

Plan des cours de méthodes numériques

Comment interpoler
une fonction ?

Comment approximer
une fonction ?

Comment intégrer
numériquement
une fonction ?

Comment dériver
numériquement
une fonction ?

Comment résoudre
numériquement un problème
aux
valeurs initiales ?

Comment résoudre
numériquement un problème
aux
conditions frontières ?

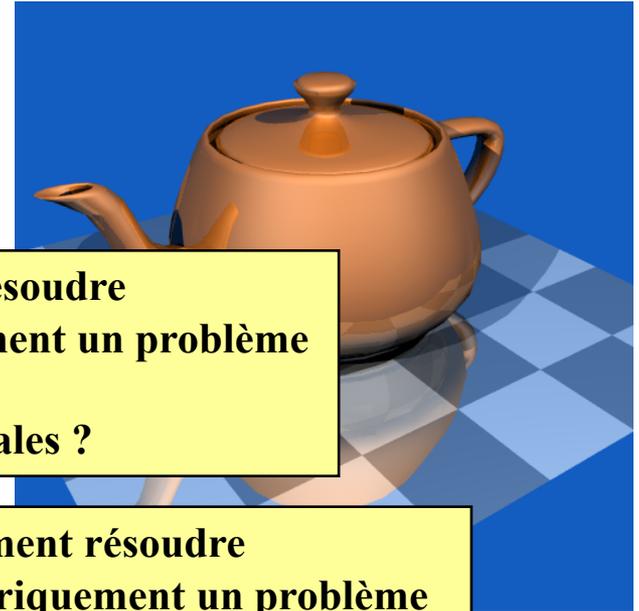
Et les équations
non linéaires ?

Et les méthodes itératives ?

Comment résoudre
numériquement une
équation aux dérivées
partielles ?

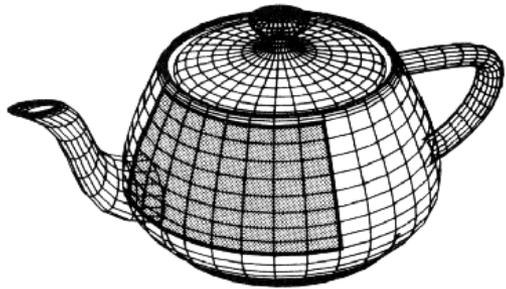
*Comment résoudre numériquement
une équation différentielle ordinaire ?*

*Comment résoudre numériquement
une équation aux dérivées partielles ?*

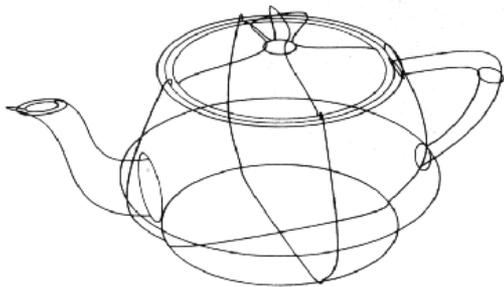


The Utah Teapot

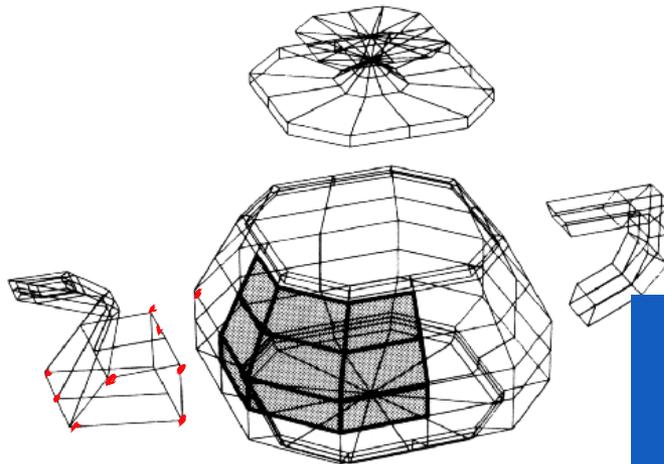
Martin Nevell (1975)



Single shaded patch



Patch edges



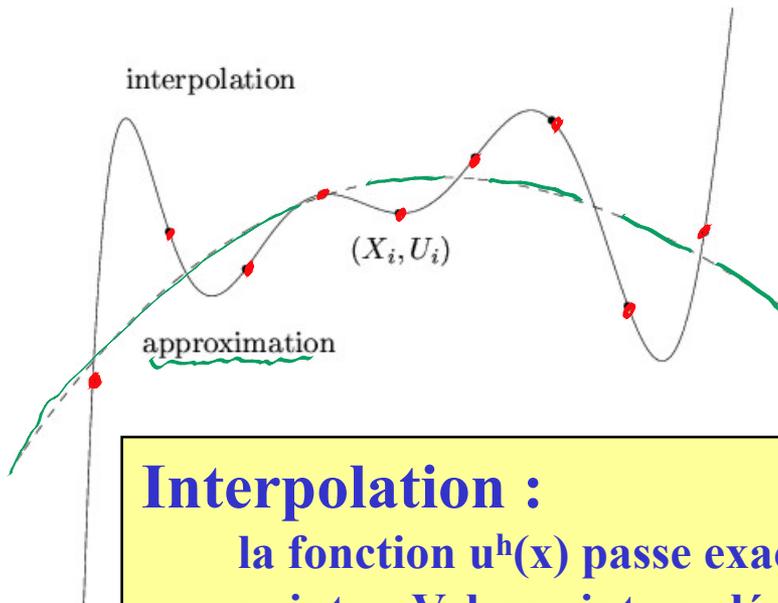
Control points



32 patches



Interpolation, approximation et extrapolation...



Interpolation :

la fonction $u^h(x)$ passe exactement par les points. Valeurs interpolées entre les points et valeurs **extrapolées** hors de l'intervalle.

Approximation :

la fonction $u^h(x)$ ne passe pas par les points, mais s'en rapproche selon un critère à définir

Fonctions de base
spécifiées a priori

$$u(x) \approx u^h(x) = \sum_{j=0}^n a_j \phi_j(x)$$

Paramètres inconnus

Beaucoup de données...

$m = 11$



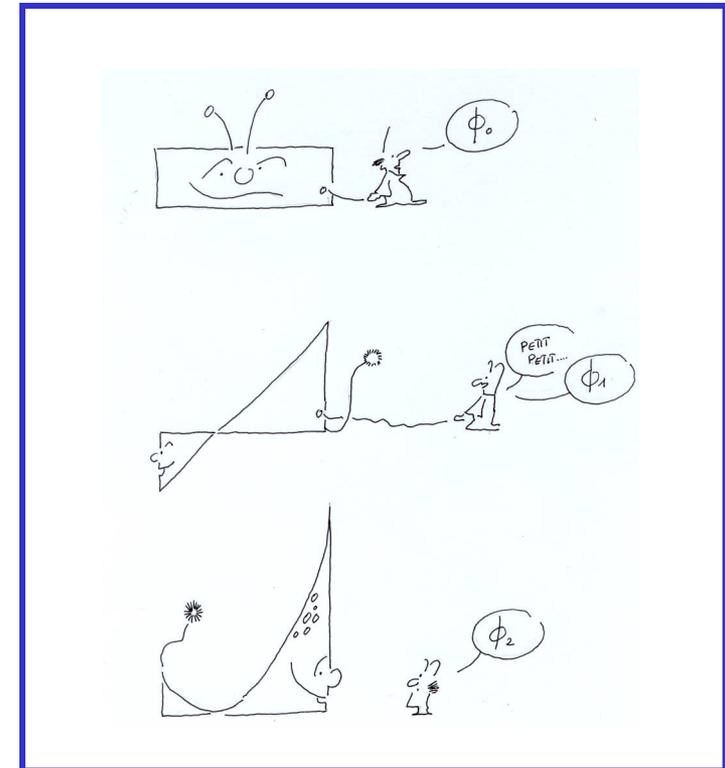
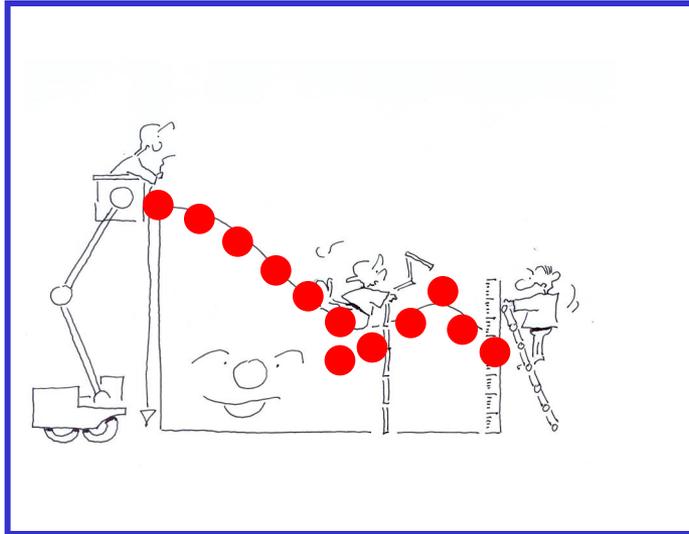
$(X_i, U_i), \quad i = 0, 1, 2, \dots, m.$

$n < m$

$n = 2$

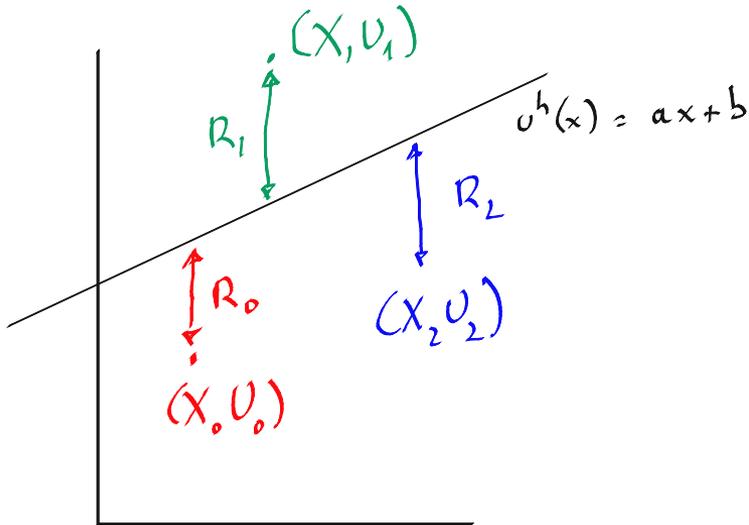
$$u^h(x) = \sum_{j=0}^n a_j \phi_j(x)$$

...peu de paramètres !



