Analyse numérique

1A UE MATH1
Quelques méthodes itératives

Guillaume Chiavassa

Centrale Marseille

Problèmes fréquents en calcul scientifique

- Calcul des zéros d'une fonction
- Calcul des extrema libres ou avec contraintes d'une fonction
- Calcul de la solution d'un système linéaire

La plupart des méthodes classiques sont des méthodes itératives :

x_0 donné
Itérer $x_{k+1} = H(x_k)$ jusqu'à la précision souhaitée

Questions :

- Choix de x_0
- Fonction d'itération $H()$
- Critère d'arrêt
- Convergence quand $k \to +\infty$
- Vitesse de convergence

Equations algébriques non-linéaires

Résolution de $f(x) = 0$

La fonction f peut être :

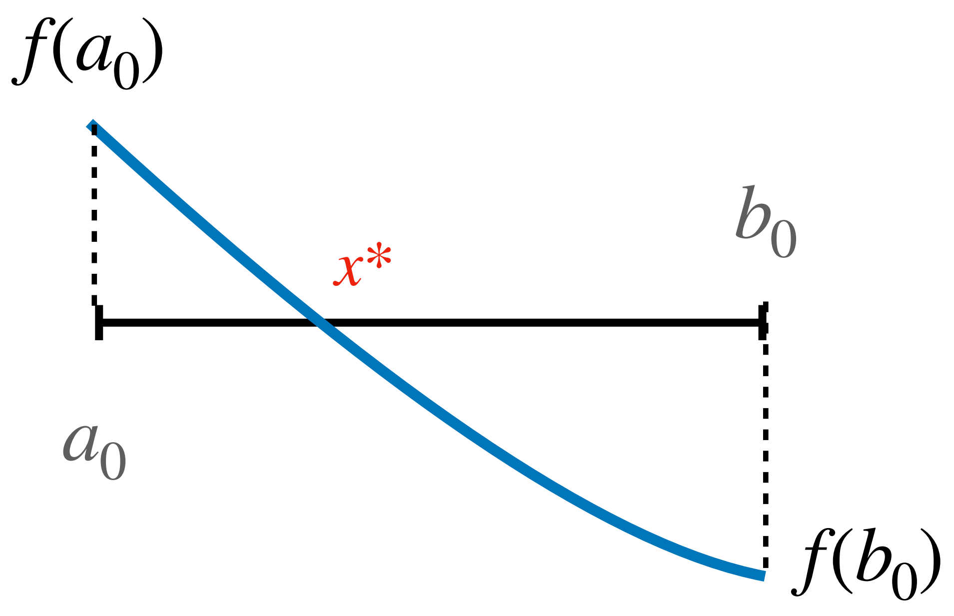
- à variable réelle ou complexe
- à valeurs réelles ou complexes,
- de $\mathbb{R}^p$ dans $\mathbb{R}^q$,
- connue explicitement ou non,
- de dérivées calculables ou non,
- polynomiale, etc…

Algorithmes de base pour une fonction f continue de variable réelle à valeurs dans $\mathbb{R}$

- Dichotomie
- Point fixe
- Newton et Sécante

Méthode de dichotomie/bissection

Intervalle $[a_0, b_0]$ tel que $f(a_0)f(b_0) < 0$



Pour $k = 0,\ldots,N$

$$x = \frac{a_k + b_k}{2}$$

Si $f(a_k)f(x) < 0$ alors $a_{k+1} = a_k$ et $b_{k+1} = x$

Sinon $a_{k+1} = x$ et $b_{k+1} = b_k$

➡ $x^* \simeq \dfrac{a_N + b_N}{2}$

- Convergence assurée si $f(a_0)f(b_0) < 0$
- Choix de N pour une précision donnée ϵ :

$$N \log 2 \geq \log(\frac{b_0 - a_O}{\epsilon})$$

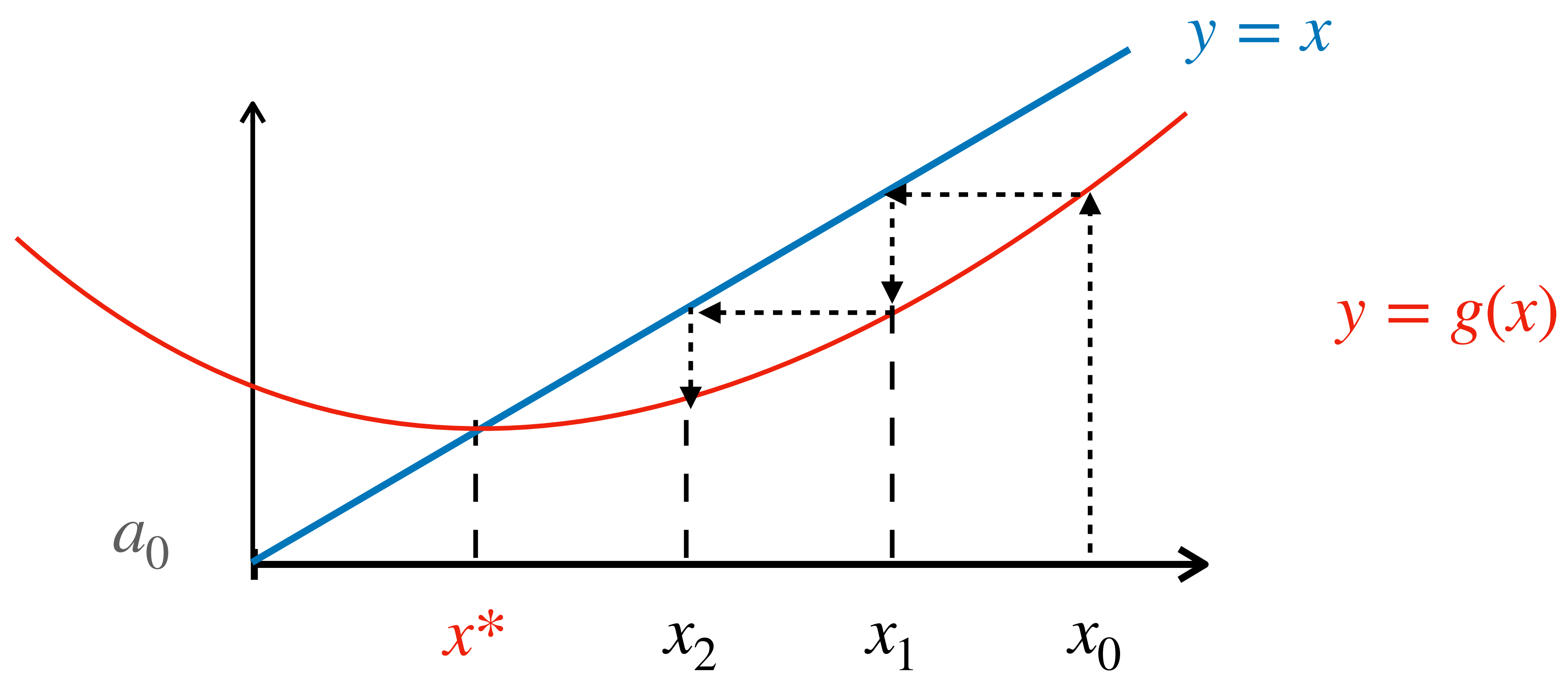
- 1 seule évaluation de f par itération
- Pas généralisable en dimension >1

