Distance et approximation

Moindres carrés et fonctions fondamentales

MAT-2930 Algèbre linéaire appliquée
Jean-François Lalonde

Image

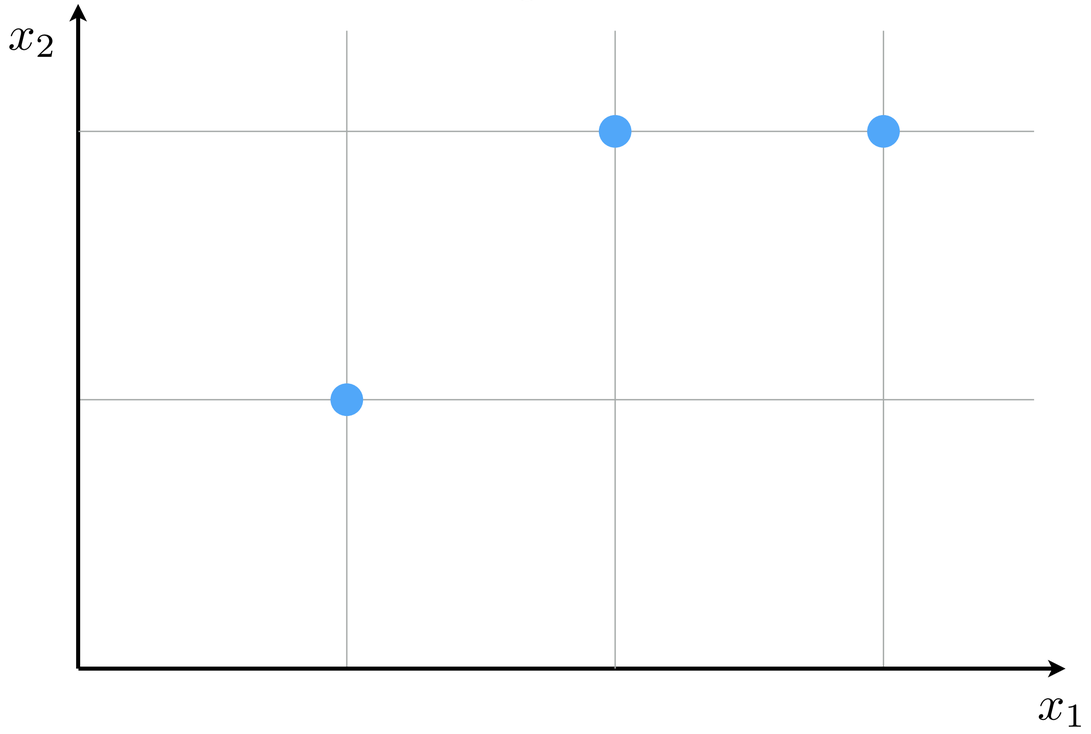
Rappel

Exemple : droite

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

Trouvons la meilleure droite qui approxime les trois points suivants.