Approximation aux moindres carrés...

$$\begin{aligned} U_0 &\simeq aX_0 + b \\ U_1 &\simeq aX_1 + b \\ U_2 &\simeq aX_2 + b \end{aligned}$$

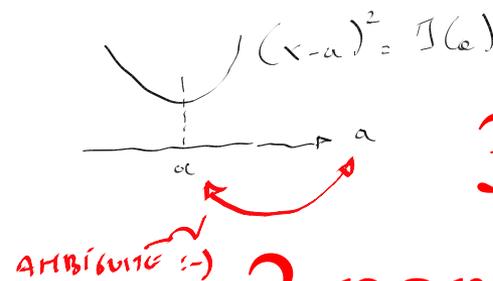
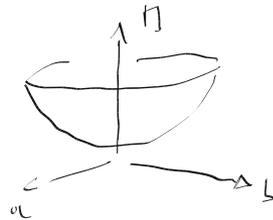


A MINIMISER

$$R_0^2 + R_1^2 + R_2^2 = (U_0 - aX_0 - b)^2 + (U_1 - aX_1 - b)^2 + (U_2 - aX_2 - b)^2$$

$$J(a, b) = (U_0 - aX_0 - b)^2 + (U_1 - aX_1 - b)^2 + (U_2 - aX_2 - b)^2$$

$$\begin{cases} \frac{\partial J}{\partial a} \Big|_{(a,b)} = 0 \\ \frac{\partial J}{\partial b} \Big|_{(a,b)} = 0 \end{cases}$$



3 points

2 paramètres

Equations normales

$$J(a,b) = (U_0 - aX_0 - b)^2 + (U_1 - aX_1 - b)^2 + (U_2 - aX_2 - b)^2$$

$$\begin{cases} \frac{\partial J}{\partial a} \Big|_{(a,b)} = 0 \\ \frac{\partial J}{\partial b} \Big|_{(a,b)} = 0 \end{cases}$$

$$0 = \frac{\partial J}{\partial a} = \cancel{2} \left[(U_0 - aX_0 - b)X_0 + (U_1 - aX_1 - b)X_1 + (\dots)X_2 \right]$$

$$0 = \frac{\partial J}{\partial b} = \cancel{2} \left[(\dots - b) + (\dots) + (\dots) \right]$$

$$\begin{cases} a \sum X_i^2 + b \sum X_i = \sum X_i U_i \\ a \sum X_i + b \sum 1 = \sum U_i \end{cases}$$

$$\begin{bmatrix} \sum X_i^2 & \sum X_i \\ \sum X_i & \sum 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum X_i U_i \\ \sum U_i \end{bmatrix}$$

EQUATIONS
NORMALES

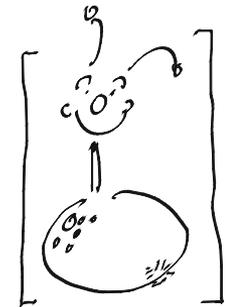
Généralisons un peu...

$$v^b(x) = a_0 \underbrace{\phi_0(x)}_x + a_1 \underbrace{\phi_1(x)}_1$$

$$\begin{bmatrix} \sum_i \phi_0(x_i) \phi_0(x_i) & \sum_i \phi_0(x_i) \phi_1(x_i) \\ \sum_i \phi_1(x_i) \phi_0(x_i) & \sum_i \phi_1(x_i) \phi_1(x_i) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_i v_i \phi_0(x_i) \\ \sum_i v_i \phi_1(x_i) \end{bmatrix}$$

$$\sum_{j=0}^n \sum_{l=0}^m \phi_k(x_i) \phi_l(x_i) a_j = \sum_{l=0}^m \phi_k(x_i) v_i$$

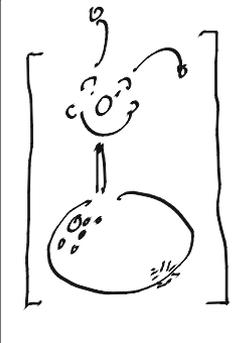
$k = 0 \dots n$



$$\begin{bmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & \sum 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum x_i v_i \\ \sum v_i \end{bmatrix}$$

EQUATIONS
NORMALES

$$\sum_{j=0}^n \sum_{l=0}^m \underbrace{\phi_k(x_i)}_{\text{row } i} \underbrace{\phi_j(x_i)}_{\text{col } j} a_j = \sum_{k=0}^m \phi_k(x_i) v_i$$



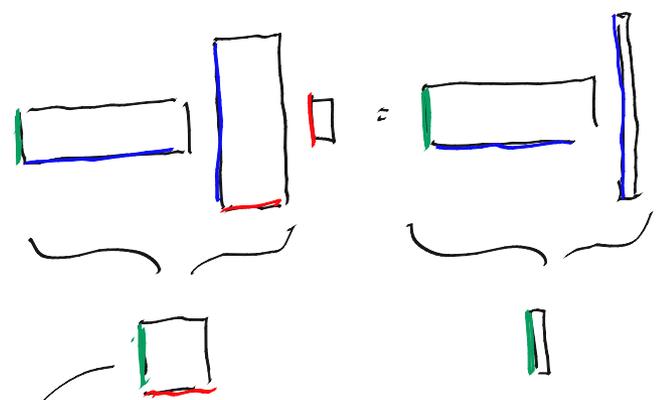
$$\underline{\underline{A}}_{ik}$$

$$\underline{\underline{A}}_{ij}$$

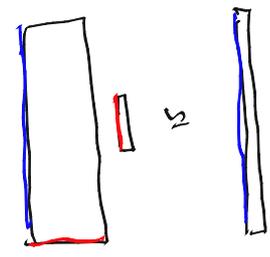
LIGNES $m=3$
COLONNES $n=2$

$$\underline{\underline{A}}^T \cdot \underline{\underline{A}} \cdot s_a = \underline{\underline{A}}^T \cdot s_c$$

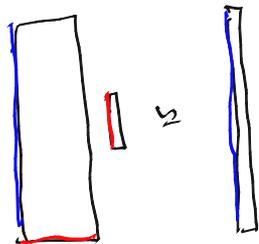
$$\underline{\underline{A}}_{ij} = \begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) \\ \phi_0(x_1) & \phi_1(x_1) \\ \phi_0(x_2) & \phi_1(x_2) \end{bmatrix}$$



$$\begin{bmatrix} \sum \phi_0 \phi_0 & \sum \phi_0 \phi_1 \\ \sum \phi_1 \phi_0 & \sum \phi_1 \phi_1 \end{bmatrix}$$



La fonction $u^h(x)$ ne
 passe pas par les points,
 mais s'en rapproche
 selon un critère
 à définir...



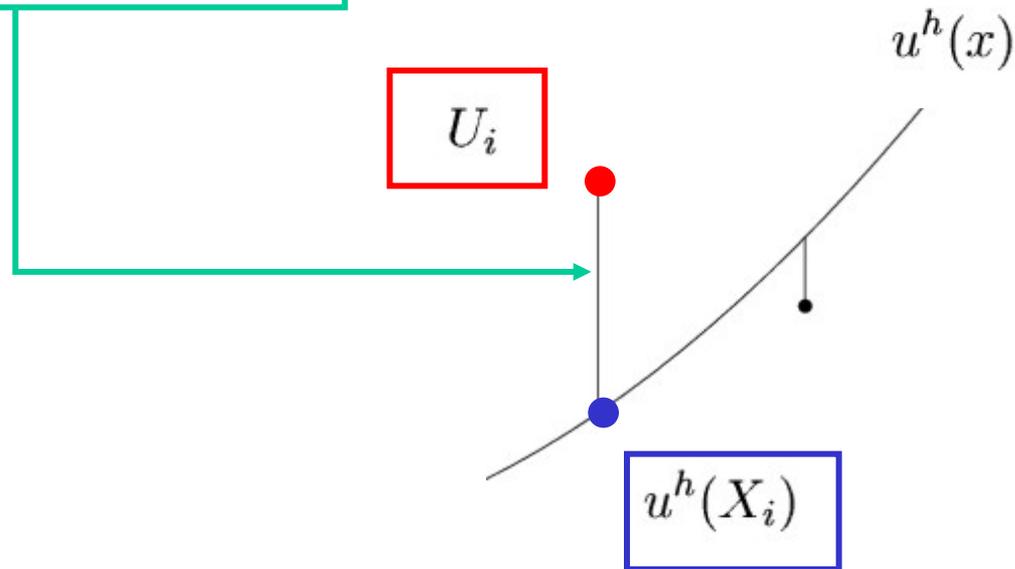
$$u^h(X_i) \approx u(X_i)$$

$$\begin{bmatrix}
 \phi_0(X_0) & \phi_1(X_0) & \dots & \phi_n(X_0) \\
 \phi_0(X_1) & \phi_1(X_1) & \dots & \phi_n(X_1) \\
 \phi_0(X_2) & \phi_1(X_2) & \dots & \phi_n(X_2) \\
 \phi_0(X_3) & \phi_1(X_3) & \dots & \phi_n(X_3) \\
 \phi_0(X_4) & \phi_1(X_4) & \dots & \phi_n(X_4) \\
 \phi_0(X_5) & \phi_1(X_5) & \dots & \phi_n(X_5) \\
 \phi_0(X_6) & \phi_1(X_6) & \dots & \phi_n(X_6) \\
 \phi_0(X_7) & \phi_1(X_7) & \dots & \phi_n(X_7) \\
 \vdots & \vdots & & \vdots \\
 \phi_0(X_m) & \phi_1(X_m) & \dots & \phi_n(X_m)
 \end{bmatrix}
 \begin{bmatrix}
 a_0 \\
 a_1 \\
 \vdots \\
 a_n
 \end{bmatrix}
 \approx
 \begin{bmatrix}
 U_0 \\
 U_1 \\
 U_2 \\
 U_3 \\
 U_4 \\
 U_5 \\
 U_6 \\
 U_7 \\
 \vdots \\
 U_m
 \end{bmatrix}$$

Minimisons les résidus

$$R_i = U_i - \underbrace{\sum_{j=0}^n \phi_j(X_i) a_j}_{u^h(X_i)}$$

$$i = 0, 1, 2, \dots, m.$$



Problème de l'approximation

Trouver $(a_0, \dots, a_n) \in \mathbb{R}^{n+1}$ tels que

$$\underbrace{\sum_{i=0}^m \left(U_i - \sum_{j=0}^n \phi_j(X_i) a_j \right)^2}_{J(a_0, \dots, a_n)} \text{ soit minimal.}$$

Il s'agit donc de minimiser J qui est une fonction à $n+1$ variables


$$\frac{\partial J}{\partial a_k} \Big|_{(a_0, \dots, a_n)} = 0$$

Il faut que le vecteur gradient s'annule et aussi que les conditions du second ordre sur la matrice hessienne soient satisfaites

Calculons...

$$\underbrace{\sum_{i=0}^m \left(U_i - \sum_{j=0}^n \phi_j(X_i) a_j \right)^2}_{J(a_0, \dots, a_n)}$$

$$\left. \frac{\partial J}{\partial a_k} \right|_{(a_0, \dots, a_n)} = 0$$

$$k = 0, 1, \dots, n$$

$$\sum_{i=0}^m -2\phi_k(X_i) \left(U_i - \sum_{j=0}^n \phi_j(X_i) a_j \right) = 0$$

$$k = 0, 1, \dots, n$$

$$\sum_{i=0}^m \phi_k(X_i) \sum_{j=0}^n \phi_j(X_i) a_j = \sum_{i=0}^m \phi_k(X_i) U_i$$

$$k = 0, 1, \dots, n$$

$$\sum_{j=0}^n \left(\sum_{i=0}^m \phi_k(X_i) \phi_j(X_i) \right) a_j = \sum_{i=0}^m \phi_k(X_i) U_i$$

$$k = 0, 1, \dots, n$$

Et pratiquement...

$$\begin{bmatrix} \sum_{i=0}^m \phi_0(X_i)\phi_0(X_i) & \sum_{i=0}^m \phi_0(X_i)\phi_1(X_i) & \dots & \sum_{i=0}^m \phi_0(X_i)\phi_n(X_i) \\ \sum_{i=0}^m \phi_1(X_i)\phi_0(X_i) & \sum_{i=0}^m \phi_1(X_i)\phi_1(X_i) & \dots & \sum_{i=0}^m \phi_1(X_i)\phi_n(X_i) \\ \sum_{i=0}^m \phi_2(X_i)\phi_0(X_i) & \sum_{i=0}^m \phi_2(X_i)\phi_1(X_i) & \dots & \sum_{i=0}^m \phi_2(X_i)\phi_n(X_i) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^m \phi_n(X_i)\phi_0(X_i) & \sum_{i=0}^m \phi_n(X_i)\phi_1(X_i) & \dots & \sum_{i=0}^m \phi_n(X_i)\phi_n(X_i) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^m \phi_0(X_i)U_i \\ \sum_{i=0}^m \phi_1(X_i)U_i \\ \sum_{i=0}^m \phi_2(X_i)U_i \\ \vdots \\ \sum_{i=0}^m \phi_n(X_i)U_i \end{bmatrix}$$

Equations normales

$$A_{ik}$$

$$A_{ki}^T$$

$$A_{ij}$$

Problème de l'approximation

Trouver $(a_0, \dots, a_n) \in \mathbb{R}^{n+1}$ tels que

$$\sum_{j=0}^n \left(\sum_{i=0}^m \phi_k(X_i) \phi_j(X_i) \right) a_j = \sum_{i=0}^m \phi_k(X_i) U_i \quad k = 0, 1, \dots, n$$

Equations normales sous forme matricielle $A^T A \mathbf{a} = A^T \mathbf{u}$

Problème de l'interpolation

A est une matrice carrée

$$\begin{bmatrix} \phi_0(X_0) & \phi_1(X_0) & \dots & \phi_n(X_0) \\ \phi_0(X_1) & \phi_1(X_1) & \dots & \phi_n(X_1) \\ \phi_0(X_2) & \phi_1(X_2) & \dots & \phi_n(X_2) \\ \phi_0(X_3) & \phi_1(X_3) & \dots & \phi_n(X_3) \\ \phi_0(X_4) & \phi_1(X_4) & \dots & \phi_n(X_4) \\ \vdots & \vdots & & \vdots \\ \phi_0(X_n) & \phi_1(X_n) & \dots & \phi_n(X_n) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ \vdots \\ U_n \end{bmatrix}$$

Equations de l'interpolation sous forme matricielle $A a = u$

On voit bien que dans le cas $n=m$, les équations normales fournissent la même solution que les équations de l'interpolation...

