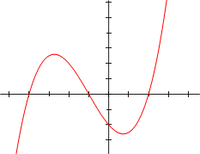
**דוגמא לפתרון "סגור" (או אנליטי) של בעיה רציפה:**

**מציאת פתרונות של משוואה מדרגה שלוש**

מציאת פתרון למשוואה מדרגה שלוש הינה מציאת השורשים הממשיים של המשוואה

***a1x3 + a2x2 +a3x + a4 = 0***

למשוואה הזו יש או פתרון ממשי אחד או שלוש פתרונות ממשיים. העובדה הזו נובעת ממשפט ערך הביניים וממשפטים אחרים על פולינומים.



ידוע שלמציאת שורשים לפולינומים מדרגה 5 ומעלה אין פתרון אנליטי ועד 4 יש, אם כי הפתרונות לדרגות 3 ו-4 הן די מסובכות. כולם מכירים את הפתרון למשוואת ריבועיות.

את הפתרון הבא (שיטת קרדאנו) למשוואות משולשות, אחת הפשוטות יחסית, נציג באופן הבא:

בהנחה ש- a1 שונה מאפס, אפשר לפשט ע"י חלוקת המשוואה ב- a1לצורה:

***x3 + ax2 +bx + c = 0***

אפשר "להפטר" מהמקדם הריבועי בנוסחה ע"י ההעתקה הלינארית

***x = t – a/3***

אם נציב אותה למשוואה במקום x נקבל:

***(t-a/3)3 +a (t-a/3)2 +b (t-a/3) + c = 0***

אחרי פתיחת סוגריים וסידור לפי t נקבל:

***t3 + pt + q = 0 (\*)***

כאשר

***p = b – a2/3***

***q = c + (2a3 – 9ab)/27***

עכשיו יש לנו משווה לפי t אם נפתור אותה נקבל את ערכי ה-x והמשוואה ב-t איננה מכילה גורם שלt2 .

לדוגמא, אם הפולינום המקורי שלנו הוא

***2x3-30x2+162x-350 = 0***

אחרי חלוקה ב-2

***x3-15x2+81x-175 = 0***

***a/3 = -15/3 =-5***

בעזרת ההשמה

***t = x+5***

נקבל

***t3 + 6t =20***

משוואה מדרגה 3 בצורה הזו ניתן לפתור בדרך אנליטית ע"י רדוקציה למשוואה ריבועית.

השיטה היא כלהלן: מצא 2 מספריםu,v המקיימים

***3uv = p (1)***

***u3 – v3 = q (2)***

ברגע שנמצא כאלי, אזי t=u-v הוא פתרון משוואה (\*). אפשר לוודא זאת ע"י הצבה במקום t, p ו-q ב-(\*).

***t3 +pt + q = (u-v)3 +3uv(u-v)*** + ***u3 – v3*** = ***0***

השאלה היא איך למצוא את u ו-v. נחלץ את v ממשוואה (1):

***v = p/3u***

נציב אותה במקום s ב-(2):

***u3 – p3/(27u3) = q***

ע"י פעולות אלגבריות נקבל

***u6 – q u3 - p3/27 = 0 (\*\*)***

קבלנו משוואה ריבועית ב-3u, נסמן ,u3 = w נקבל

***w2 –q w – p3/27 =0***

נניח שאותו ניתן לפתור בדרך הרגילה, ולחשב שורש שלישי מהפתרון

***u = w1/3***

מכאן אפשר לחשב את u, כל אחד מהפתרונות יכול לשמש אותנו, ולחלץ את v מתוך

את v אפשר לחלץ מתוך

***v3 = q + w***

***v = (q+w)1/3***

מכאן נחלץ את t ו-\*x מתוך התנאים

***t =u –v***

***x\* = t – a/3 (\*\*\*)***

לדוגמא, עבור

***t3 + 6t =20***

נקבל

***3uv = 6 (1)***

***u3 – v3 = 20 (2)***

***v = 6/(3u)=2/u***

***u6 -20u3-8=0***

***w2 -20w-8=0***

***w = u3 = 10+√108 = 20.3923048***

***u = 2.7320508***

***v3 = -10+√108 = 0. 3923048***

***v = 0. 7320508***

***t = u-v = 2.0***

***x\* = t+5 = 7.0***

x\* המחושב ב-(\*\*\*) חייב להיות ממשי, וברגע שיש לנו פתרון אנחנו יכולים לחלק את הפולינום המקורי ב-(x-x\*) ולקבל פולינום ריבועי, אותו אנחנו יכולים לפתור לפי הנוסחה הרגילה. את החלוקה נעשה לפי האלגוריתם הידוע לכך לדוגמא

***(x3-15x2+81x-175)//(x-7) = x2-8x + 25***

הערות:

- הערך x שיחושב בצורה כזו תמיד יהיה ממשיי. אולם גם אם אנחנו מחפשים רק שורשים ממשייים או שלמשוואה המקורית יש רק שורשים ממשיים עדיין יתכן של-(\*\*) יהיו מורכבים, וגם החישוב של שורש שלישי שלw ו-B+w יצטרך להביא בחשבון אריתמטיקה של מספרים מורכבים. שורש שלישי של מספר מורכב ניתן לחשב לפי הנוסחה

z1/3 = r1/3(cos((θ+2π)/3) +i sin((θ+2π)/3))

* ישנם 2 מקרים פרטיים שצריך לחשב בנפרד משום שהשיטה הכללית אינה מביאה אותם בחשבון:

1. אם p = 0 וגם q = 0 אזי –a/3 x= הוא פתרון ממשי מסדר שולש של המשוואה המקורית

2. אם p = 0 ו- q ≠ 0 אזי u = q1/3.

התוכנית המממשת את הפתרון האנליטי:

#define EPS 0.0000000000001

#define PI 3.14159265

int solve\_quadratic(double a, double b,

double c,

double \*r1, double \*i1,

double \*r2, double \*i2)

{

double delta, delta\_root, xv, qv;

if ( a == 0)

if ( b == 0)

return 0;

else

{

\*r1 = -c/b;

\*i1 = 0;

return 1;

} /\* else \*/

xv = -b/(2\*a); /\* Take care of square quad \*/

qv = a\*xv\*xv + b\*xv + c;

if(fabs(qv) < EPS)

{

\*r1 = \*r2 = xv;

\*i1 = \*i2 = 0.0;

return 2;

} /\* if \*/

delta = b\*b - 4\*a\*c;

if ( delta < 0.0 )

{

\*r1 = \*r2 = -b/(2\*a); /\* real part only \*/

delta\_root = sqrt(-delta);

\*i1 = - delta\_root/(2.0\*a);

\*i2 = delta\_root/(2.0\*a);

return 0;

} /\* if \*/

else

{

delta\_root = sqrt(delta);

\*r1 = (-b - delta\_root)/(2.0\*a);

\*r2 = (-b + delta\_root)/(2.0\*a);

\*i1 = 0;

\*i2 = 0;

return 2;

} /\* else \*/

} /\* solve\_quadratic \*/

void divide\_cubic(double a, double \*b,

double \*c, double x)

{

\*b = \*b + a\*x;

\*c = \*c + (\*b)\*x;

} /\* divide\_cubic \*/

void cubic\_root(double x, double y, double \*r, double \*i )

{

double radius, theta, temp;

if ( x == 0.0 && y == 0.0)

{

\*r = \*i = 0.0;

return;

} /\* if \*/

radius = sqrt(x\*x + y\*y);

theta = (asin(fabs(y/radius)));

temp = exp(log(radius)/3.0);

if ((x < 0)&&(y>=0))

theta = 3\*PI - theta;

else

if ((x<0)&&(y<=0))

theta = 3\*PI + theta;

else

if ((x>=0)&&(y<0))

theta=4\*PI-theta;

\*r = temp\*cos(theta/3.0);

\*i = temp\*sin(theta/3.0);

} /\* cubic\_root \*/

int solve\_cubic(double a1, double a2,

double a3, double a4,

double \*x1, double \*x2, double \*x3)

{

double p, q;

double a, b, c;

double u1, u2, wr,wi;

double x, t, ur, ui, vr, vi;

double b2, c2;

double i1, i2;

int n;

if ( a1 == 0.0)

{

n = solve\_quadratic(a2, a3, a4, x1, &i1, x2, &i2);

return n;

} /\* if \*/

a = a2/a1;

b = a3/a1;

c = a4/a1;

p = b - a\*a/3.0;

q = c + (2.0\*a\*a\*a - 9.0\*a\*b)/27.0 ;

if ( ( p==0.0 ) && (q == 0.0))

{

\*x1 = \*x2 = \*x3 = -a/3;

return 3;

} /\* if \*/

if (p == 0.0) /\* q != 0 \*/

{

cubic\_root(q,0, &ur, &ui);

x = ur - (a/3.0);

\*x1 = x;

b2 = a;

c2 = b;

divide\_cubic(1.0, &b2, &c2, x);

n = solve\_quadratic(1.0, b2, c2, x2,&i1, x3, &i2);

return n+1;

} /\* if \*/

solve\_quadratic(1.0, -q, -((p\*p\*p)/27.0), &u1, &i1, &u2, &i2);

if (u1 > u2)

{

wr = u1;

wi = i1;

} /\* if \*/

else

{

wr = u2;

wi = i2;

