(MAT-2910) Analyse numérique - Devoir 2

Dan Lévy – 536 762 873

Hiver 2025

Introduction

Ce rapport présente l'application de diverses méthodes numériques pour l'approximation des racines de la fonction

$$f(x) = (x+1)(x-1)^2.$$

Les racines exactes de cette fonction sont $r_1 = -1$ et $r_2 = 1$.

1. Méthode du point fixe

Nous utilisons la fonction :

$$g(x) = x - \frac{f(x)}{5}$$

a) Analyse théorique

La méthode du point fixe converge si $|g'(x)| < 1$ au voisinage de la racine.

$$g'(x) = 1 - \frac{f'(x)}{5}$$

Avec :

$$f'(x) = \frac{\delta(x^3 - x^2 - x + 1)}{\delta x} = 3x^2 - 2x - 1$$

Pour $r_1 = -1$:

$$g'(-1) = 1 - \frac{f'(-1)}{5} = 1 - \frac{4}{5} = \frac{1}{5} = 0.2$$

Pour $r_2 = 1$:

$$g'(1) = 1 - \frac{f'(1)}{5} = 1 - \frac{0}{5} = 1$$

On voit donc que la méthode du point fixe converge pour la racine $r_1 = -1$ puisque $|g'(-1)| = 0.2 < 1$. On voit également que pour la racine $r_2 = 1$, la méthode converge mais bien plus lentement (à un taux de $|g(1)| = 1$).

b) Figures 1 et 2

Voir les figures 1 et 2 en annexe, montrant l'erreur $E_n = |x_n - r|$ et le rapport $\frac{E_{n+1}}{E_n}$.

c) Commentaire

Les figures confirment que la méthode converge pour r_1 assez rapidement, on obtient une tolérance $\approx 2*10^{-8}$ pour un $n = 11$. Les figures montrent également que la convergence pour r_2 est extrêmement lente (tolérance $\approx 4*10^{-2}$ pour un $n = 50$).

2. Méthode de Newton

a) Analyse théorique

La méthode de Newton est quadratique pour une racine simple et linéaire pour une racine double.

$$g_{Newton}(x) = x - \frac{f(x)}{f'(x)}$$

Pour $r_1 = -1$ (racine simple) : Convergence quadratique.
Pour $r_2 = 1$ (racine double) : Convergence linéaire, taux proche de 0.5.

b) Figures 3, 4 et 5

Voir les figures 3, 4 et 5 en annexe.

c) Commentaire

Les figures montrent une convergence quadratique pour r_1, tandis que la convergence est linéaire pour r_2 à un taux de 0.5.

3. Méthode de Steffenson

a) Figures 6 et 7

Voir les figures 6 et 7 en annexe.

b) Commentaire

La méthode de Steffenson a une convergence quadratique pour r_1. Cela est conforme aux propriétés connues puisque c'est un ordre de plus que l'orde de convergence de la méthode du point fixe pour r_1.

4. Méthode de la sécante

a) Implementation dans secante.py

Fonction de la sécante implémentée à partir de pointfixe.py (voir annexe).

b) Figures 8 et 9

Voir les figures 8 et 9 en annexe.

c) Commentaire

La méthode de la sécante a un ordre de convergence égal au nombre d'or $\alpha \approx 1.618$, ce qui est visible dans la figure 9.

