Politechnika Śląska **Gliwice**
Wydział Automatyki Elektroniki i Informatyki **Rok akademicki 2007/2008**
Kierunek: Automatyka i Robotyka sem.3 **Semestr zimowy**

Metody Numeryczne
Laboratorium

Ćw.9 : Przybliżone rozwiązywanie równań nieliniowych

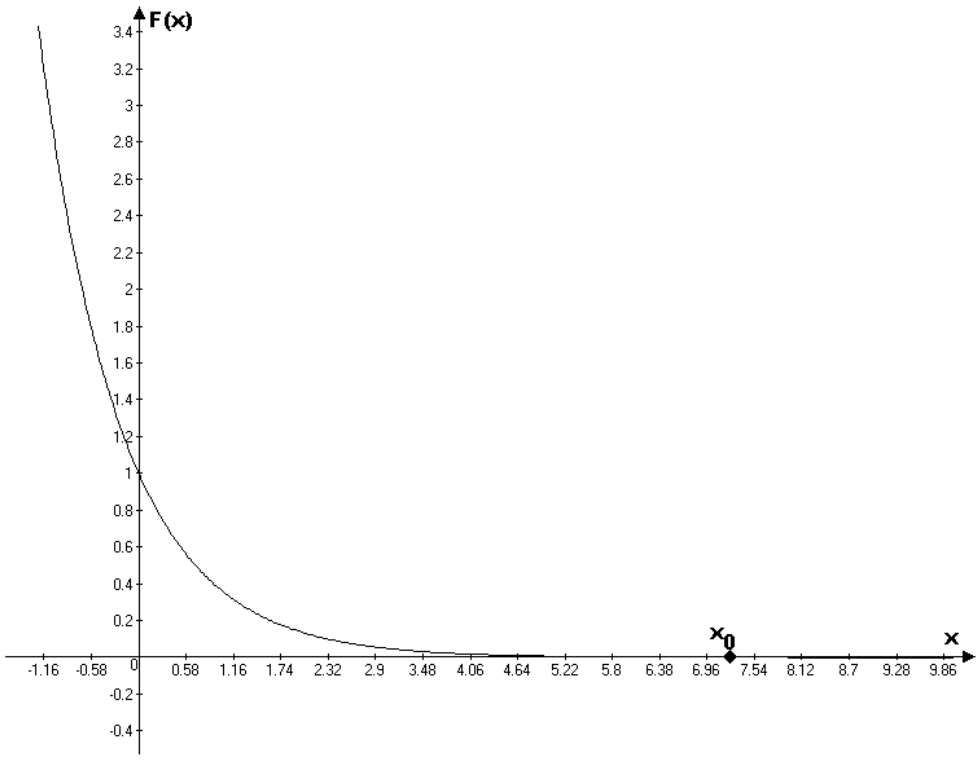
Wykonali:
Adam DZIENDZIEL
Adrian BIELEC

Grupa 4 Sekcja 1

Data odbycia ćwiczenia:
30.11.07

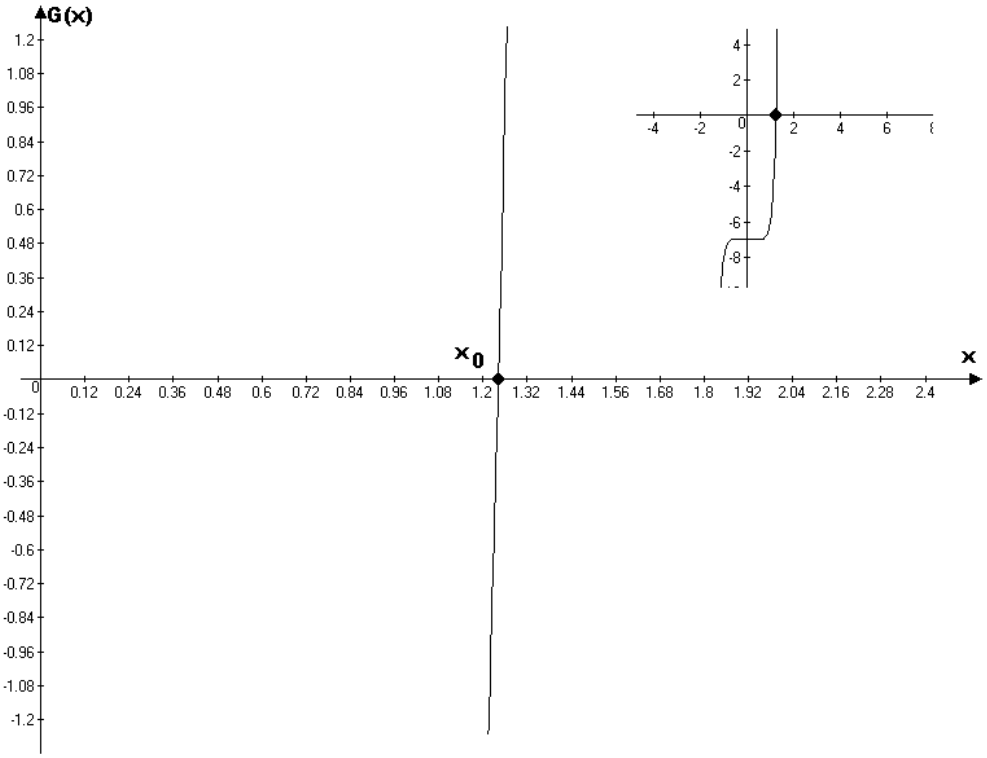
Funkcja I (wykres łagodny, małe nachylenie wykresu względem osi OX α → 0II)

