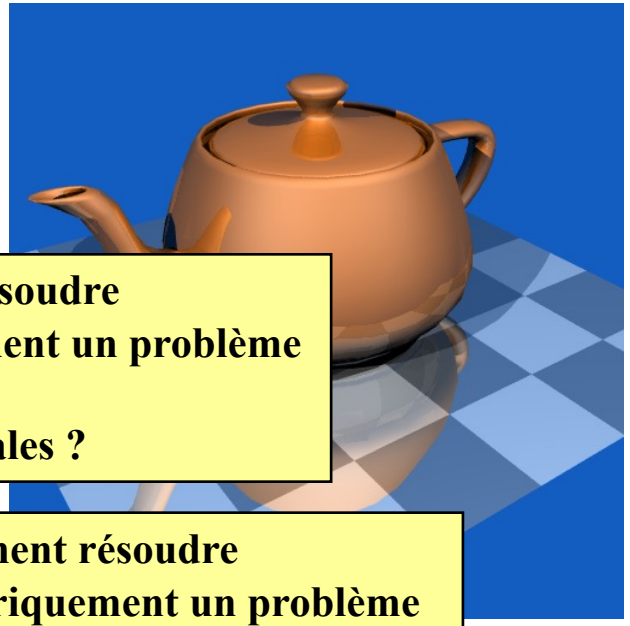
Plan des cours de méthodes numériques



Comment résoudre numériquement un problème aux valeurs initiales ?

Comment interpoler une fonction ?

Comment dériver numériquement une fonction ?

Comment résoudre numériquement un problème aux conditions frontières ?

Comment approximer une fonction ?

Comment intégrer numériquement une fonction ?

Et les équations non linéaires ?

Et les méthodes itératives ?

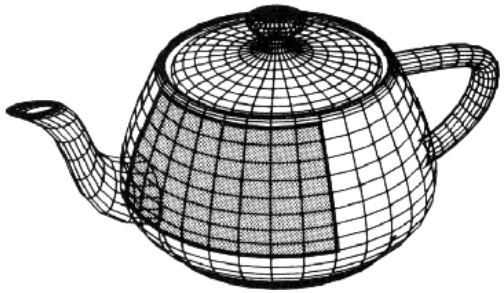
Comment résoudre numériquement une équation différentielle ordinaire ?

Comment résoudre numériquement une équation aux dérivées partielles ?

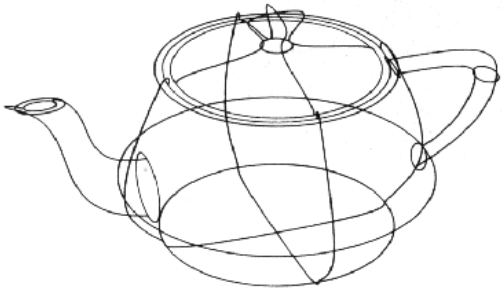
Comment résoudre numériquement une équation aux dérivées partielles ?

The Utah Teapot

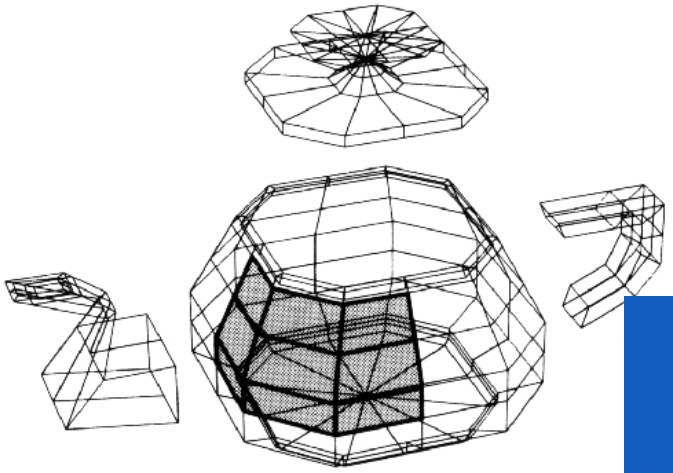
Martin Nevell (1975)



Single shaded patch



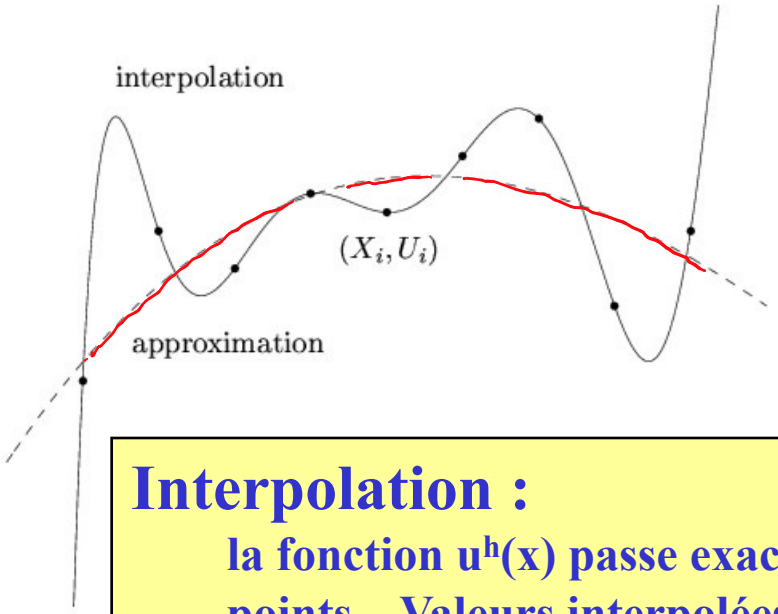
Patch edges



Control points

32 patches

Interpolation, approximation et extrapolation…



Fonctions de base spécifiées a priori

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

Paramètres inconnus

Interpolation :

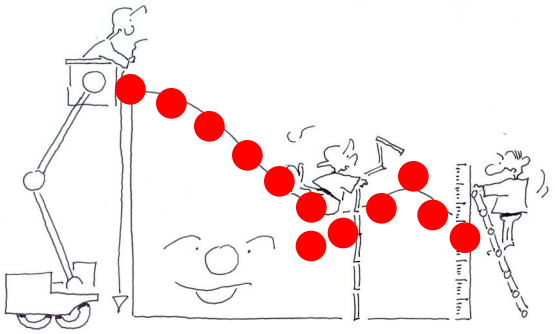
la fonction $u^h(x)$ passe exactement par les points. Valeurs interpolées entre les points et valeurs extrapolées hors de l'intervalle.

Approximation :

