# ECN 6338 Cours 5

## Résolution de systèmes d'équations non-linéaires

William McCausland

2022-02-10

# Les problèmes univarié et multivarié

- Problème univarié : trouvez $x \in \mathbb{R}$ qui vérifie
$$f(x) = 0,$$
où $f : \mathbb{R} \to \mathbb{R}$.
- Problème multivarié : trouvez $x \in \mathbb{R}^n$ qui vérifie
$$f(x) = 0_n,$$
où $f : \mathbb{R}^n \to \mathbb{R}^n$.
- Problème multivarié, élément par élément : trouvez $(x_1, \ldots, x_n)$ qui vérifie
$$\begin{aligned} f^1(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f^n(x_1, \ldots, x_n) &= 0 \end{aligned}$$

# La résolution de systèmes d'équations et l'optimisation

La solution $x^*$ au problème d'optimisation

$$\max_{x \in \mathbb{R}} f(x),$$

où $f\colon \mathbb{R} \to \mathbb{R}$ et $f \in C^2$, est aussi la solution du système

$$\frac{\partial f(x)}{\partial x^\top} = 0.$$

Cependant, la résolution du système $g(x) = 0$, $g \in C^1$, est plus générale :

- La matrice jacobienne de $g$ n'est pas forcément symmétrique
- La matrice jabobienne de $\nabla f$ est la matrice hessienne symmétrique de $g$.

# Systèmes non-linéaires et le nombre de solutions

Dans le cas spécial $f(x) = Ax - b = 0$, où $A$ est une matrice $n \times n$,

- si le rang de $A$ est de $n$, il y a une solution unique;
- si le rang de $A$ est moins grand, il n'y a aucune solution :

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

  ou il y a un nombre infini de solutions :

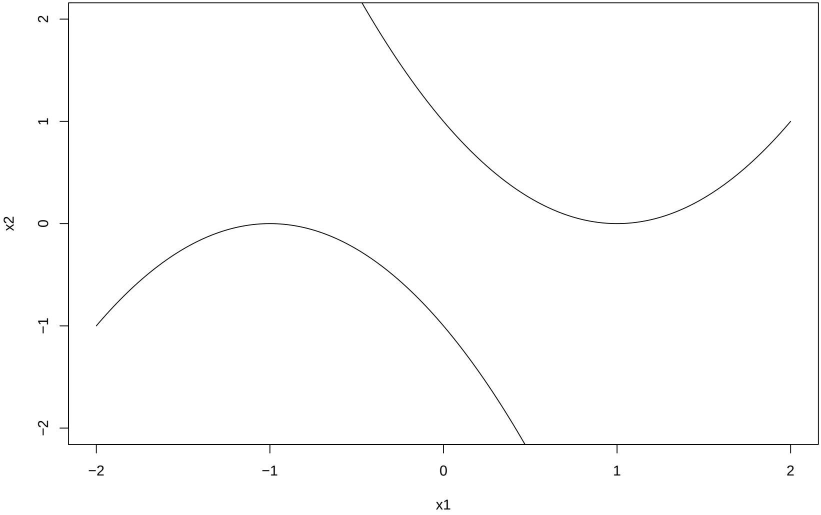
$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

Dans le cas général, il peut y avoir

- aucune solution, même pour les fonctions $f^i$ très différentes,
- un nombre fini arbitraire de solutions,
- un nombre infini de solutions.

# Exemple : absence d'une solution

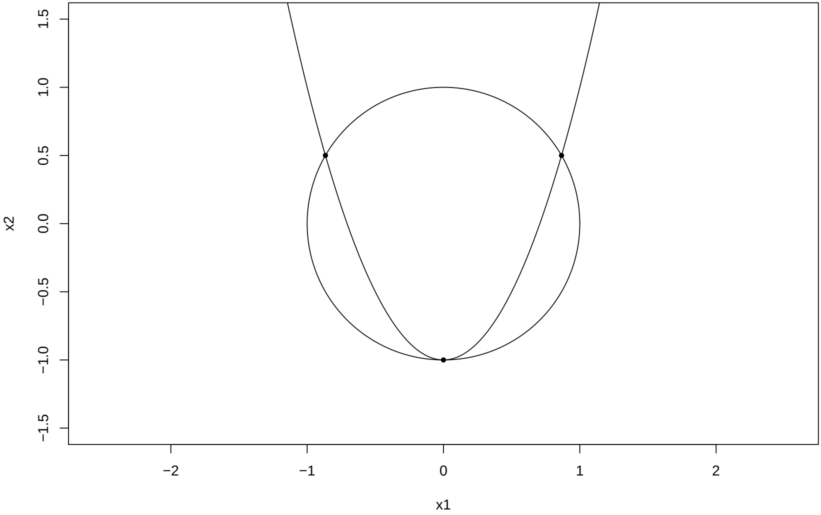
$$f^1(x_1, x_2) = x_2 - (x_1 + 1)^2, \quad f^2(x_1, x_2) = x_2 - (x_1 - 1)^2.$$