$$x_2 = \frac{2}{3} + \frac{1}{2}x_1$$

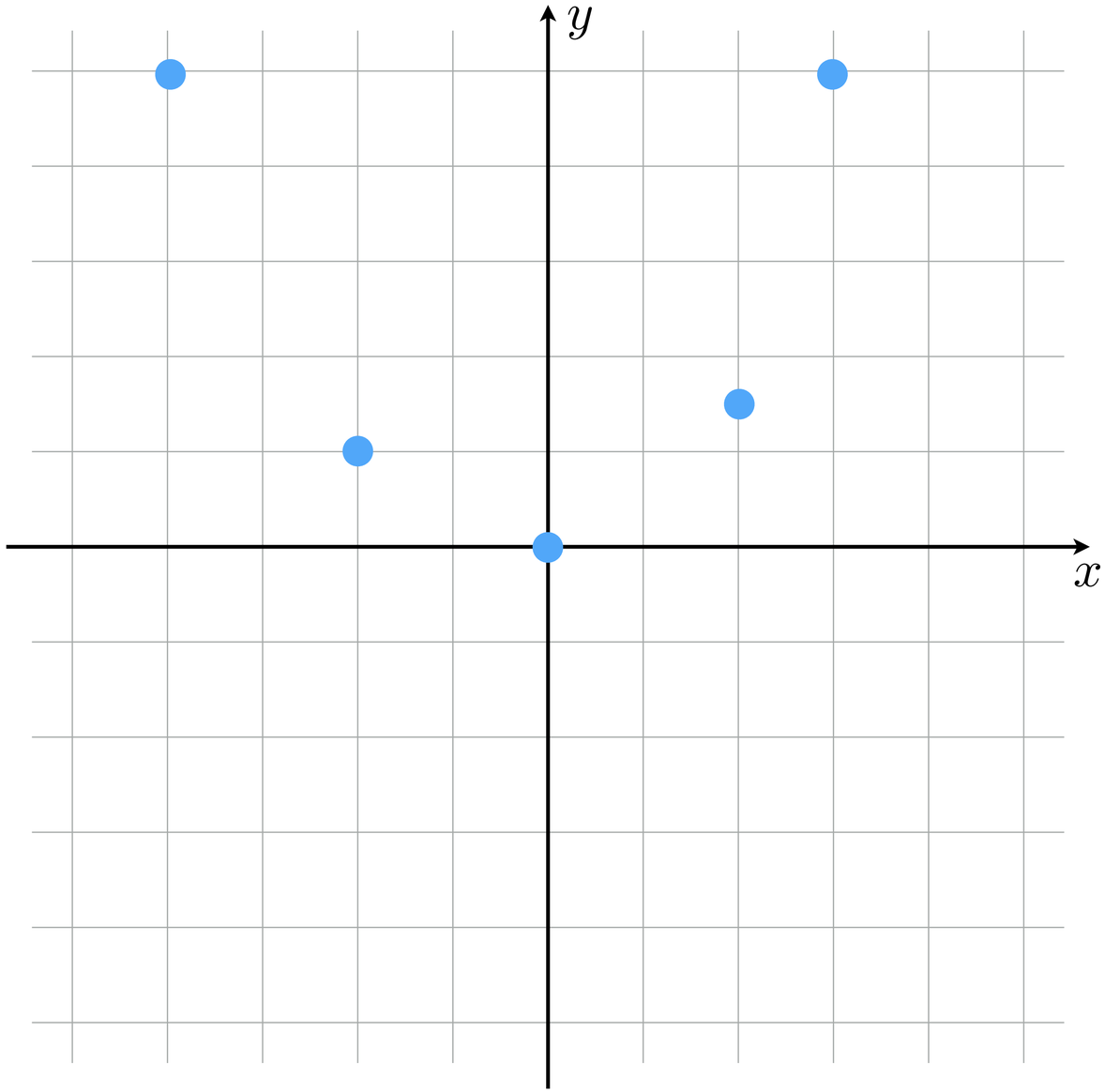


Peut-on approximer autre chose qu'une droite ?

