Résolution de système d'équations non linéaires avec R

Ousseynou GUEYE, Marie Agathe SECK, Dieynaba KA

2023-07-01

Présentation des packages et fonctions

Le package rootSolve

Le package "rootSolve" est une bibliothèque de fonctions disponible dans le logiciel R qui fournit des outils pour la résolution numérique de systèmes d'équations non linéaires.

rootSolve::multiroot()

Cette fonction est utilisée pour résoudre des systèmes d'équations non linéaires multivariées. Elle prend en entrée un vecteur de fonctions d'équation multivariée, ainsi qu'un vecteur initial de valeurs approchées pour les variables, et retourne une approximation des valeurs des variables qui satisfont le système d'équations. l'algorithme a été implémenté avec la méthode de Newton-Raphson.

```
library(rootSolve)
# Définition du système d'équations
model <- function(x){ c(F1 = x[1]^2+ x[2]^2 -1,
                        F2 = x[1]^2- x[2]^2 +0.5)
  }
solution <- multiroot(f = model, start = c(1, 1))

# Affichage des solutions
print(solution$root)
```

```
## [1] 0.5000000 0.8660254
```

```
print(solution$f.root)
```

```
##           F1           F2
## 2.323138e-08 2.323308e-08
```

```
print(solution$iter)
```

```
## [1] 5
```

Le package nleqslv

Le package "nleqslv" est une bibliothèque de fonctions disponible dans le logiciel R qui permet de résoudre des systèmes d'équations non linéaires multivariées. Il offre des fonctionnalités avancées pour trouver les solutions numériques de ces systèmes en utilisant des méthodes itératives basées sur la méthode de Newton et de broyden(méthode par défaut).

nleqslv::nleqslv()

Cette fonction est le cœur du package. Elle permet de résoudre numériquement des systèmes d'équations non linéaires multivariées en utilisant les méthode de Newton et de Broyden . Elle prend en entrée une fonction d'équation multivariée, un vecteur initial de valeurs approchées pour les variables, et retourne une approximation des valeurs des variables qui satisfont le système d'équations.

Exemple:

```
library(nleqslv)
# Définition du système d'équations
model <- function(x){ c(F1 = x[1]^2+ x[2]^2 -1,
                        F2 = x[1]^2- x[2]^2 +0.5)
  }
solution <- nleqslv(c(1, 1), model,method="Newton" )
```

```r
# Affichage des solutions
print(solution$x)
```

```
## [1] 0.5000000 0.8660254
```

```r
print(solution$fvec)
```

```
## [1] 8.881784e-16 1.110223e-15
```

```r
print(solution$iter)
```

```
## [1] 5
```

Le package pracma

Le package "pracma" aussi propose des méthodes numériques pour résoudre des équations non linéaires. Par exemple, les fonctions **`fsolve()`** et **`broyden()`** en utilisant la méthode de Newton ou la méthode de Broyden, respectivement.

Les méthodes indirectes

On peut aussi résoudre un système d'équations non linéaires à l'aide d'un problème d'optimisation . Cela peut se faire on Transforme le système d'équations en un problème d'optimisation en définissant une fonction objectif à minimiser ou à maximiser. Cette fonction objectif est généralement construite en utilisant une mesure de l'écart entre les valeurs réelles des équations et les valeurs calculées à partir des variables inconnues.

On Choisit ensuite une méthode d'optimisation : Il existe plusieurs méthodes d'optimisation disponibles pour résoudre le problème. Certaines des méthodes couramment utilisées incluent la méthode de Newton, la méthode du gradient, la méthode de recherche linéaire, la méthode de Gauss-Newton, etc.

Sous R, propose pour cela les fonctions comme **optim** () dans le package **stats , optimx()** dans le package **optimx.**

```r
# Chargement du package stats
library(stats)

# Définition des équations
model <- function(x){ c(F1 = x[1]^2+ x[2]^2 -1,
                        F2 = x[1]^2- x[2]^2 +0.5)
  }


# Fonction objectif (somme des carrés des équations)
objective <- function(x) {
  eqs <- (x[1]^2 + x[2]^2 - 1)^2 + (x[1]^2 + x[2]^2 - 0.5)^2
  return(eqs)
}

# Variables initiales
initial_vars <- c(1, 1)

# Résolution du problème d'optimisation
solution <- optim(initial_vars, objective, method = "Nelder-Mead")
                  #control = list(maxit = 1000) )

# Affichage des résultats
print(solution$par)
```

```
## [1] 0.5866119 0.6370758
```

```
print(solution$value)
```

```
## [1] 0.125
```

Cas Pratique

Dans cette partie (voir Script R) du travail nous résolvons le système d'équation non linéaire suivant :

(a*w1/e1)/(a*w1/e1 + b*w2/e2 + c*w3/e3) = 2/5

(b*w2/e2)/(a*w1/e1 + b*w2/e2 + c*w3/e3) = 2/5

a + b + c = Yf