# Exemple : solutions multiples

Les racines de ce système sont $(0, -1)$, $(\pm\sqrt{3/4}, 1/2)$ :

$$f^1(x_1, x_2) = x_1^2 + x_2^2 - 1, \quad f^2(x_1, x_2) = 2x_1^2 - x_2 - 1.$$



# Jeu du duopole (Judd, pages 162-3)

- Dans un duopole, la firme 1 produit un bien en quantité $Y$, et la firme 2 produit un bien en quantité $Z$.
- Les coûts de production sont linéaires

$$c_Y(Y) = C_Y Y, \quad c_Z(Z) = C_Z Z,$$

  où $C_Y = 0.07$ et $C_Z = 0.08$.
- La demande est celle d'un consommateur avec utilité

$$U(Y,Z) = u(Y,Z) + M = (1 + Y^\alpha + Z^\alpha)^{\eta/\alpha} + M,$$

  où $\alpha = 0.999$, $\eta = 0.2$ et $M$ représente les dépenses en autres biens.
- La demande pour $Y$, $Z$ est donnée par les équations

$$p_Y = u_Y(Y,Z), \quad p_Z = u_Z(Y,Z),$$

  où $p_Y$ et $p_Z$ sont les prix de $Y$ et $Z$.

# Jeu du duopole (cont.)

- On cherche une équilibre de Nash $(Y^*, Z^*)$ où
  - $Y^*$ maximise le profit de la firme 1, pour $Z^*$ donnée,
  - $Z^*$ maximise le profit de la firme 2, pour $Y^*$ donnée.
- La meilleure réponse $Y$ à $Z$ la maximise le profit :

$$\begin{aligned}\Pi^Y(Y,Z) &= Y u_Y(Y,Z) - C_Y Y \\ &= \eta(1 + Y^\alpha + Z^\alpha)^{\eta/\alpha - 1} Y^\alpha - C_Y Y \\ &= \eta(1 + e^{\alpha y} + e^{\alpha z})^{\eta/\alpha - 1} e^{\alpha y} - C_Y e^y,\end{aligned}$$

  où $y = \log Y$, $z = \log Z$.
- Une condition de première ordre nécessaire pour la firme 1 :

$$\begin{aligned}\Pi_1^Y(Y,Z) &= \alpha\eta(\tfrac{\eta}{\alpha} - 1)(1 + e^{\alpha y} + e^{\alpha z})^{\eta/\alpha - 2} e^{2\alpha y} \\ &\quad + \alpha\eta(1 + e^{\alpha y} + e^{\alpha z})^{\eta/\alpha - 1} e^{\alpha y} - C_Y e^y = 0.\end{aligned}$$

- La même démarche pour la firme 2 donne une expression analogue $\Pi_2^Z(Y,Z) = 0$.

# Le problème de computation pour le jeu du monopole

L'équilibre du jeu du monopole est

$$(Y^*, Z^*) = (e^{x_1^*}, e^{x_2^*}),$$

où $x^* = (x_1^*, x_2^*)$ est la solution du système $f(x) = 0$, où

$$f^1(x_1, x_2) = \Pi_1^Y(e^{x_1}, e^{x_2}), \qquad f^2(x_1, x_2) = \Pi_2^Z(e^{x_1}, e^{x_2}).$$

# Illustration univariée I : méthode de Newton

Considérons la fonction $f$ et sa dérivée, définie sur l'intervalle $[0, 1]$ :

$$f(x) = (1-x)^3 - \log(1+x), \quad f'(x) = -3(1-x)^2 - (1+x)^{-1}.$$

Si on prend le point initial $x_0 = 0$, la droite de tangente est

$$g(x) = f(0) + f'(0)(x-0) = 1 - 4x,$$

et le point $x_1$ de l'itération de Newton est l'intersection de cette droite et l'axe des abscisses :

$$x_1 = x_0 - \frac{f'(0)}{f(0)} = 0 - \frac{1}{-3-1} = \frac{1}{4}.$$

Un pas de Newton de plus donne

$$x_2 = x_1 - \frac{f'(1/4)}{f(1/4)} \approx 0.329892,$$

très près de la racine unique.