} /\* else \*/

cubic\_root(wr, wi, &ur, &ui);

cubic\_root(-q + wr, wi, &vr, &vi);

t = vr - ur;

x = t - (a/3.0);

\*x1 = x;

b2 = a;

c2 = b;

divide\_cubic(1.0, &b2, &c2, x);

n = solve\_quadratic(1.0, b2, c2, x2,&i1, x3, &i2);

return n+1;

} /\* solve\_cubic \*/

**הגרסה הבאה של התוכנית מדווחת על כל הפתרונות (גם הקומפלקסיים):**

#define EPS 0.0000000000001

#define PI 3.14159265

int solve\_quadratic(double a, double b,

double c,

double \*r1, double \*i1,

double \*r2, double \*i2)

{

double delta, delta\_root, xv, qv;

if ( a == 0)

if ( b == 0)

return 0;

else

{

\*r1 = -c/b;

\*i1 = 0;

return 1;

} /\* else \*/

xv = -b/(2\*a); /\* Take care of square quad \*/

qv = a\*xv\*xv + b\*xv + c;

if(fabs(qv) < EPS)

{

\*r1 = \*r2 = xv;

\*i1 = \*i2 = 0.0;

return 2;

} /\* if \*/

delta = b\*b - 4\*a\*c;

if ( delta < 0.0 )

{

\*r1 = \*r2 = -b/(2\*a); /\* real part only \*/

delta\_root = sqrt(-delta);

\*i1 = - delta\_root/(2.0\*a);

\*i2 = delta\_root/(2.0\*a);

return 0;

} /\* if \*/

else

{

delta\_root = sqrt(delta);

\*r1 = (-b - delta\_root)/(2.0\*a);

\*r2 = (-b + delta\_root)/(2.0\*a);

\*i1 = 0;

\*i2 = 0;

return 2;

} /\* else \*/

} /\* solve\_quadratic \*/

void divide\_cubic(double a, double \*b,

double \*c, double x)

{

\*b = \*b + a\*x;

\*c = \*c + (\*b)\*x;

} /\* divide\_cubic \*/

void cubic\_root(double x, double y, double \*r, double \*i )

{

double radius, theta, temp;

if ( x == 0.0 && y == 0.0)

{

\*r = \*i = 0.0;

return;

} /\* if \*/

radius = sqrt(x\*x + y\*y);

theta = (asin(fabs(y/radius)));

temp = exp(log(radius)/3.0);

if ((x < 0)&&(y>=0))

theta = 3\*PI - theta;

else

if ((x<0)&&(y<=0))

theta = 3\*PI + theta;

else

if ((x>=0)&&(y<0))

theta=4\*PI-theta;

\*r = temp\*cos(theta/3.0);

\*i = temp\*sin(theta/3.0);

} /\* cubic\_root \*/

int solve\_cubic(double a1, double a2,

double a3, double a4,

double \*x1, double \*y1,

double \*x2, double \*y2,

double \*x3, double \*y3)

{

double p, q;

double a, b, c;

double u1, u2, wr,wi;

double x, t, ur, ui, vr, vi;

double b2, c2;

double i1, i2;

int n;

if ( a1 == 0.0)

{

n = solve\_quadratic(a2, a3, a4, x1, &i1, x2, &i2);

return n;

} /\* if \*/

a = a2/a1;

b = a3/a1;

c = a4/a1;

p = b - a\*a/3.0;

q = c + (2.0\*a\*a\*a - 9.0\*a\*b)/27.0 ;

if ( ( p==0.0 ) && (q == 0.0))

{

\*x1 = \*x2 = \*x3 = -a/3;

\*y1 = \*y2 = \*y3 = 0;

return 3;

} /\* if \*/

if (p == 0.0) /\* q != 0 \*/

{

cubic\_root(q,0, &ur, &ui);

x = ur - (a/3.0);

\*x1 = x;

\*y1 = 0;

b2 = a;

c2 = b;

divide\_cubic(1.0, &b2, &c2, x);

n = solve\_quadratic(1.0, b2, c2, x2,y2, x3, y3);

return n+1;

} /\* if \*/

solve\_quadratic(1.0, -q, -((p\*p\*p)/27.0), &u1, &i1, &u2, &i2);

if (u1 > u2)

{

wr = u1;

wi = i1;

} /\* if \*/

else

{

wr = u2;

wi = i2;

} /\* else \*/

cubic\_root(wr, wi, &ur, &ui);

cubic\_root(-q + wr, wi, &vr, &vi);

t = vr - ur;

x = t - (a/3.0);

\*x1 = x;

\*y1 = 0;

b2 = a;

c2 = b;

divide\_cubic(1.0, &b2, &c2, x);

n = solve\_quadratic(1.0, b2, c2, x2,y2, x3, y3);

return n+1;

} /\* solve\_cubic \*/