Cas particulier : Régression polynomiale

$$\phi_j(x) = x^j \quad j = 0, 1, 2, \dots, n.$$


$$u^h(x) = a_0 + a_1x + a_2x^2$$

$$\begin{bmatrix} (m+1) & \sum_{i=0}^m X_i & \sum_{i=0}^m X_i^2 \\ \sum_{i=0}^m X_i & \sum_{i=0}^m X_i^2 & \sum_{i=0}^m X_i^3 \\ \sum_{i=0}^m X_i^2 & \sum_{i=0}^m X_i^3 & \sum_{i=0}^m X_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^m U_i \\ \sum_{i=0}^m X_i U_i \\ \sum_{i=0}^m X_i^2 U_i \end{bmatrix}$$

Exemple

```
from numpy import *  
from matplotlib import pyplot as plt
```

```
X = [ -55, -25,  5,  35,  65]
```

```
U = [3.25, 3.20, 3.02, 3.32, 3.10]
```

```
a = polyfit(X,U,1)
```

DEGRÉ DE L'APPROXIMATION

```
x = linspace(X[0],X[-1],100)
```

```
uh = polyval(a,x)
```

```
plt.plot(x,uh,'-b',X,U,'or')
```

```
plt.show()
```

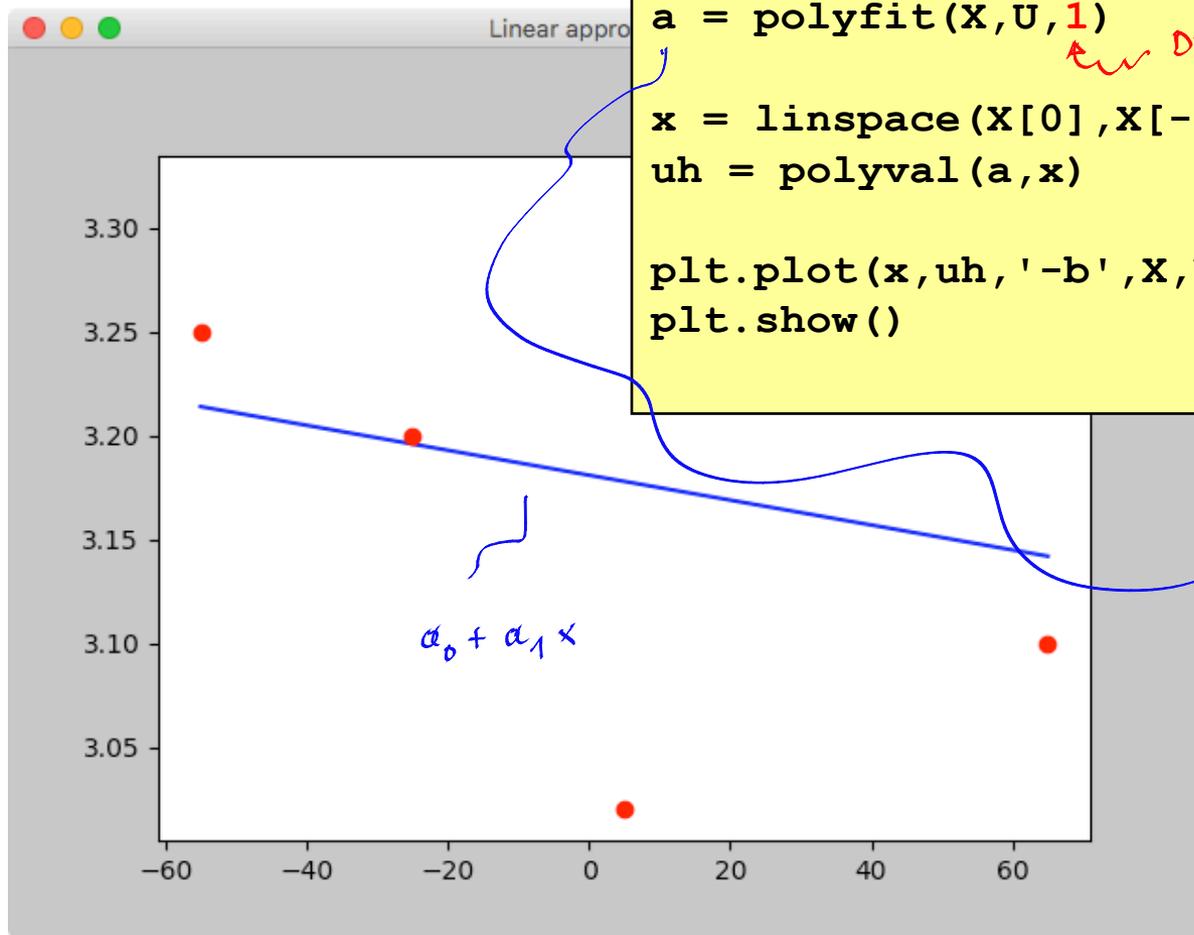


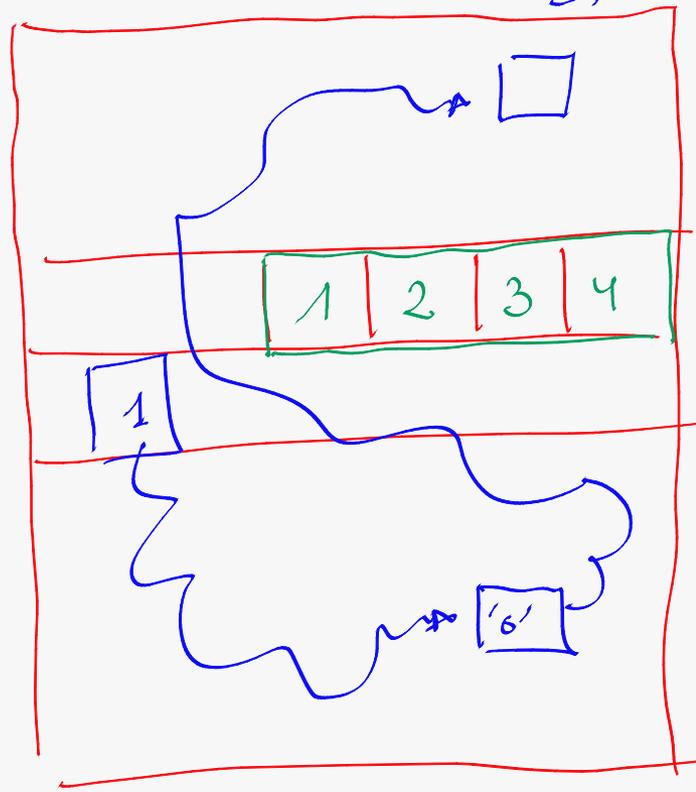
TABLEAU
coef([1,2,3])

$[a_0 a_1]$

LISTE

$[1, 'a00', [1,2,3], \dots]$

c:\v\ } 'dao'

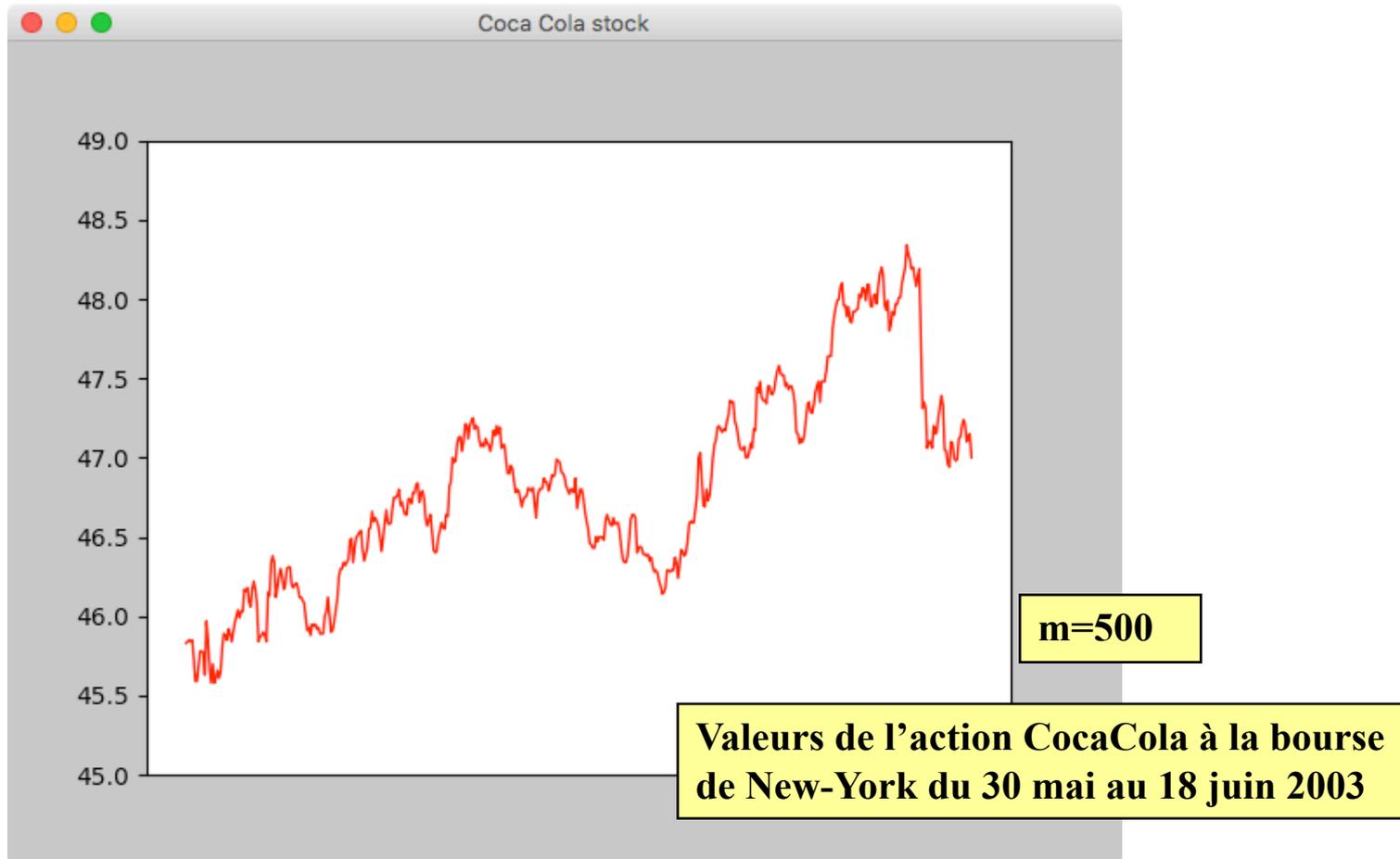


array([1, 2, 3, 4])