$F(x) = e^{-x} - \sin(x/10000)$ $F(x_0) = 0 \rightarrow x_0 = 7.23185$ szukane w <4;10>



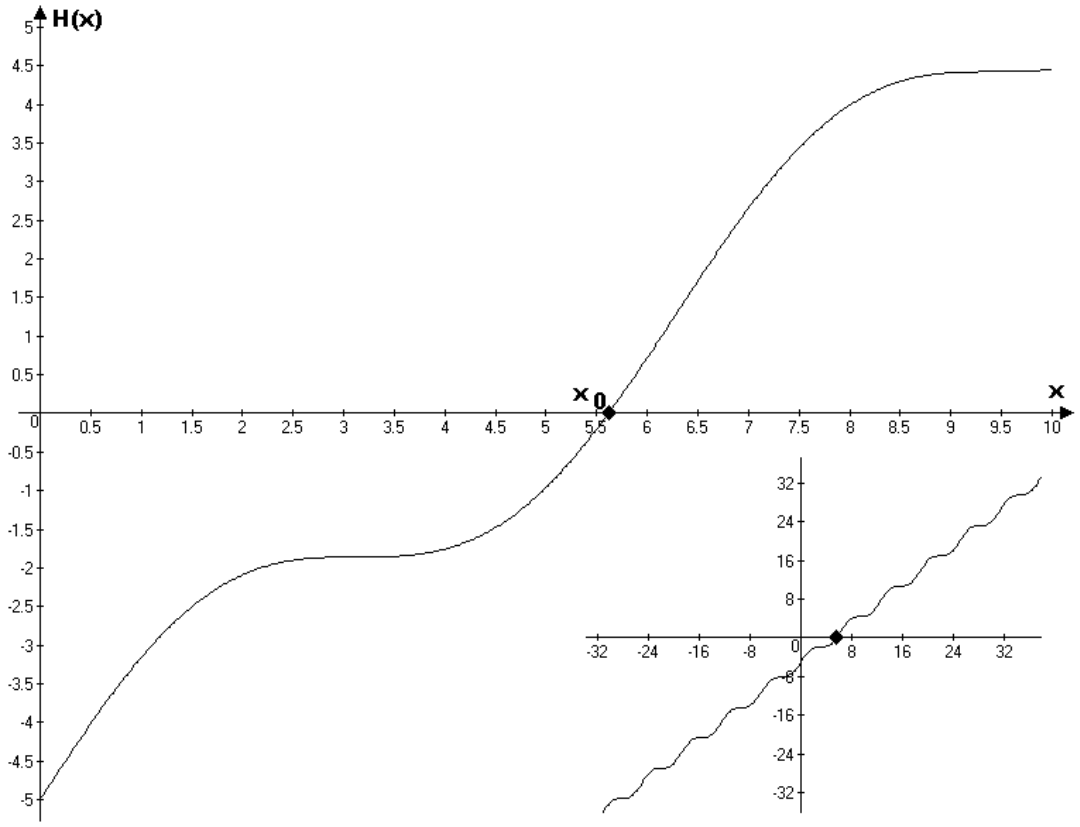
Funkcja II (wykres ostry, duże nachylenie wykresu względem osi OX α → II/2)

