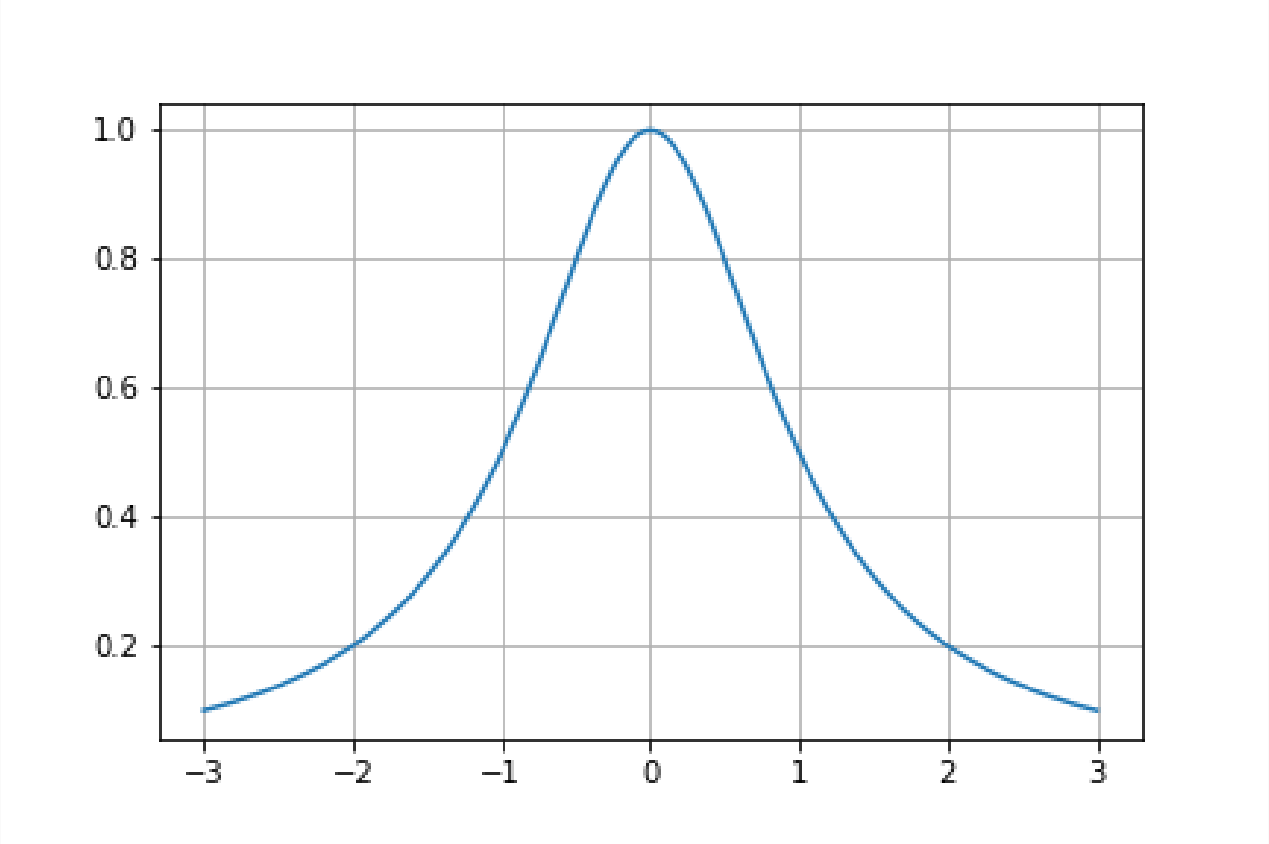
TP3: Introduction à l'interpolation

Bastien Nony, Alban Gossard

09 novembre 2020

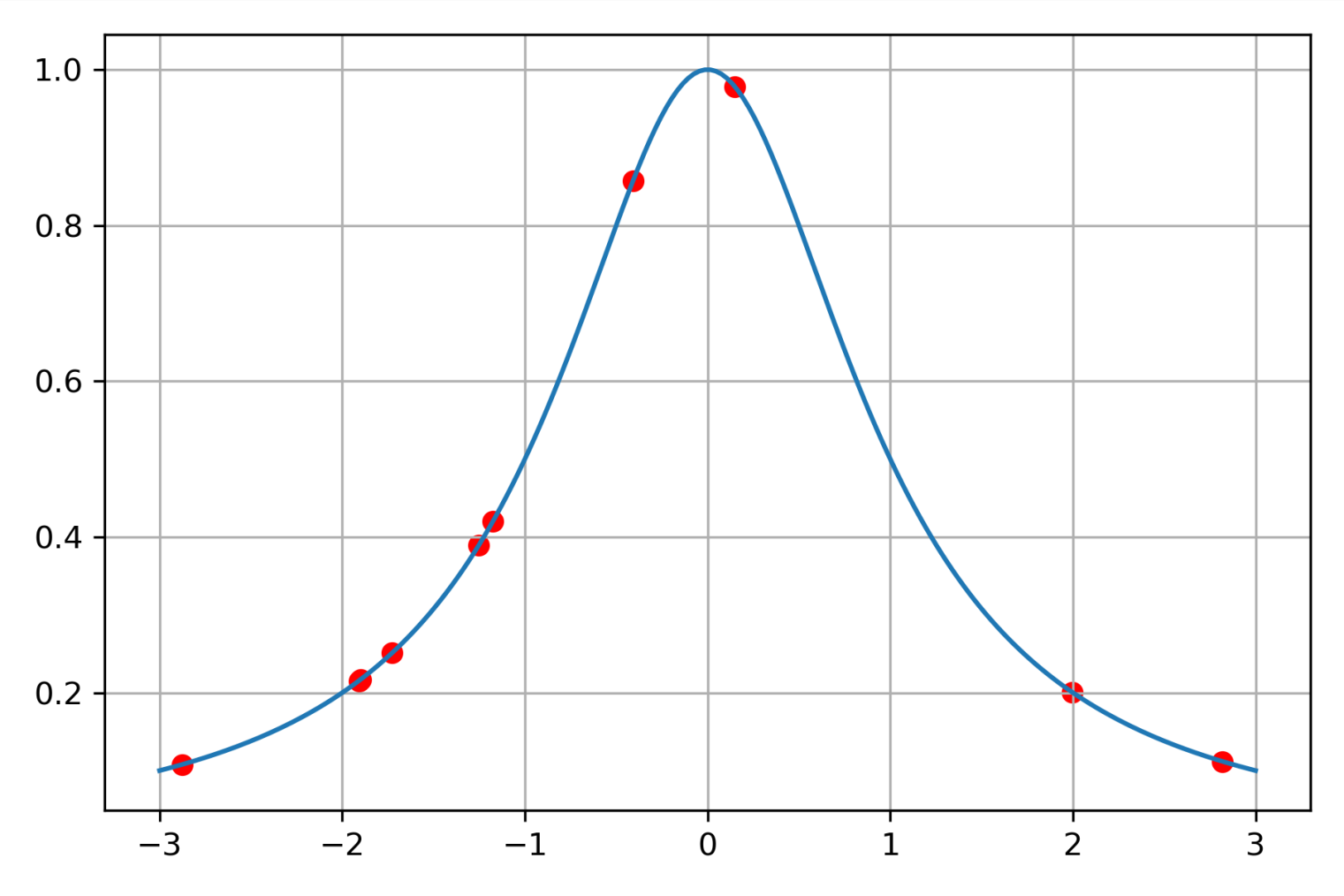
Analyse Numérique et Optimisation

Fonction à interpoler




```
def Runge(x):
    return 1./(1.+x**2.)
x=np.linspace(-3,3,200)
plt.plot(x,Runge(x))
plt.grid(True)
plt.show()
```