Annexe - Figures

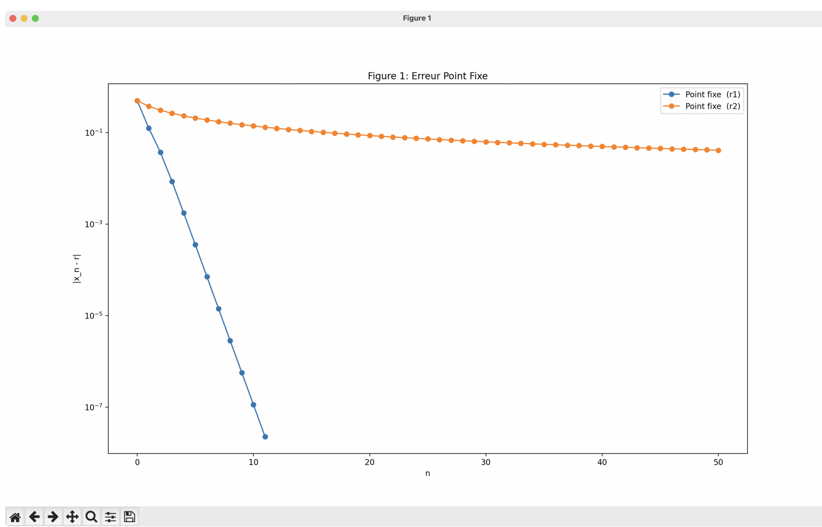


Figure 1: Erreur en fonction des itérations - Méthode du point fixe pour r_1 et r_2

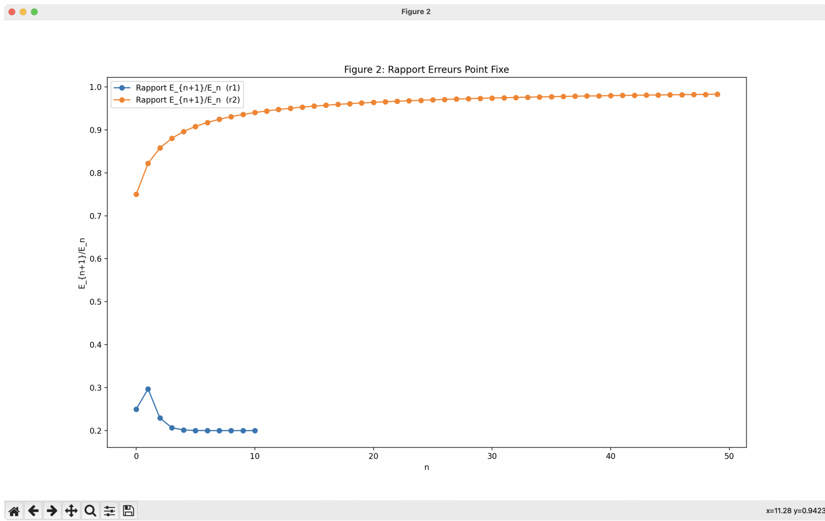


Figure 2: Rapport des erreurs successives - Méthode du point fixe pour r_1 et r_2