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

Beaucoup de données…

m = 11

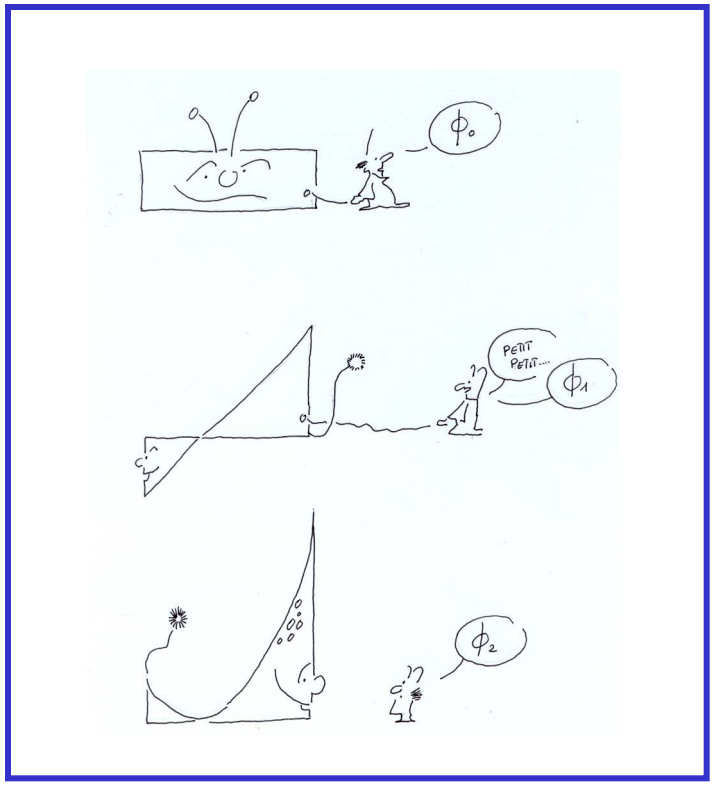


$$(X_i, U_i), \quad i = 0, 1, 2, \ldots, m.$$

$$n < m$$

n = 2

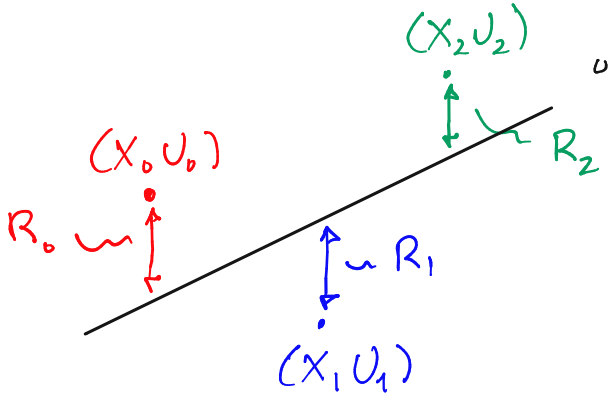
$$u^h(x) = \sum_{j=0}^{n} a_j \, \phi_j(x)$$



…peu de paramètres !

Approximation aux moindres carrés…

$$U_0 \simeq a X_0 + b$$
$$U_1 \simeq a X_1 + b$$
$$U_2 \simeq a X_2 + b$$



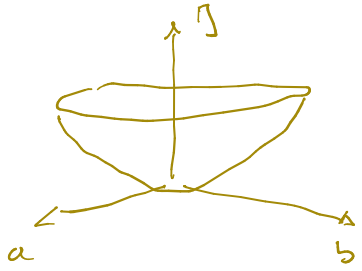
A MINIMISER

$$R_0^2 + R_1^2 + R_2^2$$

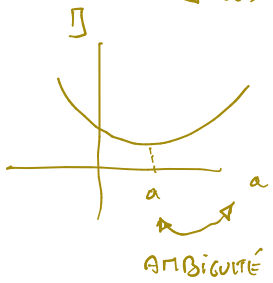
$$(U_0 - aX_0 - b)^2 \quad (U_1 - aX_1 - b)^2 \quad (U_2 - aX_2 - b)^2$$

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

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



$J(a) = (x - a)^2$



3 points
2 paramètres

Equations normales

$$J(a,b) = (v_0 - aX_0 - b)^2 + (v_1 - aX_1 - b)^2 + (v_2 - aX_2 - b)^2$$

$$0 = \frac{\partial J}{\partial a} = -2\left[(v_0 - aX_0 - b)X_0 + (v_1 - aX_1 - b)X_1 + \dots\right]$$

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

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

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

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

EQUATIONS NORMALES

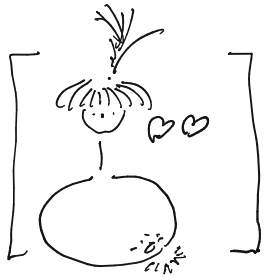
Généralisons un peu…

$$u_h(x) = a_0 \underbrace{\varphi_0(x)}_{1} + a_1 \underbrace{\varphi_1(x)}_{x}$$

$$\begin{bmatrix} \sum_{i=0}^{m} \phi_0(x_i)\,\phi_0(x_i) & \sum_{i} \phi_0(x_i)\,\phi_1(x_i) \\ \sum_{i=0}^{m} \phi_1(x_i)\,\phi_0(x_i) & \sum_{i} \phi_1(x_i)\,\phi_1(x_i) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{i} u_i\,\phi_0(x_i) \\ \sum_{i} u_i\,\phi_1(x_i) \end{bmatrix}$$

$$\sum_{j=0}^{n} \sum_{i=0}^{m} \phi_k(x_i)\,\phi_j(x_i)\,a_j = \sum_{i=0}^{m} u_i\,\phi_k(x_i)$$

$$k = 0 \dots n$$

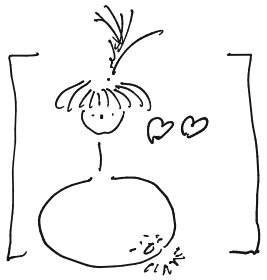


$$\begin{bmatrix} \Sigma x_i^2 & \Sigma x_i \\ \Sigma x_i & \Sigma 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \Sigma x_i u_i \\ \Sigma u_i \end{bmatrix}$$

ÉQUATIONS NORMALES

$$\sum_{j=0}^{n} \sum_{i=0}^{m} \phi_k(x_i)\, \phi_j(x_i)\, a_j = \sum_{i=0}^{m} u_i\, \phi_k(x_i)$$

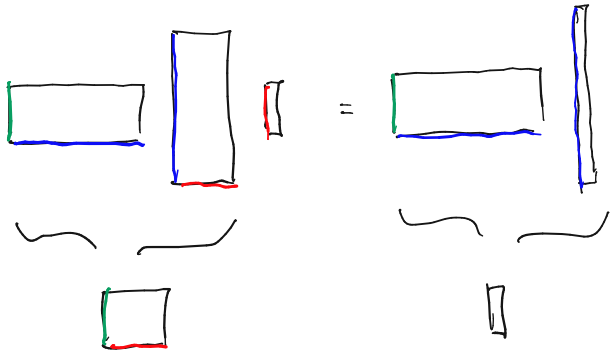
$k = 0 \dots n$



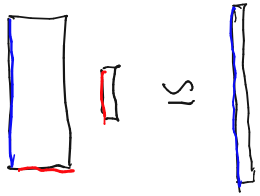
$\underline{\underline{A}}_{ik}$

$\underline{\underline{A}}_{ij}$ — COLONNES $n+1 = 2$ — LIGNES $m+1 = 3$

$$\underline{\underline{A}}^T \cdot \underline{\underline{A}} \cdot a = \underline{\underline{A}}^T \cdot u$$



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



…pour conclure
par un petit dessin

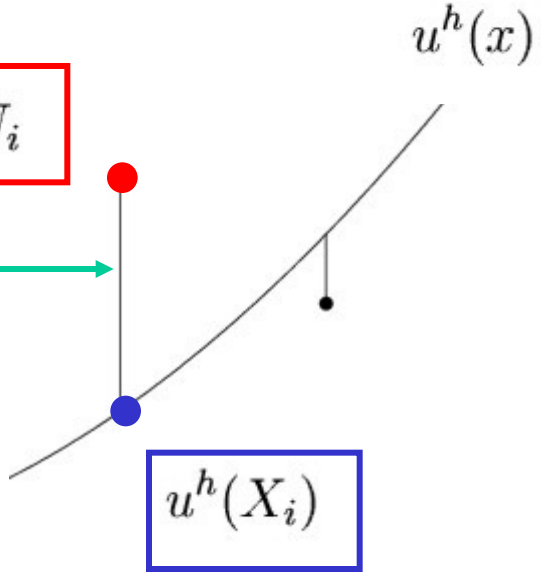
La fonction $u^h(x)$ ne passe pas par les points, mais s'en rapproche selon un critère à définir…

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

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

Minimisons les résidus

$$R_i = U_i - \underbrace{\sum_{j=0}^{n} \phi_j(X_i)\ a_j}_{u^h(X_i)} \qquad i = 0, 1, 2, \ldots, m.$$