Choix des points d'interpolation



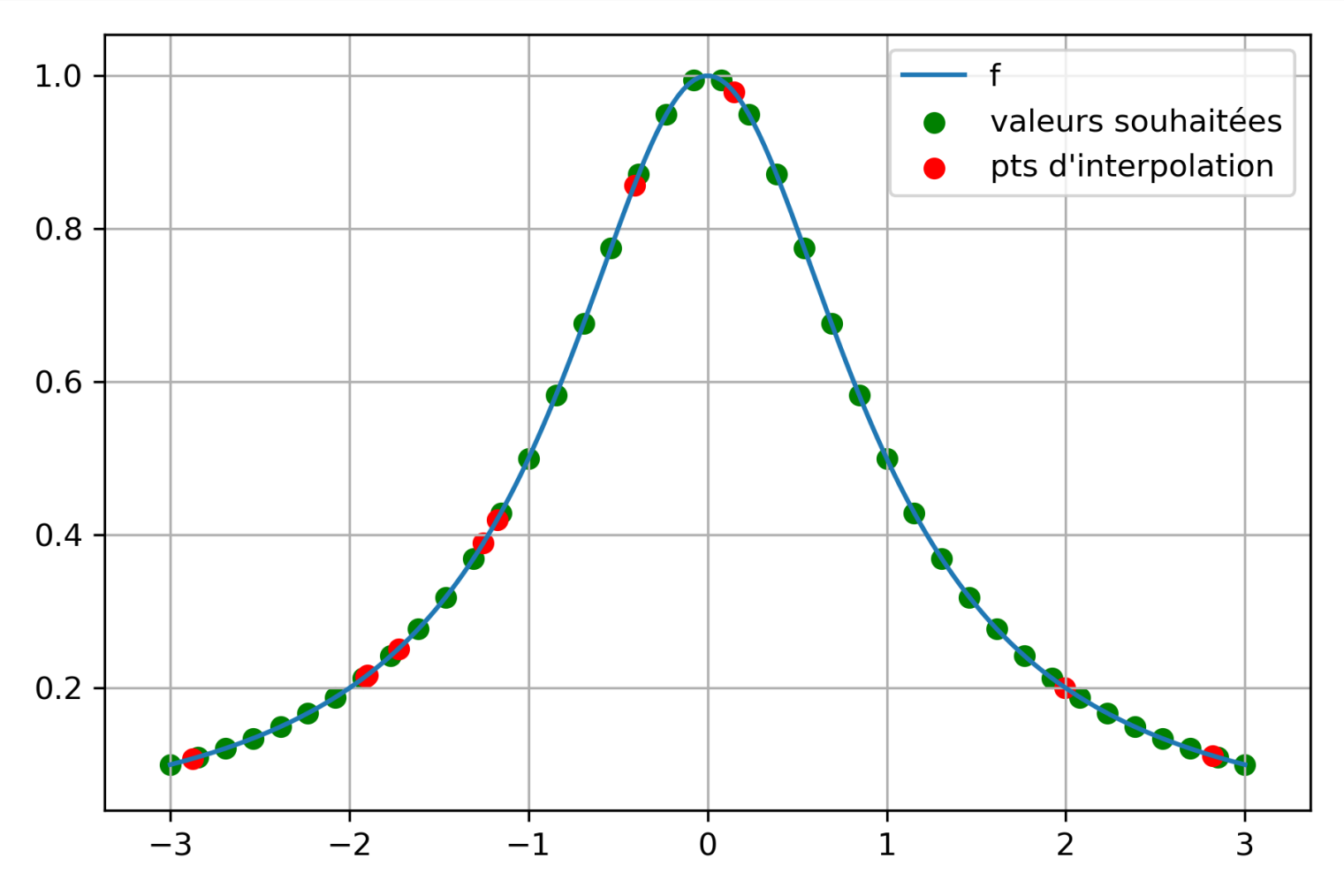
Interpolation

A partir de points d'interpolation $(x_i, y_i)_{i=0\dots n-1}$ de la fonction f on veut construire une fonction interpolante notée p.
La fonction p doit correspondre à f aux points d'interpolation :

$$y_i = p(x_i) \qquad \forall i = 0 \dots n-1 \tag{1}$$

⚠L'écriture de ces contraintes revient toujours à écrire un système linéaire !

Utilité de l'interpolation



Construction de points d'interpolation équirépartis

```
def Interp_Equi(a,b,m):
    return np.linspace(a,b,m)
print(Interp_Equi(0,1,5))
```

Méthode d'interpolation directe

On considère des fonctions interpolantes sous forme de polynômes :

$$p(X) = \sum_{k=0}^{n-1} a_k X^k \tag{2}$$

Les coefficients (a_k) sont des valeurs qui dépendent de f la fonction à interpoler.
Ce sont ces valeurs que l'on cherche à calculer pour déterminer la fonction interpolante.

