**Báo cáo chi tiết về phương pháp   
Newton-Raspson**

# 1. Bài toán

…

Bài toán dẫn đến một hệ phương trình tuyến tính 6 ẩn được biểu diễn dưới đây:

Trong đó:

- là các hằng số.

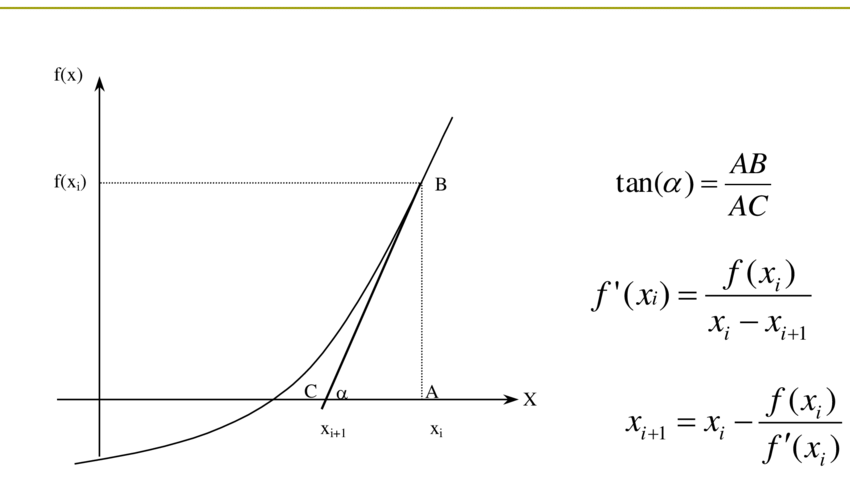
- là các tham số được xác định bằng các công thức khi có giá trị ER, .

Kết quả thực nghiệm cho thấy các giá trị có các kì vọng như sau:

trong đó, .

# 2. Thuật toán Newton Raspson

## 2.1 Tư tưởng thuật toán



- Cho hàm số có đồ thị như trên, yêu cầu đặt ra là tìm nghiệm của phương trình .

- Tại điểm bất kì thuộc hàm f(x) ta có các thông tin sau:

* Phương trình tiếp tuyến với điểm có công thức:

* Góc tạo bởi tiếp tuyến tại điểm và trục hoành (Ox) có giá trị được tính bằng công thức:

Với A là hình chiếu của B lên trục hoành (Ox), C là giao điểm của tiếp tuyến (d).

-

## 2.2 Chi tiết thuật toán

- Đầu vào của thuật toán (Input):

* Vector là nghiệm khởi tạo.
* : sai số cho phép.
* N: số lượng vòng lặp giới hạn.

- Đầu ra của thuật toán (Output):

- Vector nghiệm tìm được.

A close up of text on a black background

Description automatically generated- Mã giả (pseudocode) của thuật toán:

* Bước 1: Khởi tạo
* Bước 2: While ( thực hiện buớc 3 đến bước 7.
* Bước 3: Tính giá trị và , trong đó , với .
* Bước 4: Giải hệ phương trình tuyến tính , với là vector cần tìm.
* Bước 5: Cập nhật nghiệm bằng công thức:
* Bước 6: Nếu :

Trả về giá trị nghiệm thu được

// Thành công

* Bước 7: .
* Bước 8: Trả về nghiệm thu được

// Không thành công

*Mã giã là một cách biểu diễn thuật toán bằng ngôn ngữ, các kí hiệu, quy ước sao cho dễ dàng trình bày được từng bước của thuật toán và khi đọc mã giả, ta có thể hiểu và cài đặt lại thuật toán.*

## 2.2 Ví dụ

Giải hệ phương trình tuyến tính sau:

Sử dụng thuật toán Newton-Raspson để giải hệ phương trình tuyến tính trên như sau:

* Đặt:
  + là nghiệm của hệ phương trình.