Problème de l'approximation

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

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

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

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

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

Calculons…

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

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

$$\downarrow$$

$$\sum_{i=0}^{m} -2\phi_k(X_i)\left(U_i - \sum_{j=0}^{n}\phi_j(X_i)\ a_j\right) = 0 \qquad k = 0,1,\ldots,n$$

$$\downarrow$$

$$\sum_{i=0}^{m}\phi_k(X_i)\ \sum_{j=0}^{n}\phi_j(X_i)\ a_j = \sum_{i=0}^{m}\phi_k(X_i)U_i \qquad k = 0,1,\ldots,n$$

$$\downarrow$$

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

Et pratiquement…

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

Equations normales

A_{ik}

A^T_{ki}

A_{ij}

Problème de l'approximation

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

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

Equations normales sous forme matricielle $A^T A\, a = A^T\, u$

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

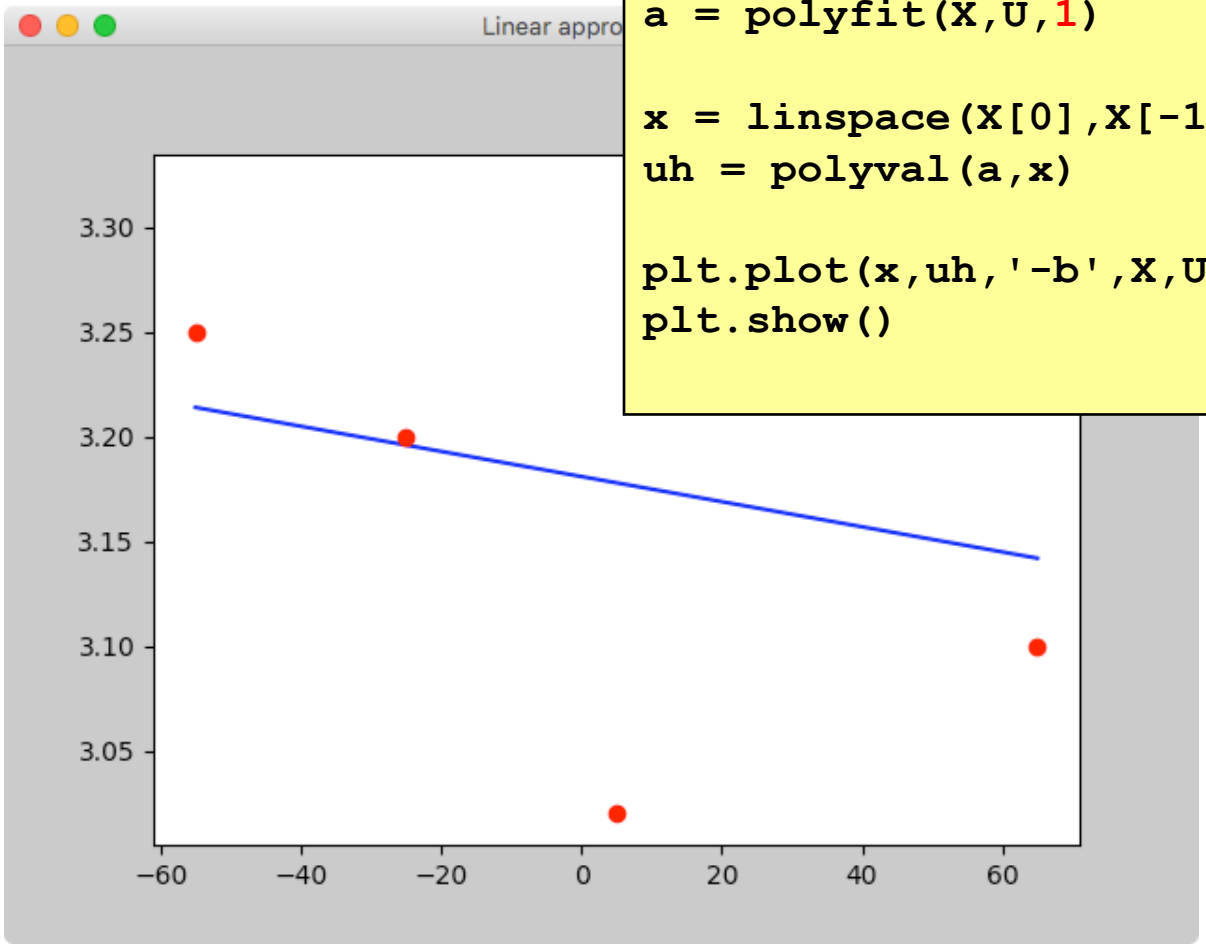
$$\phi_j(x) = x^j \qquad j = 0, 1, 2, \ldots, n.$$

$$u^h(x) = a_0 + a_1 x + a_2 x^2$$

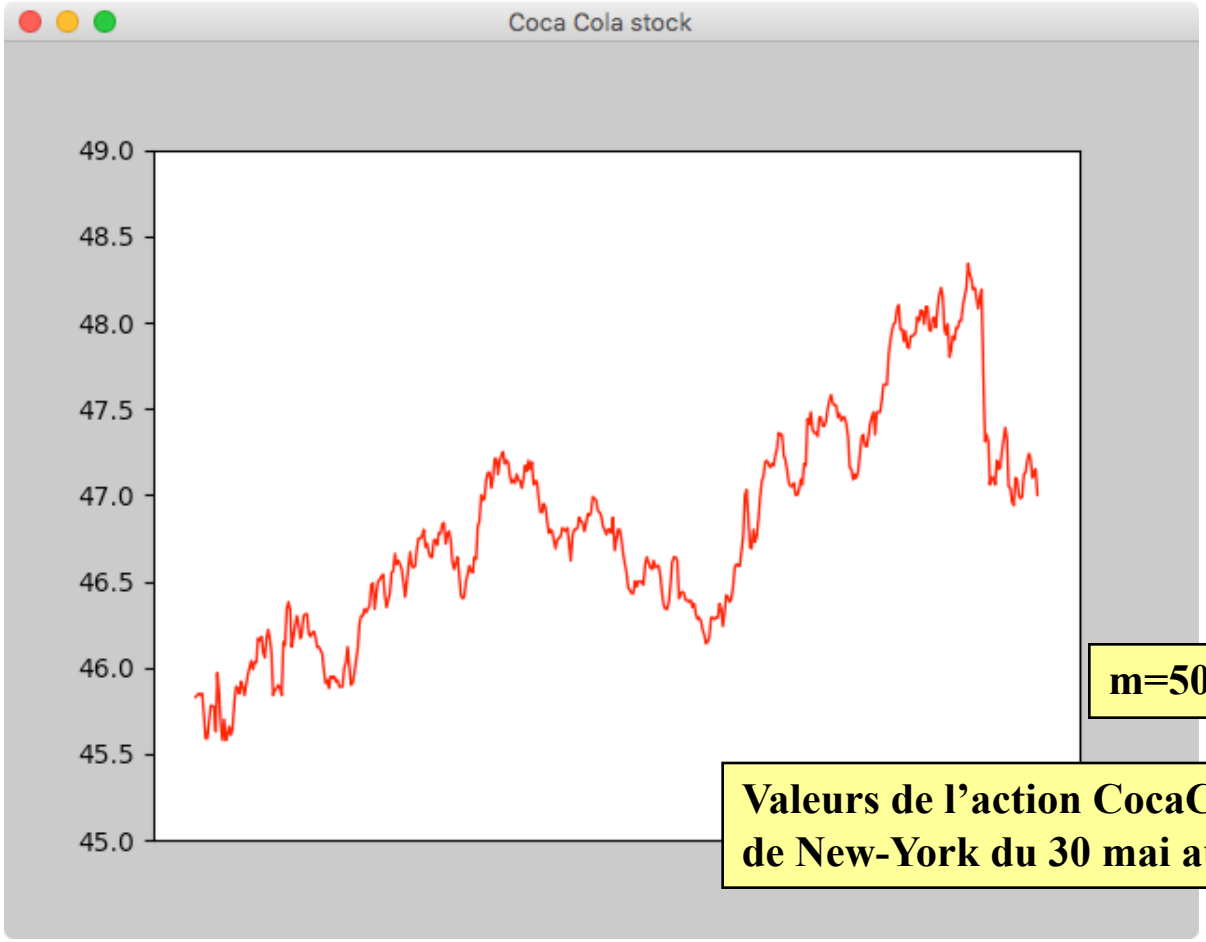
$$\begin{bmatrix} (m+1) & \sum_{i=0}^{m} X_i & \sum_{i=0}^{m} X_i^2 \\ \sum_{i=0}^{m} X_i & \sum_{i=0}^{m} X_i^2 & \sum_{i=0}^{m} X_i^3 \\ \sum_{i=0}^{m} X_i^2 & \sum_{i=0}^{m} X_i^3 & \sum_{i=0}^{m} X_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{m}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