Méthode de point fixe

1) On transforme l'équation $f(x) = 0$ en une équation de type $g(x) = x$ ayant la (les) même(s) solution(s)

Par exemple $g(x) = f(x) + x$, mais il n'y a évidemment pas d'unicité

2) On choisit une valeur x_0 et on itère la suite $x_{k+1} = g(x_k)$



Justification: Si $\lim_{k \to +\infty} x_n = x^*$ on a forcément $g(x^*) = x^*$ et donc $f(x^*) = 0$

Théorème du point fixe

Soit $g : [a, b] \to [a, b]$, dérivable et telle que $\forall x \in [a, b],\ |g'(x)| \leq K$, avec $0 \leq K < 1$,
Alors pour tout $x_0 \in [a, b]$ la suite $x_{k+1} = g(x_k)$ converge vers l'unique point fixe de g.

Vitesse de convergence

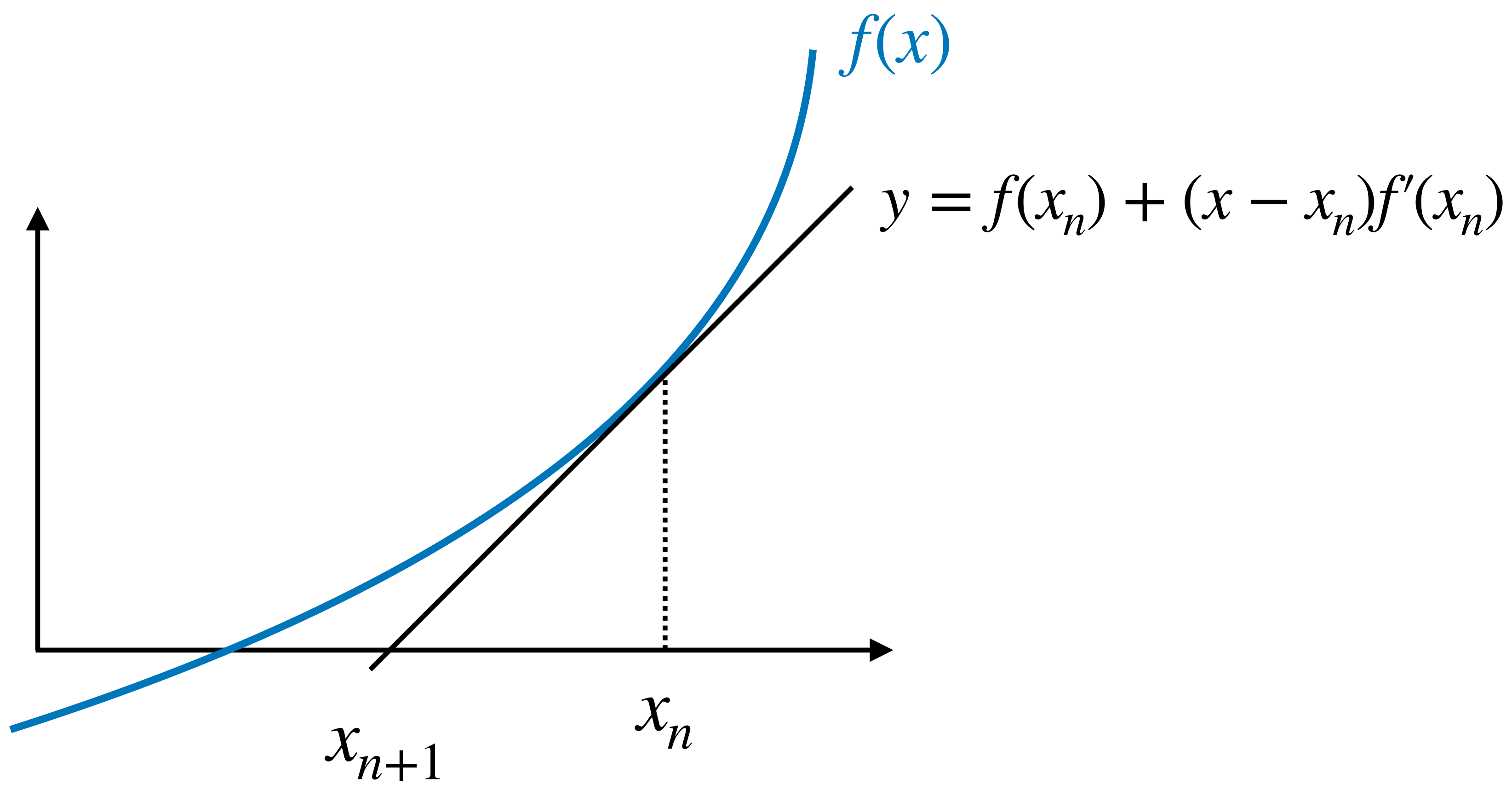
Soit x_k une suite convergeant vers x^*. Si il existe $C > 0$ et un entier $p > 0$ tels que $\lim_{k \to +\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|^p} = C$

Alors la convergence est d'ordre p.