* Giá trị đầu vào:
  + Nghiệm khởi tạo:
  + Sai số cho phép:
  + Số lượng vòng lặp giới hạn: N = 1000
* Khởi tạo .
* (đúng)

  + Giải hệ phương trình tuyến tính , ta được:
  + Cập nhật nghiệm:
  + Kiểm tra độ dài vector thặng dư.  
    Vì , nên ta thực hiện tiếp vòng lặp thứ 2.
* (đúng)
  + …
  + …
* …

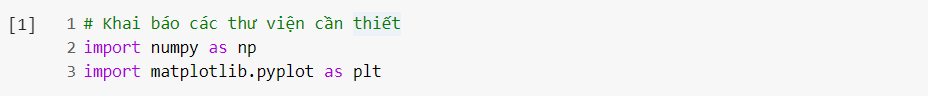
Lặp lại tương tự các bước trên cho cho đến khi vector thặng dư tại thời điểm vòng lặp thứ có độ lớn nhỏ hơn hoặc là thuật toán đã lặp được lần nhưng vẫn chưa tìm được nghiệm thỏa mãn sai số cho trước.

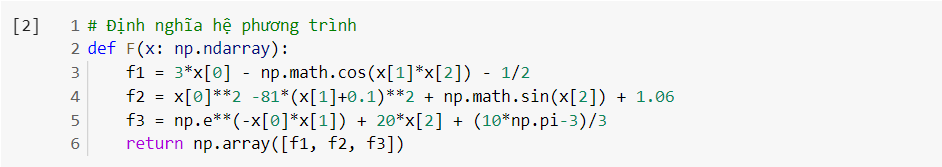
Kết quả sau 5 vòng lặp được ghi nhận trong bảng đưới đây:

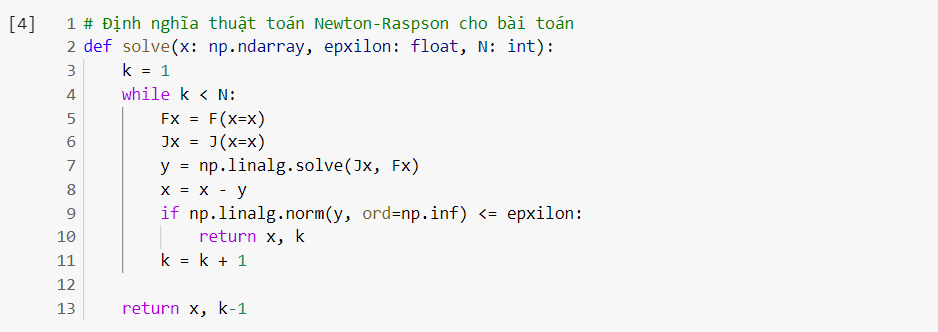
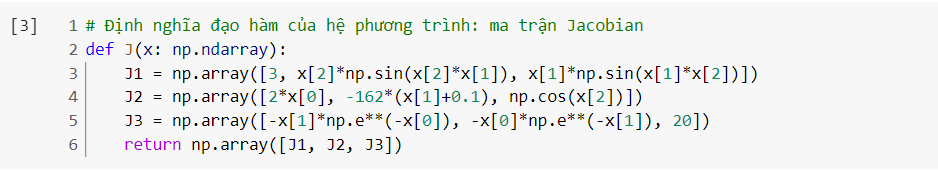
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| 0 | 0.1 | 0.1 | -0.1 |  |
| 1 | 0.49987 | 0.019463 | -0.521657 | 0.586665344512809 |
| 2 | 0.500014 | 0.001588 | -0.523553 | 0.017975385119342 |
| 3 | 5.000001e-01 | 1.243978e-05 | -5.235984e-01 | 0.001576447946574 |
| 4 | 5.000000e-01 | 7.752669e-10 | -5.235988e-01 | 1.244464609669e-05 |
| 5 | 5.000000e-01 | -1.308796e-17 | -5.235988e-01 | 7.756181058021e-10 |