NUMPY
TABLEAU

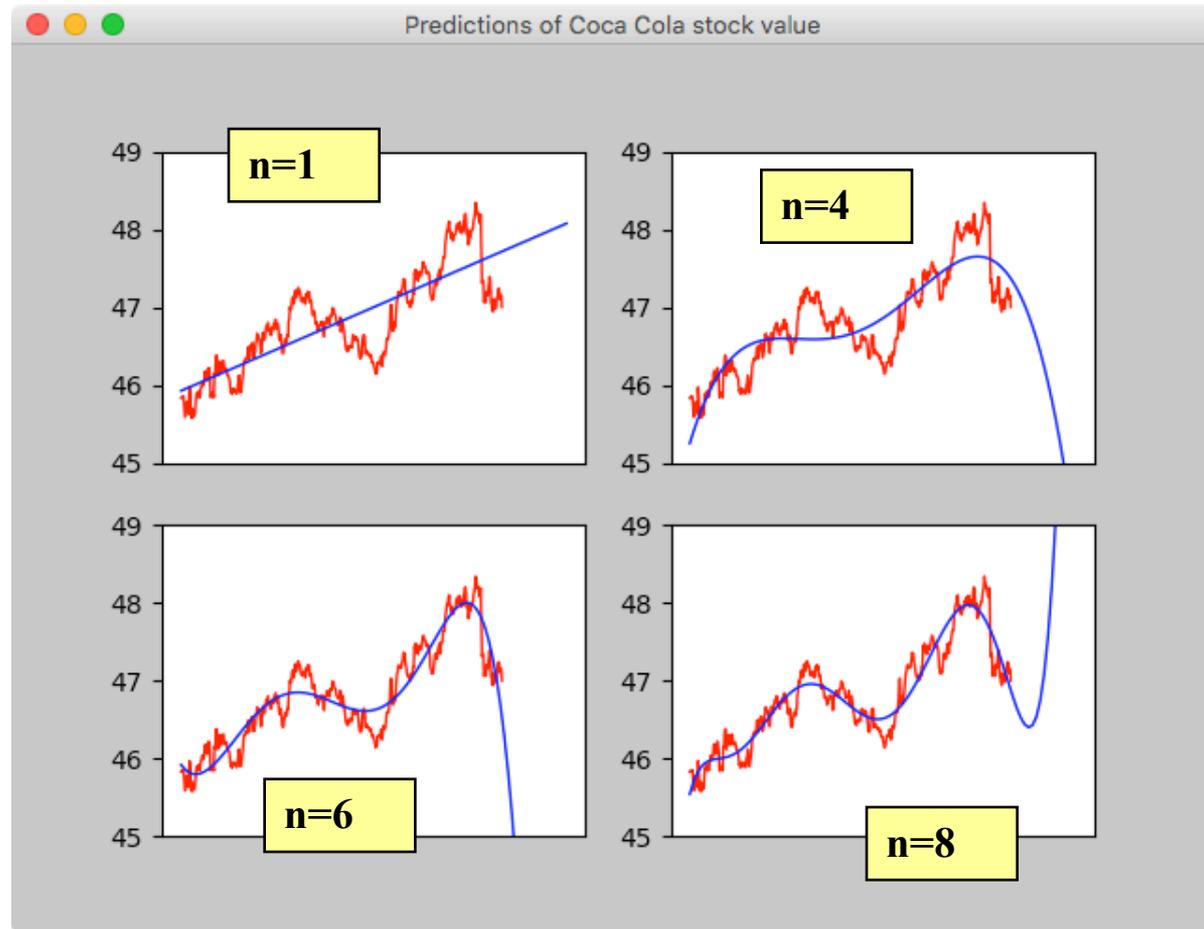
[1, 'a', [1, 2]]

Beaucoup de données

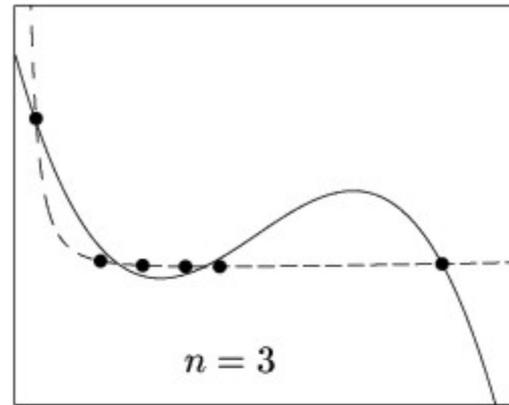
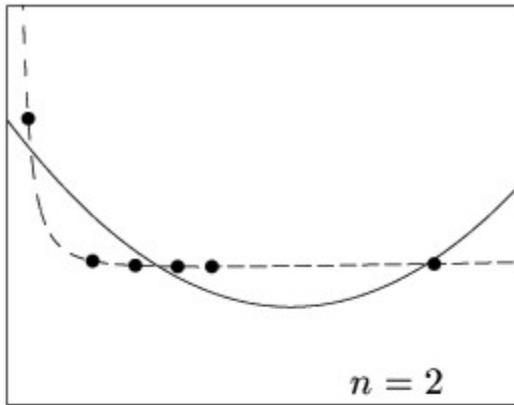


Faisons
des
régressions
et

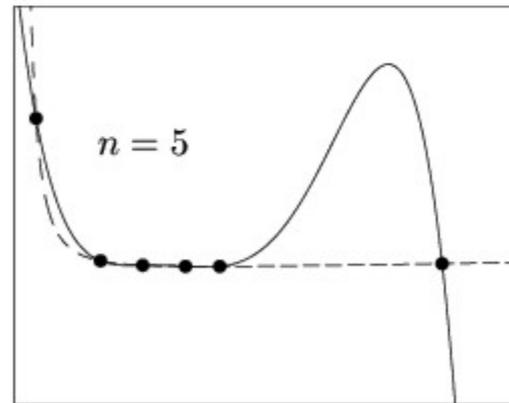
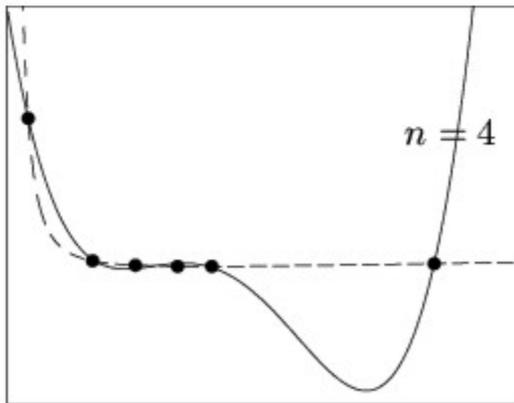
extrapolons les valeurs pour
devenir riches :-)



...les approximations polynomiales divergent aussi !



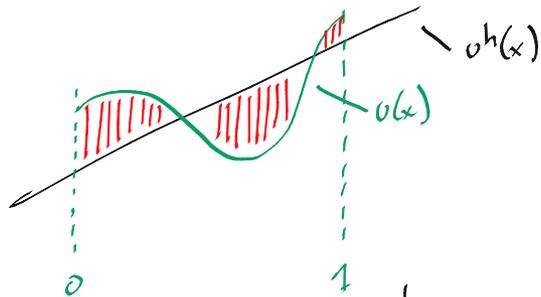
$$u(x) = 1.44x^{-2} + 0.24$$



Et si nous avions $u(x)$...

$$\begin{bmatrix} \int x^2 & \int x \\ \int x & \int 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \int u(x)x \\ \int u(x) \end{bmatrix}$$

EQUATIONS
NORMALES



$$J(a,b) = \int_0^1 (u(x) - ax - b)^2 dx$$

$$0 = \frac{\partial J}{\partial a} \quad \begin{cases} \int_0^1 \cancel{2x} (u(x) - ax - b) dx = 0 \\ \int_0^1 \cancel{2} (u(x) - ax - b) dx = 0 \end{cases}$$

$$\begin{cases} a \int x^2 + b \int x = \int u(x)x \\ a \int x + b \int 1 = \int u(x) \end{cases}$$

$$\begin{bmatrix} \sum X_i^2 & \sum X_i \\ \sum X_i & \sum 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum X_i U_i \\ \sum U_i \end{bmatrix}$$

EQUATIONS
NORMALES

$$\sum_{j=0}^n \int_0^1 \phi_k(x) \phi_j(x) dx \quad a_j = \int_0^1 \phi_k(x) u(x) dx$$

$k = 0, n$



$$\int_0^1 \cancel{2x} (v(x) - ax - b) dx = 0$$

$$\int_0^1 \cancel{2} (v(x) - ax - b) dx = 0$$

$$\begin{cases} a \int x^2 + b \int x = \int v(x)x \\ a \int x + b \int 1 = \int v(x) \end{cases}$$

$$\int_0^1 \cancel{2x} [v(x) - ax - b] dx = 0$$

$$\int_0^1 x v(x) - ax^2 - bx dx = 0$$

$$\int_0^1 x v(x) dx - a \int_0^1 x^2 dx - b \int_0^1 x dx = 0$$

$$a \int_0^1 x^2 dx + b \int_0^1 x dx = \int_0^1 x v(x) dx$$

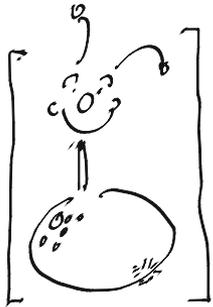
$$\sum_{j=0}^n \int_0^1 \phi_k(x) \phi_j(x) dx a_j = \int_0^1 \phi_k(x) u(x) dx$$

$k = 0, n$

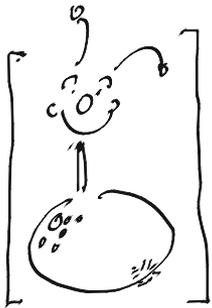


$$\sum_{j=0}^n \sum_{l=0}^m \phi_k(x_l) \phi_j(x_l) a_j = \sum_{l=0}^m \phi_k(x_l) u_l$$

$k = 0, \dots, n$



EST
APPROCHÉ DE
MANIÈRE
SOMMAIRE
PAR...



