ΕΡΓΑΣΤΗΡΙΟ 1

05/03/2021

Η ΜΕΘΟΔΟΣ NEWTON – RAPHSON

Ντουνέτας Δημήτρης
ΑΜ: 2016030141

Εισαγωγή

Σκοπός του πρώτου σετ ασκήσεων είναι η εξοικείωση με την τις επαναληπτικές μεθόδους που χρησιμοποιούνται στα προγράμματα CAD και πιο συγκεκριμένα η εξοικείωση με την μέθοδο Newton-Raphson.Μας ζητείτε να φτιάξουμε ένα πρόγραμμα που να προσεγγίζει τη ρίζα μιας πολυωνυμικής συνάρτησης μέχρι $5^{ου}$ βαθμού με αποδεκτό σφάλμα $\varepsilon = 10^{-3}$. Οι πράξεις γίνονται με αριθμητική κινητής υποδιαστολής απλής ακρίβειας(float).

Λειτουργία του προγράμματος

Το πρόγραμμα ξεκινά ζητώντας από τον χρήστη να εισάγει τον βαθμό του πολυωνύμου που πρόκειται να προσθέσει στη συνέχεια. Αν ο βαθμός είναι στα αποδεκτά όρια(0 – 5) το πρόγραμμα συνεχίζει, αλλιώς ζητάει ξανά είσοδο απο το χρήστη. Στη συνέχεια ο χρήστης προσθέτει μια-μια τις τιμές των συντελεστών του πολυωνύμου του ξεκινώντας από το μεγαλύτερο βαθμό σε φθίνουσα σειρά βαθμού.Αφού τελειώσει αυτή η διαδικασία ο χρήστης λαμβάνει τα αποτελέσματα στη κονσόλα. Αυτά είναι το πολυώνυμο που πρόσθεσε και επαναληπτικά για κάθε επανάληψη το X της επανάληψης καθώς και το P(x) της επανάληψης. Αφού τελειώσουν οι επαναλήψεις το πρόγραμμα είτε θα συγκλίνει μέσα σε 20 επαναλήψεις είτε θα φτάσει το μέγιστο αριθμό επαναλήψεων που είναι το 20 και θα σταματήσει χωρίς να συγκλίνει και να βρεί αποτέλεσμα. Η διαδικασία αυτή θα γίνει 2 φορές. Τη πρώτη με υπολογισμό πρώτης παραγώγου με αναλυτική μέθοδο και τη δέυτερη με υπολογισμό παραγώγου αριθμητικά. Για τον αριθμητικό υπολογισμό της παραγώγου έχει οριστεί ένα δ=0.00001. Στο τέλος της κάθε μεθόδου υπολογισμού εκτυπώνονται στη κονσόλα τα στατιστικά της μεθόδου. Εκτυπώνεται ο αριθμός επαναλήψεων, αριθμός προσθέσεων/αφαιρέσεων, αριθμός πολλαπλασιασμών και αριθμός διαιρέσεων.

Ανάλυση του κώδικα

Αρχικοποιήσεις συναρτήσεων και μεταβλητών:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <math.h>

void reqA(int deg, float coef[]);//Requirment A function
void reqB(int deg, float coef[]);//Requirment B function
float polynomial_derivative(int deg, float coef[], float x, bool is_der);//Find polynomial value or derivative
int repetitions,add_subs,multiplications,divisions;
```

Η main συνάρτηση:

```c
int main(){

    int deg; // Polynomial's degree
    int max_deg = 5; // maximum degree of input polynomial
    float coef[6];//coefficients array
    while (deg <=0 || deg > max_deg) // if polynomial degree is 0 to 5 is accepted else enter again degree
    {
        printf("Enter polynomial's degree: ");
        scanf("%d",&deg);
    }
    printf("Enter Polynomial's coefficients: \n");
    for (int i = 0; i <= deg; i++)//get coefficients
    {
        printf("Give coefficient x^%d: ",deg-i);
        scanf("%f",&coef[i]);
    }
    printf("\nPolynomial is: ");//print polynomial
    for (int i = 0; i <= deg; i++)
    {
        printf("(%f)*x^%d",coef[i],deg-i);
        if (i<deg)
        {
           printf(" + ");
        }
    }
    printf("\n\n------------------Requirment A--------------------\n");
    reqA(deg,coef);// Requirement A, with analytical derivative
    printf("\n\n------------------Requirment B--------------------\n");
    reqB(deg,coef);// Requirement B, with arethmetic derivative

}
```

Έχει αρχικοποιήσεις μεταβλητών και την κύρια λειτουργία για την επικοινωνία με το χρήστη και την κλήση των συναρτήσεων υπολογισμού της ρίζας με τις 2 διαφορετικές μεθόδους εύρεσης της παραγώγου.

Η συνάρτηση για το ερώτημα Α:

