

DOMACI RAD 3

Newtonova metoda i metoda bisekcije

1. Zadatak

U prvom zadatku programski je ostvarena realizacije metode bisekcije pomoću koje pronalazimo nultočke funkcije. Konkretno odabir nultočaka ove metode se svodi na crtanje grafa, pronalaska odgovarajućih intervala (procjena) u kojima se nalazi nultočka, te odabrane točke (intervale) proslijediti programu koji pronalazi nultočke između danih intervala ukoliko postoje.

U ovom zadatku se tražio pronalazak nultočaka za jednadžbu :

$$f(\theta) \equiv \frac{2V_0^2 \cos \theta \sin \theta}{g} - d = 0.$$

Ukoliko uzmemo podatke :

$$V_0 = 100 \text{ m/s}$$

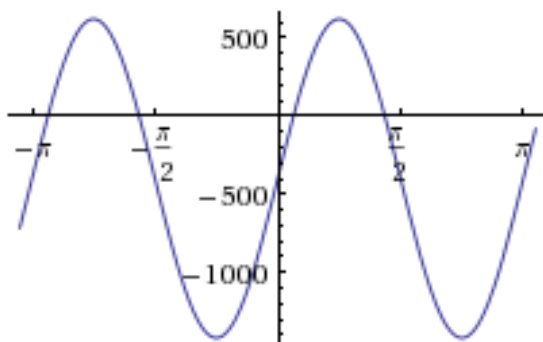
$$g = 9.81 \text{ m/s}^2$$

$$d = 400 \text{ m}$$

$$\frac{100 \times 100 \sin(2x)}{9.81} - 400$$

dobijemo jednadžbu oblika :

gdje je $2\sin x \cos x = \sin 2x$. a graf za danu jednadžbu izgleda :



Ako malo bolje pogledamo graf vidimo kako prva nultočka ulazi u intervalu : $[0, \pi/8]$ dok se druga nultočka nalazi u intervalu između : $[3\pi/8, \pi/2]$. Također kako se radi o sinusnoj funkciji i kako je kosi hitac u pitanju ograničio sam se isključivo na prvi kvadrant.

Programski kod:

```
#include <stdio.h>
#include <math.h>

#define POJETNA_BRZINA 100
#define TOLERANCIJA 0.000005
#define AKCELERACIJA_SLOBODNOG_PADA 9.81
#define UDALJENOST 400

struct data
{
    double nulTocka;
    int brojIteracija;
};

double funkcija (double x)
{
    return (((POJETNA_BRZINA*POJETNA_BRZINA)/AKCELERACIJA_SLOBODNOG_PADA) * sin(2*x))
    - UDALJENOST;
}

void bisekcija(double x1,double x2,struct data *Temp)
{
    double pomocna;
    Temp->brojIteracija = 0;
    if( funkcija(x1) * funkcija(x2) > 0) // Ako ne alterniraju u predznaku
        return;

    while(abs((x2-x1)) > TOLERANCIJA)
    {
        Temp->brojIteracija++;
        pomocna = (x1+x2)/2;

        if( funkcija(x1) * funkcija(pomocna) < 0)
            x2 = pomocna;

        else
            x1 = pomocna;
    }

    Temp->nulTocka = pomocna;
}

void main(int argc, char *argv[])
{
    double x1,x2,x3,x4;
    struct data rezultat;
    x1 = 0;
    x2 = 0.3927;
    x3 = 1.1781;
    x4 = 1.5707;

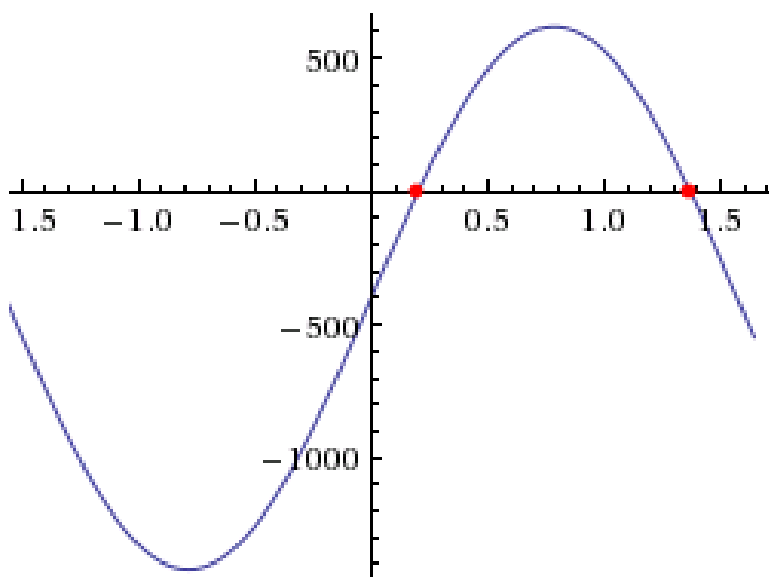
    bisekcija(x1,x2,&rezultat);
    printf("Za interval [0, 0.3927] imamo %d iteracija, a nultocka je:
    %f\n\n",rezultat.brojIteracija,rezultat.nulTocka);

    bisekcija(x3,x4,&rezultat);
    printf("Za interval [1.1781, 1.5707] imamo %d iteracija, a nultocka je:
    %f\n\n",rezultat.brojIteracija,rezultat.nulTocka);
}
```