$G(x) = x^9 - 7$ $G(x_0) = 0 \rightarrow x_0 = 1.24137$ szukane w <1;5>



Funkcja III (optymalne nachylenie wykresu względem osi OX α → 0Π/4)

H(x) = x + sin(x) - 5 $H(x_0) = 0 \rightarrow x_0 = 5.61756$ szukane w <4;10>



W trakcie testowania programów będziemy stosować 2 sposoby stopu (zakończenia procesu iteracji) . Oto one:

$$\mathbf{S_1} \rightarrow |x_i - x_j| < \xi \; ; \; j = i-1$$
$$\mathbf{S_2} \rightarrow |f(x_i)| < \xi$$

S_2 będzie stosowany tylko dla metody stycznych i siecznych a S_1 do wszystkich trzech.

Metoda połowienia przedziałów (S_1)		F(x)		G(x)		H(x)	
		Il. iteraacji	Błąd	Il. iteraacji	Błąd	Il. iteraacji	Błąd
ξ	0.1	6	0.0493999	6	0.05387	6	0.0238099
	0.01	10	0.00333452	9	0.0008175	10	0.00548697
	0.001	13	0.000327587	12	0.000159063	13	0.000360012
	0.0001	16	5.292×10^{-5}	16	2.404×10^{-5}	16	8.535×10^{-5}
	0.00001	20	1.430×10^{-6}	19	1.154×10^{-6}	20	9.430×10^{-6}

Metoda siecznych (S_1)		F(x)		G(x)		H(x)	
		Il. iteraacji	Błąd	Il. iteraacji	Błąd	Il. iteraacji	Błąd
ξ	0.1	12	0.584549	2	0.241345	4	0.0359964
	0.01	28	0.0497265	2	0.241345	12	0.0038271
	0.001	42	0.00528288	2	0.241345	20	0.000416756
	0.0001	56	0.000553608	2	0.241345	29	2.908×10^{-5}
	0.00001	71	4.577×10^{-5}	7234	0.160322	37	1.430×10^{-6}

Metoda stycznych (S_1)		F(x)		G(x)		H(x)	
		Il. iteraacji	Błąd	Il. iteraacji	Błąd	Il. iteraacji	Błąd
ξ	0.1	6	2.765×10^{-5}	13	0.00944051	2	0.000237942
	0.01	6	2.765×10^{-5}	14	0.000276109	8	4.768×10^{-6}
	0.001	7	3.814×10^{-6}	15	3.93×10^{-6}	9	4.768×10^{-6}
	0.0001	7	3.814×10^{-6}	16	3.93×10^{-6}	9	4.768×10^{-6}
	0.00001	8	3.814×10^{-6}	16	4.182×10^{-7}	10	4.768×10^{-6}

Metoda siecznych (S_2)		F(x)		G(x)		H(x)	
		Il. iteraacji	Błąd	Il. iteraacji	Błąd	Il. iteraacji	Błąd
ξ	0.1	1	2.46462	>10000	0.134081	3	0.0458302
	1	1	2.46462	>10000	0.134081	11	0.00502357
	0.001	1	2.46462	>10000	0.134081	19	0.000538632
	0.0001	22	0.128376	>10000	0.134081	28	4.940×10^{-5}
	0.00001	37	0.0117911	>10000	0.134081	36	9.786×10^{-6}

Metoda stycznych (S_2)		F(x)		G(x)		H(x)	
		Il. iteraacji	Błąd	Il. iteraacji	Błąd	Il. iteraacji	Błąd
ξ	0.1	1	2.259	14	0.000276109	34	0.0238421
	0.01	1	2.259	15	3.93×10^{-6}	35	9.051×10^{-5}
	0.001	3	0.583781	15	3.93×10^{-6}	35	9.051×10^{-5}
	0.0001	5	0.00737995	15	3.93×10^{-6}	36	4.993×10^{-6}
	0.00001	5	0.00737995	16	4.182×10^{-7}	36	4.993×10^{-6}

Wpływ numerycznego różniczkowania na metodę stycznych (pierwsza i druga pochodna liczone numerycznie)