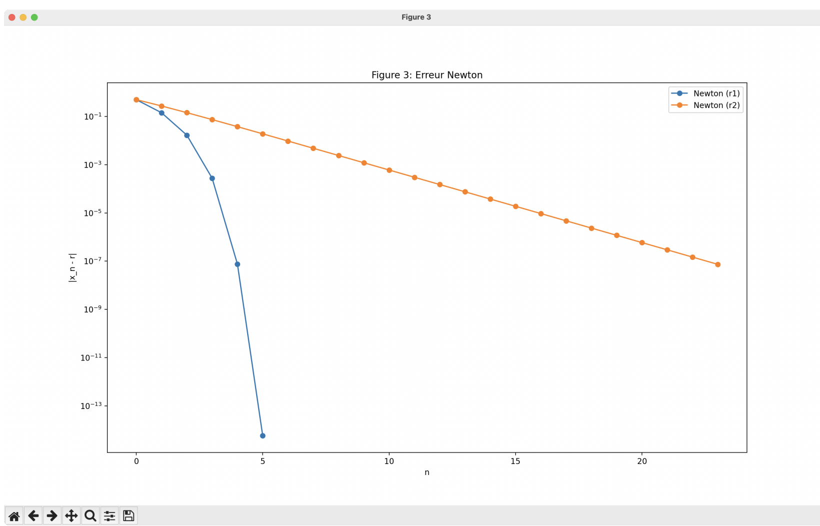


Figure 3: Erreur en fonction des itérations - Méthode de Newton pour r_1 et r_2

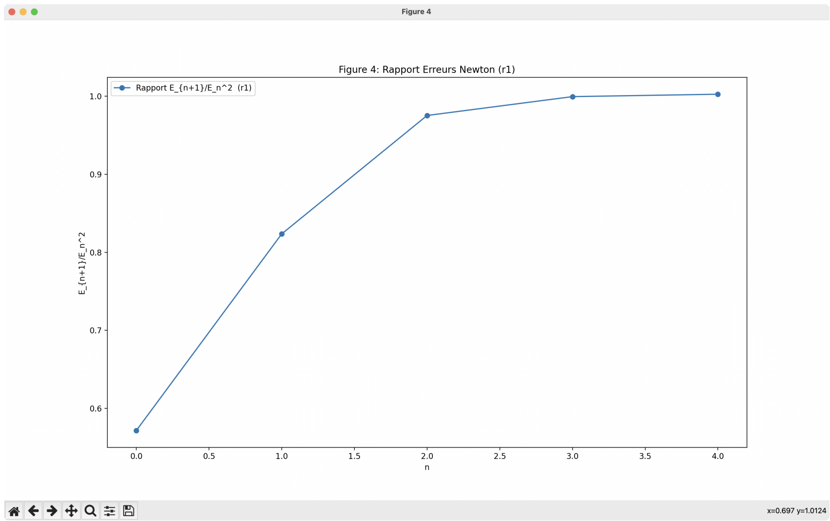


Figure 4: Erreur en fonction des itérations - Méthode de Newton pour r_2

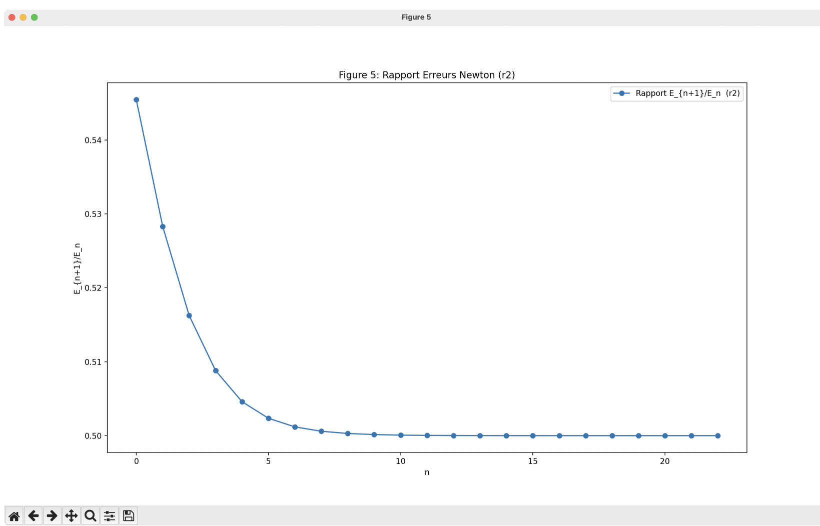


Figure 5: Rapport des erreurs successives - Méthode de Newton pour r_2

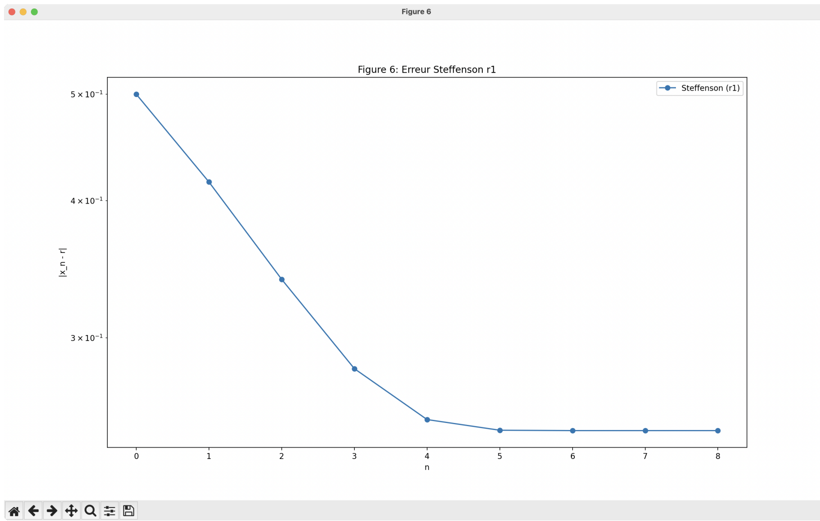


Figure 6: Erreur en fonction des itérations - Méthode de Steffenson pour r_1