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)

x = linspace(X[0],X[-1],100)
uh = polyval(a,x)

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



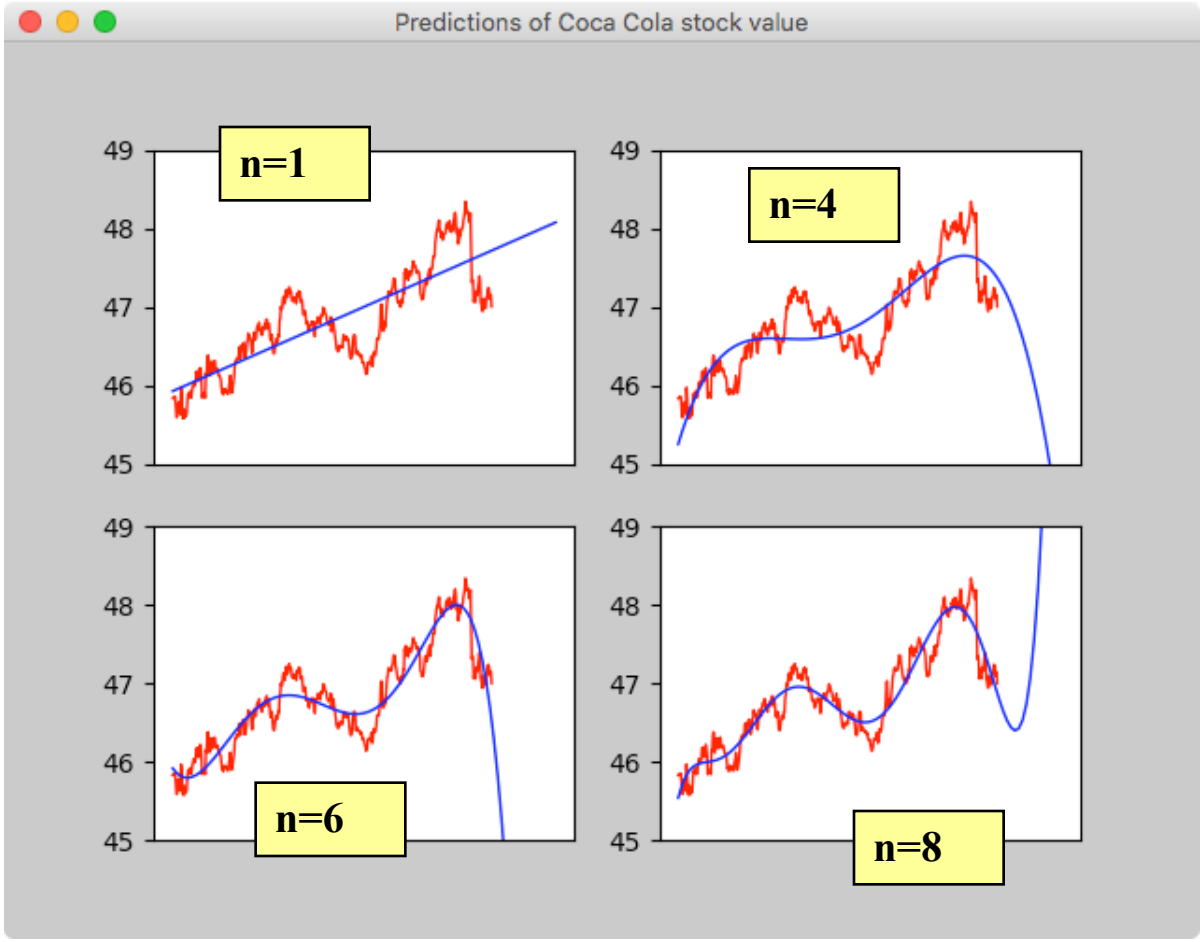
Beaucoup de données



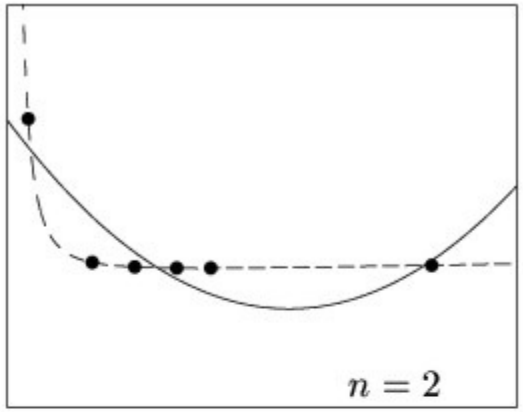
m=500

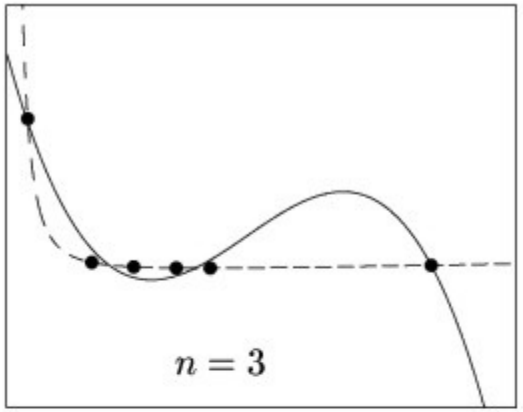
Valeurs de l'action CocaCola à la bourse de New-York du 30 mai au 18 juin 2003

Faisons des régressions et extrapolons les valeurs pour devenir riches :-)

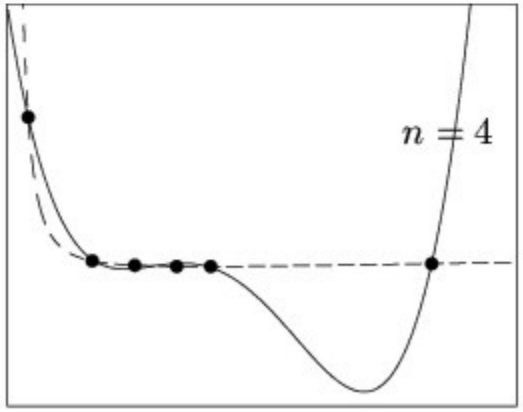


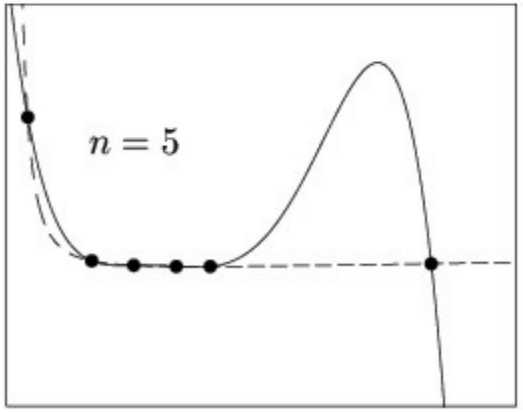
…les approximations polynomiales divergent aussi !