CAR $\int \approx \sum_{l=0}^m$

Et si nous
avons la
fonction
 $u(x)$...

$$J(a_0, a_1, \dots, a_n) = \int_a^b \left(u(x) - \sum_{j=0}^n \phi_j(x) a_j \right)^2 dx.$$

$$\left. \frac{\partial J}{\partial a_k} \right|_{(a_0, \dots, a_n)} = 0 \quad k = 0, 1, \dots, n$$

$$\int_a^b -2\phi_k(x) \left(u(x) - \sum_{j=0}^n \phi_j(x) a_j \right) dx = 0 \quad k = 0, 1, \dots, n$$

$$\sum_{j=0}^n \left(\int_a^b \phi_k(x) \phi_j(x) dx \right) a_j = \int_a^b \phi_k(x) u(x) dx \quad k = 0, 1, \dots, n$$

$$\mathbf{B} \mathbf{a} = \mathbf{c}$$

Problème intégral de l'approximation

Trouver $(a_0, \dots, a_n) \in \mathbb{R}^{n+1}$ tels que

$$\sum_{j=0}^n \left(\int_a^b \phi_k(x) \phi_j(x) dx \right) a_j = \int_a^b \phi_k(x) u(x) dx \quad k = 0, 1, \dots, n$$

Les équations normales ne sont qu'une approximation de la forme intégrale par une somme finie

Trouver $(a_0, \dots, a_n) \in \mathbb{R}^{n+1}$ tels que

$$\sum_{j=0}^n \left(\sum_{i=0}^m \phi_k(X_i) \phi_j(X_i) \right) a_j = \sum_{i=0}^m \phi_k(X_i) U_i \quad k = 0, 1, \dots, n$$

Et pratiquement...

$$\begin{bmatrix} \int_a^b \phi_0(x)\phi_0(x)dx & \int_a^b \phi_0(x)\phi_1(x)dx & \dots & \int_a^b \phi_0(x)\phi_n(x)dx \\ \int_a^b \phi_1(x)\phi_0(x)dx & \int_a^b \phi_1(x)\phi_1(x)dx & \dots & \int_a^b \phi_1(x)\phi_n(x)dx \\ \int_a^b \phi_2(x)\phi_0(x)dx & \int_a^b \phi_2(x)\phi_1(x)dx & \dots & \int_a^b \phi_2(x)\phi_n(x)dx \\ \vdots & \vdots & \vdots & \vdots \\ \int_a^b \phi_n(x)\phi_0(x)dx & \int_a^b \phi_n(x)\phi_1(x)dx & \dots & \int_a^b \phi_n(x)\phi_n(x)dx \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \int_a^b \phi_0(x)u(x)dx \\ \int_a^b \phi_1(x)u(x)dx \\ \int_a^b \phi_2(x)u(x)dx \\ \vdots \\ \int_a^b \phi_n(x)u(x)dx \end{bmatrix}$$

*Calcul de la projection
orthogonale de $u(x)$ pour le
produit scalaire L^2*

Une jolie interprétation...

$$\langle f, g \rangle = \int_0^1 f(x) g(x) dx$$

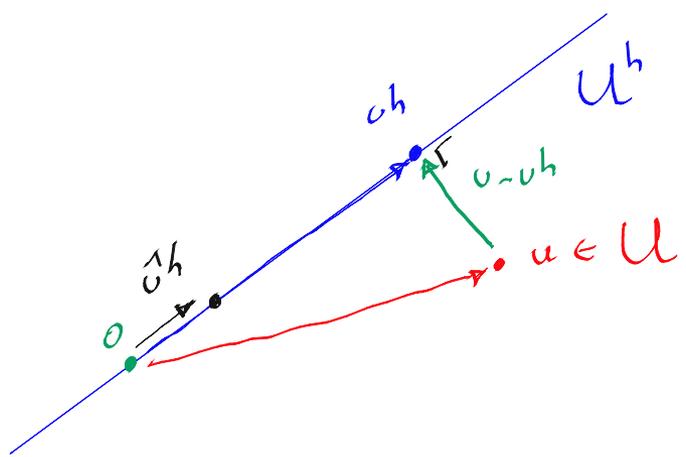
PRODUIT SCALAIRE DE

$$L^2([0,1[) = \left\{ f(x) \text{ tel que } \int_0^1 f^2 < \infty \right\}$$

ESPACE VECTORIEL
 - COMPLET
 - AVEC PRODUIT SCALAIRE

ESPACE D'HILBERT

ESPACE DES FONCTIONS CARRÉ INTEGRABLES
 ESPACE DE SOBOLEV



$$\langle u^h, (u - u^h) \rangle = 0 \quad \forall u^h \in U^h$$

PROJECTION ORTHOGONALE

Projection orthogonale...

$$\langle \hat{u}^h, (v - u^h) \rangle = 0 \quad \forall \hat{u}^h \in U^h$$

PROJECTION
ORTHOOGONALE

$$U^h = \{ ax + b \quad a, b \in \mathbb{R} \}$$

$$\dim(U^h) = 2$$

$$\text{base} = \{ x ; 1 \}$$

$$\forall \hat{u}^h \in U^h$$

$$\hat{u}^h = \hat{a}x + \hat{b}$$

$$\Leftrightarrow \forall (\hat{a}, \hat{b}) \Leftrightarrow (1, 0) \text{ et } (0, 1)$$

$$\langle x (v(x) - u^h(x)) \rangle = 0$$

$$\langle (v(x) - u^h(x)) \rangle = 0$$

$$\int_0^1 x (v(x) - u^h(x)) dx = 0$$

$$\int_0^1 (v(x) - u^h(x)) dx = 0$$



$$\langle 68x (\quad) \rangle$$

$$\langle 1x (\quad) \rangle$$

$$\int_0^1 x (u(x) - u^h(x)) dx = 0$$

$$\int_0^1 (u(x) - u^h(x)) dx = 0$$



$$\sum_{j=0}^n \int_0^1 \phi_k(x) \phi_j(x) dx a_j = \int_0^1 \phi_k(x) u(x) dx$$

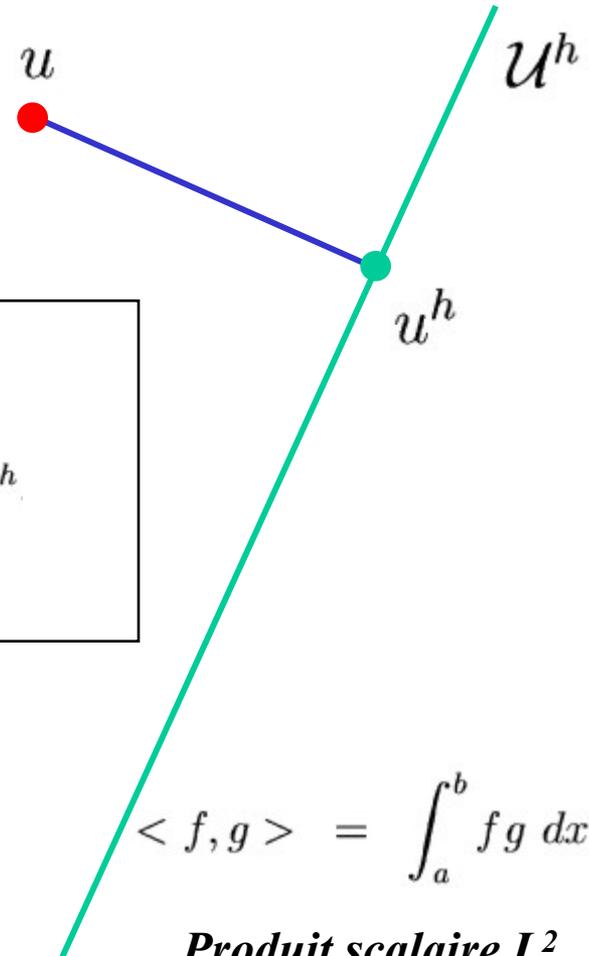
$k = 0, n$



...oui enfin !

L'approximation au sens
des moindres carrés
est une projection
orthogonale...

*Sous-espace
vectoriel dont une
base est donnée par
les fonctions ϕ_i*



$$\langle \hat{u}^h, \underbrace{(u - u^h)}_{e^h} \rangle = 0, \quad \forall \hat{u}^h \in \mathcal{U}^h$$

$$\langle f, g \rangle = \int_a^b f g \, dx$$

Produit scalaire L^2

Et il s'agit bien du même $u^h(x)$.

$$\langle \hat{u}^h, \underbrace{(u - u^h)}_{e^h} \rangle = 0, \quad \forall \hat{u}^h \in \mathcal{U}^h,$$

*Imposer pour toute fonction de base ϕ_i
est équivalent à
l'imposer pour toute fonction de l'espace*

$$\int_a^b \phi_k(x) \left(u(x) - \sum_{j=0}^n \phi_j(x) a_j \right) dx = 0, \quad k = 0, 1, \dots, n$$

$$\sum_{j=0}^n \left(\int_a^b \phi_k(x) \phi_j(x) dx \right) a_j = \int_a^b \phi_k(x) u(x) dx \quad k = 0, 1, \dots, n$$

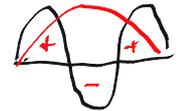
*Equations intégrales de l'approximation au
sens des moindres carrés !*



Une base orthogonale pour ce produit scalaire !

$$\phi_i(x) = \cos(ix) \quad i = 1, 2, \dots, n$$

$$[-\pi, \pi].$$

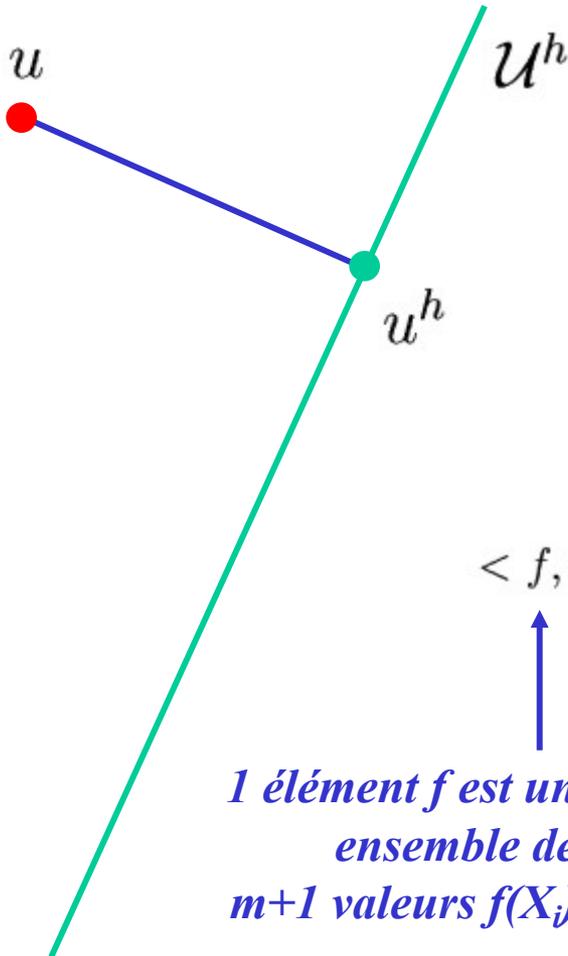



$$\int_{-\pi}^{\pi} \cos(ix) \cos(jx) = 0 \quad i \neq j$$
$$= \pi \quad i = j$$


$$a_i = \frac{1}{\pi} \int_{-\pi}^{\pi} u(x) \cos(ix) dx$$

*Sous-espace vectoriel
dont une base est
donnée par les $n+1$
vecteurs $\phi_j(X_j)$*

Approximer $m+1$
points au sens des
moindres carrés avec
 $n+1$ fonctions de base



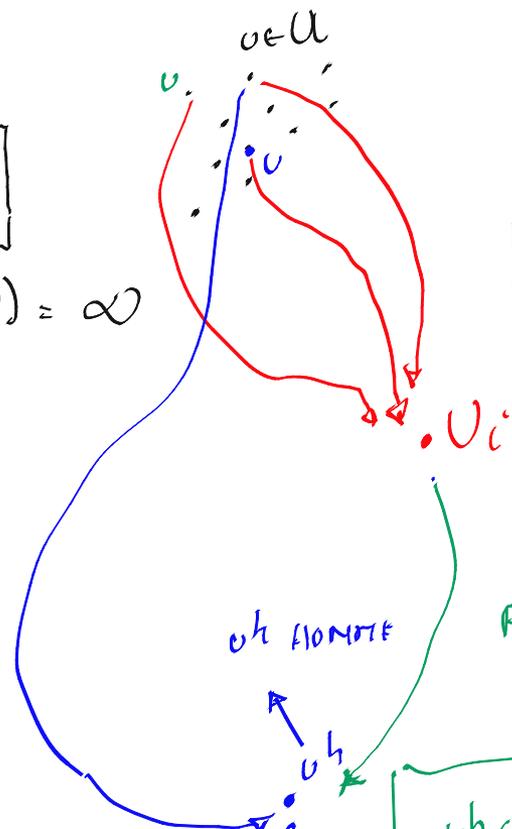
$$\langle f, g \rangle_* = \sum_{j=0}^m f(X_j)g(X_j)$$

*1 élément f est un
ensemble de
 $m+1$ valeurs $f(X_j)$*

C'est réaliser une projection orthogonale
d'un élément de \mathbb{R}^{m+1}
dans un sous-espace de dimension $n+1$

$$v \in U$$

$$\dim(U) = \infty$$

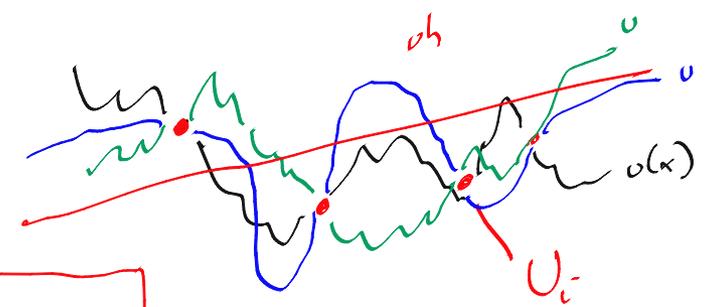


$$U_i$$

m MESURES

$$U_i \in \mathbb{R}^m$$

$\dim(\mathbb{R}^m) = m$



PROJECTION
ORTHO

$$\langle \rangle = \sum f(x_i) g(x_i)$$

$$\langle \rangle = \int_0^1 f g \, dx$$

u_h FONCTE
 u_h
 u_h
 u_h FEMME

$$u_h \in \mathcal{U}^h$$

$$\dim(\mathcal{U}^h) = 2 \approx m$$

$$m < \infty \text{ :-)}$$

Fonction Mesures

Approximation

En fait, on réalise
deux approximations
successives...