$p = 1$ convergence linéaire, $p = 2$ quadratique

Méthode de Newton

On prend la fonction $g(x) = x - \dfrac{f(x)}{f'(x)}$



Si elle converge la méthode de Newton est quadratique

Démo

Remarques

- Si $f'(x^*) = 0$ la convergence sera au mieux linéaire
- Ici $g'(x^*) = 0$, on est donc localement dans le cadre du théorème du point fixe
- Si x_0 est choisit « assez près » de x^* la méthode convergera
- Problème : choix du « assez près » en pratique !
- On appelle bassin d'attraction de x^* l'ensemble des valeurs initiales x_0 pour lesquelles une méthode de point fixe converge vers x^*.
- Newton nécessite l'évaluation de f et f' à chaque évaluation

Méthode de la sécante

- On remplace $f'(x_k)$ dans Newton par le taux d'accroissement $\dfrac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$

$$x_{k+1} = x_k - f(x_k)\,\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

- Choix de x_0 et x_1 ?
- Si elle converge la méthode est d'ordre $p = \dfrac{1+\sqrt{5}}{2} = 1{,}618...$

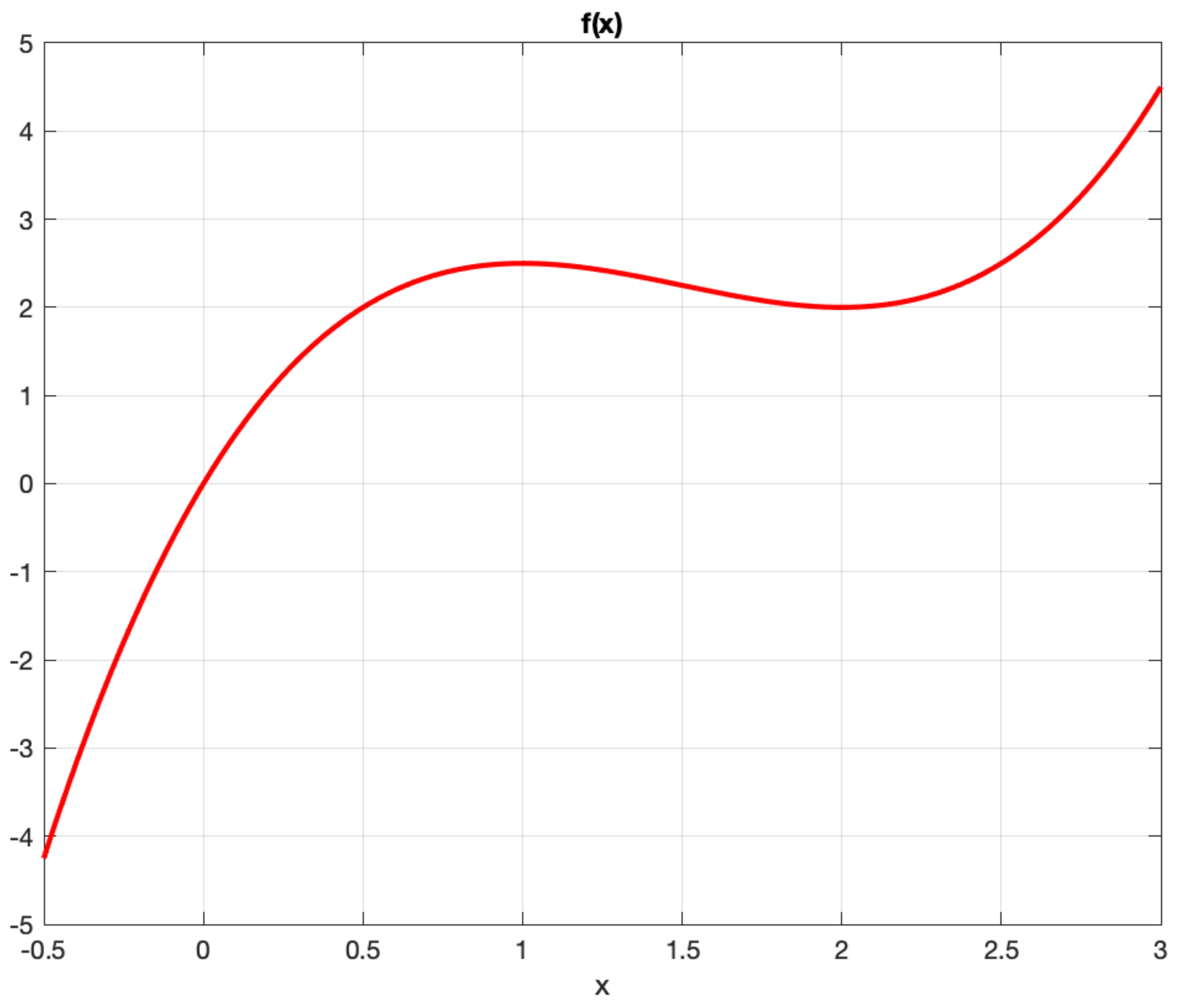
Critères d'arrêt

- Plusieurs choix possibles pour une précision donnée ϵ
- $|x_{k+1} - x_k| \leq \epsilon$ pas toujours efficace $(x_k = 1 + 1/2 + \ldots + 1/k)$
- $|f(x_k)| \leq \epsilon$ pas efficace si f très « plate » autour de sa racine
- $|f(x_k) - f(x_{k-1})| \leq \epsilon$ évite division par zéro dans la sécante

En pratique on combine plusieurs critères et on prend si possible des erreurs relatives
Et on met toujours un nombre maximum d'itérations!

