Εφαρμογή της μεθόδου του Newton στην έμμεση μέθοδο του Euler

Διατύπωση του προβλήματος

Αναζητάμε συνάρτηση $y : [0,5] \to \mathbb{R}$, η οποία να είναι λύση του προβλήματος αρχικών τιμών

$$y'(t) = e^{-y(t)}, \qquad t \in [0,5], \tag{1}$$
$$y(0) = 1. \tag{2}$$

Η ακριβής λύση του (1)-(2) είναι $y(t) = \ln(t+e)\,,\ t \in [0,5]$.

Έμμεση μέθοδος του Euler

Δοσμένου ενός φυσικού αριθμού N, θεωρούμε την ομοιόμορφη διαμέριση του $[a,b]$ με πλάτος $h = \frac{5}{N}$ με κόμβους τα σημεία $t^n = 0 + nh$, για $n = 0,\dots,N$. Η έμμεση μέθοδος του Euler υπολογίζει την προσέγγιση $Y^n \in \mathbb{R}$ της $y(t^n)$ από

$$Y^{n+1} = Y^n + h\,e^{-Y^{n+1}}, \quad n = 0,\dots,N-1, \tag{3}$$
$$Y^0 = 1. \tag{4}$$

Η Y^{n+1} είναι λύση μιας μη-γραμμικής εξίσωσης. Επομένως, μια ιδέα είναι να εφαρμόσουμε την μέθοδο του Newton για την προσέγγιση του Y^{n+1}.

Η μέθοδος του Newton

Η μέθοδος του Newton παράγει προσεγγίσεις μιας ρίζας μιας πραγματικής συνάρτησης $g(x)$, $x \in \mathbb{R}$, για την οποία είναι γνωστή η παράγωγος $g'(x)$. Οι προσεγγίσεις της μεθόδου του Newton, παράγονται από

$$\begin{aligned} x^{(m+1)} &= x^{(m)} - \frac{g(x^{(m)})}{g'(x^{(m)})}, \quad m \geq 0, \\ x^{(0)} &= x_0, \end{aligned} \tag{5}$$

με $x_0 \in \mathbb{R}$ μια αρχική προσέγγιση της ρίζας που θέλουμε να προσεγγίσουμε, για την οποία $g'(x_0) \neq 0$.

Εφαρμογή της μεθόδου του Newton για την μη-γραμμική εξίσωση (3)-(4)

Θέλουμε να προσεγγίσουμε την λύση της εξίσωσης (3). Αντικαθιστούμε την τιμή Y^{n+1} με x και μεταφέρουμε όλους τους όρους στο αριστερό μέλος. Τότε,

$$x - Y^n - h\,e^{-x} = 0. \tag{6}$$

Συνεπώς, $g(x) := x - Y^n - h\,e^{-x},\ x \in \mathbb{R}$. Για αρχική συνθήκη στην (5) επιλέγουμε $x_0 = Y^n$, όπου Y^n η προσέγγιση της ακριβής λύσης στον κόμβο t^n. Σε κάθε βήμα της μεθόδου του Euler, η Y^{n+1} προσεγγίζεται κάνοντας M επαναλήψεις με την μέθοδο του Newton σύμφωνα με την (5) και την g όπως ορίζεται παραπάνω. Στο τέλος, ορίζουμε $Y^{n+1} := x^M$. Επιγραμματικά,

Για $n = 0,\dots,N-1$,
Βήμα 1: Θέτουμε $x_0 := Y^n$
Βήμα 2: Για $m = 0,\dots,M-1$ υπολογίζουμε διαδοχικά

$$\begin{aligned} x^{(m+1)} &= x^{(m)} - \frac{g(x^{(m)})}{g'(x^{(m)})} \\ x^{(0)} &= Y^n, \end{aligned}$$

Βήμα 3: Θέτουμε $Y^{n+1} := x^{(M)}$.

Υλοποίηση στην Python

```
import numpy as np

# Define the right hand side of the initial value problem
def f(t, y) :
  z = np.exp(-y)
  return z

# Define the exact solution of the initial value problem
def exact(t) :
  z = np.log(t + np.exp(1))
  return z

# Define the function g of the Newton iteration
def g(t, yold, x, h) :
  z = x - yold - h*f(t,x)
  return z

# Define the derivative of the function of the Newton iteration
def dg(t, yold, x, h) :
  z = 1 - h*(-np.exp(-x))
  return z


# Main program
def myeuler(a, b, y0, N, M) :
  h = (b-a)/N                       # Time step
  t = a                             # Start from the left endpoint
  yold = y0                         # Define the Y^n as yold and set at the begining as y0
  err = 0                           # Define the error variable at zero
  for n in range(N) :               # Loop over all nodes
    xold = yold
    for m in range(M) :             # At each time step, approximate Y^{n+1} by the Neuton method
      xnew = xold - g(t+h, yold, xold, h)/dg(t+h, yold, xold, h)
      xold = xnew
    ynew = xnew                     # Define the Y^{n+1} as the last approximation of Newton method
    t = t + h                       # Update time
    yold = ynew                     # Update the old approximation
    error = abs(ynew - exact(t))    # Calculate the error
    if error > err :                # Check if the current error is greater than the previous error
      err = error
  return err

# Define the parameters of the current problem and print the convergence rate for a given vector N

a = 0.0
b = 5.0
t0 = a
y0 = exact(t0)
N = np.array([20,40])
M = 3                               # Stopping criterion for Newton method. Maximun number of
    iterations
m = len(N)
error = np.zeros([m])
p = np.zeros([m])
for i in range(m) :
  err = myeuler(a, b, y0, N[i], M)
  error[i] = err
  if i > 0 :
    p[i] = (np.log(error[i-1]/error[i])/(np.log((float(b)/N[i-1])/(float(b)/N[i]))));
print(error)
print(p)
```