```
void reqA(int deg, float coef[]){
    //Initialize statistics
    repetitions=0;
    add_subs=0;
    multiplications=0;
    divisions=0;
    float value,derivative,h,e=0.001;
    //starting X0
    float x=10;
    //Basic funtion characteristics
    for (int i = 0; i < 20; i++)
    {
        repetitions++;
        value = polynomial_derivative(deg,coef,x,false);//find the P(x)
        derivative = polynomial_derivative(deg-1,coef,x,true);//find the P'(x)
        h = -(value/derivative);//find the value to move x
        multiplications++;
        divisions++;
        x += h;//move x by adding h
        add_subs++;
        printf("\nx%d = %f",i,x);
        printf("\nP(%f)=%f",x,value);
        if (fabs(h)<e)//if h is less than error we found an acceptable solution
            break;

    }
        printf("\nAnalytical derivation: x = %f",x);
        printf("\nRepetitions are: %d\nAdditions and Substractions are: %d\nMultiplications are: %d\nDivisions are: %d",repetitions,add_subs,multiplications,divisions);
}
```

Βρίσκει τη ρίζα του πολυωνύμου με χρήση της αναλυτικής εύρεσης παραγώγου. Αρχικά μηδενίζει όλα τα στατιστικά που θέλουμε να λάβουμε για την εφαρμογή και επαναληπτικά υπολογίζει την τιμή του P(x) και του P’(x). Στη συνέχεια υπολογίζει το h και το προσθέτει στο x για να λάβουμε το νέο x για την επόμενη επανάληψη. Στο τέλος εμφανίζει το τελικό x και εμφανίζει τα στατιστικά από τις πράξεις που έτρεξε το πρόγραμμα.

Η συνάρτηση για το ερώτημα Β:

```
void reqB(int deg, float coef[]){
    //Initialize statistics
    repetitions=0;
    add_subs=0;
    multiplications=0;
    divisions=0;
    float value1,value2,derivative,h,e=0.001,delta=0.01;
    //starting X0
    float x=10;
    //Basic funtion characteristics
    for (int i = 0; i < 20; i++)
    {
        repetitions++;
        value1 = polynomial_derivative(deg,coef,x,false);//find the P(x)
        value2 = polynomial_derivative(deg,coef,x+delta,false);//find the P(x+delta)
        derivative = ((value2-value1)/delta);//find the P'(x)
        divisions++;
        add_subs++;
        h = -(value1/derivative);//find the value to move x
        multiplications++;
        divisions++;
        x += h;//move x by adding h
        add_subs++;
        printf("\nx%d = %f",i,x);
        printf("\nP(%f)=%f",x,value1);
        if (fabs(h)<e)//if h is less than error we found an acceptable solution
            break;

    }
        printf("\nArithmetic derivation x = %f\n",x);
        printf("\nRepetitions are: %d\nAdditions and Substractions are: %d\nMultiplications are: %d\nDivisions are: %d",repetitions,add_subs,multiplications,divisions);
```

Όμοια με την προηγούμενη αρχικοποιεί και υπολογίζει το P(x) και το P’(x) απλά αντί για αναλυτική μέθοδο χρησιμοποιεί αριθμητική μέθοδο για υπολογισμό του P’(x).Τέλος, επίσης εμφανίζει το τελικό x και τα στατιστικά από τις πράξεις που έτρεξε το πρόγραμμα.

Η συνάρτηση υπολογισμού τιμής συνάρτηση P(x) και τιμής παραγώγου P’(x):

```
float polynomial_derivative(int deg, float coef[], float x ,bool is_der){
    //initialization
    float value=0,power=1;
    //do the above for every coefficient
    for (int i = deg; i >= 0; i--)
    {
        power=1;
        //calculate the power of x
        for (int j = 0; j < i; j++)
        {
            power  *= x;
            multiplications++;
        }
        //if is_der == True calculate derivative else calcualte polynomial.
        if (is_der)
        {
            //calculate P(x)
            value += power * coef[deg-i] * (float)(i+1);
            multiplications +=2;
            add_subs+=2;
        }
        else
        {
            //calculate P'(x)
            value += power * coef[deg-i];
            add_subs++;
            multiplications++;
        }