Cet élément de \mathbb{R}^m représente
toutes les fonctions dont
les m valeurs en X_i sont identiques

$$\langle \hat{u}^h, \underbrace{(u - u^h)}_{e^h} \rangle_* = 0,$$

$$\forall \hat{u}^h \in \mathcal{U}^h,$$

$$\sum_{i=0}^m \phi_k(X_i) \left(u(X_i) - \sum_{j=0}^n \phi_j(X_i) a_j \right) = 0,$$

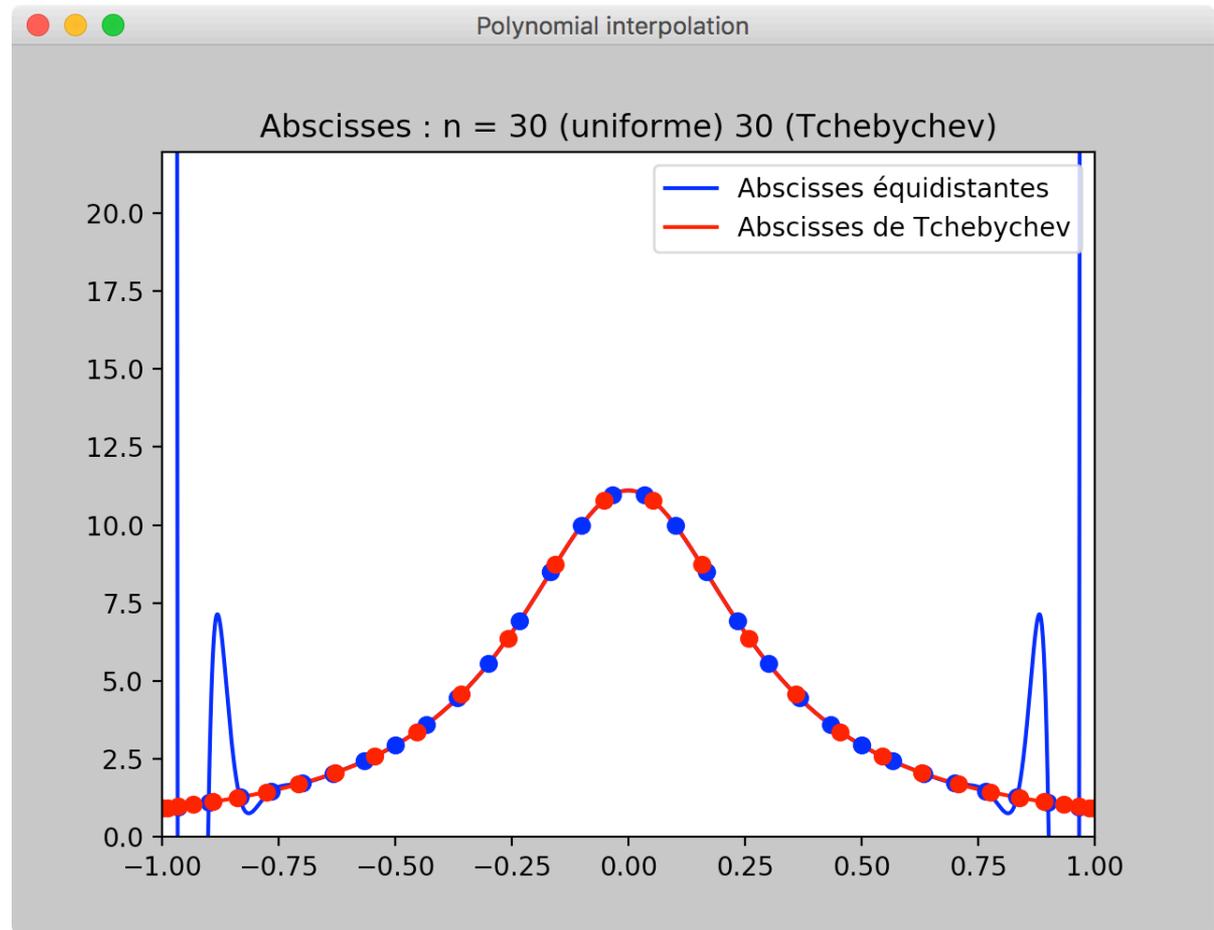
$$k = 0, 1, \dots, n$$

$$\sum_{j=0}^n \left(\sum_{i=0}^m \phi_k(X_i) \phi_j(X_i) \right) a_j = \sum_{i=0}^m \phi_k(X_i) U_i \quad k = 0, 1, \dots, n$$

Equations normales !



Implémenter l'interpolation polynomiale...



Une implémentation naïve mais qui fournit la bonne réponse...

```
def lagrange_naive(x,X,U):
```

```
    n = min(len(X),len(U))
```

```
    m = len(x)
```

```
    uh = zeros(m)
```

```
    phi = zeros(n)
```

```
    for j in range(m):
```

```
        uh[j] = 0
```

```
        for i in range(n):
```

```
            phi[i] = 1.0
```

```
            for k in range(n):
```

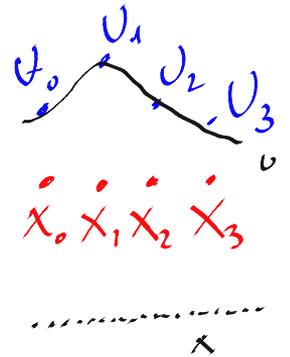
```
                if i != k:
```

```
                    phi[i] = phi[i]*(x[j]-X[k])/(X[i]-X[k])
```

```
                uh[j] = uh[j] + U[i] * phi[i]
```

```
    return uh
```

*Pas si naïf que cela :
Implémentation robuste !
Pas d'erreur possible !*



$$u^h(x) = \sum U_i \phi_i(x)$$

$$\phi_i(x) = \frac{(x - X_0)(x - X_1) \dots (x - X_{i-1})(x - X_{i+1}) \dots (x - X_n)}{(X_i - X_0)(X_i - X_1) \dots (X_i - X_{i-1})(X_i - X_{i+1}) \dots (X_i - X_n)}$$

```
def lagrange_naive(x,X,U):
```

```
    n = min(len(X), len(U))
```

```
    m = len(x)
```

```
    uh = zeros(m)
```

```
    phi = zeros(n)
```

```
    for j in range(m):
```

```
        uh[j] = 0
```

```
        for i in range(n):
```

```
            phi[i] = 1.0
```

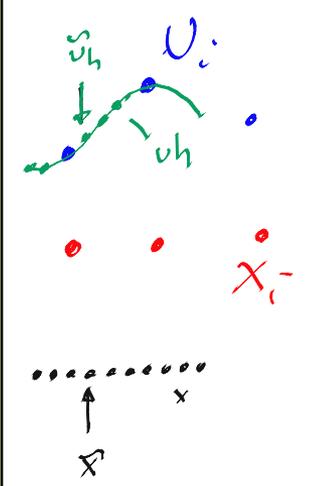
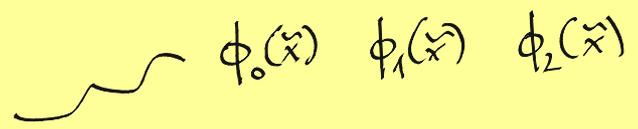
```
            for k in range(n):
```

```
                if i != k:
```

```
                    phi[i] = phi[i]*(x[j]-X[k])/(X[i]-X[k])
```

```
                uh[j] = uh[j] + U[i] * phi[i]
```

```
    return uh
```



$$u^h(x) = \sum_{i=0}^n U_i \phi_i(x)$$

```

def lagrange_naive(x,X,U):

    n = min(len(X),len(U))
    ...
    for j in range(m):
        uh[j] = 0
        for i in range(n):
            ...
            uh[j] = uh[j] + U[i] * phi[i]
    return uh

```

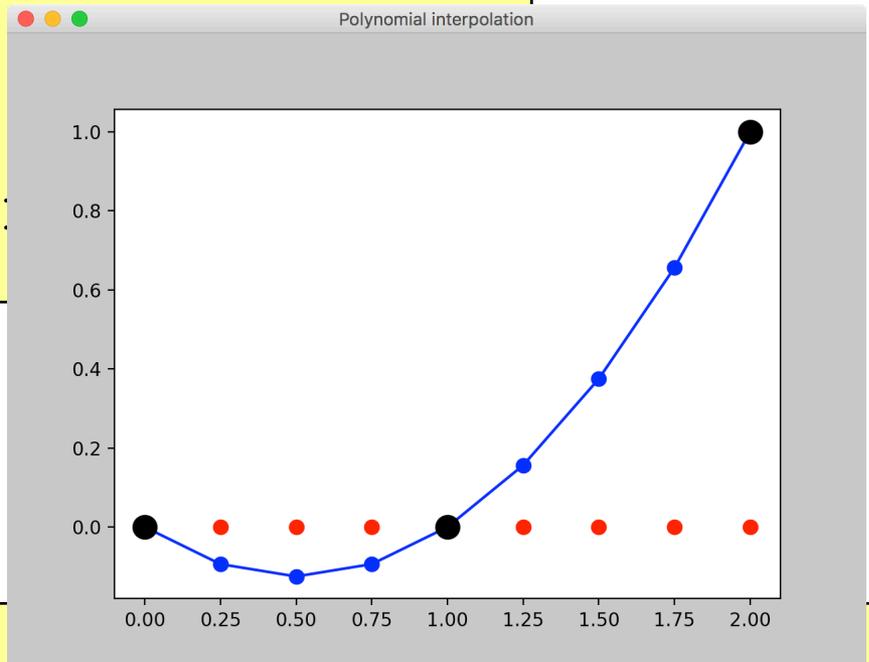
x : c'est quoi ?

```

X = [0,1,2]
U = [0,0,1]
x = linspace(0,2,9)
uh = lagrange_naive(x,X,U);

plt.plot(x,zeros(size(x)),'.r',markersize=15)
plt.plot(x,uh,'.-b',markersize=15)
plt.plot(X,U,'.k',markersize=25)

```



Pourquoi est-ce naïf ?

```
x = linspace(-1,1,2000)
tic()
lagrange(x,[0,1,2],[0,1,2])
toc()
tic()
lagrange(x,[0,1,2],[0,1,2])
toc()
```