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

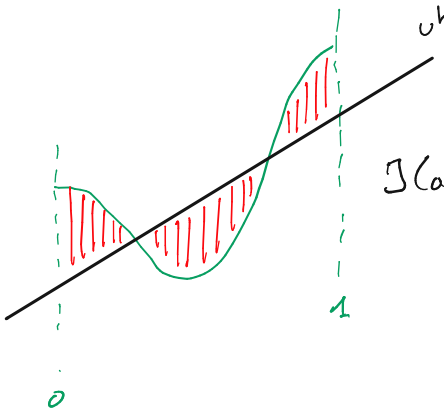




Et si nous avions u(x)…

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

EQUATIONS NORMALES



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

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

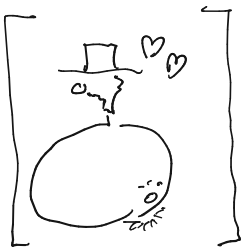
EQUATIONS NORMALES

$$0 = \frac{\partial J}{\partial a} = \int_0^1 -2x \, (u(x) - ax - b) \, dx$$

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

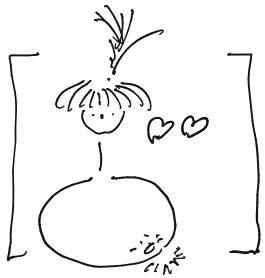
$$\sum_{j=0}^{n} \int_0^1 \phi_k(x) \, \phi_j(x) \, dx \; a_j = \int_0^1 \phi_k(x) \, u(x) \, dx$$

$k = 0 \ldots n$



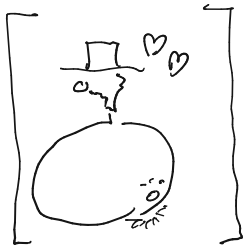
$$\sum_{j=0}^{n} \sum_{i=0}^{m} \phi_k(x_i)\, \phi_j(x_i)\, a_j = \sum_{i=0}^{m} u_i\, \phi_k(x_i)$$

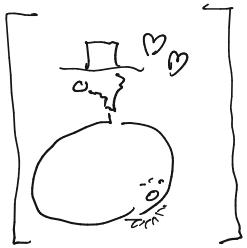
$$k = 0 \dots n$$



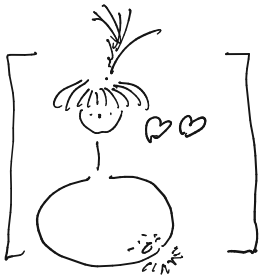
$$\sum_{j=0}^{n} \int_0^1 \phi_k(x)\, \phi_j(x)\, dx\, a_j = \int_0^1 \phi_k(x)\, u(x)\, dx$$

$$k = 0 \dots n$$





EST BETEMENT APPROCHÉ PAR UN ETRE FAIBLE



CAR $\int = \int$ $\Leftrightarrow$ $\sum = \sum$

Et si nous avions la fonction u(x)…

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

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

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

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

$$B\ a = c$$

Problème intégral de l'approximation

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

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

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

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

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

Et pratiquement…

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

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

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

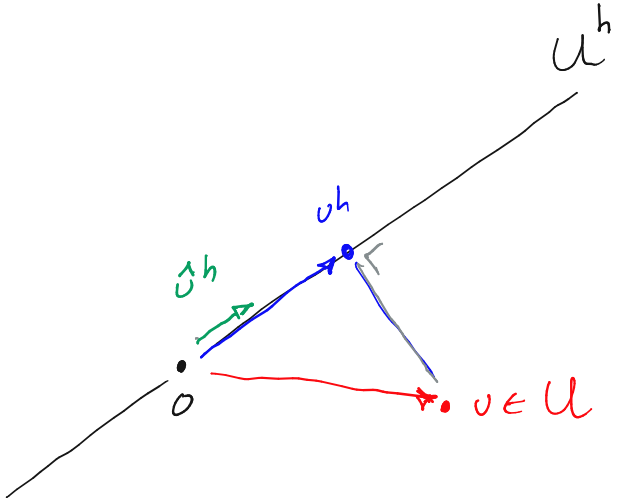
PRODUIT SCALAIRE DE $L^2(]0,1[) = \left\{ f(x) \text{ tel que } \int_0^1 f^2 < \infty \right\}$

ESPACE DES FONCTIONS CARRÉ-INTEGRABLES

ESPACE VECTORIEL
- AVEC PRODUIT SCALAIRE
- COMPLET

ESPACE D' HILBERT

ESPACE DE SOBOLEV



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

PROJECTION ORTHOGONALE

Une jolie interprétation algébro-géométrique…

Projection orthogonale

$$U^h = \{ \underbrace{ax+b}_{u^h(x)} \quad a,b \in \mathbb{R} \}$$

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

$$\text{BASE} = \{1, x\}$$

$$< \hat{u}^h, (u - u^h) > = 0 \quad \forall \hat{u}^h \in U^h$$

PROJECTION ORTHOGONALE

$$\forall \hat{u}^h \in U^h \qquad \hat{u}^h = \hat{a}x + \hat{b}$$

$$\Leftrightarrow \quad \forall \hat{a}, \hat{b}$$

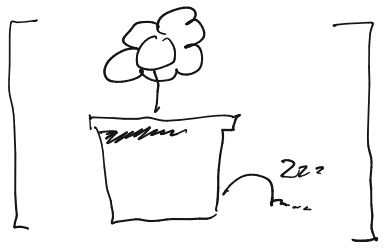
$$\Leftrightarrow \quad (0,1) \quad (1,0)$$

$$< x\,(u(x) - ax - b) > = 0$$

$$< \quad (u(x) - ax - b) > = 0$$

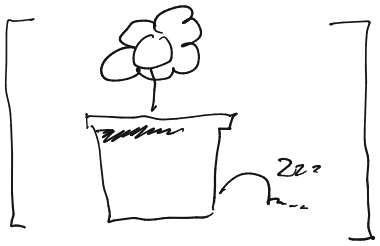
$$\int_0^1 x\,(u(x) - ax - b)\,dx = 0$$

$$\int_0^1 (u(x) - ax - b)\,dx = 0$$



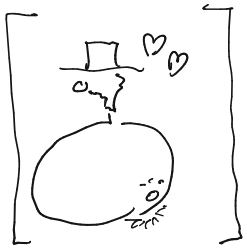
$$\int_0^1 x\,(v(x) - ax - b)\,dx = 0$$

$$\int_0^1 (v(x) - ax - b)\,dx = 0$$



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

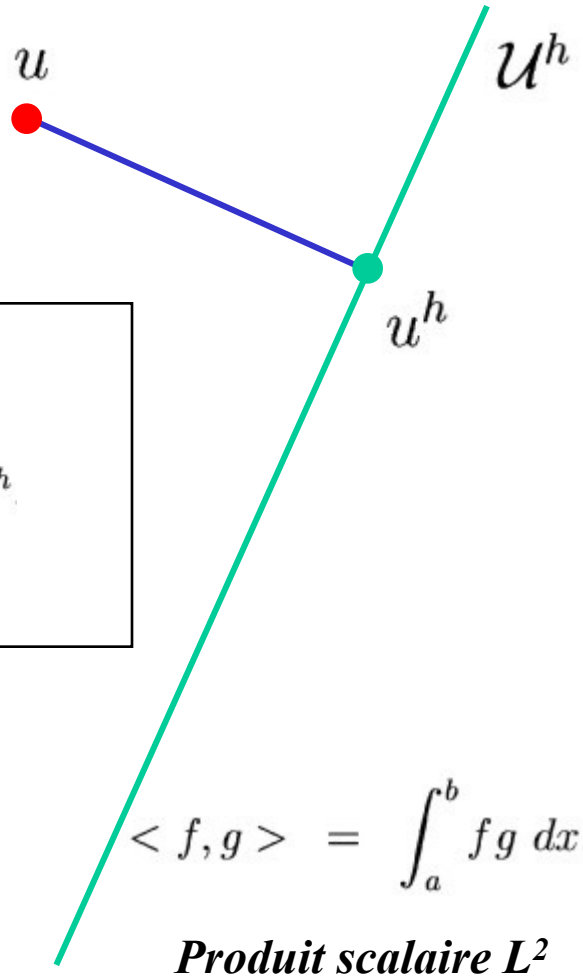
$k = 0 \dots n$



…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



$$< \widehat{u}^h \, , \underbrace{(u - u^h)}_{e^h} > \; = \; 0, \qquad \forall \widehat{u}^h \in \mathcal{U}^h$$

$$< f, g > \; = \; \int_a^b f g \; dx$$

Produit scalaire L^2

Et il s'agit bien du même $u^h(x)$.

$$< \widehat{u}^h \, , \underbrace{(u - u^h)}_{e^h} > \quad = \quad 0, \qquad \forall \widehat{u}^h \in \mathcal{U}^h,$$

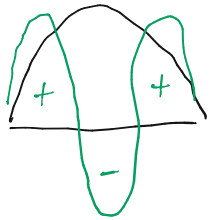
Imposer pour toute fonction de base ϕ_i est équivalent à l'imposer pour toute fonction de l'espace

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

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

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

Une base orthogonale pour ce produit scalaire !