Méthode d'interpolation directe

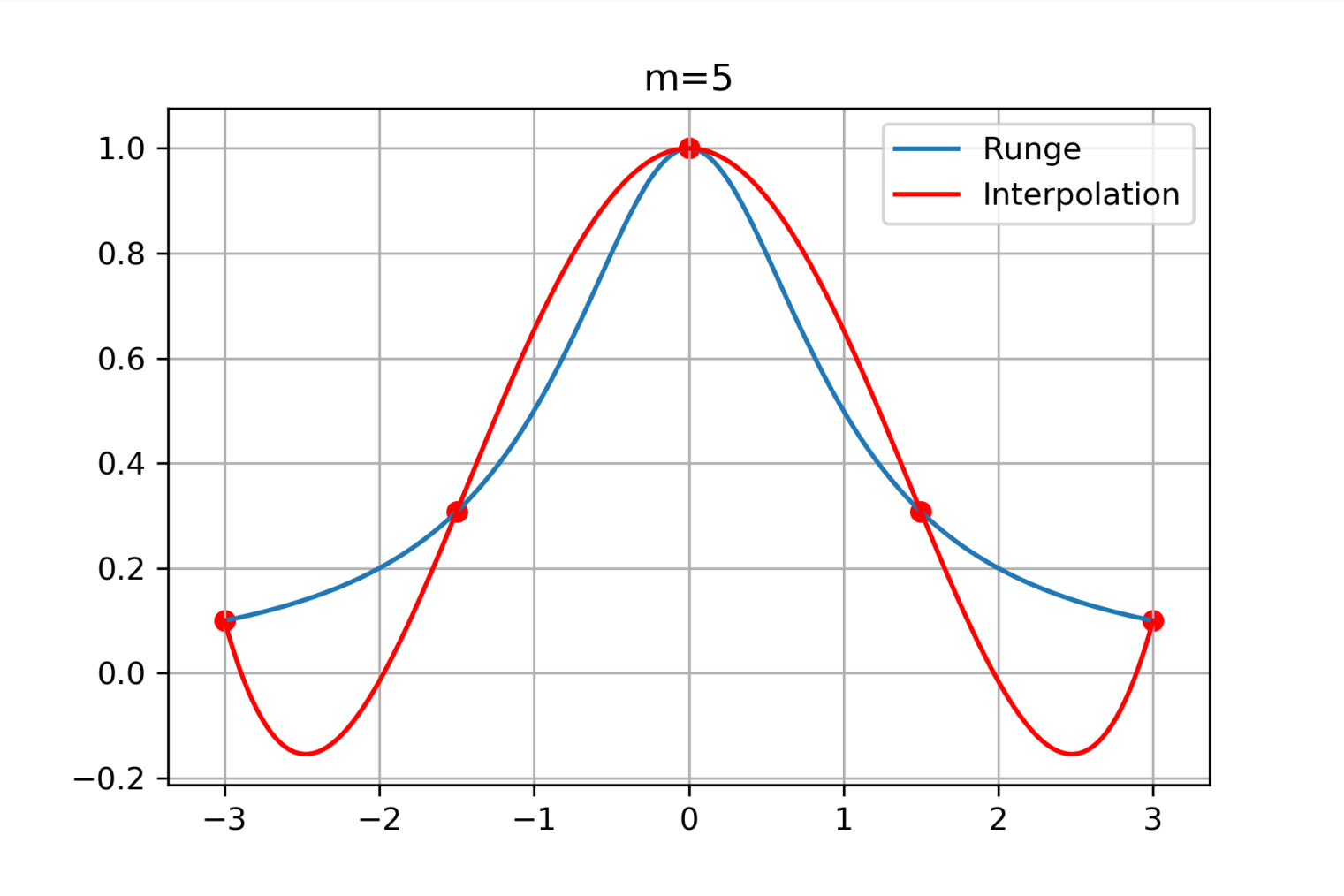
Calcul des valeurs de (a_k)

On écrit les contraintes aux différents points d'interpolation :

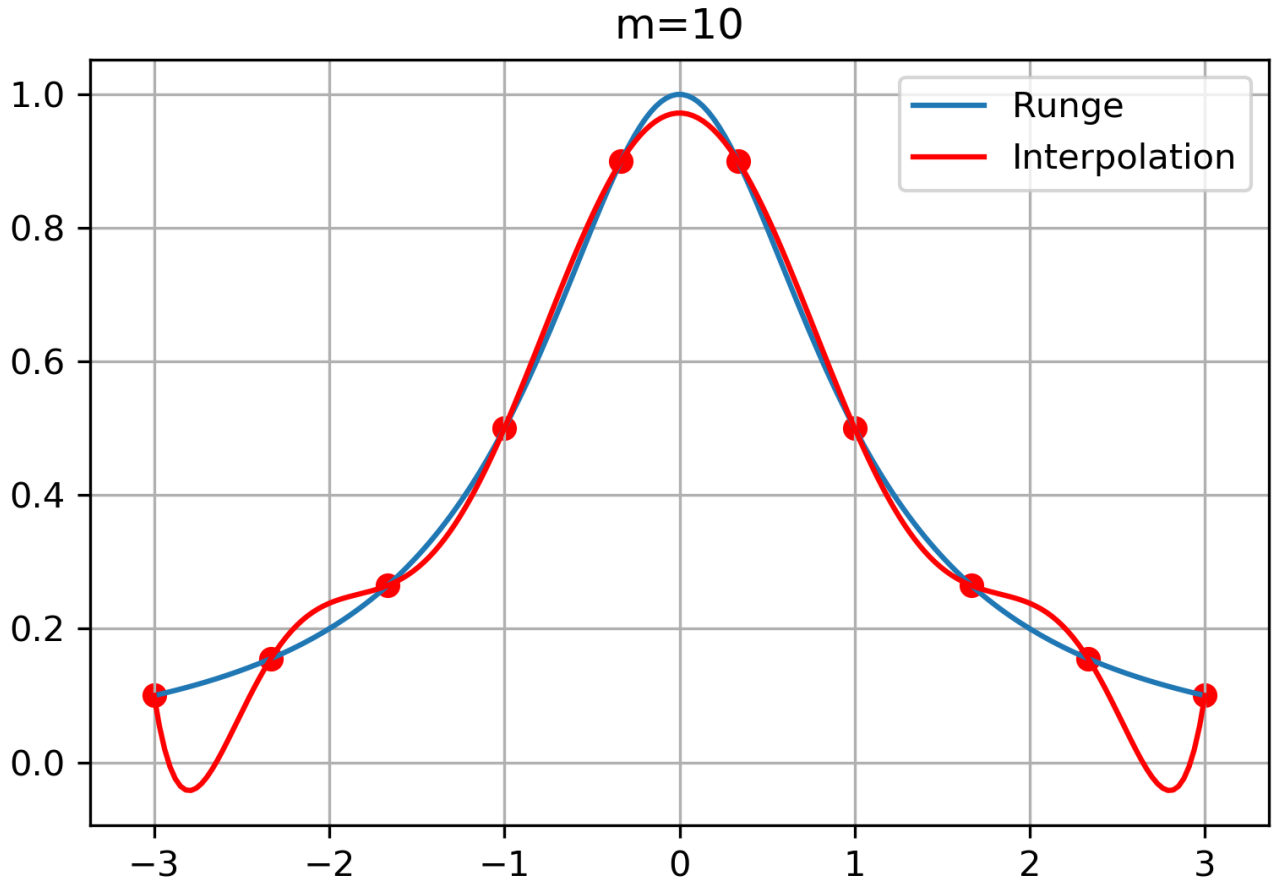
$$\begin{cases} p(x_0) = \sum_{k=0}^{n-1} a_k x_0^k = y_0 \\ \vdots \\ p(x_i) = \sum_{k=0}^{n-1} a_k x_i^k = y_i \\ \vdots \\ p(x_{n-1}) = \sum_{k=0}^{n-1} a_k x_{m-1}^k = y_{n-1} \end{cases} \quad \Longrightarrow \sum_{k=0}^{n-1} a_k x_i^k = y_i \qquad \forall i = 0 \dots n-1 \quad (3)$$

