}(k, i)$ 
17:       Fin Pour
18:        $\mathbb{L}(j, i) \leftarrow (\mathbb{A}_{j,i} - S_2) / \mathbb{U}(i, i)$ .
19:     Fin Pour
20:   Fin Pour
21: Fin Fonction
```

# Factorisation de Cholesky

## Définition

Soit  $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$  une matrice symétrique. On dit que  $\mathbb{A}$  est symétrique définie positive si

$$\langle \mathbb{A}\mathbf{x}, \mathbf{x} \rangle > 0, \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

## Théorème (Factorisation de Cholesky)

*Une matrice  $\mathbb{A} \in \mathcal{M}_n(\mathbb{R})$  est symétrique définie positive si et seulement si il existe une matrice  $\mathbb{L} \in \mathcal{M}_n(\mathbb{R})$  inversible triangulaire inférieure telle que*

$$\mathbb{A} = \mathbb{L}\mathbb{L}^t.$$

*Si les éléments diagonaux de  $\mathbb{L}$  sont choisis strictement positifs, la factorisation est unique.*

## Exercice

### Exercice

*Ecrire une fonction `CHOLESKY` permettant à partir d'une matrice symétrique définie positive d'obtenir sa factorisation positive de Cholesky.*

### Exercice

*Ecrire une fonction `RSLCHOLESKY` permettant de résoudre un système linéaire en utilisant la factorisation positive de Cholesky. On supposera les fonctions `RESTRINF` et `RESTRISUP`, permettant respectivement la résolution d'un système triangulaire inférieur et triangulaire supérieur, déjà écrites.*

### Exercice

*Proposer un(des) programme(s) de validation des différentes fonctions.*