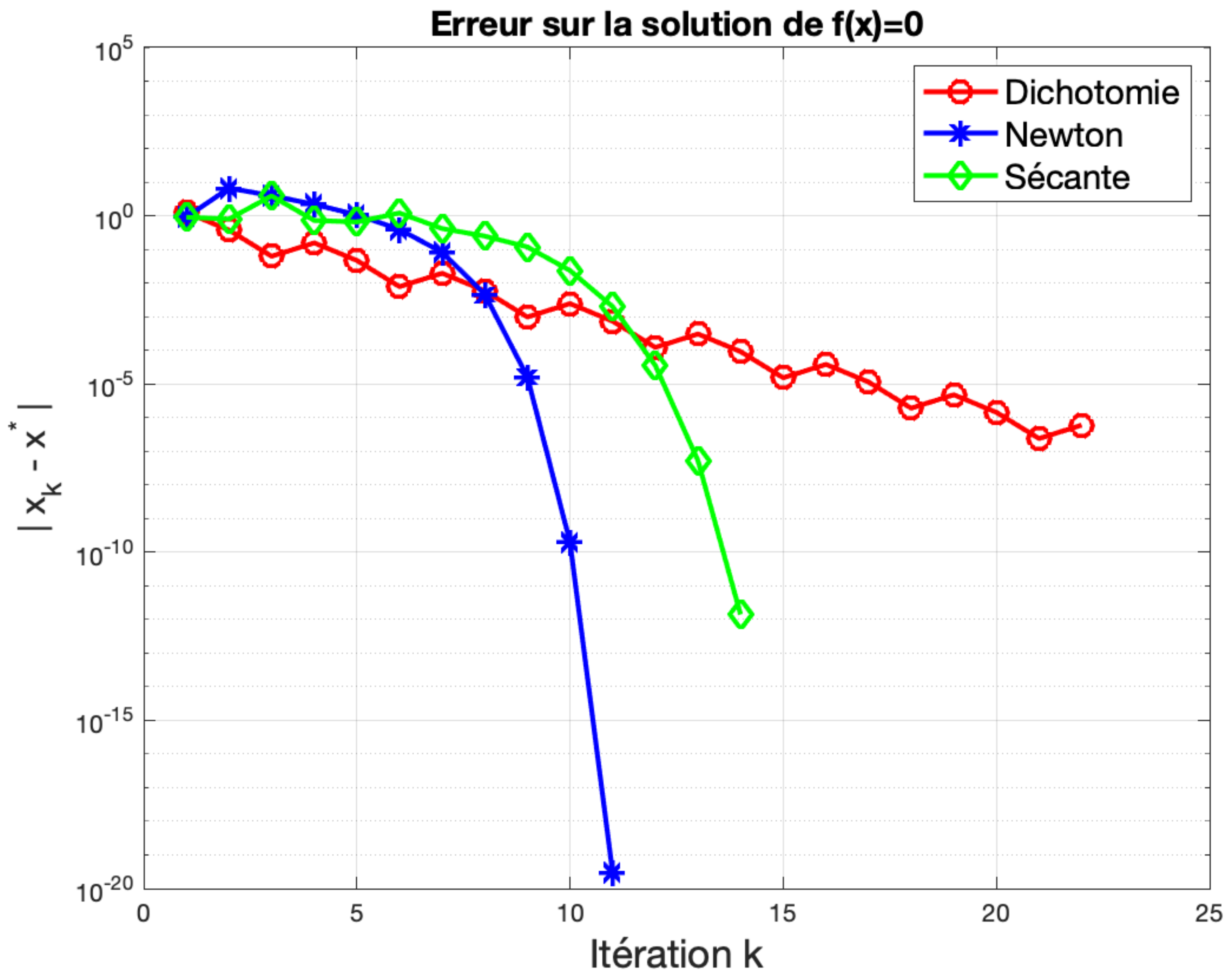
Exemple

$f(x) = x^3 - 9/2x^2 + 6x$
Racine $x^* = 0$

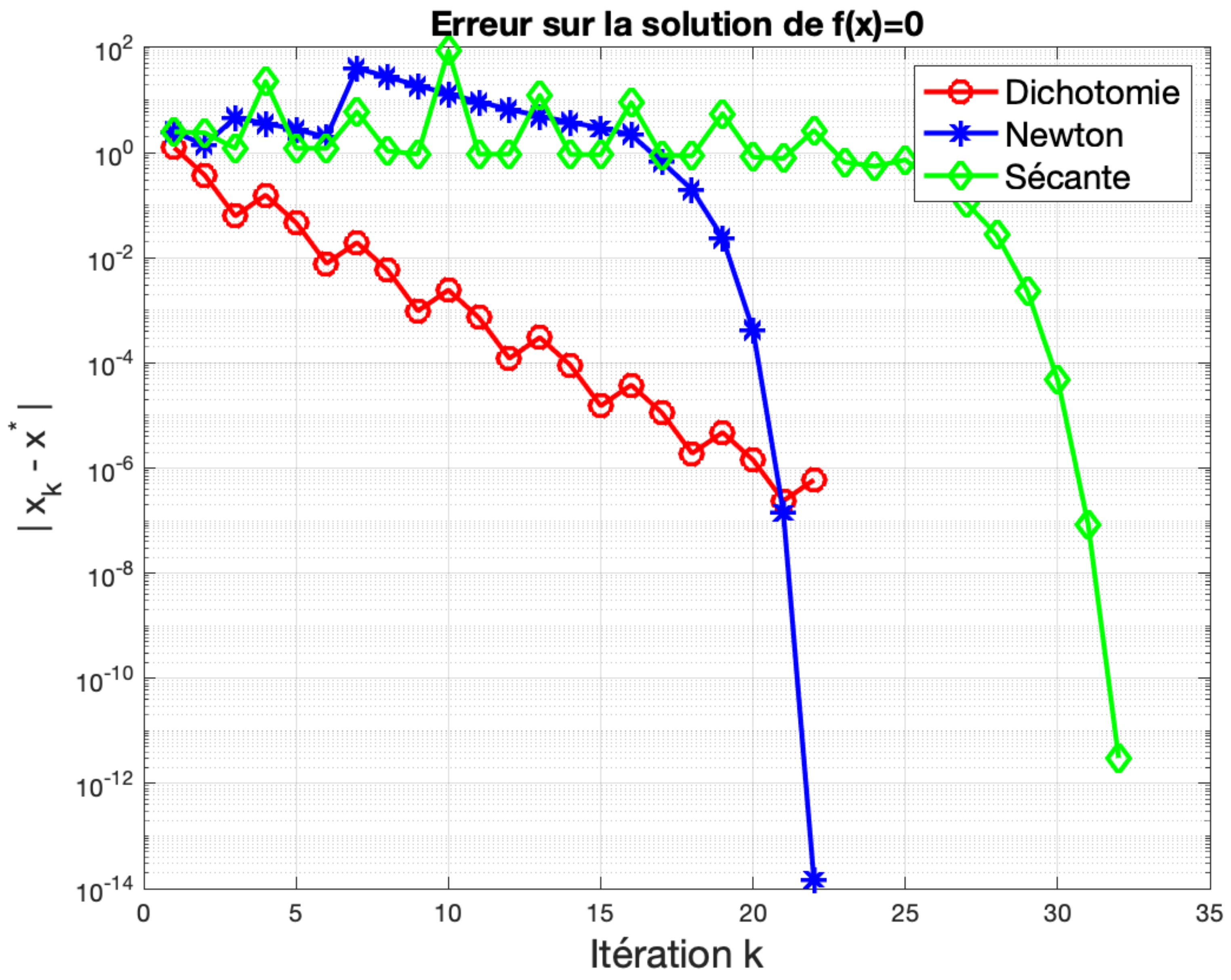


Programme equation_nonlineaire.m

Dichotomie : $[a_0, b_0] = [-0.5,3]$
Newton : $x_0 = 0.9$
Sécante: $x_0 = 0.9$ $x_1 = 0.8$



Dichotomie : $[a_0, b_0] = [-0.5,3]$
Newton : $x_0 = 2.5$
Sécante: $x_0 = 2.5$ $x_1 = 2.4$



En pratique

- Méthodes d'accélération de la convergence : relaxation, Aitken, Steffensen,…
- Méthodes spécifiques pour les polynômes, racines complexes : Bairstow, Müller
- Dimension supérieure à 1 : Newton-Raphson, calcul ou approximation du gradient