Elapsed time is 0.000219 seconds

Elapsed time is 0.017949 seconds

```
x = linspace(-1,1,2000000)
tic()
lagrange_naive(x,[0,1,2],[0,1,2])
...
```

Elapsed time is 1.415670 seconds

Elapsed time is 159.760674 seconds

Pas de vectorisation :-(
Mais pré-allocation :-)



CTRL-C

Suite de Pisano Fibonacci

A man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair if it is supposed that every month each pair begets a new pair which from the second month on becomes productive...

$$f_n = f_{n-1} + f_{n-2}$$

```
def fibonacci(n):  
    f = zeros(n)  
    f[0] = 1  
    f[1] = 2  
    for k in range(2,n):  
        f[k] = f[k-1] + f[k-2]  
    return f
```

Dans numpy, il faut toujours préallouer les tableaux...

```
def fibonacci(n):  
    f = zeros(n)  
    f[0] = 1  
    f[1] = 2  
    for k in range(2,n):  
        f[k] = f[k-1] + f[k-2]  
    return f
```

```
def fibonaccon(n):  
  
    f[0] = 1  
    f[1] = 2  
    for k in range(2,n):  
        f[k] = f[k-1] + f[k-2]  
    return f
```

```
bash-3.2$ python fibonacci.py  
Traceback (most recent call last):  
  File "fibonacci.py", line 69, in <module>  
    fibonacci(n)  
  File "fibonacci.py", line 50, in fibonacci  
    f[0] = 1  
NameError: name 'f' is not defined
```

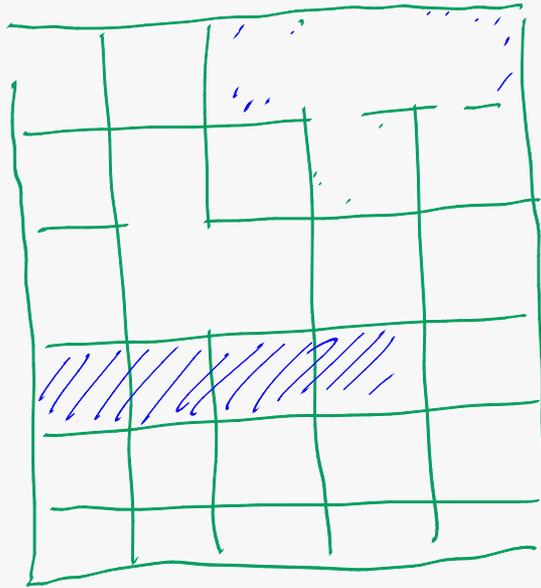
Et si tu veux vraiment pas pré-allouer les tableaux...

```
def fibonacci(n):  
    f = zeros(n)  
    f[0] = 1  
    f[1] = 2  
    for k in range(2,n):  
        f[k] = f[k-1] + f[k-2]  
    return f
```

```
def fibonaccon(n):  
  
    f = [1,2]  
  
    for k in range(2,n):  
        f = append(f,[f[k-1] + f[k-2]])  
    return f
```

```
bash-3.2$ python fibonacci.py  
Elapsed time is 0.070130 seconds (fibonacci)  
Elapsed time is 10.754738 seconds (fibonaccon)
```

```
n = 200000  
tic()  
fibonacci(n)  
toc('fibonacci')
```

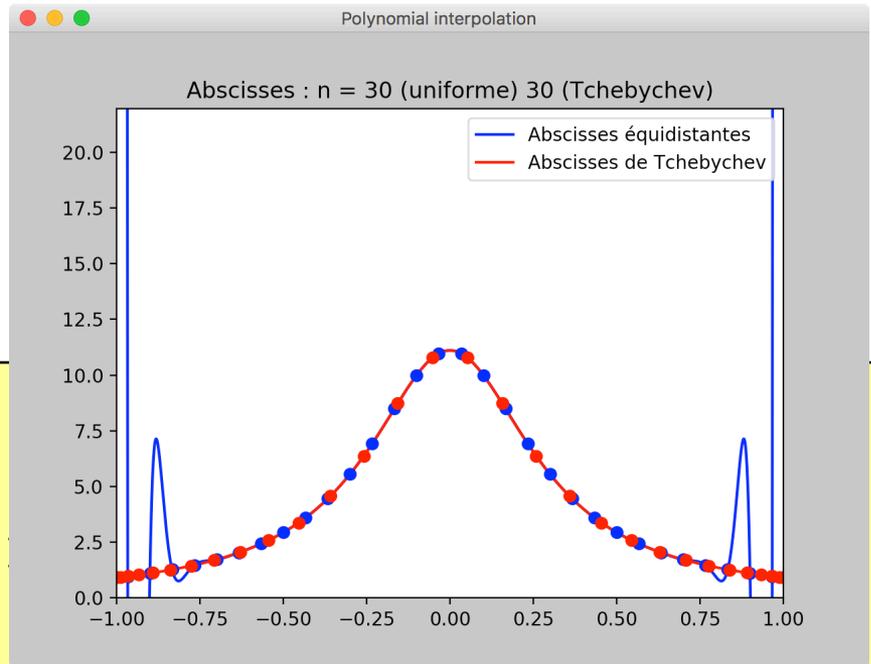


Obtenir la figure !

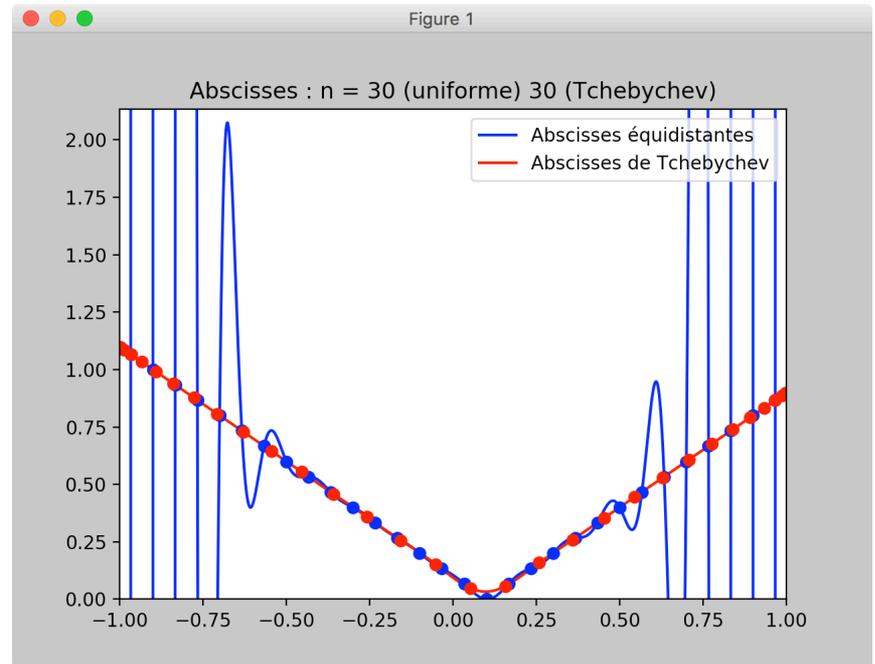
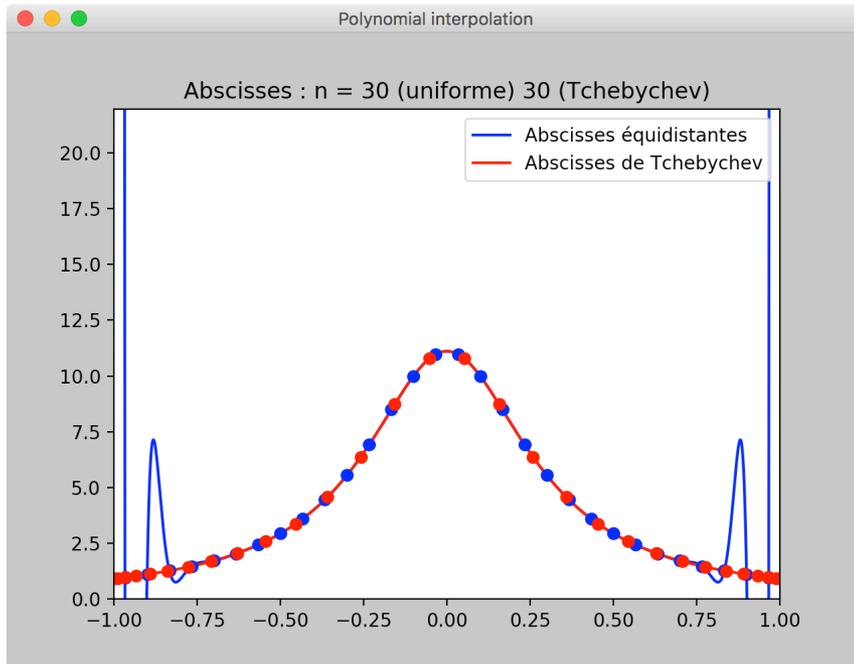
```
h = 1/n
x = linspace(-1,1,2000)
Xunif = arange(-1+h/2,1+h/2,h)
Xcheb = cos(pi*(2*arange(0,2*n)+1))

u = lambda x : 1/(x**2 + 0.09)
Uunif = u(Xunif)
Ucheb = u(Xcheb)

plt.figure('Polynomial interpolation')
plt.plot(x,lagrange(x,Xunif,Uunif),'-b',label='Absc. équadistantes')
plt.plot(x,lagrange(x,Xcheb,Ucheb),'-r',label='Absc. de Tchebychev')
plt.plot(Xunif,Uunif,'ob',Xcheb,Ucheb,'or')
plt.xlim((-1,1))
plt.ylim((0,max(Uunif)*2))
plt.title('Abscisses : n = %d (uniforme) %d (Tchebychev) '
          '% (len(Xunif),len(Xcheb)))')
plt.legend(loc='upper right')
```



Obtenir deux figures...



Cool : ctrl-c / ctrl-v 😊

Ne jamais dupliquer du code !

```
functions = [lambda x : 1/(x**2 + 0.09),  
             lambda x : abs(x - 0.1)]  
  
for u in functions:  
    Unif = u(Xunif)  
    Ucheb = u(Xcheb)  
    plt.figure('Polynomial interpolation')  
    plt.plot(x, lagrange(x, Xunif, Unif), '-b', ...  
            plt.plot(x, lagrange(x, Xcheb, Ucheb), '-r', ...
```

*Les listes de Python peuvent contenir
n'importe quoi et donc aussi des fonctions*

Il faudra maintenir deux fois plus
de lignes de code : c'est cher !