Système linéaire à résoudre : $Ma = y$ avec M la matrice de Vandermonde, $y = (y_k)$ et l'inconnue $a = (a_k)$.

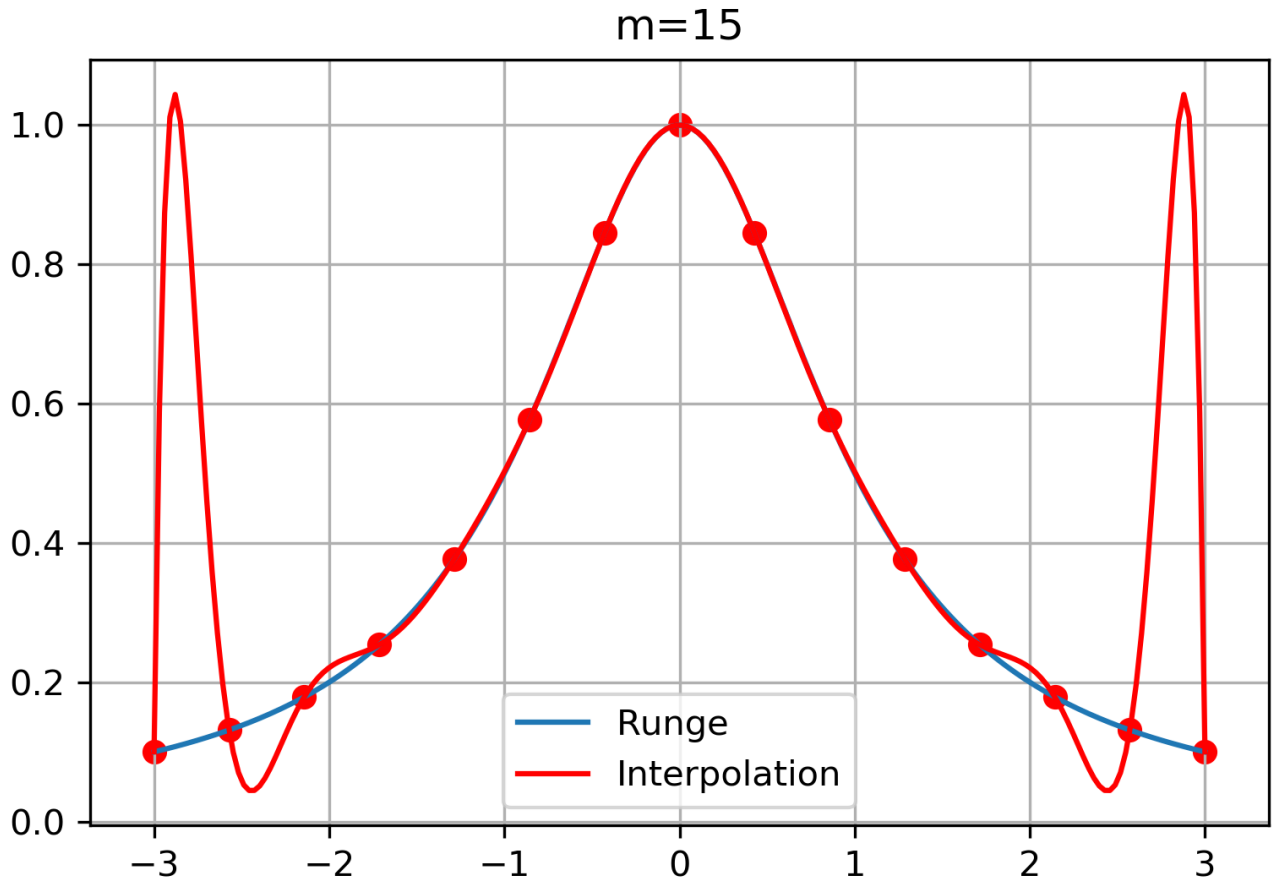
Méthode d'interpolation directe



Méthode d'interpolation directe



Méthode d'interpolation directe



Méthode d'interpolation de Lagrange

Interpolation par d'autres fonctions → **les polynômes de Lagrange**

$$p(X) = \sum_{j=0}^{n-1} y_j L_{j,n}(X) \tag{4}$$

$$\text{avec} \quad L_{j,n}(X) = \prod_{\substack{k=0 \\ k \neq j}}^{n-1} \frac{X - x_k}{x_j - x_k} \tag{5}$$

Ici l'égalité $p(x_i) = y_i \quad \forall i = 0 \dots n-1$ est toujours vérifiée (propriété des polynômes de Lagrange).

Méthode d'interpolation de Lagrange

On veut évaluer p en un ensemble de valeurs $\hat{x} = (\hat{x}_i)_{i=0\dots m-1}$:

$$p(X) = \sum_{j=0}^{n-1} y_j L_{j,n}(X)$$

$$p(\hat{x}_i) = \sum_{j=0}^{n-1} y_j L_{j,n}(\hat{x}_i)$$

On pose $\mathscr{L}_{i,j} = L_{j,n}(\hat{x}_i)$

$$\hat{y}_i = p(\hat{x}_i) = \sum_{j=0}^{n-1} \mathscr{L}_{i,j} y_j$$

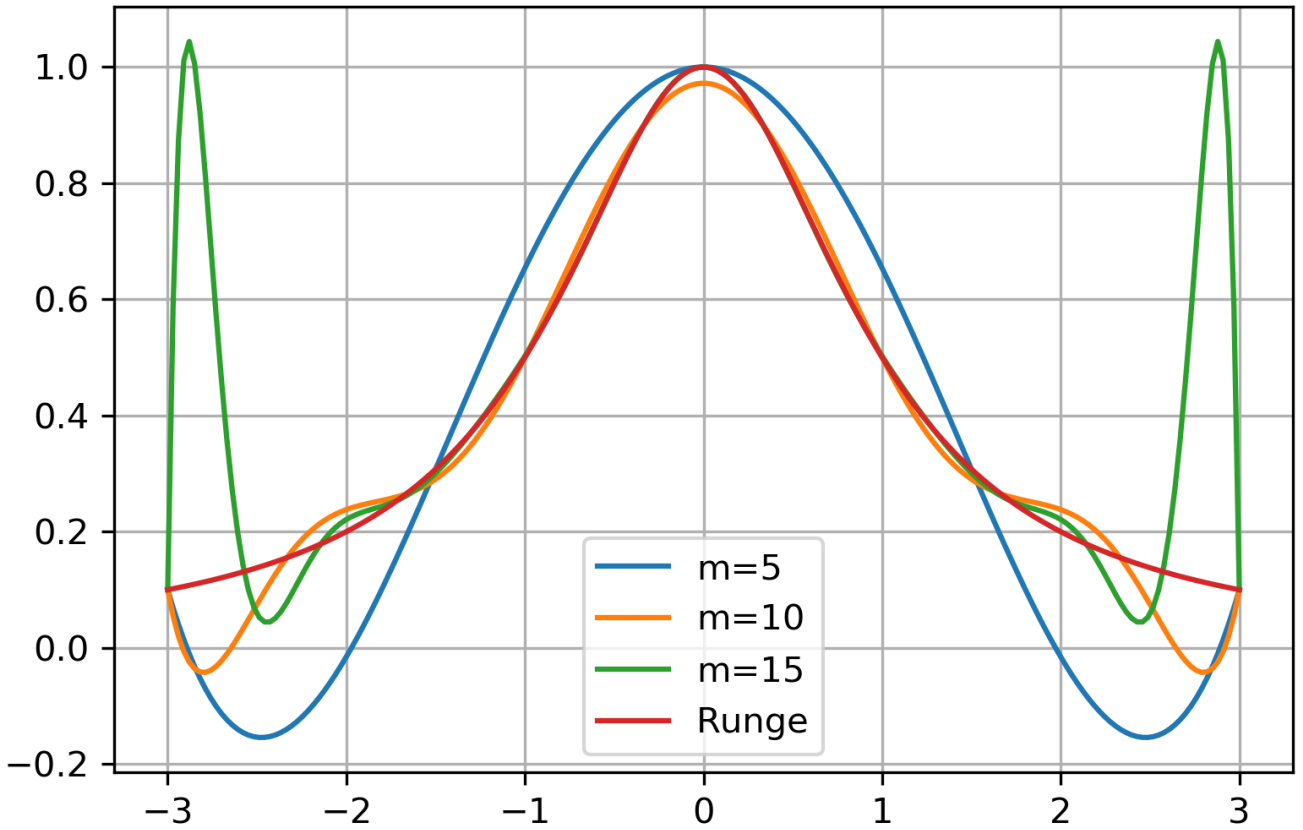
$$\hat{y} = \mathscr{L} y$$

→ simple multiplication matrice-vecteur

Ici

$$\mathscr{L}_{i,j} = \prod_{\substack{k=0 \\ k \neq j}}^{n-1} \frac{\hat{x}_i - x_k}{x_j - x_k}$$

Méthode d'interpolation de Lagrange



Méthode d'interpolation de Newton

Interpolation par d'autres fonctions → **les polynômes de Newton**

$$p(X) = \sum_{j=0}^{n-1} z_j N_{j,n}(X) \tag{6}$$

avec

$$N_{j,n}(X) = \prod_{k=0}^{j-1} (X - x_k)$$