- Dans Matlab outils très puissants : roots, fzero, fsolve

$$f(x) = x^3 - 9/2x^2 + 6x$$

```
>> p=[1 -9/2 6 0];
>> roots(p)

ans =

   0.0000 + 0.0000i
   2.2500 + 0.9682i
   2.2500 - 0.9682i
```

$$f(x) = \sin(x) - \cos(x)$$

```
>> fzero(@(x) sin(x)-cos(x),1)

ans = 0.7854
>> ans-pi/4

ans = 1.1102e-16
```

```
>> fzero(@(x) sin(x)-cos(x),3)

ans = 3.9270
>> ans-5*pi/4

ans = 0
```

fsolve : résolution de systèmes d'équations non-linéaires

avec des méthodes numériques de pointe

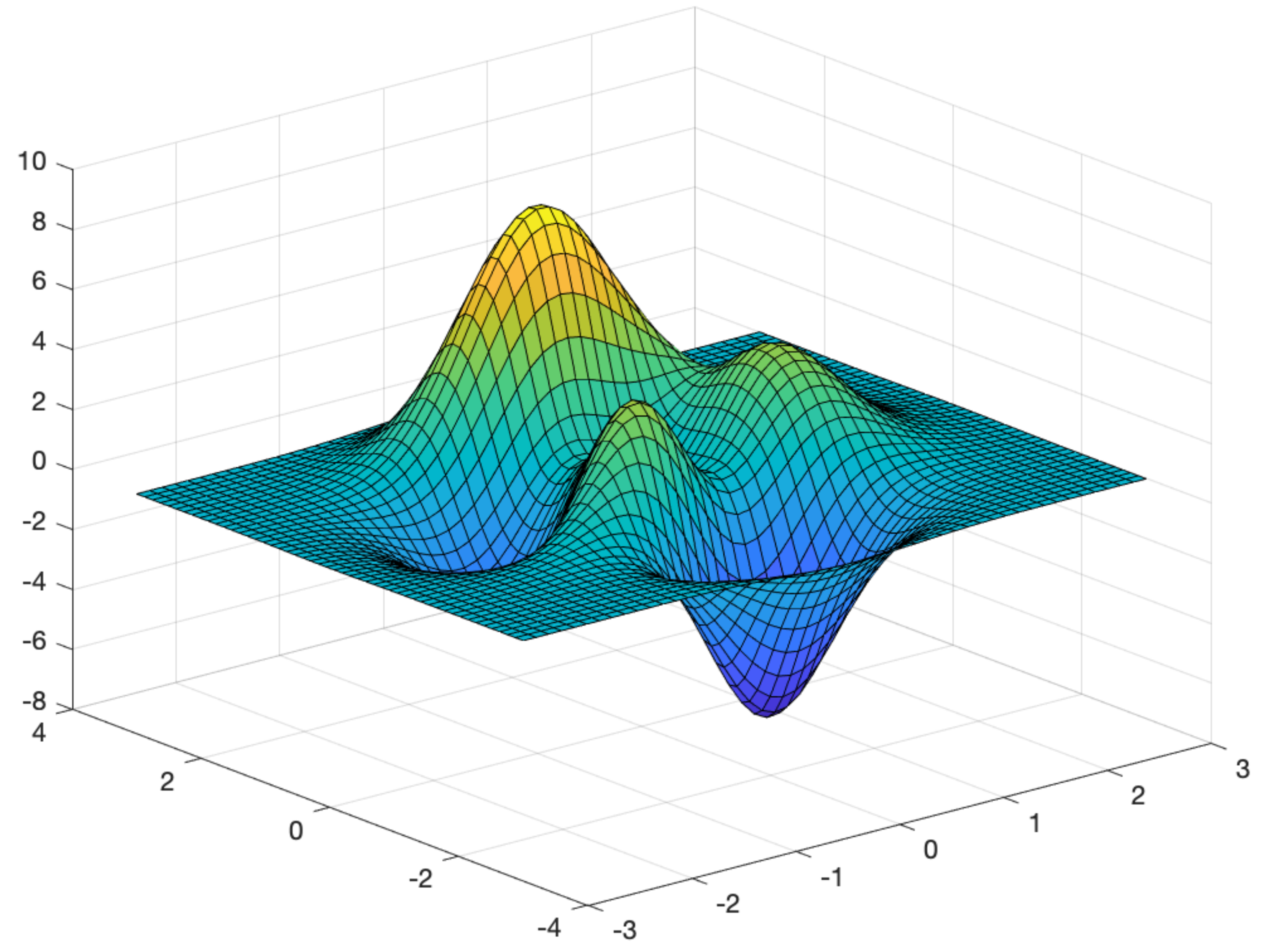
Optimisation numérique

Algorithmes de descente

Optimisation sans contrainte :