Polynôme de degré 2

Écrivons les équations

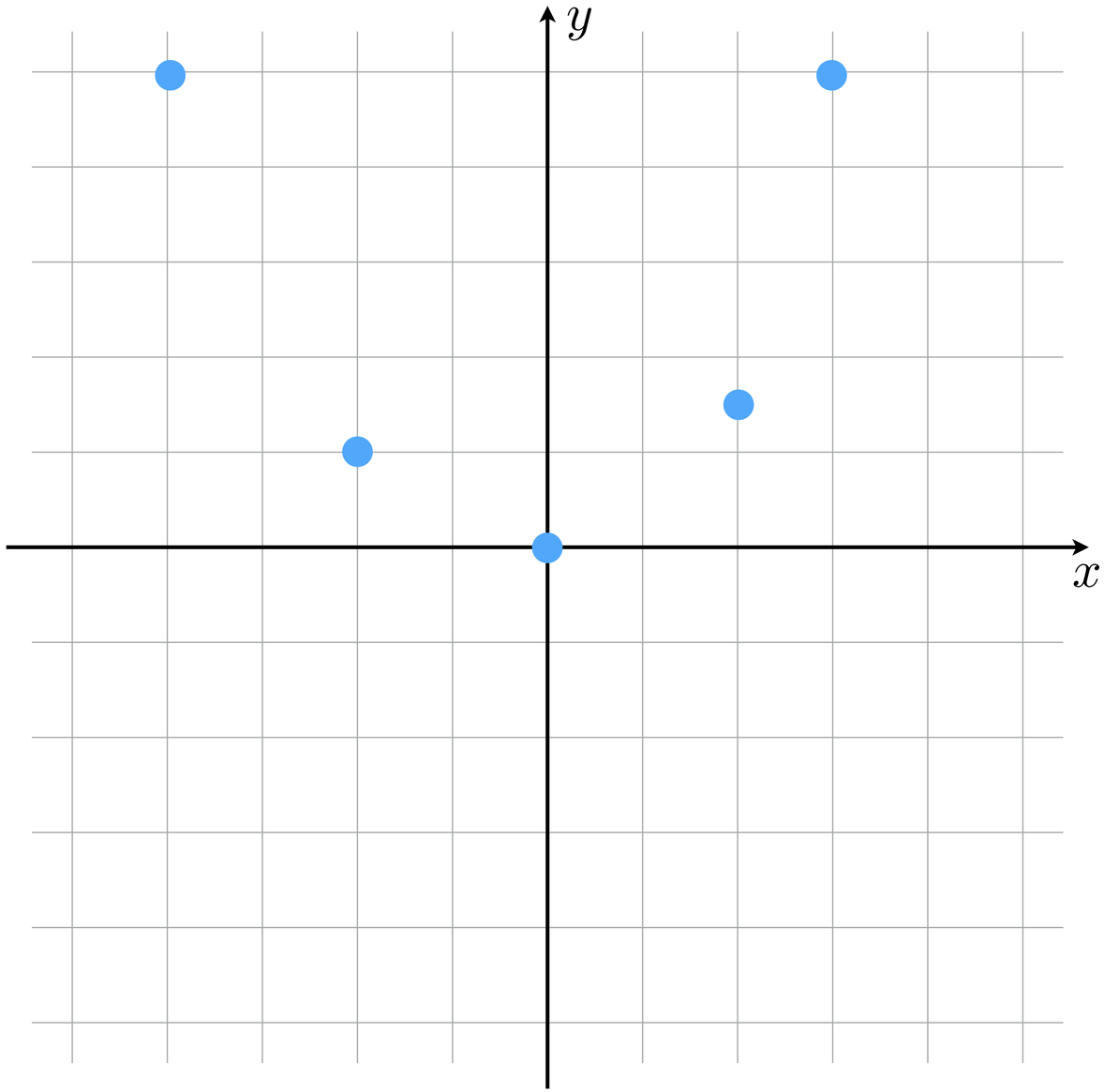
$$y = ax^2 + bx + c$$



Polynôme de degré 2

Exprimons sous forme matricielle

$$y = ax^2 + bx + c$$



Non-linéaire pour x, mais…
linéaire pour les coefficients !

Fonctions fondamentales

Forme matricielle pour un polynôme de degré 2 :

$$\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Fonctions fondamentales

Forme plus générale

$$\begin{bmatrix} f_1(x_1) & f_2(x_1) & f_3(x_1) & \dots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & f_3(x_2) & \dots & f_m(x_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_1(x_n) & f_2(x_n) & f_3(x_n) & \dots & f_m(x_n) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Dans python (voir PAX)

Quels sont les coefficients de ce polynôme de degré 3 ?