# Illustration univariée II : échec de la méthode de Newton

- On peut commencer à $x_0 = 1$ où la courbature est plus prononcée.
- On évalue $f(x_0) = -\log 2$, $f'(x_0) = -\frac{1}{2}$ et on calcule

$$x_1 = x_0 - \frac{f'(x_0)}{f(x_0)} \approx -0.3862944,$$

  beaucoup plus loin de la racine et hors de l'intervalle $[0, 1]$.
- À voir aussi : "Pathological Examples", page 153 de Judd.

# Illustration univariée III : méthode d'intérpolation linéaire

- Note : fonction $f(x)$ en bleu, droites de tangente en rouge, droite de sécante en vert.
- Pour la première itération, où on calcule $x_1$, on n'a pas encore deux valeurs précédentes et on utilise la méthode de Newton.
- Une fois qu'on a $x_0$ et $x_1$, on peut construire la droite de sécante entre $(x_0, f(x_0))$ et $(x_1, f(x_1))$ :

$$h(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0)$$

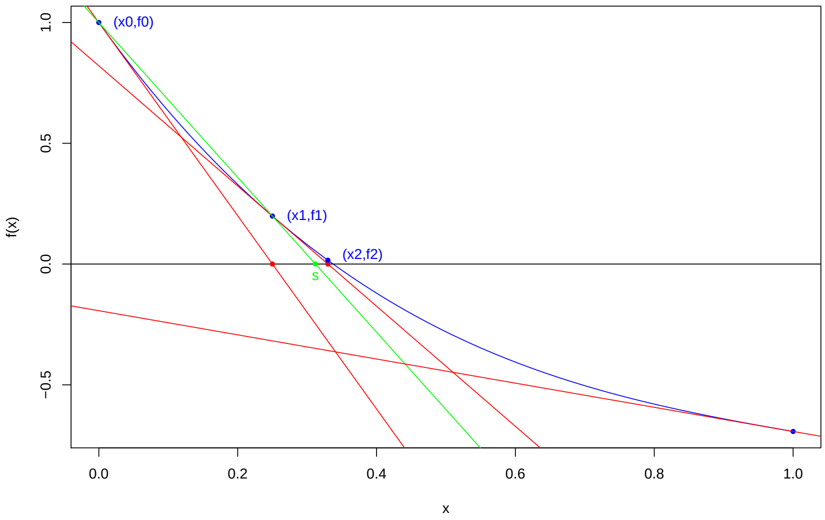
- Le point $s$ de l'itération par interpolation linéaire est l'intersection de cette droite et l'axe des abscisses :

$$s = x_0 + \frac{(x_1 - x_0)}{f(x_1) - f(x_0)} f(x_0) \approx 0.3120053,$$

un peu plus loin de la racine.

# Illustration (Newton et interpolation linéaire)

```
source('root_uni.R')
```



# Méthode de dichotomie

Intrants à l'itération $k+1$ : points $a_k$, $b_k$, valeurs $f(a_k)$, $f(b_k)$ tels que

1. $a_k < b_k$,
2. $f(a_k)f(b_k) < 0$. (signes opposés)

À l'itération $k+1$ :