On souhaite calculer numériquement le/les points $x \in \mathbb{R}^n$ qui réalisent le minimum d'une fonction $f : \mathbb{R}^n \to \mathbb{R}$: $\min_{x \in \mathbb{R}^n} f(x)$

- Existence, unicité, etc : cours et TDs d'Analyse théorique
- Calcul des points critiques + conditions du second ordre
- Cas général: extrema locaux, globaux, points selles



Principe général : Générer une suite $(x_k)_{k \in \mathbb{N}}$ telle que $f(x_{k+1}) \leq f(x_k)$

Direction de descente

Le vecteur $d \in \mathbb{R}^n$ est une *direction de descente* pour f au point x si $t \to f(x + td)$ est décroissante en $t = 0$,

ie $\exists \eta > 0$ tel que $\forall t \in]0,\eta]$, $f(x + td) < f(x)$

Algorithmes de descente

Proposition

Soit $f : \mathbb{R}^n \to \mathbb{R}$ *différentiable* et $x \in \mathbb{R}^n$ tel que $\nabla f(x) \neq 0$.

- Si $\langle \nabla f, d \rangle < 0$ alors d est une direction de descente
- Si d est une direction de descente alors $\langle \nabla f, d \rangle \leq 0$

Démo

Remarques importantes

En choisissant $d = -\nabla f(x)$ on est sur d'avoir une direction de descente

C'est la direction de descente maximale pour les vecteurs de norme $||\nabla f||$

Démo

Algorithme du gradient

Algorithme général

- Choix de x_0
- Choix d'un critère d'arrêt

- Tant que le critère n'est pas satisfait:

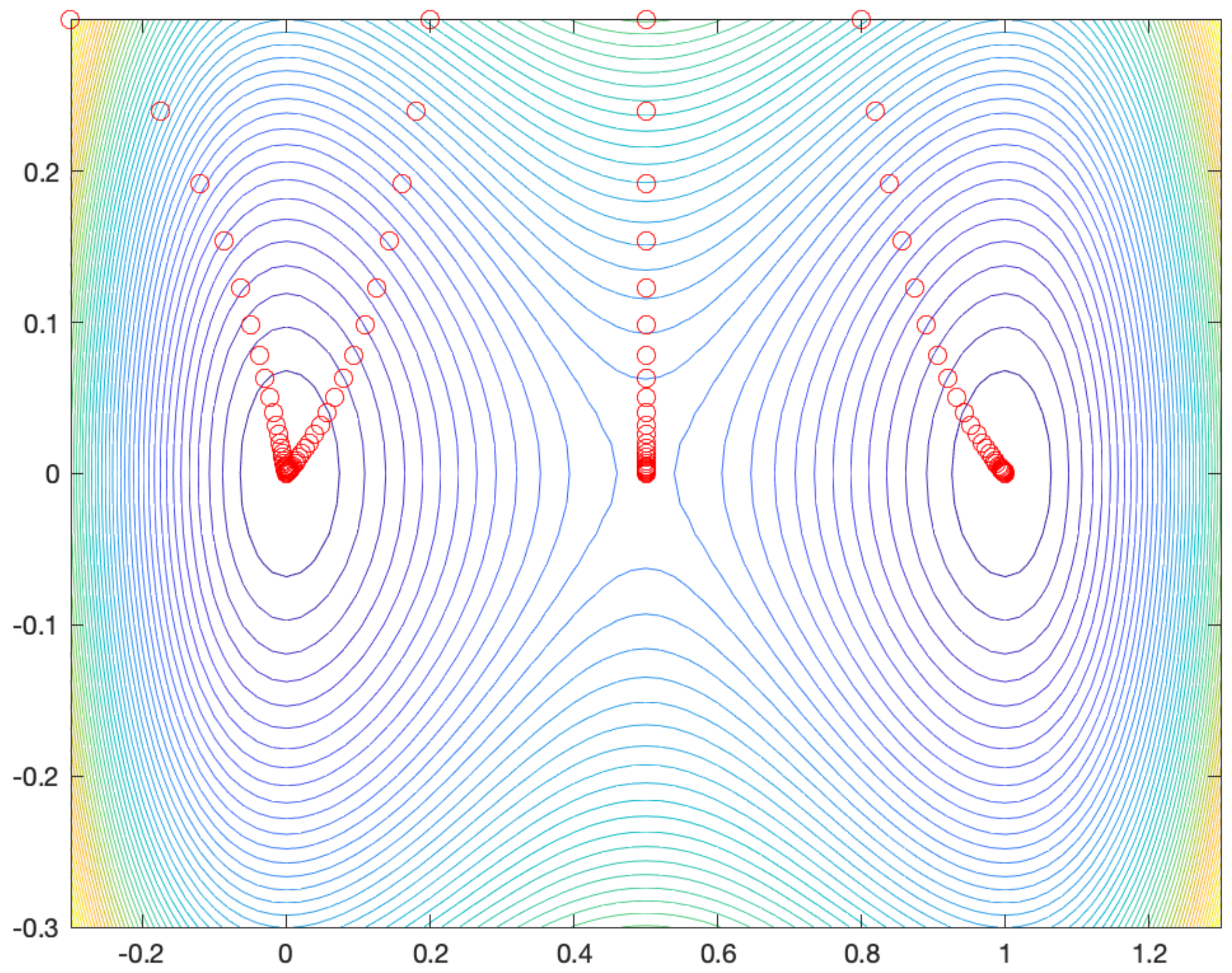
  Calcul de la direction de descente: $-\nabla f(x_k)$