    }
    return value;
}
```

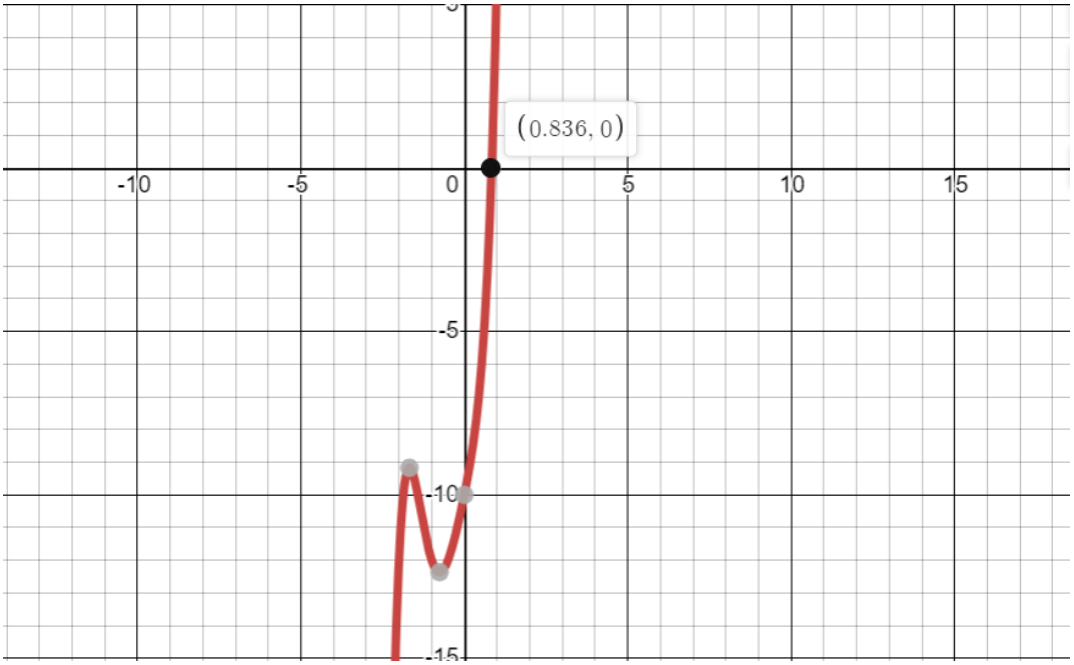
Υπολογίζει την τιμή της συνάρτησης για δεδομένο x και την τιμή της παραγώγου για δεδομένο x. Για λειτουργία υπολογισμού συνάρτησης δίνουμε ώς είσοδο στην συνάρτηση το is_der ως False και για υπολογισμού παραγώγου δίνουμε το is_der ως True. Επιστρέφει στο τέλος την τιμή που υπολόγισε.

Ανάλυση της διαδικασίας

Στις μετρήσεις χρησιμοποιούμε το πολυώνυμο $\mathbf{y=2x^5+5x^4+2x^3+2x^2+5x-10}$

Όπως βλέπουμε στο desmos.com έχει τη παρακάτω γραφική παράσταση:

Υπάρχει μια μοναδική λύση στο 0.836 που μας κάνει ως προσομοιώση ενός συστήματος που θα χρησιμοποιήσουμε ένα εργαλείο CAD αφού είναι μοναδική και θετική λύση.



Α Ερώτημα. Αναλυτική μέθοδος υπολογσμού Παραγώγου

Πολυώνυμο : $y=2x^5+5x^4+2x^3+2x^2+5x-10$				
X0 = 5				
Επαναλήψεις	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί**	**Διαιρέσεις**	**Αποτέλεσμα**
10	**170**	**420**	**10**	**0,835817**

X0 = 10				
Επαναλήψεις	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί**	**Διαιρέσεις**	**Αποτέλεσμα**
13	**221**	**546**	**13**	**0,835817**

Παρατηρούμε ότι όσο πιο κοντά στην πραγματική λύση είναι το αρχικό X0 τόσο λιγότερες επαναλήψεις χρειάζεται για να συγκλίνει η διαδικασία και έτσι μας δίνει πιο γρήγορα αποτέλεσμα και με λιγότερες πράξεις.

Β Ερώτημα. Αριθμητική μέθοδος υπολογσιμού Παραγώγου

Πολυώνυμο : $y=2x^5+5x^4+2x^3+2x^2+5x-10$				
X0 = 5,δ=0,1				
Επαναλήψεις	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί**	**Διαιρέσεις**	**Αποτέλεσμα**
11	**154**	**473**	**22**	**0,835940**
X0 = 5,δ=0,01				
Επαναλήψεις	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί**	**Διαιρέσεις**	**Αποτέλεσμα**
10	**140**	**430**	**20**	**0,835819**
X0 = 5,δ=0,001				
Επαναλήψεις	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί**	**Διαιρέσεις**	**Αποτέλεσμα**
10	**140**	**430**	**20**	**0,835817**
X0 = 5,δ=0,0001				
Επαναλήψεις	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί**	**Διαιρέσεις**	**Αποτέλεσμα**
10	**140**	**430**	**20**	**0,835817**

X0 = 10,δ=0,1				
Επαναλήψεις	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί**	**Διαιρέσεις**	**Αποτέλεσμα**
14	**196**	**602**	**28**	**0,835940**
X0 = 10,δ=0,01				
Επαναλήψεις	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί**	**Διαιρέσεις**	**Αποτέλεσμα**
13	**182**	**559**	**26**	**0,835819**
X0 = 10,δ=0,001				
Επαναλήψεις	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί**	**Διαιρέσεις**	**Αποτέλεσμα**
13	**182**	**559**	**26**	**0,835817**
X0 = 10,δ=0,0001				
Επαναλήψεις	**Προσθέσεις/Αφαιρέσεις**	**Πολλαπλασιασμοί**	**Διαιρέσεις**	**Αποτέλεσμα**
13	**182**	**559**	**26**	**0,835817**

Παρατηρούμε ότι και στο δεύτερο ερώτημα ισχύει το συμπέρασμα που βγάλαμε από το προηγούμενο ερώτημα και όσο πιο κοντά είναι το Χ0 στην πραγματική τιμή της ρίζας τόσο λιγότερες επαναλήψεις χρειαζόμαστε.

Επιπρόσθετα, παρατηρούμε ότι για δ μικρό λαμβάνουμε πιο ακριβή αποτελέσματα με λιγότερες επαναλήψεις αφού συγκλίνουμε πιο γρήγορα. Επίσης για δ αρκετά μεγάλο, για παράδειγμα δ=1, η διαδικασία δεν συγκλίνει και δε μπορούμε να λάβουμε στοιχεία.

Παράδειγμα εκτύπωσης αποτελέσματος από το πρόγραμμα που δημιουργήθηκε