1. Calculer $m = \frac{1}{2}(a_k + b_k)$.
2. Évaluer $f(m)$, si $f(m) = 0$, terminer avec la racine $m$.
3. Si $f(m)$ a la même signe que $a_k$,

$$a_{k+1} = m, \quad b_{k+1} = b_k.$$

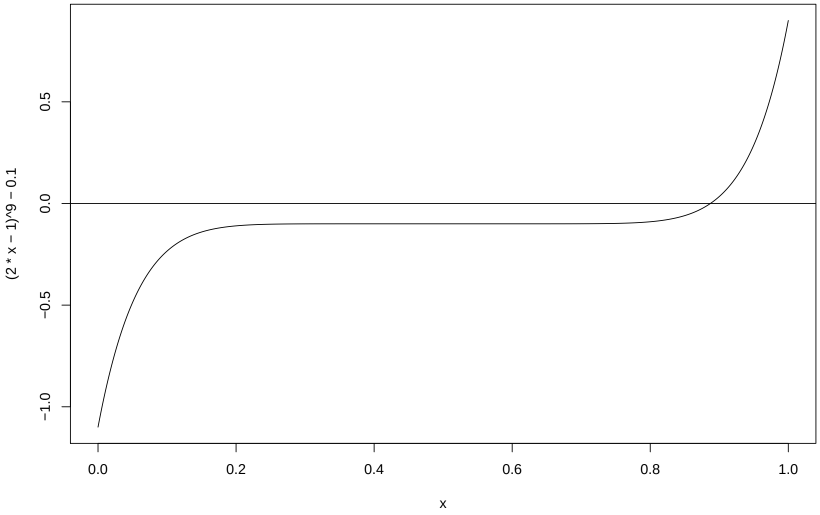
   sinon

$$a_{k+1} = a_k \quad b_{k+1} = m.$$

   Si $b_{k+1} - a_{k+1} < \delta$, terminer avec $\frac{1}{2}(a_{k+1} + b_{k+1})$.

# Quand la méthode de dichotomie marche relativement bien

```
x = seq(0, 1, by=0.0001)
plot(x, (2*x-1)^9 - 0.1, type='l')
abline(h=0)
```



# Discussion, méthode de dichotomie

- Pour la méthode de dichotomie,
  - on gagne 1 bit de précision à chaque itération (pas beaucoup, mais sûr),
  - on sait à l'avance combien d'itérations il faut pour atteindre ces deux conditions : $b_k - a_k < \delta$, $[a_k, b_k]$ contient une racine.
- Considérez des jeux zéro-somme entre
  - joueur 1 qui choisit une fonction continue $f(x)$ avec $f(a_0)f(b_0) < 0$, et veut maximiser le nombre d'itérations pour trouver un intervalle $[a, a+\delta]$ qui contient une racine.
  - joueur 2 qui choisit un algorithme pour trouver un intervalle $[a, a+\delta]$ contenant une racine.
- Conjecture : Si joueur 2 joue en premier, la méthode de dichotomie est optimale (minmax) pour joueur 2.
- Mais la méthode de dichotomie est très sous-optimale pour les fonctions habituelles.
- On veut accélérer la convergence et en même temps garantir un intervalle court en un nombre borné d'itérations.

# Méthodes du type Dekker-Brent

Intrants à l'itération $k+1$ : points $a_k$, $b_k$, $b_{k-1}$ ($b{-}1 = a_0$) et valeurs $f(a_k)$, $f(b_k)$ et $f(b_{k-1})$ tels que

1. $|f(a_k)| \leq |f(b_k)|$ (point $b_k$, contrepoint $a_k$)
2. $f(a_k)f(b_k) < 0$.

À l'itération $k+1$ :

1. Calculer $m = \frac{1}{2}(a_k + b_k)$.
2. Calculer $s$ comme fonction de $a_k$, $b_k$, $f(a_k)$, $f(b_k)$, $b_{k-1}$, $f(b_{k-1})$. (détails à venir)
3. Choisir entre $b_{k+1} = s$ et $b_{k+1} = m$. (détails à venir)
4. Évaluer $f(b_{k+1})$, si $f(b_{k+1}) = 0$, terminer avec $b_{k+1}$.
5. Choisir entre $a_{k+1} = a_k$ et $a_{k+1} = b_k$ tel que $f(a_{k+1})f(b_{k+1}) < 0$. (Condition 2.)
6. Si $|f(a_{k+1})| < f(b_{k+1})|$, échanger $a_{k+1}$ et $b_{k+1}$. (Condition 1.)
7. Si $|a_{k+1} - b_{k+1}| < \delta$, terminer avec $b_{k+1}$.

# Calculer $s$ (étape 2) par interpolation linéaire (droite sécante)

$$s = b_k - \frac{b_k - b_{k-1}}{f(b_k) - f(b_{k-1})} f(b_k)$$

Notes :

1. $s$ n'est pas une fonction de $a_k$.
2. Si on choisit $s$ par interpolation linéaire, une condition nécessaire pour choisir $b_{k+1} = s$ (étape 3) est que $s$ se trouve entre $m$ et $b_k$.