1) Calcul de (z_j) tel que $y_i = p(x_i) \quad \forall i = 0 \dots n-1$

2) Evaluation de la fonction interpolante p en des points $(\hat{x}_i)_{i=0\dots m-1}$

① Calcul de (z_j)

Comme pour la méthode directe on cherche à calculer (z_j) tel que $y_i = p(x_i)$.

Cela revient à résoudre le système linéaire $Nz = y$ avec N la matrice $n \times n$ dont les éléments sont donnés par :

$$N_{i,j} = \prod_{k=0}^{j-1} (x_i - x_k) \tag{7}$$

Méthode d'interpolation de Newton

② Evaluation de la fonction interpolante

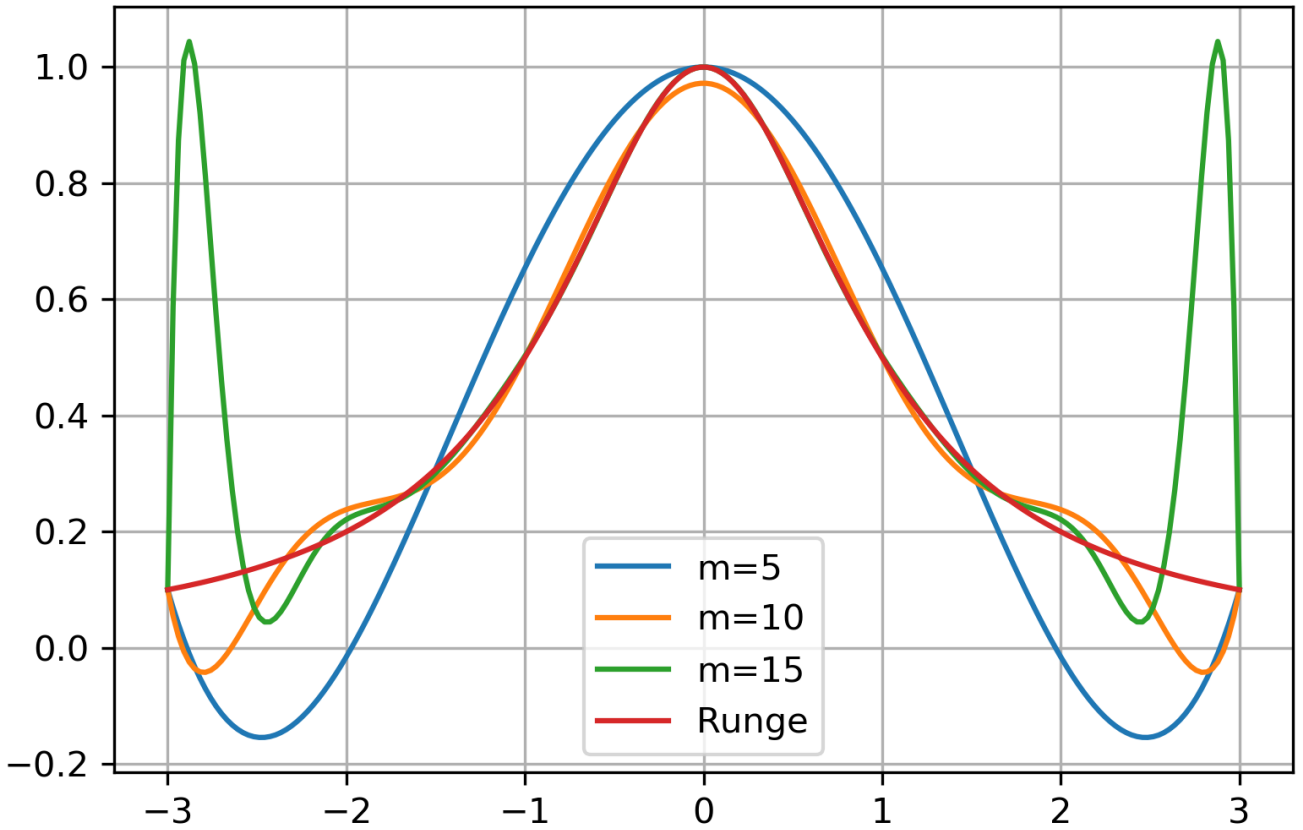
L'interpolation en m points $\hat{x}_i$ est donnée par l'expression des polynômes de Newton en fonction de z.