Votre premier devoir a été corrigé !

perso.uclouvain.be/vincent.legat/zouLab/epl1104

U L zouLab 1.0.0 : epl1104

EPL1104 News Documents Mon profil Hello Fick Deconnexion

Liste des étudiants Equipe didactique Groupes et canaux Canaux Teams et horaires des exercices Choisir son canal et son tuteur :-)

Soumettre le devoir 0 solveRatio Soumettre le devoir 2 cubic

Evaluation devoir 1 diophantine

Evaluation du devoir 1 : diophantine

- 1 Test 0 (/1) : Est-ce que le programme ne se plante pas ?
- Test 1 (/5) : Implémentation de base avec [a,b,c] = [2, 2, 2] n = 20 et norm = 400
- 1 Test 1.1 (/1) : Taille du tableau de sortie : len(X) = 3
- 0 Test 1.2 (/1) : Cohérence du tableau de sortie
- 0 Test 1.3 (/1) : La solution est correcte
- 0 Test 1.4 (/1) : La solution est celle avec le plus grand x
- 0 Test 1.5 (/1) : Efficacite et précision du code : timeObs <= 3*timeRef avec X = Xref
- Test 2 (/4) : Test avec [a,b,c] = [4, 3, 2] n = 30 et norm = 800
- 0 Test 2.1 (/1) : Cohérence du tableau de sortie
- 0 Test 2.2 (/1) : La solution est correcte
- 0 Test 2.3 (/1) : La solution est celle avec le plus grand x
- 0 Test 2.4 (/1) : Efficacite et précision du code : timeObs <= 3*timeRef avec X = Xref

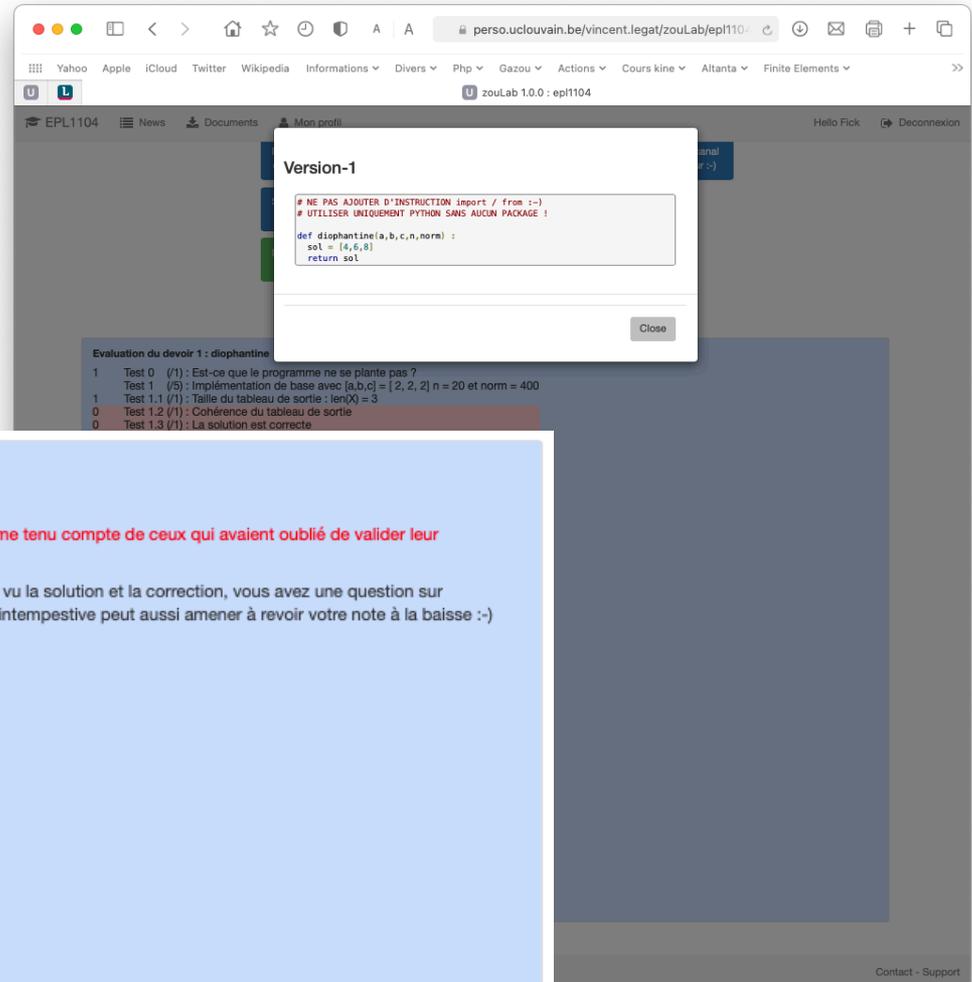
Note obtenue : 2/10

$52 = 16 + 36 = 4 \cdot 4 + 6 \cdot 6 = 8 \cdot 8 = 64$

Diagnostic détaillé Soumission testée Solution :-)

© 2020 Vincent Legat Contact - Support

Votre premier devoir a été corrigé !



La correction et la solution du devoir 1 sont disponibles :-)

La correction et la [solution](#) de premier problème sont disponibles :-)

On a été très gentils même pour ceux qui ont effectué la soumission en gardant les zéros et on a même tenu compte de ceux qui avaient oublié de valider leur soumission : attention, c'est l'unique et la dernière fois où on sera aussi gentils :-)

Vous avez accès à votre note dans l'onglet [Evaluation devoir 1](#) après identification. Si vraiment après vu la solution et la correction, vous avez une question sur l'évaluation, il vaut s'adresser exclusivement [Nathan Coppin](#). Attention, toute réclamation abusive et intempestive peut aussi amener à revoir votre note à la baisse :-)

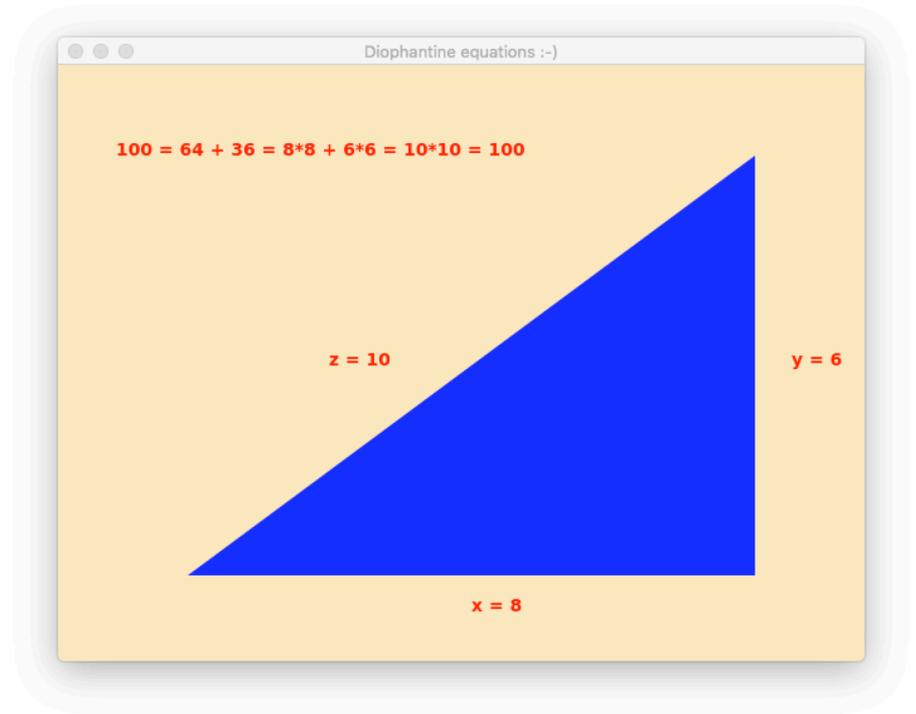
Il y a eu que 437 soumissions : c'est bien et il ne faut pas vous décourager !

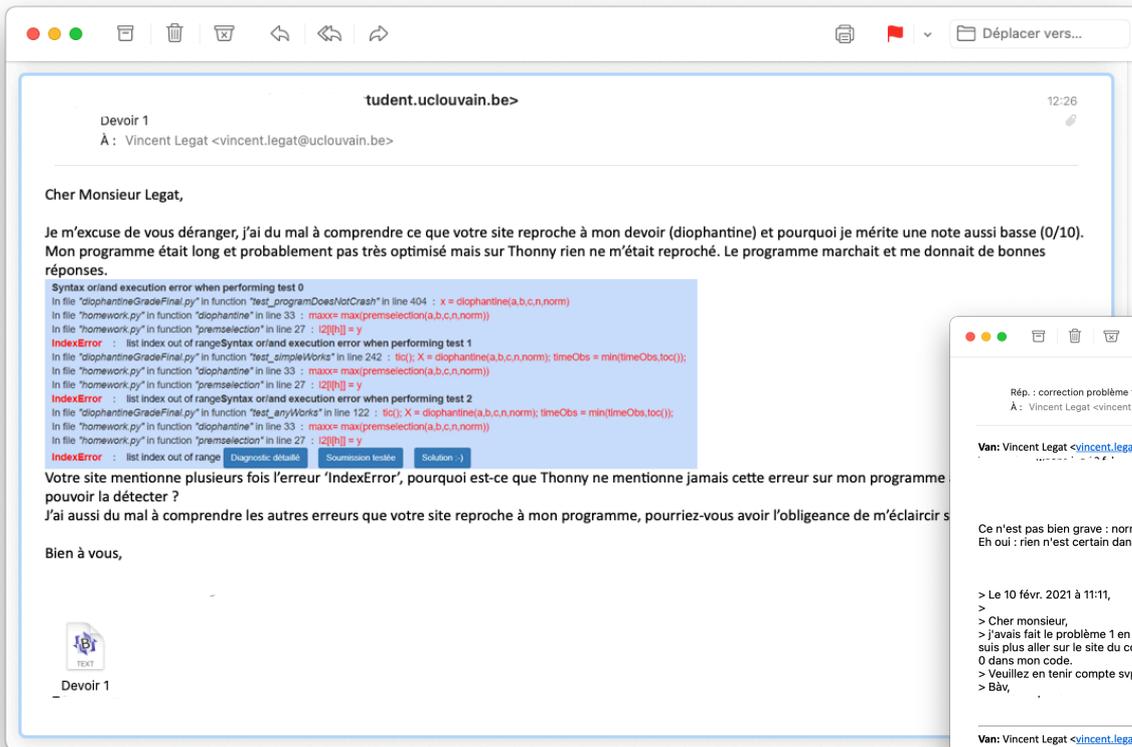
Les statistiques des résultats pour ce problème sont :

10/10 : 222 étudiant(e)s
09/10 : 40 étudiant(e)s
08/10 : 58 étudiant(e)s
07/10 : 15 étudiant(e)s
06/10 : 19 étudiant(e)s
05/10 : 8 étudiant(e)s
04/10 : 30 étudiant(e)s
03/10 : 5 étudiant(e)s
02/10 : 11 étudiant(e)s
01/10 : 18 étudiant(e)s
00/10 : 11 étudiant(e)s

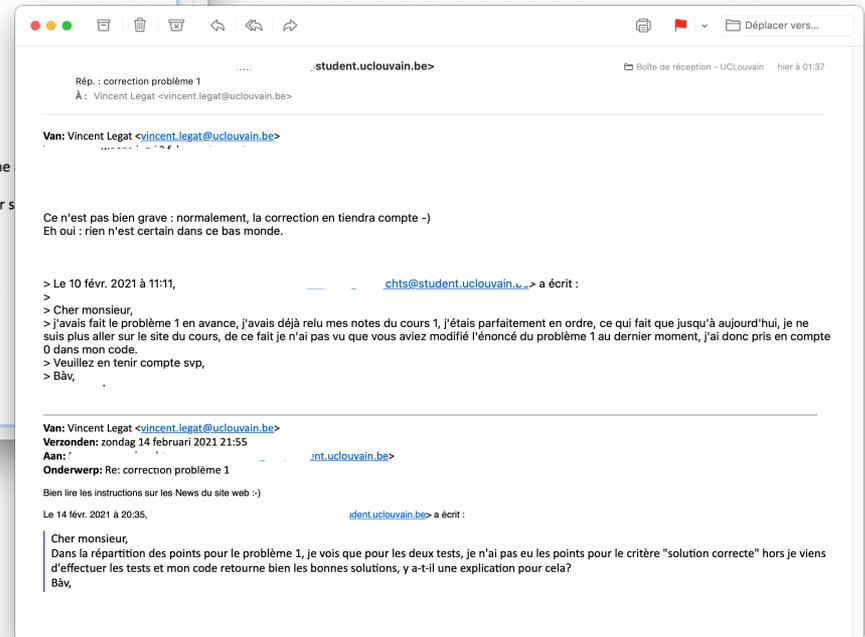
(2021-02-14 16:24:05)

Monsieur !
Où est le SAV ?
Y a un problème !
I want my money back !





2
0,2
0,02



Service après vente
Nathan est là !