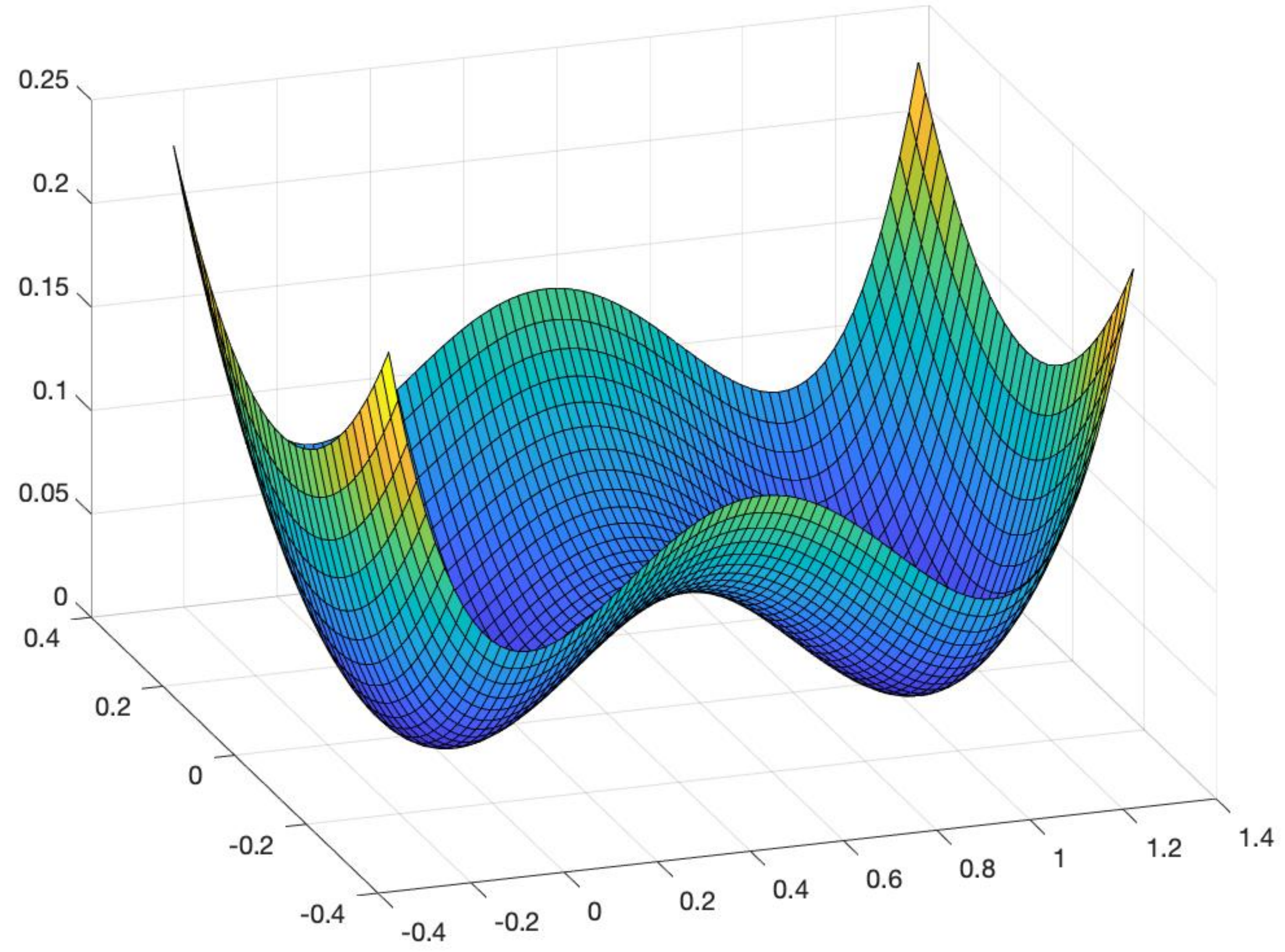
  Choix d'un pas dans cette direction : $s_k > 0$