Metoda stycznych (S_1)		F(x)		G(x)		H(x)	
		Il. iteraacji	Błąd	Il. iteraacji	Błąd	Il. iteraacji	Błąd
ξ	0.1	6	2.813×10^{-5}	13	0.01044051	2	0.000237942
	0.01	6	2.813×10^{-5}	14	0.000276109	8	4.771×10^{-6}
	0.001	7	3.934×10^{-6}	15	3.94×10^{-6}	9	4.771×10^{-6}
	0.0001	7	3.934×10^{-6}	16	3.94×10^{-6}	9	4.771×10^{-6}
	0.00001	8	3.934×10^{-6}	16	4.191×10^{-7}	10	4.771×10^{-6}

Metoda stycznych (S_2)		F(x)		G(x)		H(x)	
		Il. iteraacji	Błąd	Il. iteraacji	Błąd	Il. iteraacji	Błąd
ξ	0.1	1	2.259	14	0.000287109	34	0.0238421
	0.01	1	2.259	15	3.94×10^{-6}	35	5.012×10^{-5}
	0.001	3	0.583781	15	3.94×10^{-6}	35	5.012×10^{-5}
	0.0001	5	0.00737995	15	3.94×10^{-6}	36	5.007×10^{-6}
	0.00001	5	0.00737995	16	4.187×10^{-7}	36	5.007×10^{-6}

Jak widać wpływ różniczkowania numerycznego na wyniki poszukiwań pierwiastków jest bardzo mały. Różnice w błędach pojawiają się dopiero na 4-5 miejscu po przecinku. Należy pamiętać jednak że różnice tę będą zależały od odpowiedniego doboru współczynników n (ilości podziałów) i h (szerokość pojedynczego przedziału) dla algorytmu różniczkowania oraz od wykresu.

Modyfikacja metody siecznych (S_1)		F(x)		G(x)		H(x)	
		Il. iteraacji	Błąd	Il. iteraacji	Błąd	Il. iteraacji	Błąd
ξ	0.1	8 / 12	2.894×10^{-6}	2 / 2	0.241345	3 / 4	0.0790738
	0.01	8 / 28	2.894×10^{-6}	2 / 2	0.241345	4 / 12	4.994×10^{-6}
	0.001	8 / 42	2.894×10^{-6}	2 / 2	0.241345	4 / 20	4.994×10^{-6}
	0.0001	9 / 56	3.234×10^{-6}	2 / 2	0.241345	4 / 29	4.994×10^{-6}
	0.00001	9 / 71	3.234×10^{-6}	2/ 7234	0.241345	4 / 37	4.994×10^{-6}

po ukośniku ilość iteracji w klasycznej metodzie siecznych

Jak widać dzięki modyfikacji metody siecznych dostajemy wyniki o zdecydowanie lepszej dokładności w o wiele mniejszej liczbie kroków gdy funkcja jest płaska, natomiast gdy funkcja jest nachylona do osi x pod dużym kątem modyfikacja nie przynosi znacznych korzyści ale dla wyższych dokładności obliczeń jest o wiele bardziej stabilna. Dla wykresu pośredniego zachowanie jest proporcjonalne dla nachylenia wykresu. Różnica w liczbie kroków jest wyraźniejsza wtedy gdy weźmiemy szerszy przedział poszukiwania pierwiastka.