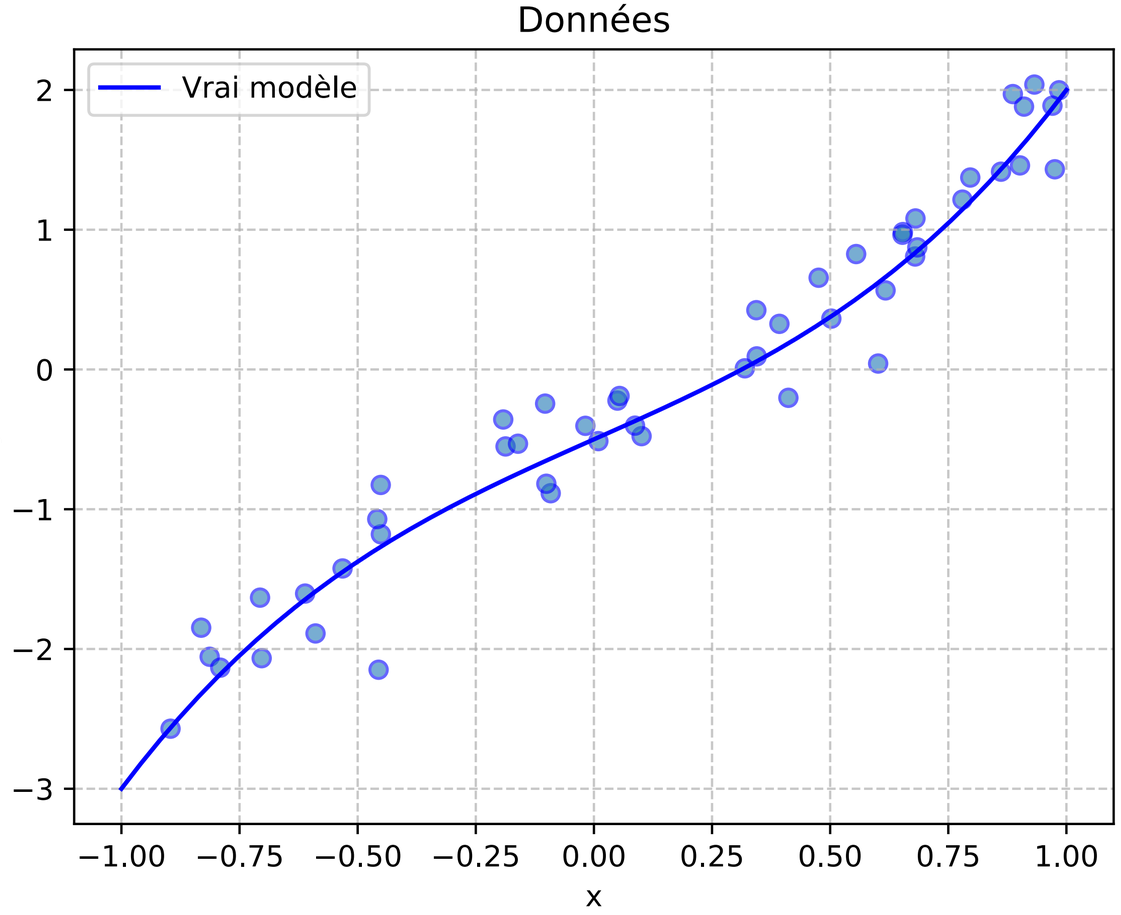
```
shape = (50,)
sigma = 0.25 # pour le bruit

p = np.array([1, 0, 1.5, -0.5])

x = np.random.uniform(-1, 1, shape)
e = np.random.normal(0, sigma, shape)

y = np.polyval(p, x) + e

# évaluer le polynôme grâce à np.polyval
xTest = np.linspace(-1, 1, 50)
yTest = np.polyval(p, xTest)
```