# Calculer $s$ (étape 2) par interpolation inverse quadratique

- Supposez que $f(a_k)$, $f(b_k)$ et $f(b_{k-1})$ sont distinctes.
- Voici une fonction quadratique $g(y)$ qui passe par les points $(f(a_k), a_k)$, $(f(b_k), b_k)$ et $(f(b_{k-1}), b_{k-1})$ :

$$
\begin{aligned}
g(y) = &\frac{(y - f(a_k))(y - f(b_k))}{(f(b_{k-1}) - f(a_k))(f(b_{k-1} - f(b_k))} b_{k-1} \\
&+ \frac{(y - f(a_k))(y - f(b_{k-1}))}{(f(b_k) - f(a_k))(f(b_k) - f(b_{k-1}))} b_k \\
&+ \frac{(y - f(b_{k-1}))(y - f(b_k))}{(f(a_k) - f(b_{k-1}))(f(a_k) - f(b_k))} a_k
\end{aligned}
$$

- La fonction inverse $x = f^{-1}(y)$ passe par les mêmes points.
- Défine $s = g(0)$, un zéro de la fonction $g^{-1}(x)$

# Calculer $s$ par interpolation inverse quadratique (cont.)

$$
\begin{aligned}
s = {} & \frac{f(a_k)f(b_k)}{(f(b_{k-1}) - f(a_k))(f(b_{k-1} - f(b_k))} b_{k-1} \\
& + \frac{f(a_k)f(b_{k-1})}{(f(b_k) - f(a_k))(f(b_k) - f(b_{k-1}))} b_k \\
& + \frac{f(b_{k-1})f(b_k)}{(f(a_k) - f(b_{k-1}))(f(a_k) - f(b_k))} a_k
\end{aligned}
$$

Notes :

1. Habituellement, c'est une amélioration, mais on peut toujours utiliser l'interpolation linéaire quand $k = 1$ où quand deux des valeurs $f(a_k)$, $f(b_k)$ et $f(b_{k-1})$ sont très près l'une à l'autre.
2. Si on choisit $s$ par interpolation inverse quadratique, une condition nécessaire habituelle pour choisir $b_{k+1} = s$ (étape 3) est que $s$ se trouve entre $\frac{3}{4}b_k + \frac{1}{4}a_k$ et $b_k$.

# Choisir entre $s$ et $m$ (étape 3)

- $b_{k+1} = m$ est plus sécure que $b_{k+1} = s$, mais le deuxième est habituellement meilleur.
- On ajoute aux conditions nécessaires déjà mentionnées pour choisir $s$ d'autres conditions :
  - Après un pas de bisection (pour $b_k$), on ajoute les conditions $|b_k - b_{k-1}| > \delta$ et $\frac{1}{2}|b_k - b_{k-1}| > |s - b_k|$.
  - Après un pas d'interpolation, on ajoute les conditions $|b_{k-1} - b_{k-2}| > \delta$ et $\frac{1}{2}|b_{k-1} - b_{k-2}| > |s - b_k|$.
- Avec ces conditions, le nombre maximale d'itérations est de $M^2$, où $M$ est le nombre d'itérations nécessaires pour la méthode de dichotomie.

# Méthode de Gauss-Seidel (exemple)

- Rappelons l'exemple avec trois racines :

$$f^1(x_1, x_2) = x_1^2 + x_2^2 - 1, \quad f^2(x_1, x_2) = 2x_1^2 - x_2 - 1.$$

- Résoudre $f_i(x_1, x_2) = 0$ pour $x_i$, $i = 1, 2$, donne une mise à jour possible de Seidel :

$$x_1^{k+1} = \pm\sqrt{1 - (x_2^k)^2}, \qquad x_2^{k+1} = 2(x_1^{k+1})^2 - 1$$

- La version linéaire de Seidel donne

$$x_1^{k+1} = x_1^k - \frac{f^1(x_1^k, x_2^k)}{f_1^1(x_1^k, x_2^k)} = x_1^k - \frac{(x_1^k)^2 + (x_2^k)^2 - 1}{2x_1^k},$$

et $x_2^{k+1}$ comme dans la version non-linéaire.

