**נספח**

התוכנית הבאה מוצאת את השורש של הפונקציה f(x) = x3 – 100 כלומר מוצאת השורש השלישי של 100:

/\* newton1.c - newton's method \*/

#include <stdio.h>

#include <math.h>

long double newton(long double (\*fun)(long double),

long double (\*fd)(long double),

long double x0, long double eps)

{

 long double fdv, f0;

 int i;

 do {

 f0 = (\*fun)(x0);

 fdv = (\*fd)(x0);

 printf("x0 = %LF, f0 = %LF, fdv = %LF\n", x0, f0, fdv);

 x0 = x0 - f0/fdv;

 } while (fabsl(f0) > eps);

return x0;

} /\* newton \*/

long double poly(long double x)

{

 return (x\*x\*x - 100.0);

} /\* poly \*/

long double polyd(long double x)

{

 return (3\*x\*x);

}

int main ()

{

long double x;

x = newton(poly, polyd, 3.0, 0.0000001);

printf("Solution to poly(x) = 0, x = %Lf, f(%Lf) = %Lf\n", x, x,

poly(x));

return 0;

} /\* main \*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

C:\> newton1.exe

x0 = 3.000000, f0 = -73.000000, fdv = 27.000000

x0 = 5.703704, f0 = 85.554235, fdv = 97.596708

x0 = 4.827094, f0 = 12.475318, fdv = 69.902505

x0 = 4.648626, f0 = 0.455553, fdv = 64.829184

x0 = 4.641599, f0 = 0.000688, fdv = 64.633337

x0 = 4.641589, f0 = 0.000000, fdv = 64.633041

Solution to poly(x) = 0, x = 4.641589, f(4.641589) = 0.000000

C:\>

התוכנית הבאה פותרת את אותה הבעיה בערת שיטת מיתר:

/\* secant1.c - secant method \*/

#include <stdio.h>

#include <math.h>

long double secant(long double (\*fun)(long double),

long double x1,

long double x0, long double eps)

{

 long double f0, f1, oldx1;

 int i;

 do {

 f0 = (\*fun)(x0);

 f1 = (\*fun)(x1);

 oldx1 = x1;

 printf("x1 = %LF, f1 = %LF, x0 = %LF, f0 = %LF\n",

 x1, f1, x0, f0);

 x1 = x1 - ((x1-x0)/(f1-f0))\*f1;

 x0 = oldx1;

 } while (fabsl(f1) > eps);

return x1;

} /\* secant \*/

long double poly(long double x)

{

 return (x\*x\*x - 100.0);

} /\* poly \*/

long double polyd(long double x)

{

 return (3\*x\*x);

}

int main ()

{

long double x;

x = secant(poly, 3.0, 2.0, 0.0000001);

printf("\nSolution to poly(x) = 0, x = %Lf, f(%Lf) = %Lf\n",

x, x, poly(x));

return 0;

} /\* main \*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

C:\> secant1.exe

x1 = 3.000000, f1 = -73.000000, x0 = 2.000000, f0 = -92.000000

x1 = 6.842105, f1 = 220.309083, x0 = 3.000000, f0 = -73.000000

x1 = 3.956239, f1 = -38.077615, x0 = 6.842105, f0 = 220.309083

x1 = 4.381520, f1 = -15.884811, x0 = 3.956239, f0 = -38.077615

x1 = 4.685921, f1 = 2.892761, x0 = 4.381520, f0 = -15.884811

x1 = 4.639027, f1 = -0.165513, x0 = 4.685921, f0 = 2.892761

x1 = 4.641565, f1 = -0.001572, x0 = 4.639027, f0 = -0.165513

x1 = 4.641589, f1 = 0.000001, x0 = 4.641565, f0 = -0.001572

x1 = 4.641589, f1 = -0.000000, x0 = 4.641589, f0 = 0.000001

Solution to poly(x) = 0, x = 4.641589, f(4.641589) = -0.000000

C:\>

**שיטת Steffensen**

התוכנית הבאה מחשבת שורש שלישי של 100 בעזרת שיטת Steffensen:

/\* steff.c - Steffensen method \*/

#include <stdio.h>

#include <math.h>

long double steffensen(long double (\*fun)(long double),

long double x0, long double eps)

{

 long double f0, f1, g;

 int i;

 f0 = (\*fun)(x0);

 while (fabsl(f0) > eps)

 {

 f1 = (\*fun)(x0 + (\*fun)(x0));

 g = (f1 - f0)/f0;

 x0 = x0 - f0/g ;

 f0 = (\*fun)(x0);

 } // while

return x0;

} /\* steffensen \*/

long double poly(long double x)

{

 return (x\*x\*x - 100.0);

} /\* poly \*/

long double polyd(long double x)

{

 return (3\*x\*x);

}

int main ()

{

long double x;

x = steffensen(poly, 2.0, 0.0000001);

printf("\nSolution to poly(x) = 0, x = %Lf, f(%Lf) = %Lf\n",

x, x, poly(x));

return 0;

} /\* main \*/

**פלט ריצה:**

C:\>steff.exe

Solution to poly(x) = 0, x = 4.641589, f(4.641589) = 0.000000

C:\>

תוכנית דומה שעושה הדפסות ביניים:

/\* steff.c - Steffensen method \*/

#include <stdio.h>

#include <math.h>

long double steffensen(long double (\*fun)(long double),

long double x0, long double eps)