REZULTAT:

```
C:\WINDOWS\system32\cmd.exe
Za interval [0, 0.3927] imamo 17 iteracija, a nultocka je: 0.201620
Za interval [1.1781, 1.5707] imamo 17 iteracija, a nultocka je: 1.369179
Press any key to continue . . . _
```

Ukoliko u nekom od programa (wolfram alpha) nacrtamo naš graf vidjet cemo kako stvarno ovaj proračun daje približno točne rezultate, što se vidi na sljedećoj slici:



2. Zadatak

U ovoj metodi imamo sličan pristup kao i kod metode bisekcije (određivanje intervala iz grafa) samo što je postupak traženja nultočka drugačiji nego kod metode bisekcije.

Prvo trebamo izračunati prvu i drugu derivaciju funkcije, te za traženje nultočka trebamo izabrati jednu rubnu točku danog intervala po kriteriju:

$$f(\varphi_0) * f''(\varphi_0) > 0$$

U mom primjeru za prvi interval [0, 0.3927] odabirem točku 0.1 jer vrijedi

$f(0.1) * f''(0.1) > 0$, dok je za rubnu točku 0.3927 $\Rightarrow f(0.3927) * f''(0.3927) < 0$ što ne ispunjava gornji uvjet. Također nije odabrana rubna točka 0 jer je $f''(0) = 0$.

Za drugi interval [1.1781, 1.5707] odabirem točku 1.5707 jer je za nju ispunjen gornji uvjet, dok za točku 1.1781 nije.

Nakon što smo odabrali rubne točke njih proslijeđujemo programu koji pronalazi nultočke.