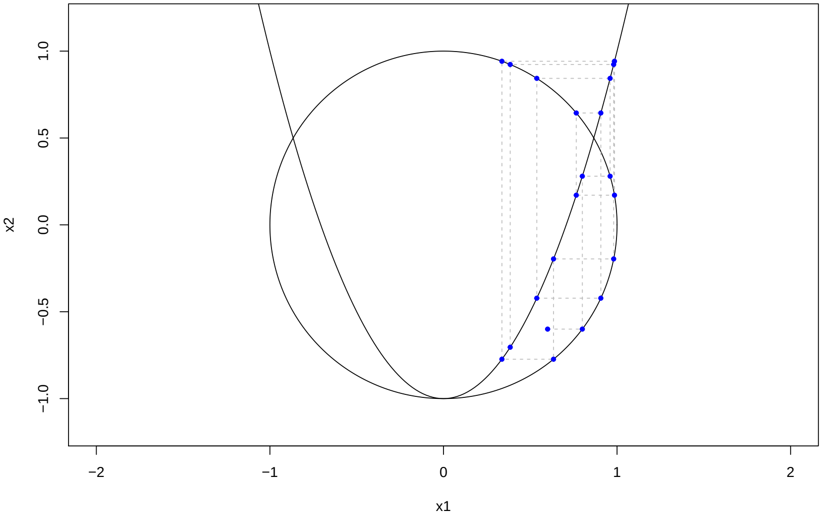
# La Méthode de Gauss-Seidel avec une permutation

- L'ordre des variables et l'ordre des équations importent.
- Résoudre $f_1(x_1, x_2)$ pour $x_2$ et $f_2(x_1, x_2)$ pour $x_1$ donne une autre mise à jour de Seidel :

$$x_1^{k+1} = \sqrt{\tfrac{1}{2}(1 + x_2^k)}, \quad x_2^{k+1} = \pm\sqrt{1 - (x_1^k)^2}.$$

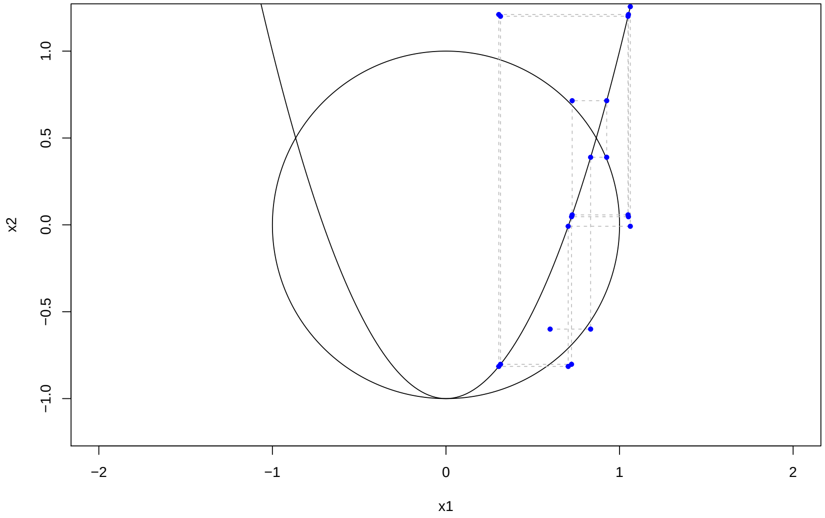
# Illustration Gauss-Seidel

```
x0 = c(0.6, -0.6); name = 'seidel'
source('cerc_parab.R')
```