{

 long double f0, f1, g;

 int i=0;

 f0 = (\*fun)(x0);

 while (fabsl(f0) > eps)

 {

 f1 = (\*fun)(x0 + (\*fun)(x0));

 g = (f1 - f0)/f0;

 x0 = x0 - f0/g ;

 f0 = (\*fun)(x0);

 i++;

 printf("%d:x0 = %Lf, f(%Lf) = %Lf\n", i, x0, x0, f0);

 } // while

return x0;

} /\* steffensen \*/

long double poly(long double x)

{

 return (x\*x\*x - 100.0);

} /\* poly \*/

long double polyd(long double x)

{

 return (3\*x\*x);

}

int main ()

{

long double x;

x = steffensen(poly, 3.5, 0.0000001);

printf("\nSolution to poly(x) = 0, x = %Lf, f(%Lf) = %Lf\n",

x, x, poly(x));

return 0;

} /\* main \*/

פלט ריצה:

C:\>staff1.exe

1:x0 = 3.521156, f(3.521156) = -56.342815

2:x0 = 3.542689, f(3.542689) = -55.536959

3:x0 = 3.564625, f(3.564625) = -54.705896

4:x0 = 3.586992, f(3.586992) = -53.847914

5:x0 = 3.609821, f(3.609821) = -52.961102

6:x0 = 3.633148, f(3.633148) = -52.043309

7:x0 = 3.657011, f(3.657011) = -51.092105

8:x0 = 3.681458, f(3.681458) = -50.104727

9:x0 = 3.706538, f(3.706538) = -49.078013

10:x0 = 3.732312, f(3.732312) = -48.008310

11:x0 = 3.758850, f(3.758850) = -46.891371

12:x0 = 3.786233, f(3.786233) = -45.722208

13:x0 = 3.814559, f(3.814559) = -44.494894

14:x0 = 3.843943, f(3.843943) = -43.202306

15:x0 = 3.874527, f(3.874527) = -41.835760

16:x0 = 3.906487, f(3.906487) = -40.384488

17:x0 = 3.940046, f(3.940046) = -38.834877

18:x0 = 3.975489, f(3.975489) = -37.169316

19:x0 = 4.013199, f(4.013199) = -35.364358

20:x0 = 4.053699, f(4.053699) = -33.387666

21:x0 = 4.097746, f(4.097746) = -31.192588

22:x0 = 4.146493, f(4.146493) = -28.707678

23:x0 = 4.201849, f(4.201849) = -25.814124

24:x0 = 4.267378, f(4.267378) = -22.288867

25:x0 = 4.351145, f(4.351145) = -17.622099

26:x0 = 4.479486, f(4.479486) = -10.115540

27:x0 = 4.859995, f(4.859995) = 14.790873

28:x0 = 4.830722, f(4.830722) = 12.729120

29:x0 = 4.800161, f(4.800161) = 10.603101

30:x0 = 4.768438, f(4.768438) = 8.424720

31:x0 = 4.735998, f(4.735998) = 6.226933

32:x0 = 4.703989, f(4.703989) = 4.087592

33:x0 = 4.674953, f(4.674953) = 2.171958

34:x0 = 4.653394, f(4.653394) = 0.764932

35:x0 = 4.643359, f(4.643359) = 0.114437

36:x0 = 4.641632, f(4.641632) = 0.002816

37:x0 = 4.641589, f(4.641589) = 0.000002

38:x0 = 4.641589, f(4.641589) = 0.000000

Solution to poly(x) = 0, x = 4.641589, f(4.641589) = 0.000000

C:\>

ניכר בעליל שהשיטה טובה מאד בסביבות הפתרון אבל מפוקפק הרחק ממנו.

**שיטת ה-Inverse quadratic interpolation**

**(שיטת הפרבולה ההפוכה)**

התוכנית הבאה מחשבת שורש שלישי של 100 בעזרת שיטת Inverse quadratic interpolation:

/\* invquad.c - invquad method \*/

#include <stdio.h>

#include <math.h>

long double invquad(long double (\*fun)(long double),

long double x0, long double x1, long double x2, long double eps)

{

 long double f0, f1, f2, c0, c1, c2, temp;

 int i;

 f0 = (\*fun)(x0);

 f1 = (\*fun)(x1);

 f2 = (\*fun)(x2);

 while (fabsl(f2) > eps)

 {

 c0 = f2\*f1/((f0-f1)\*(f0-f2));

 c1 = f0\*f2/((f1-f0)\*(f1-f2));

 c2 = f0\*f1/((f2-f0)\*(f2-f1));

 temp = c0\*x0 + c1\*x1 + c2\*x2;

 x0 = x1;

 x1 = x2;

 x2 = temp;

 f0 = f1;

 f1 = f2;

 f2 = (\*fun)(x2);

 } // while

return x2;

} /\*invquad \*/

long double poly(long double x)

{

 return (x\*x\*x - 100.0);

} /\* poly \*/

long double polyd(long double x)

{

 return (3\*x\*x);

}

int main ()

{

long double x;

x = invquad(poly, 3.0, 4.0, 5.0, 0.0000001);

printf("\nSolution to poly(x) = 0, x = %Lf, f(%Lf) = %Lf\n",

x, x, poly(x));

return 0;

} /\* main \*/

פלט ריצה:

Solution to poly(x) = 0, x = 4.641589, f(4.641589) = 0.000000