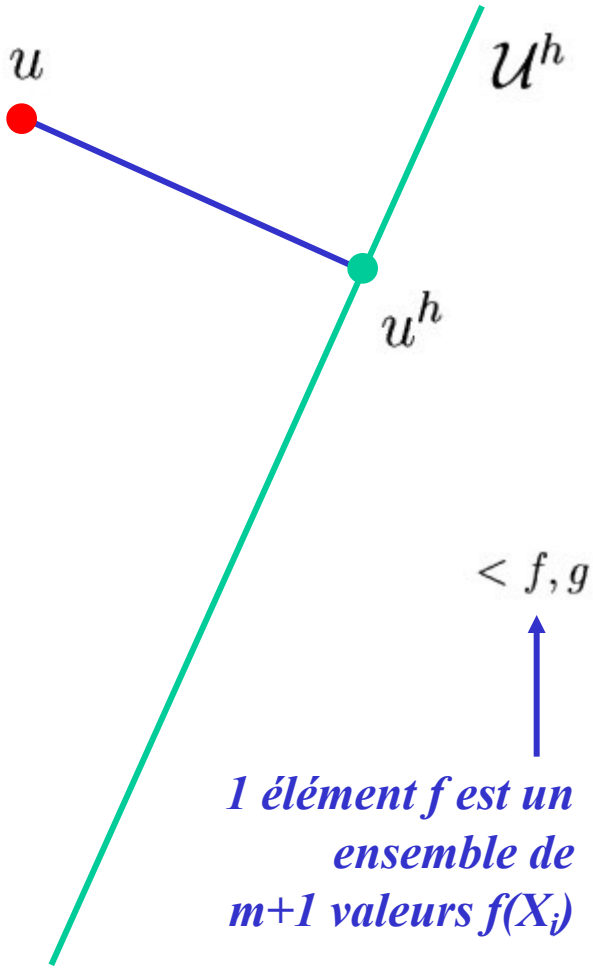
$$\phi_i(x) = \cos(ix) \qquad i = 1, 2, \ldots, n$$

$[-\pi, \pi]$.

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

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

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

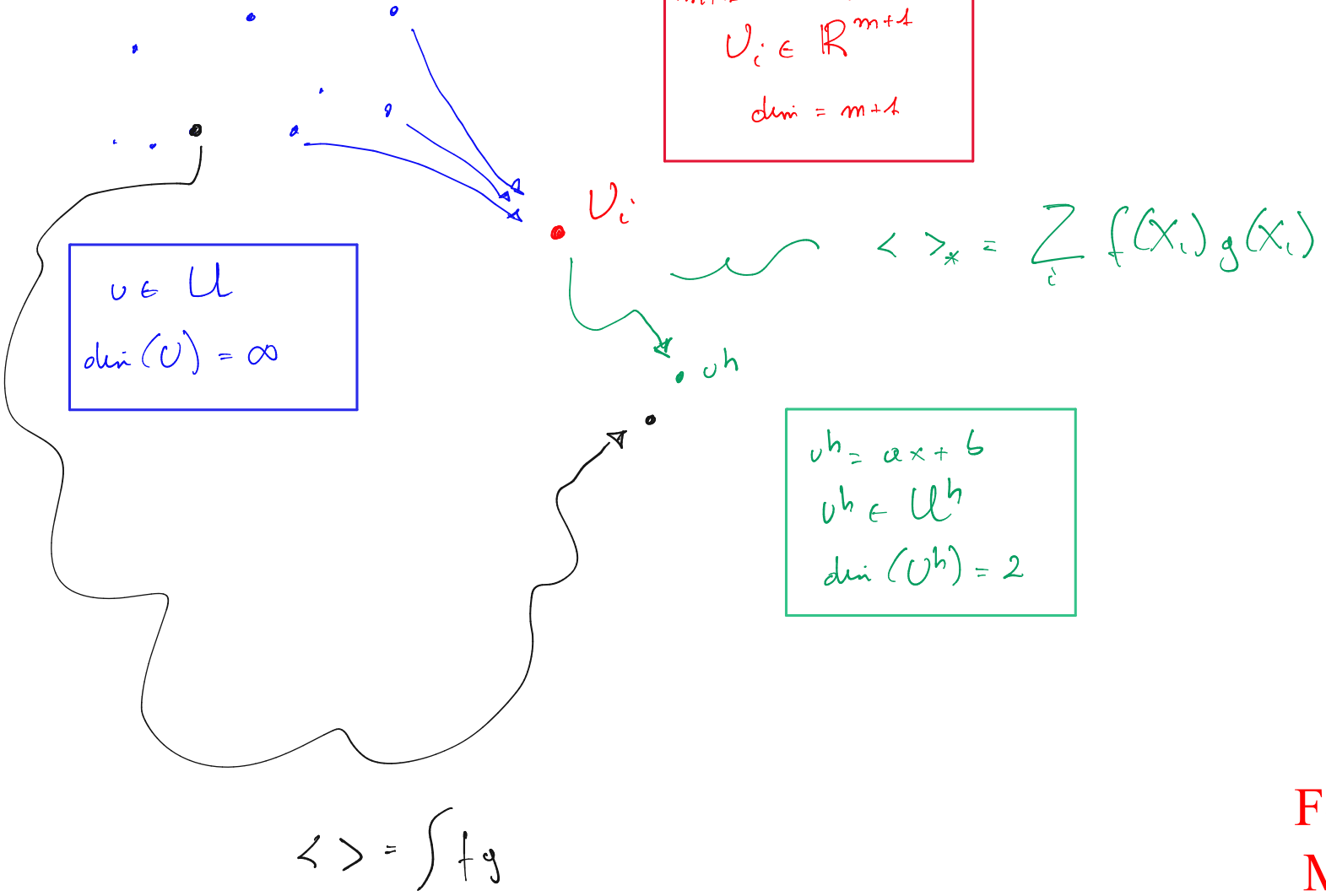


Approximer m+1 points au sens des moindres carrés avec n+1 fonctions de base

$$< f, g >_* \;=\; \sum_{j=0}^{m} f(X_j) g(X_j)$$

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

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



Fonction
Mesures
Approximation

En fait, on réalise deux approximations successives…

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

$$< \widehat{u}^h \, , \underbrace{(u - u^h)}_{e^h} >_* \quad = \quad 0, \qquad \forall \widehat{u}^h \in \mathcal{U}^h,$$