$$x_{k+1} = x_k - s_k \ \nabla f(x_k)$$

Fin

Pas $s = 0.1$





Exemple

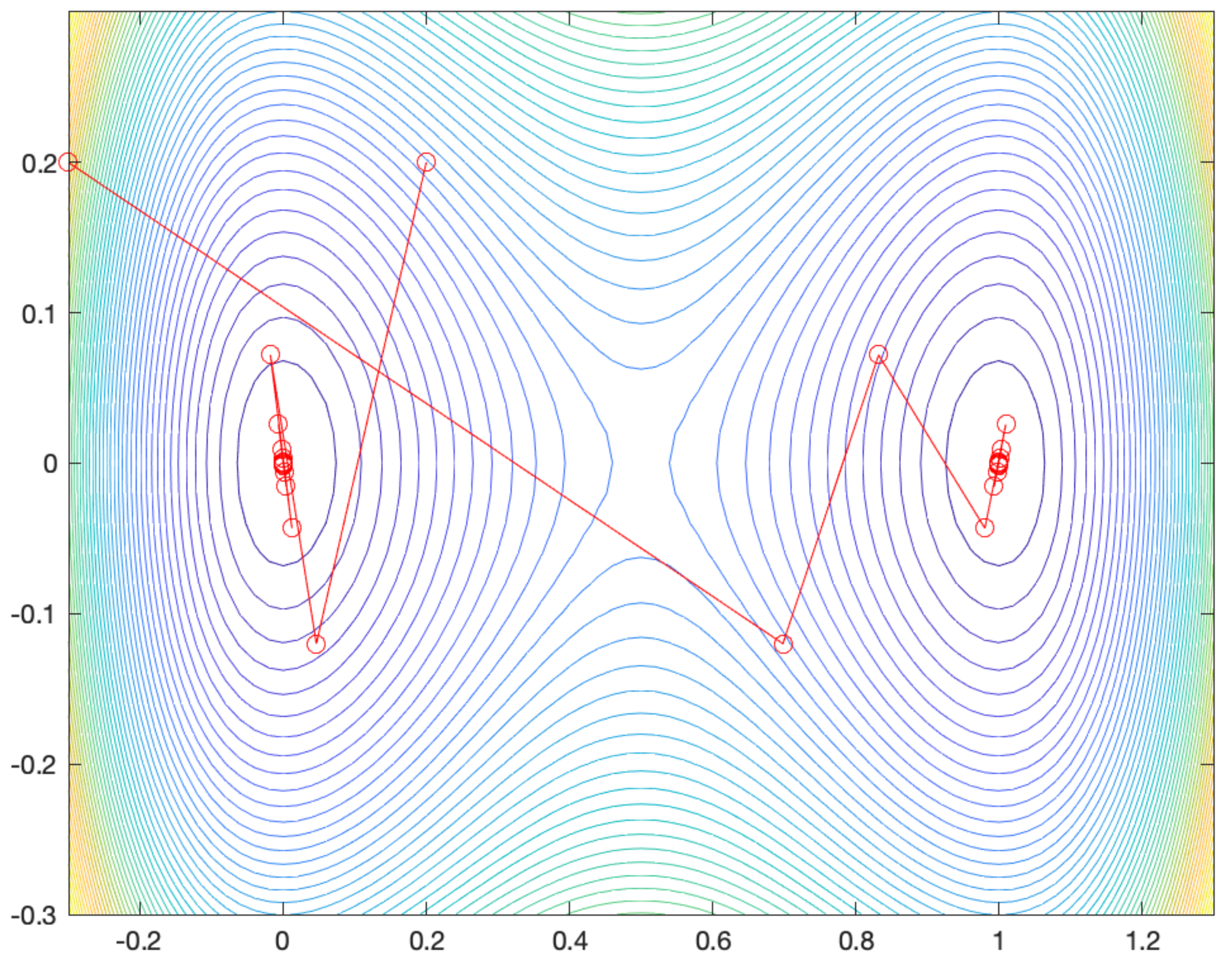
$f(x, y) = x^2(x-1)^2 + y^2$

2 minima (0,0) et (1,0), un col (0.5,0)

Gradient pas s constant

Programme methode_gradient.m

Pas $s = 0.8$



Remarques

- Choix x_0 et du pas s_k sont déterminants pour la convergence
- Pas constant : $s_k = s$ pas trop grand ni trop petit !
- Pas optimal $s_k = \min_{s>0} f(x_k + s d_k)$
- Critère d'arrêt : $\|\nabla f(x_k)\| \leq \epsilon$ couplé aux critères vu précédemment

En pratique

- Choisir x_0 le plus proche possible du minimum !
- Méthodes pour le choix du pas : Wolfe, Armijo,…
- La convergence dépend de la convexité de f
- Calcul du gradient pas toujours possible : approximation numérique, calcul partiel: gradient stochastique
- Méthodes d'ordre plus élevé : calcul de la Hessienne,…

- Optimisation avec contraintes : $\min_{x \in C} f(x)$ avec C un convexe fermé de $\mathbb{R}^n$

Gradient projeté: $x_{k+1} = p_C\left(x_k - s_k \nabla f(x_k)\right)$

Avec p_C la projection orthogonale sur C

- Sauf cas particuliers, p_C très couteux à calculer
- Très nombreux algorithmes existant :

  Uzawa, pénalisation, programmation linéaire, quadratique, algorithmes génétiques…

- Dans Matlab ToolBox Optimization très puissante : fminsearch, fmincon, …

Etude mathématique du problème indispensable -> algorithme sur mesure

```
>> f=@(x) x(1).^2.*(x(1)-1).^2 + x(2).^2;

>> fminsearch(f,[0.2,0.3])

    ans =   1.0e-04 *( -0.0173    0.3692)

>> fminsearch(f,[0.6,0.3])

    ans =   (1.0000. -0.0000)
```

Application aux systèmes linéaires

Problème : résoudre numériquement le système linéaire $Ax = b$

A matrice $N \times N$ inversible et x, b des vecteurs de $\mathbb{R}^N$

- N grand : on ne peut pas calculer x explicitement
- Nombreux résultats et méthodes existantes: cf poly 1A

    Méthodes directes : Gauss, LU, Cholevsky, etc..

    Méthodes itératives : Jacobi, Gauss-Seidel, Gradient, etc

Théorème

Si A est une matrice *Symétrique Définie Positive* alors :

Trouver $x \in \mathbb{R}^n$ tel que $Ax = b$ $\Leftrightarrow$ Trouver $\min\limits_{y \in \mathbb{R}^n} J(y)$ avec $J(y) = \frac{1}{2}\langle Ay, y\rangle - \langle b, y\rangle$

- On montre facilement que J est *strictement convexe* sur $\mathbb{R}^n$ et donc qu'il existe un unique $x^* \in \mathbb{R}^n$ qui minimize J
- On a $\nabla J(y) = Ay - b$

Méthode du gradient

Démo ∇J

On applique donc la méthode du gradient vue précédemment

Algorithme

- Choix de x_0
- Tant que $\|Ax_k - b\| > \epsilon$:

  Calcul de la direction de descente: $r_k = b - Ax_k$

  Choix d'un pas dans cette direction : $s_k > 0$

  $$x_{k+1} = x_k + s_k\,(b - Ax_k)$$

Fin

- Pas constant optimal : $s = \dfrac{2}{\lambda_{\min} + \lambda_{\max}}$
- Pas optimal : $s_k = \dfrac{\|r_k\|^2}{\langle Ar_k, r_k\rangle}$ et la méthode converge
- Amélioration: Gradient conjugué qui converge en au plus N itérations

Méthodes de point fixe pour les systèmes

Méthode générale

- Ecrire $Ax = b$ sous la forme $x = Bx + c$
- Itérer la suite : x_0 donné, $x_{k+1} = Bx_k + c$

Remarque

Point fixe : $x_{k+1} = g(x_k)$

$g : \mathbb{R}^n \to \mathbb{R}^n$ de différentielle $dg = B$

Théorème

La suite $x_{k+1} = Bx_k + c$ est convergente $\Leftrightarrow$ le rayon spectral de B vérifie $\rho(B) < 1$ $\Leftrightarrow$ $\|B\| < 1$ pour au moins une norme

Démo

Suite démo

Méthodes de point fixe pour les systèmes

Méthodes classiques : Jacobi, Gauss-Seidel, Relaxation

Exemple Jacobi

Si A est inversible et telle que $a_{ii} \neq 0 \ \forall i = 1,\ldots,N$ on choisit $B = Id - D^{-1}A$ et $c = D^{-1}b$

$$x_{k+1} = (Id - D^{-1}A)x_k + D^{-1}b$$

Démo Jacobi

- Convergence: dépend de $\rho(Id - D^{-1}A)$

Remarques

- Voir poly 1A pour les autres méthodes
- Méthode du gradient à pas fixe s:

$$B = Id - sA \ \text{ et } c = sb$$

En pratique

- Connaitre le plus possible de propriétés de la matrice A: taille, creuse, norme, valeurs propres, etc…
- Matlab : très nombreux solveurs disponibles : inv, ldivide, linsolve,…