$$\begin{aligned} p(\hat{x}_i) &= \sum_{j=0}^{n-1} z_j N_{j,n}(\hat{x}_i) \\ &= \sum_{j=0}^{n-1} L_{i,j} z_j \end{aligned}$$

Avec L la matrice $m \times n$

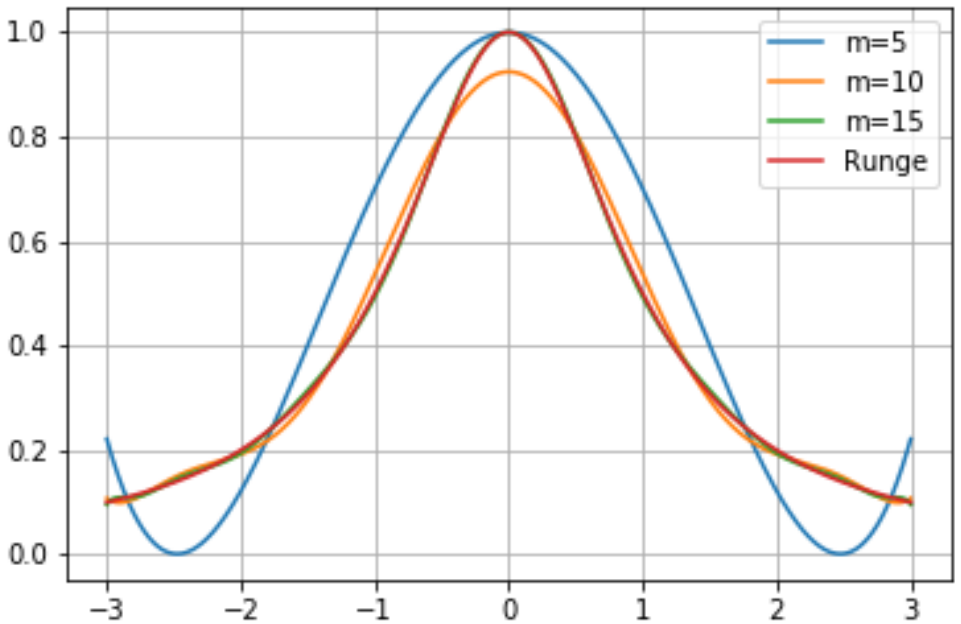
$$L_{i,j} = N_{j,n}(\hat{x}_i) = \prod_{k=0}^{j-1} (\hat{x}_i - x_k) \tag{8}$$

Méthode d'interpolation de Newton



Points d'interpolation de Chebyshev

```
def Interp_Chebyshev(a,b,m):
    n=m
    i=np.arange(n)
    y=np.cos((2*i+1)*np.pi/(2*n))
    x=(a+b)/2.+(a-b)*y/2.
    return x
print(Interp_Chebyshev(-3,3,5)) # [-2.85316955e+00 -1.76335576e+00
#  -1.83697020e-16  1.76335576e+00  2.85316955e+00]
```



Comparaison d'erreur