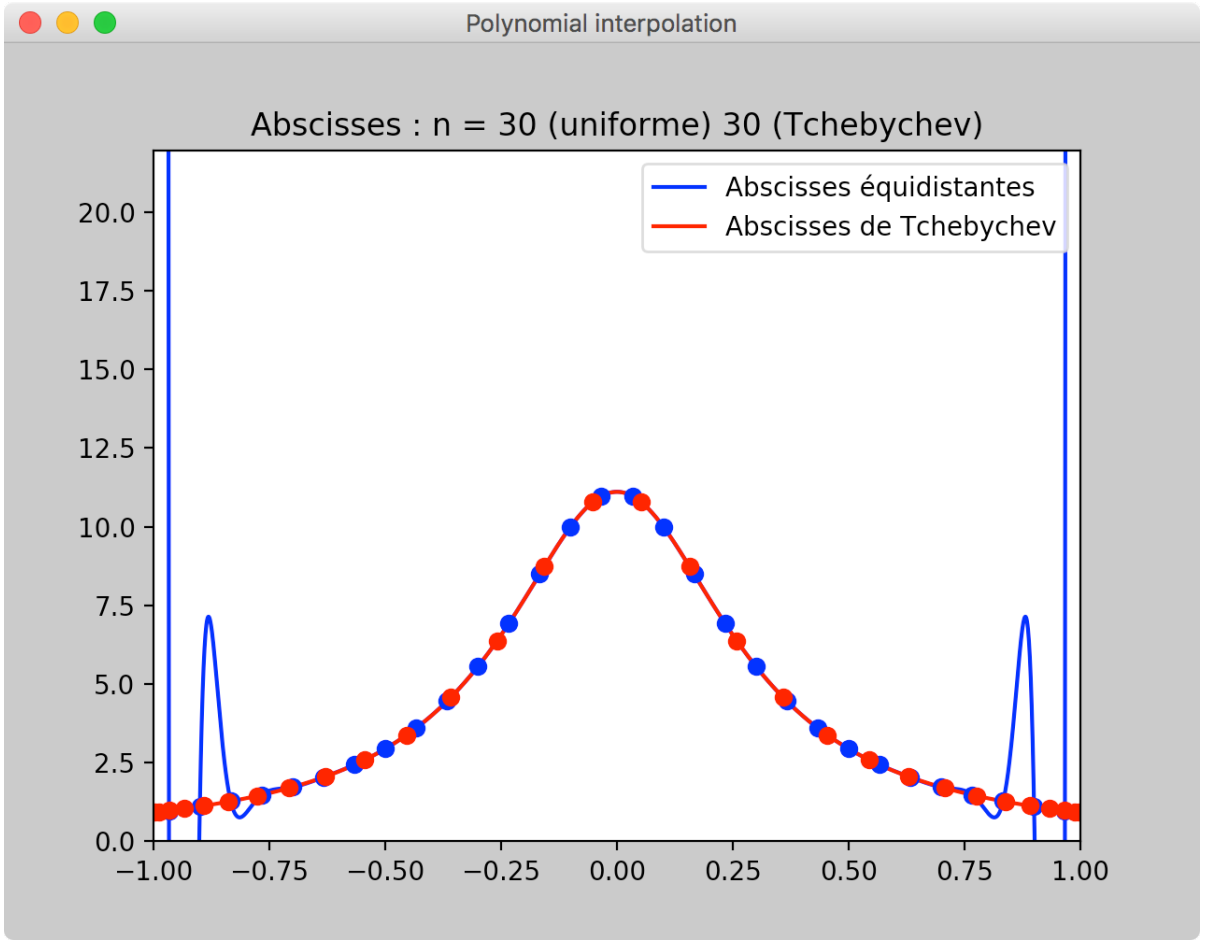
$$\sum_{i=0}^{m} \phi_k(X_i) \left(u(X_i) - \sum_{j=0}^{n} \phi_j(X_i) \; a_j \right) \quad = \quad 0, \qquad k = 0, 1, \ldots, n$$

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

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
```

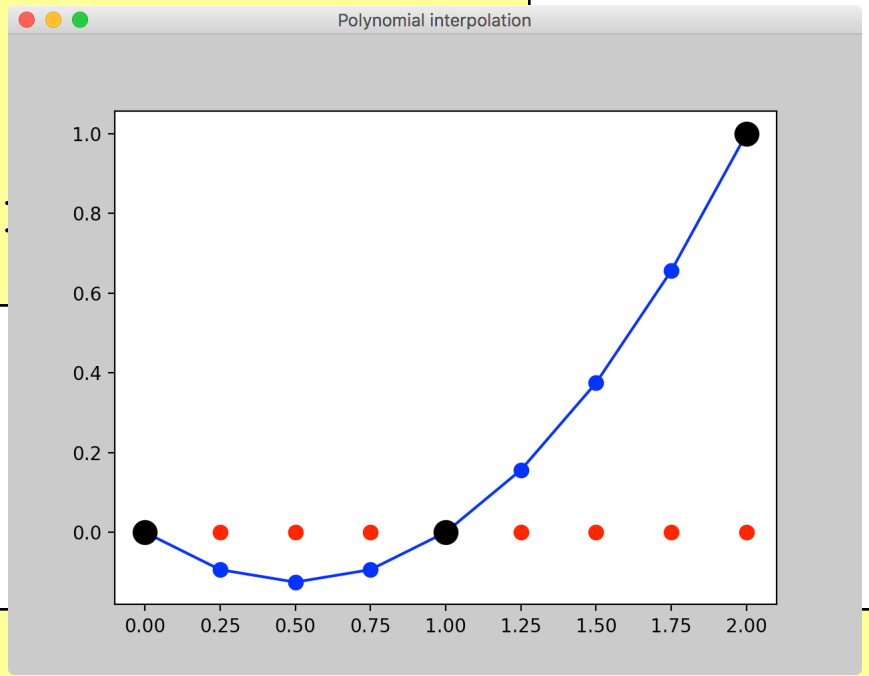
Pas si naïf que cela :
Implémentation robuste !
Pas d'erreur possible !

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

```
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_naive(x,[0,1,2],
toc()
```

```
Elapsed time is 0.000219 seconds
Elapsed time is 0.017949 seconds
```

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

```
Elapsed time is 1.415670 seconds
Elapsed time is 159.760674 seconds
```

CTRL-C

Pas de vectorisation :-(

Mais pré-allocation :-)

Suite de Pisano Fibonacci

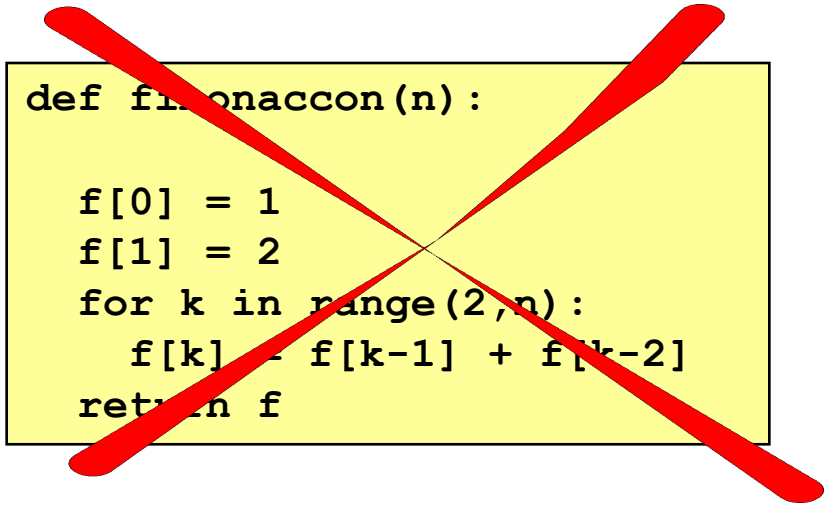
A man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair if it is supposed that every month each pair begets a new pair which from the second month on becomes productive…

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