Program 1 (pochodne liczone analitycznie)

```
#include<iostream>
#include<math.h>
#include<conio.h>
#include<stdlib.h>
using namespace std;

double funkcja(double x)
 {
 double y;
//y=1/exp(x)-sin(x/10000); //łagodna
//y=pow(x,9)-7; //ostra
 y=x+sin(x)-5; //pomiędzy
 return y;
 }

double pochodna1(double x)
 {
  double y;
//y=-1/exp(x)-cos(x/10000)/10000; //łagodna
//y=9*pow(x,8); //ostra
  y=cos(x)+1; //pomiędzy
  return y;
 }

double pochodna2(double x)
 {
  double y;
//y=1/exp(x)+sin(x/10000)/100000000; //łagodna
//y=72*pow(x,7);//ostra
  y=-sin(x); //pomiędzy
  return y;
 }

int main()
 {
  int krok=0;
  double a,aa,b,bb,x,eps,x0,x1;

  cout<<"W jakim przedziale znajduja sie pierwiastki ???"<<endl<<endl;
  cout<<"       a = "; cin>>a;
  cout<<"       b = "; cin>>b;
  cout<<endl<<"Podaj dokladnosc obliczen >>> "; cin>>eps;
//x0=7.23185; //1 wykres łagodny
//x0=1.24137; //2 wykres ostry
  x0=5.61756; //3 wykres posredni



  if(funkcja(a)*funkcja(b)<0)
  {
   aa=a;
   bb=b;

//metoda połowienia przedziałów

   do
   {
    krok++;
    x1=x;
```

```
    x=(a+b)/2.0;
    if(funkcja(a)*funkcja(x)<0) b=x;
    else a=x;
   }while(fabs(x-x1)>=eps && krok<10000);

   cout<<endl<<endl<<"Metoda polowienia przedzialow : "<<endl;
   cout<<endl<<"miejsce zerowe z tego przedzialu to >>> "<<x<<endl;
   cout<<"1 obliczone w "<<krok<<" krokow"<<endl;
   cout<<"blad >>> "<<fabs(x0-x)<<endl;

//metoda siecznych

   krok=0;
   a=aa;
   b=bb;
   if(funkcja(a)*pochodna2(a)>0)
   {
//a punktem stalym
   do
   {
    krok++;
    x1=x;
    x=a-((funkcja(a)*(b-a))/(funkcja(b)-funkcja(a)));
    b=x;
   }while(fabs(x-x1)>=eps && krok<10000);
//}while(fabs(funkcja(x))>=eps && krok<10000);
   }else
    {
//b punktem stalym
    do
     {
      krok++;
      x1=x;
      x=a-((funkcja(a)*(b-a))/(funkcja(b)-funkcja(a)));
      a=x;
     }while(fabs(x1-x)>=eps && krok<10000);
//}while(fabs(funkcja(x))>=eps && krok<10000);
     }
    cout<<endl<<endl<<"Metoda siecznych : "<<endl;
    cout<<endl<<"miejsce zerowe z tego przedzialu to >>> "<<x<<endl;
    cout<<"2 obliczone w "<<krok<<" krokow"<<endl;
    cout<<"blad >>> "<<fabs(x0-x)<<endl;

//metoda stycznych

    krok=0;
    a=aa;
    b=bb;
    if(funkcja(a)*pochodna2(a)>0)
    {
//a punktem początkowym
    do
    {
     krok++;
     x1=x;
     x=a-funkcja(a)/pochodna1(a);
     a=x;
    }while(fabs(x1-x)>=eps && krok<10000);
//}while(fabs(funkcja(x))>=eps && krok<10000);
    }else
   {
```

```
//b punktem początkowym
   do
   {
    krok++;
    x1=x;
    x=b-funkcja(b)/pochodna1(b);
    b=x;
   }while(fabs(x1-x)>=eps && krok<10000);
//}while(fabs(funkcja(x))>=eps && krok<10000);
   }

    cout<<endl<<endl<<"Metoda stycznych : "<<endl;
    cout<<endl<<"miejsce zerowe z tego przedzialu to >>> "<<x<<endl;
    cout<<"3 obliczone w "<<krok<<" krokow"<<endl;
    cout<<"blad >>> "<<fabs(x0-x)<<endl;

//metoda siecznych modyfikacja

    krok=0;
    a=aa;
    b=bb;
    if(funkcja(a)*pochodna2(a)>0)
    {
//a punktem stalym
    do
    {
     krok++;
     x1=x;
     x=a-((funkcja(a)*(b-a))/(funkcja(b)-funkcja(a)));
     b=x;
     if(krok>=2)
     {
      a=x-((funkcja(x)*(x1-x))/(funkcja(x1)-funkcja(x)));
     }
    }while(fabs(x-x1)>=eps && krok<10000);
//}while(fabs(funkcja(x))>=eps && krok<10000);
    }else
    {
//b punktem staly
    do
    {
     krok++;
     x1=x;
     x=a-((funkcja(a)*(b-a))/(funkcja(b)-funkcja(a)));
     a=x;
     if(krok>=2)
     {
      b=x-((funkcja(x)*(x1-x))/(funkcja(x1)-funkcja(x)));
     }
    }while(fabs(x1-x)>=eps && krok<10000);
//}while(fabs(funkcja(x))>=eps && krok<10000);
    }
    cout<<endl<<endl<<"Metoda siecznych modyfikacja : "<<endl;
    cout<<endl<<"miejsce zerowe z tego przedzialu to >>> "<<x<<endl;
    cout<<"2 obliczone w "<<krok<<" krokow"<<endl;
    cout<<"blad >>> "<<fabs(x0-x)<<endl;

   }else
   cout<<"W tym przedziale nie miejsc zerowych lub jest ich kilka!!!"<<endl;
  getch();
 }
```