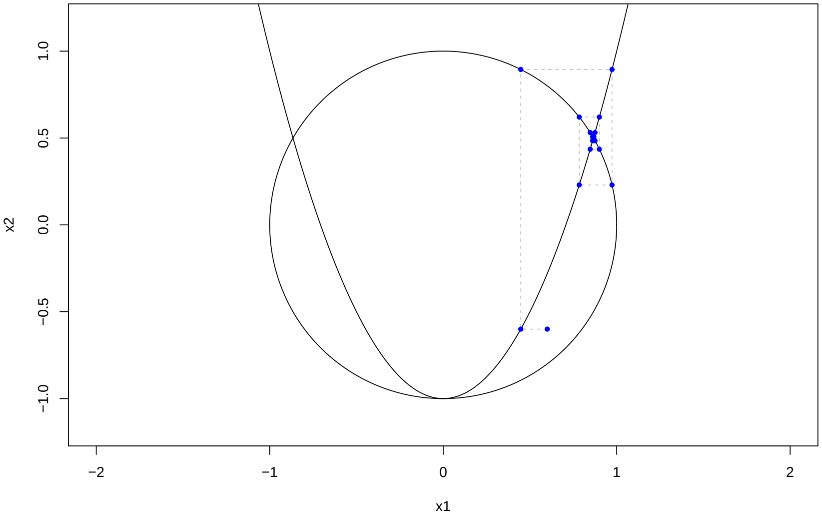
# Illustration Gauss-Seidel linéaire

```
x0 = c(0.6, -0.6); name = 'seidel_lin'
source('cerc_parab.R')
```



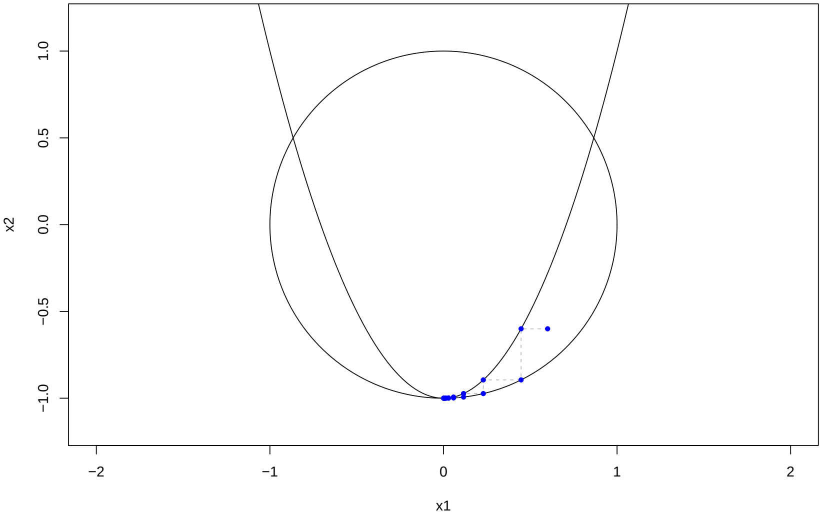
# Illustration Gauss-Seidel (permutation, $+, +$)

```
x0 = c(0.6, -0.6); name = 'perm_seidel'
source('cerc_parab.R')
```



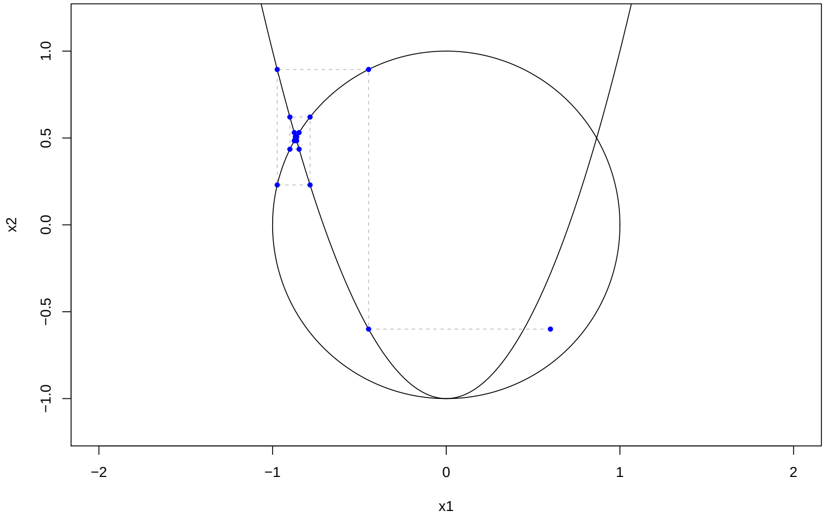
# Illustration Gauss-Seidel (permutation, $+, -$)

```
x0 = c(0.6, -0.6); name = 'perm_seidel+-'
source('cerc_parab.R')
```