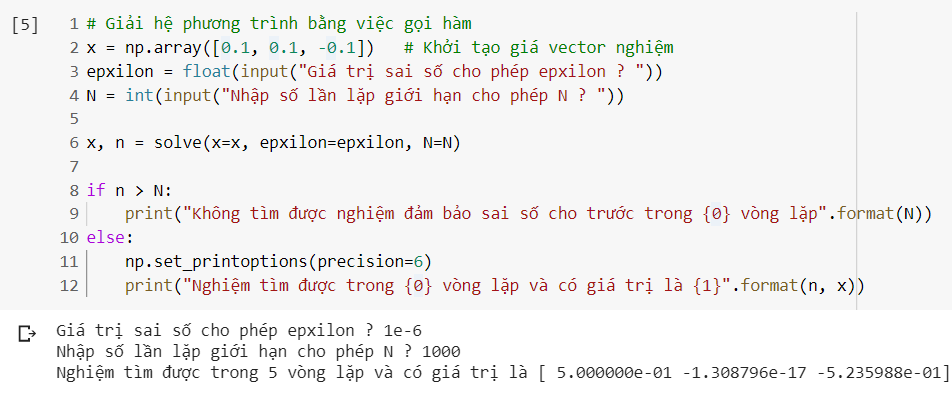
Kiểm tra lại bằng cách thay nghiệm nhận được vào hệ phương trình ban đầu ta được kết quả sau:

Dưới đây là đoạn code thực hiện chương trình được viết bằng ngôn ngữ lập trình Python.









# 3. Áp dụng thuật toán Newton – Raspson vào bài toán

## 3.1 Phân tích bài toán.

Quay lại bài toán của chúng ta, một hệ phương trình phi tuyến tính gồm 6 ẩn số .

Chúng ta vẫn áp dụng thuật toán Newton-Raspson để cài đặt lời giải cho bài toán nhưng do bài toán thực tế của chúng ta nhận giá trị đầu vào là nên cần thiết kế giải pháp cho bài toán sao cho dễ dàng cài đặt và tùy biến cho các mục đích khác, đặc biệt là dễ hiểu.

A close up of a map

Description automatically generatedChúng ta chia chương trình thành nhiều modules (files) để tiện quản lý và tránh sai sót, cụ thể như sau:

* File *constants.py* chứa tất cả các hằng số, ví dụ: a, b, w, … mà bài toán cung cấp. Các giá trị hằng số này sẽ dùng để cập nhật các tham số mà ta cần tính khi đẫ nhập hai giá trị ER và T2.
* File *parameters.py* chứa các tham số, ví dụ: q, K1, K2, … được lập luận từ bài toán. Các giá trị tham số này sẽ có gía trị xác định khi ta nhập ER, T2 (tức khi ER, T2 xác định) và các giá trị này lại tiếp tục được đưa vào hai module tiếp theo là functions.py và derivatives.   
  *Lưu ý: các giá trị bên trong module constaints.py sau khi đã đưa vào module parameters.py thì bên trong module parameters.py cũng sẽ có các biến giá trị giống như trong module constants.py. Vì vậy việc phân chia hai modules constants.py và parameters.py cũng có thể bỏ qua (gộp hai modules thành 1) nhưng vì số lượng tham số và hằng số khá nhiều và để tránh sai sót nên chia ra để dễ theo dõi và tinh chỉnh.*
* File *functions.*py chứa 6 phương trình, mỗi phương trình là một phương trình 6 ẩn số được thiết lập từ bài toán*.* Khi module *paramters.py* được cập nhật (các giá trị trị bên trong module đều được xác định) thì 6 phương trình bên trong module functions.py sẽ được xác định và không phụ thuộc vào các tham số nữa. Để lấy được giá trị của hệ phương trình ta cần cung cấp cho module này vector nghiệm .
* File *derivatives.py* chứa đạo hàm của hệ phương trình. Tương tự như file *functions.py*, ma trận Jacobian sẽ được xác định (chỉ phụ thuộc vào vector nghiệm). Để lấy được giá trị của ma trận Jacobian ta cần cung cấp cho module này vector nghiệm .
* Module *main.py* là nơi chương trình được thực thi. Chức năng của module này đó là cho phép chúng ta nhập vào giá trị ER, T2 và gửi giá trị ER, T2 này cho module paremeters.py (đã đề cập bên trên), quá trình cập nhật diễn ra. Tiếp đó, khi mà các phương trình bên trong 2 modules *functions.py* và *derivatives.py* đã được xác định thì ta tiến hành chạy thuật toán Newton – Raspson để tìm nghiệm gần đúng của hệ phương trình.

## 3.2 Quá trình giải bài toán