```
Enter polynomial's degree: 5
Enter Polynomial's coefficients:
Give coefficient x^5: 2
Give coefficient x^4: 5
Give coefficient x^3: 2
Give coefficient x^2: 2
Give coefficient x^1: 5
Give coefficient x^0: -10

Polynomial is: (2.000000)*x^5 + (5.000000)*x^4 + (2.000000)*x^3 + (2.000000)*x^2 + (5.000000)*
x^1 + (-10.000000)*x^0
```

```
------------------Requirment A--------------------

x0 = 7.909238
P(7.909238)=252240.000000
x1 = 6.238276
P(6.238276)=82612.265625
x2 = 4.903255
P(4.903255)=27052.144531
x3 = 3.837037
P(3.837037)=8856.733398
x4 = 2.985965
P(2.985965)=2898.876465
x5 = 2.307622
P(2.307622)=948.219788
x6 = 1.769978
P(1.769978)=309.423401
x7 = 1.353087
P(1.353087)=100.021515
x8 = 1.054296
P(1.054296)=31.212784
x9 = 0.887476
P(0.887476)=8.621140
x10 = 0.839258
P(0.839258)=1.613324
x11 = 0.835833
P(0.835833)=0.100565
x12 = 0.835817
P(0.835817)=0.000467
Analytical derivation: x = 0.835817
Repetitions are: 13
Additions and Substractions are: 221
Multiplications are: 546
Divisions are: 13
```

```
------------------Requirment B--------------------

x0 = 7.948836
P(7.948836)=252240.000000
x1 = 6.309402
P(6.309402)=84588.625000
x2 = 4.999425
P(4.999425)=28524.398438
x3 = 3.953021
P(3.953021)=9684.868164
x4 = 3.117507
P(3.117507)=3316.003418
x5 = 2.451036
P(2.451036)=1146.837036
x6 = 1.921284
P(1.921284)=401.095642
x7 = 1.505504
P(1.505504)=141.661606
x8 = 1.192291
P(1.192291)=50.039455
x9 = 0.982351
P(0.982351)=17.117338
x10 = 0.875739
P(0.875739)=5.223653
x11 = 0.842718
P(0.842718)=1.226748
x12 = 0.836761
P(0.836761)=0.202660
x13 = 0.835940
P(0.835940)=0.027497
Arithmetic derivation x = 0.835940

Repetitions are: 14
Additions and Substractions are: 196
Multiplications are: 602
Divisions are: 28
```

Συμπέρασμα

Η μέθοδος Newton-Raphson είναι μια επαναληπτική διαδικασία για την εύρεση ριζών πολυωνύμου. Παρόλο που δεν εγγυάται τη λύση μέσω σύγκλισης σε κάθε περίπτωση, αν χρησιμοποιηθεί σωστά και ρυθμιστεί το x0 και το δ για μια συγκεκριμένη εφαρμογή, όπως δηλαδή ενα εργαλείο CAD που γνωρίζουμε τα χαρακτηριστικά των συστημάτων που θα αναλυθούν μας δίνει γρήγορα και σχετικά εύκολα τη λύση που αναζητούμε.