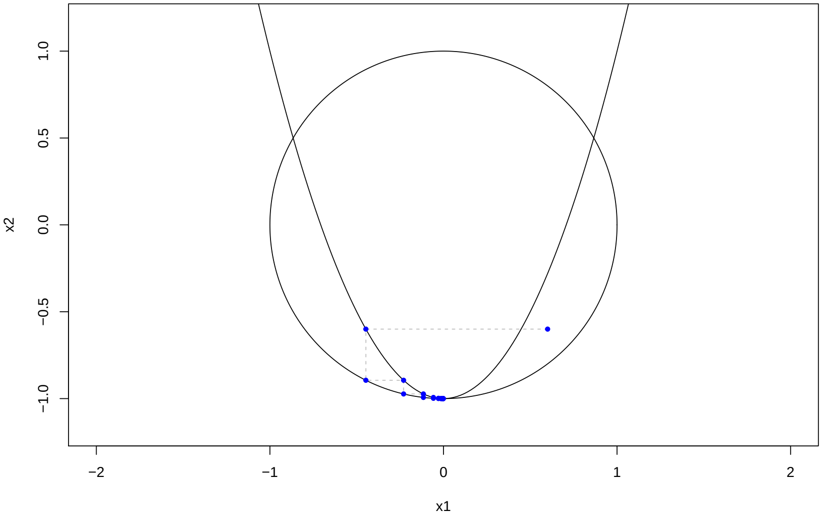
# Illustration Gauss-Seidel (permutation, $-, +$)

```
x0 = c(0.6, -0.6); name = 'perm_seidel-+'
source('cerc_parab.R')
```



# Illustration Gauss-Seidel (permutation, $-, -$)

```
x0 = c(0.6, -0.6); name = 'perm_seidel--'
source('cerc_parab.R')
```



# Méthode de Newton

- L'expansion linéaire de Taylor autour du point actuel $x^k$ est

$$g(x) = f(x^k) + J(x^k)(x - x^k).$$

- Si la matrice jacobienne est inversible, il y a un zéro de $g$ à

$$\tilde{x}^* = x^k - J(x^k)^{-1} f(x^k).$$

- La mise à jour de Newton sans modification est $x^{k+1} = \tilde{x}^*$ :

$$x^{k+1} = x^k - J(x^k)^{-1} f(x^k).$$

- La méthode converge rapidement (quadratiquement) d'un point local, mais elle n'est pas forcément globalement convergente.

# Méthode de Broyden

- Comme la méthode BFGS (B pour Broyden) pour l'optimisation multivariée, la méthode de Broyden pour résoudre les systèmes non-linéaires multivariés utilise une approximation $A_k$ de la matrice jacobienne $J(x^k)$ pour donner le pas de Broyden :

$$s^k = -A_k^{-1} f(x^k), \quad x^{k+1} = x^k + s^k.$$

- La mise à jour de $A_k$ est de rang 1 :

$$A_{k+1} = A_k + \frac{(y_k - A_k s^k)(s^k)^\top}{(s^k)^\top s_k}$$

- Cela permet la mise à jour simultanée de $A_k^{-1}$ à $A_{k+1}^{-1}$ en $O(n^2)$ opérations avec le formule Sherman-Morrison.

# Illustration, méthodes de Newton et Broyden

- Code pour la fonction $f(x_1, x_2) = (x_1^2 + x_2^2 - 1, 2x_1^2 - x_2 - 1)$

```
library(pracma)

# La fonction
f <- function(x) {
  f1 <- x[1]^2 + x[2]^2 - 1
  f2 <- 2*x[1]^2 - x[2] - 1
  c(f1, f2)
}

# Sa matrice jacobienne
J <- function(x) {
  J <- matrix(0, nrow=2, ncol=2)
  J[1, 1] <- 2*x[1]; J[1, 2] <- 2*x[2];
  J[2, 1] <- 4*x[1]; J[2, 2] <- -1;
  J
}
```

# Illustration, méthode de Newton

```r
x0 <- c(0.6, -0.6)
newtonsys(f, x0, Jfun=J)
```

```
## $zero
## [1]  7.573773e-09 -1.000000e+00
##
## $fnorm
## [1] 2.220446e-16
##
## $niter
## [1] 25
```

# Illustration, méthode de Broyden

```
x0 <- c(0.6, -0.6)
broyden(f, x0, J0 = J(x0))
```

```
## $zero
## [1]  7.032333e-05 -1.000000e+00
##
## $fnorm
## [1] 1.043668e-08
##
## $niter
## [1] 20
```