Program 2 (pierwsza i druga pochodna liczone numerycznie, metoda stycznych)

```
#include<iostream>
#include<math.h>
#include<conio.h>
#include<stdlib.h>
using namespace std;

double funkcja(double x)
 {
  double y;

//y=1/exp(x)-sin(x/10000); //łagodna
//y=pow(x,9)-7; //ostra
  y=x+sin(x)-5; //pomiędzy
  return y;
 }

float pochodna1(float x0)
 {
  int i,j,n;
  float a,x,h,fnx,y,eps;
  float **tab;

  h=0.12;
  n=9;
  eps=0.000000001;
  n++;

  tab=(float**)malloc(n*sizeof(float));
  for(i=0;i<n;i++)
  {
  tab[i]=(float*)malloc((n-i)*sizeof(float));
  }

  for(i=0;i<n;i++)
  {
   tab[0][i]=funkcja(x0+i*h);
  }

  for(i=1;i<n;i++)
  {
   for(j=0;j<n-i;j++)
   {
    tab[i][j]=tab[i-1][j+1]-tab[i-1][j];
   }
  }

  fnx=0;
  for(i=1;i<(n-1);i++)
  {
   fnx=fnx-((pow(-1,i)/float(i))*tab[i][0]);
  }
  fnx=fnx/h;

  return fnx;
 }
```

```
float pochodna2(float x0)
 {
  int i,j,n;
  float a,x,h,fnx,y,eps;
  float **tab;

  h=0.14;
  n=9;
  eps=0.000000001;
  n++;

  tab=(float**)malloc(n*sizeof(float));
  for(i=0;i<n;i++)
  {
  tab[i]=(float*)malloc((n-i)*sizeof(float));
  }

  for(i=0;i<n;i++)
  {
   tab[0][i]=funkcja(x0+i*h);
  }

  for(i=1;i<n;i++)
  {
   for(j=0;j<n-i;j++)
   {
    tab[i][j]=tab[i-1][j+1]-tab[i-1][j];
   }
  }

  fnx=(1/(h*h))*(tab[2][0]-tab[3][0]+(11*tab[4][0])/12-(10*tab[5][0])/12);

  return fnx;
 }
```

```
int main()
 {
  int krok=0;
  float a,aa,b,bb,x,eps,x0,x1;

  cout<<"W jakim przedziale znajduja sie pierwiastki ???"<<endl<<endl;
  cout<<"       a = "; cin>>a;
  cout<<"       b = "; cin>>b;
  cout<<endl<<"Podaj dokladnosc obliczen >>> "; cin>>eps;
//x0=7.23185; //1 wykres łagodny
//x0=1.24137; //2 wykres ostry
  x0=5.61756; //3 wykres posredni

  if(funkcja(a)*funkcja(b)<0)
  {
   aa=a;
   bb=b;

//metoda stycznych

   krok=0;
   a=aa;
   b=bb;
   if(funkcja(a)*pochodna2(a)>0)
   {
 //a punktem początkowym
    do
    {
     x1=x;
     krok++;
     x=a-funkcja(a)/pochodna1(a);
     a=x;
    }while(fabs(x-x1)>=eps && krok<10000);
 //}while(fabs(funkcja(x))>=eps && krok<5000);
    }else
    {
 //b punktem początkowym
    do
    {
     x1=x;
     krok++;
     x=b-funkcja(b)/pochodna1(b);
     b=x;
    }while(fabs(x-x1)>=eps && krok<10000);
 //}while(fabs(funkcja(x))>=eps && krok<5000);

   }
   cout<<endl<<endl<<"Metoda stycznych : "<<endl;
   cout<<endl<<"miejsce zerowe z tego przedzialu to >>> "<<x<<endl;
   cout<<"obliczone w "<<krok<<" krokow"<<endl;
   cout<<"z bledem "<<fabs(x-x0)<<"  "<<endl;


   }else
   cout<<"W tym przedziale nie miejsc zerowych lub jest ich kilka!!!"<<endl;

   getch();
 }
```