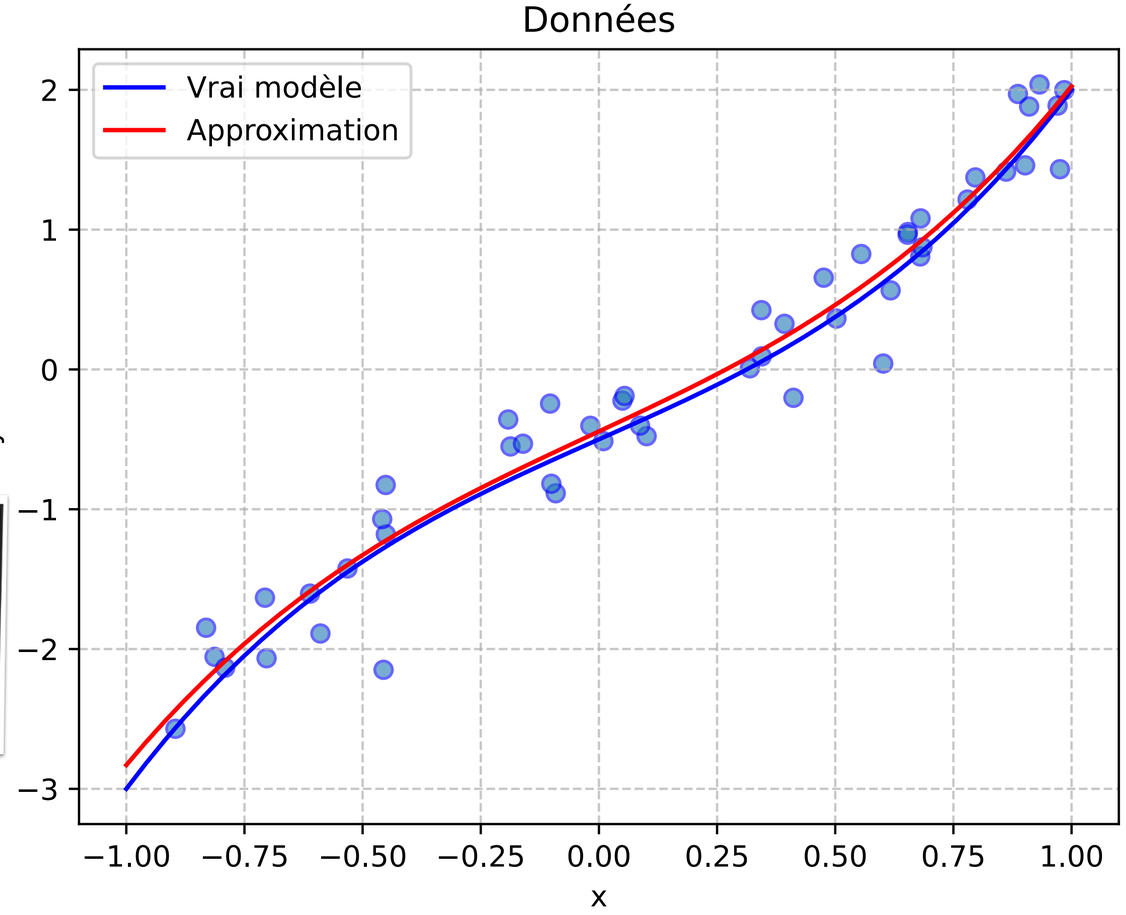
Attention : ne pas utiliser `np.polyfit`, `np.polyval` pour votre TP5 vous devez calculer la pseudo-inverse par vous-même !

Dans python (voir PAX)

Quels sont les coefficients de ce polynôme de degré 3 ?

```
phat = np.polyfit(x, y, 3)
```

```
Vrais paramètres :
[ 1.   0.   1.5 -0.5]
Paramètres approximés :
[ 0.707018 -0.057517  1.748796 -0.422526]
```



Attention : ne pas utiliser `np.polyfit` pour votre TP5, vous devez calculer la pseudo-inverse par vous-mêmes !