```
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
```

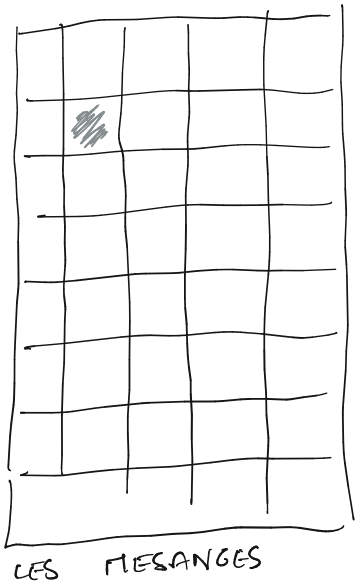
Et si tu veux vraiment pas pré-allouer les tableaux…

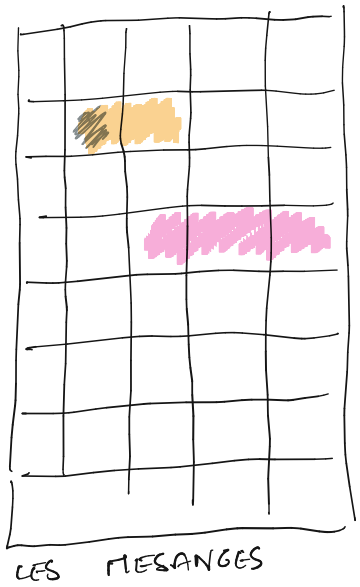
```
def fibonacci(n):
  f = zeros(n)
  f[0] = 1
  f[1] = 2
  for k in range(2,n):
    f[k] = f[k-1] + f[k-2]
  return f
```

```
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')
```





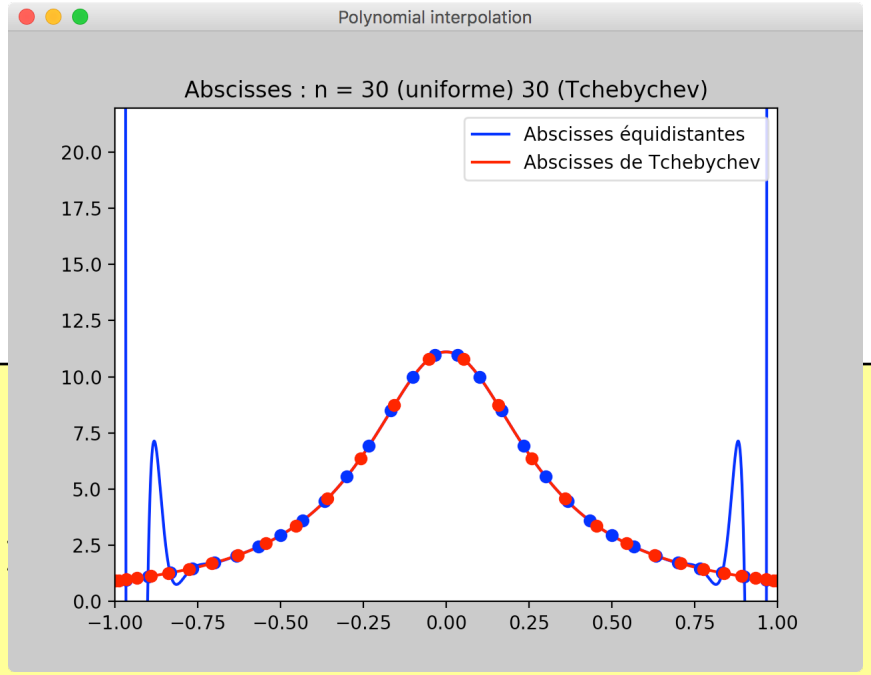
Logements sociaux à Charleroi

Ecrire le maximum d'opérations sous forme matricielle

```
U @ phi
```

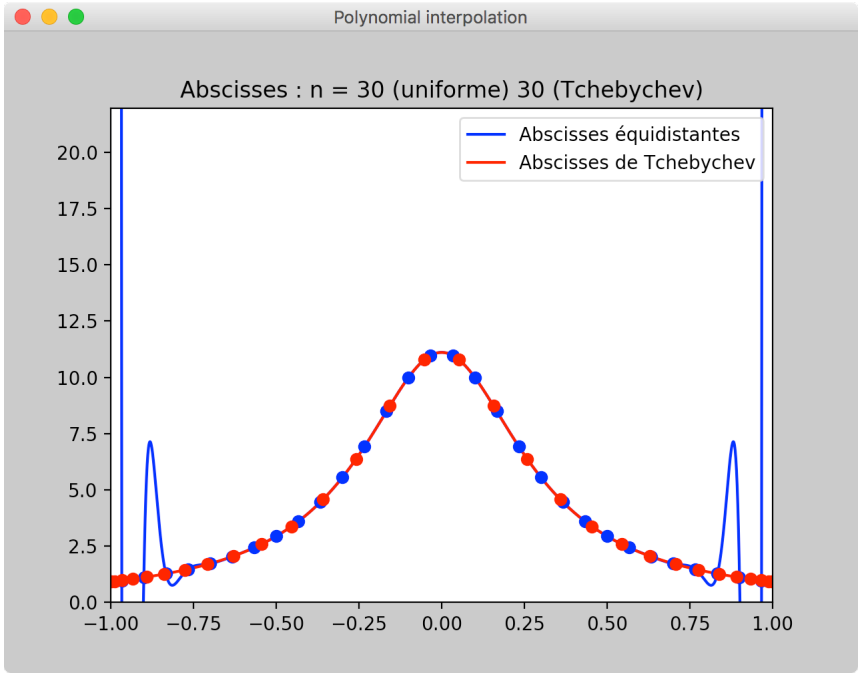
```
def lagrange(x,X,U):

  n = min(len(X),len(U))          # Evite un message d'erreur :-)
  phi = ones((n,len(x)))          # Preallocation (imposé !)
  for i in range(n):
    for k in list(range(0,i)) + list(range(i+1,n)):
      phi[i,:] = phi[i,:]*(x-X[k])/(X[i]-X[k])

  return U @ phi                  # Produit matriciel (plus rapide !)
```

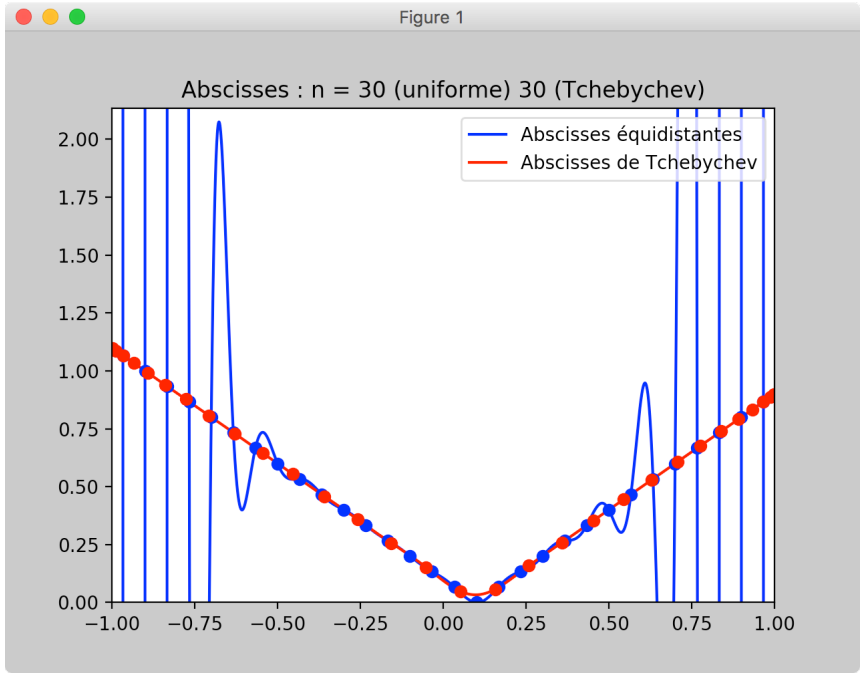
Obtenir la figure !




```
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. équidistantes')
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:
  Uunif = u(Xunif)
  Ucheb = u(Xcheb)
  plt.figure('Polynomial interpolation')
  plt.plot(x,lagrange(x,Xunif,Uunif),'-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 !