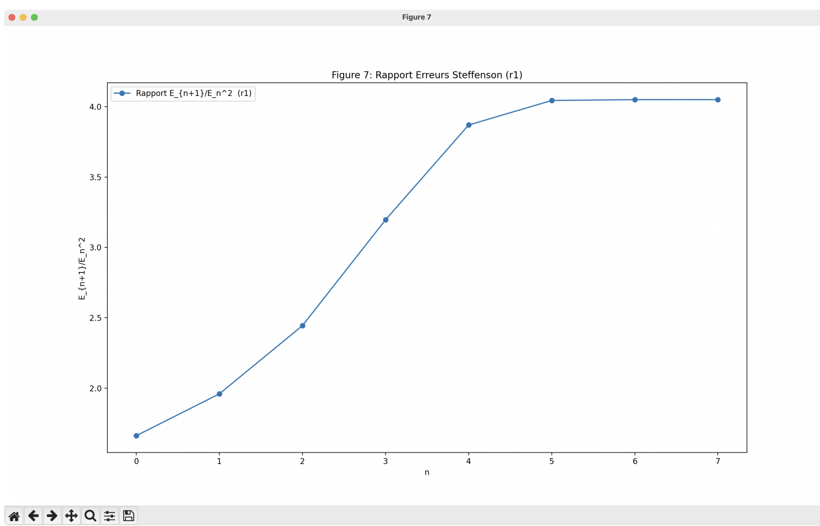


Figure 7: Rapport des erreurs successives - Méthode de Steffenson pour r_1

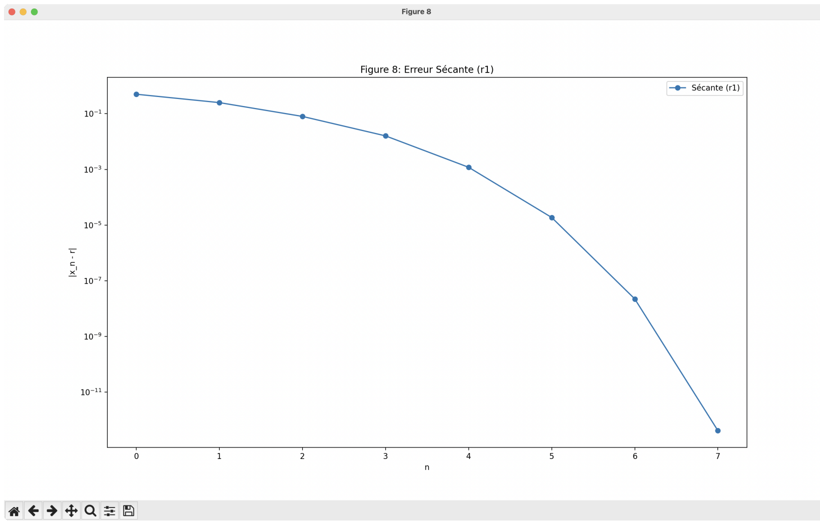


Figure 8: Erreur en fonction des itérations - Méthode de la s'ecante pour r_1

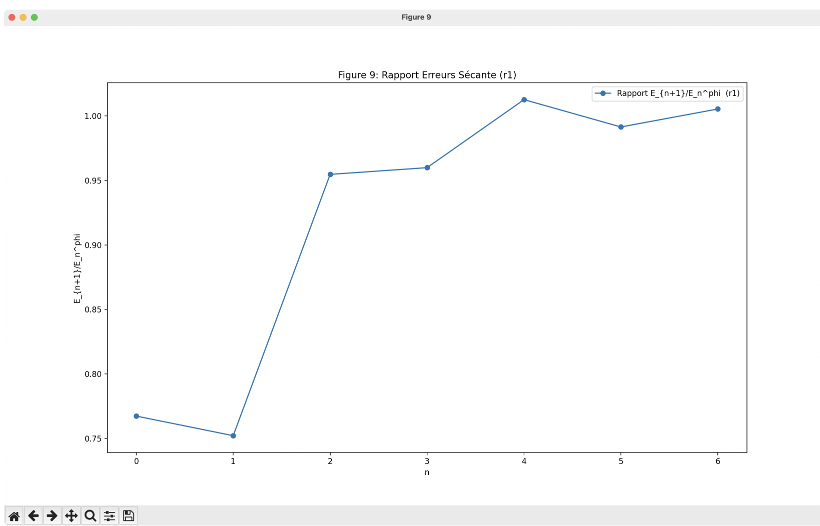
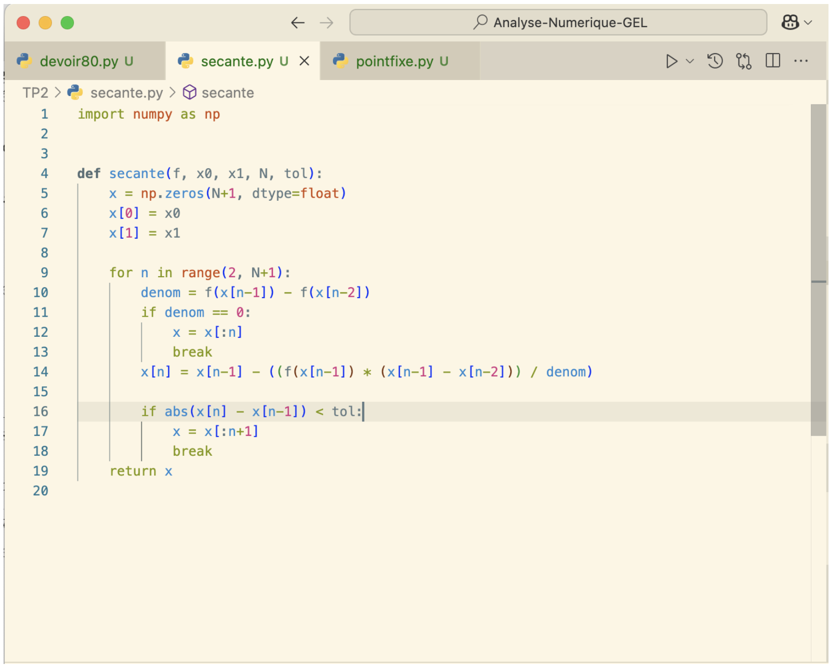


Figure 9: Rapport des erreurs successives - Méthode de la s'ecante pour r_1



```python
import numpy as np


def secante(f, x0, x1, N, tol):
    x = np.zeros(N+1, dtype=float)
    x[0] = x0
    x[1] = x1

    for n in range(2, N+1):
        denom = f(x[n-1]) - f(x[n-2])
        if denom == 0:
            x = x[:n]
            break
        x[n] = x[n-1] - ((f(x[n-1]) * (x[n-1] - x[n-2])) / denom)

        if abs(x[n] - x[n-1]) < tol:
            x = x[:n+1]
            break
    return x
```

Figure 10: Rapport - Point Fixe