Programski kod:

```
#include<stdio.h>
#include<math.h>

#define PO CETNA_BRZINA 100
#define TOLERANCIJA 0.000005
#define AKCELERACIJA_SLOBODNOG_PADA 9.81
#define UDALJENOST 400

struct data
{
    double nulTocka;
    int brojIteracija;
};

double funkcija (double x)
{
    return (((POCETNA_BRZINA*POCETNA_BRZINA)/AKCELERACIJA_SLOBODNOG_PADA) * sin(2*x))
    - UDALJENOST;
}

double derivacija(double x)
{
    return 2 * ((POCETNA_BRZINA*POCETNA_BRZINA)/AKCELERACIJA_SLOBODNOG_PADA) *
    cos(2*x);
}
```

```

void newton(double x0, struct data *Temp)
{
    Temp->brojIteracija = 0;

    while(abs(funkcija(x0)) > TOLERANCIJA)
    {
        Temp->brojIteracija++;
        x0 = x0 - funkcija(x0) / derivacija(x0);
    }
    Temp->nulTocka = x0;
}

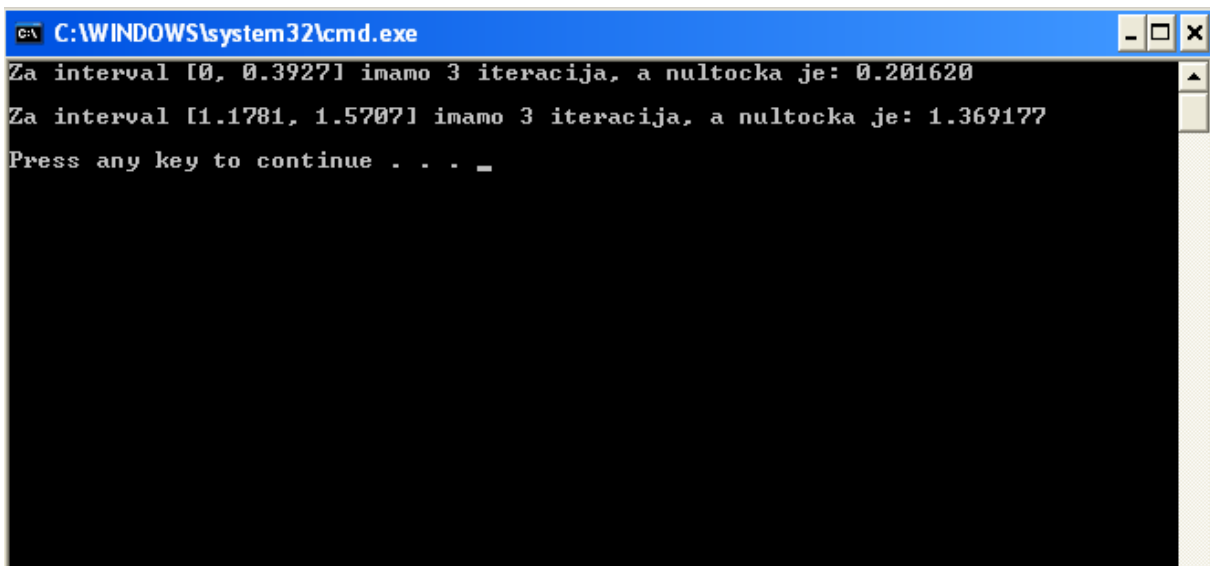
void main()
{
    double x1,x2;
    struct data rezultat;
    x1 = 0.1;
    x2 = 1.5707;

    newton(x1,&rezultat);
    printf("Za interval [0, 0.3927] imamo %d iteracija, a nultocka je:
    %f\n\n",rezultat.brojIteracija,rezultat.nulTocka);

    newton(x2,&rezultat);
    printf("Za interval [1.1781, 1.5707] imamo %d iteracija, a nultocka je:
    %f\n\n",rezultat.brojIteracija,rezultat.nulTocka);
}

```

Rezultati:



```

C:\WINDOWS\system32\cmd.exe
Za interval [0, 0.3927] imamo 3 iteracija, a nultocka je: 0.201620
Za interval [1.1781, 1.5707] imamo 3 iteracija, a nultocka je: 1.369177
Press any key to continue . . . _

```

Možemo primjetiti kako u ovom primjeru za razliku od metode bisekcije imamo puno manje koraka tj. broja iteracija potrebnih za pronalaženje nultočaka funkcije.

Referenca: Za grafove sam koristio program "Wolfram Alpha", dok sam za generiranje matematičkih formula koristio LaTeX online generator:

<http://www.codecogs.com/latex/eqneditor.php>