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 różniczkowych zwyczajnych

Wykonali:
Adam DZIENDZIEL
Adrian BIELEC

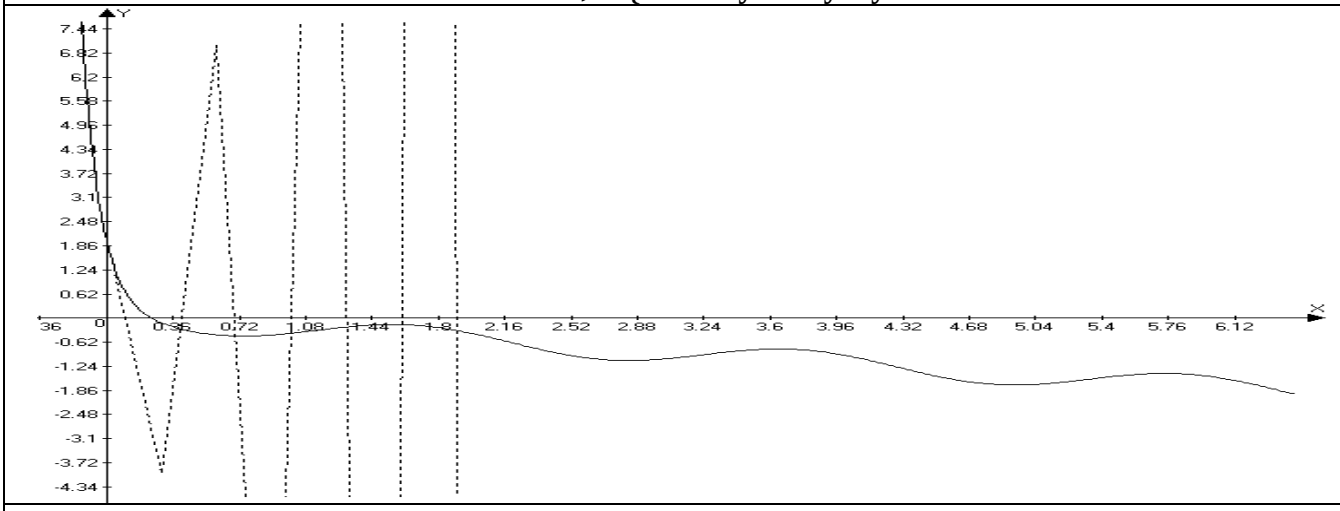
Grupa 4 Sekcja 1

Data odbycia ćwiczenia:
14.12.07

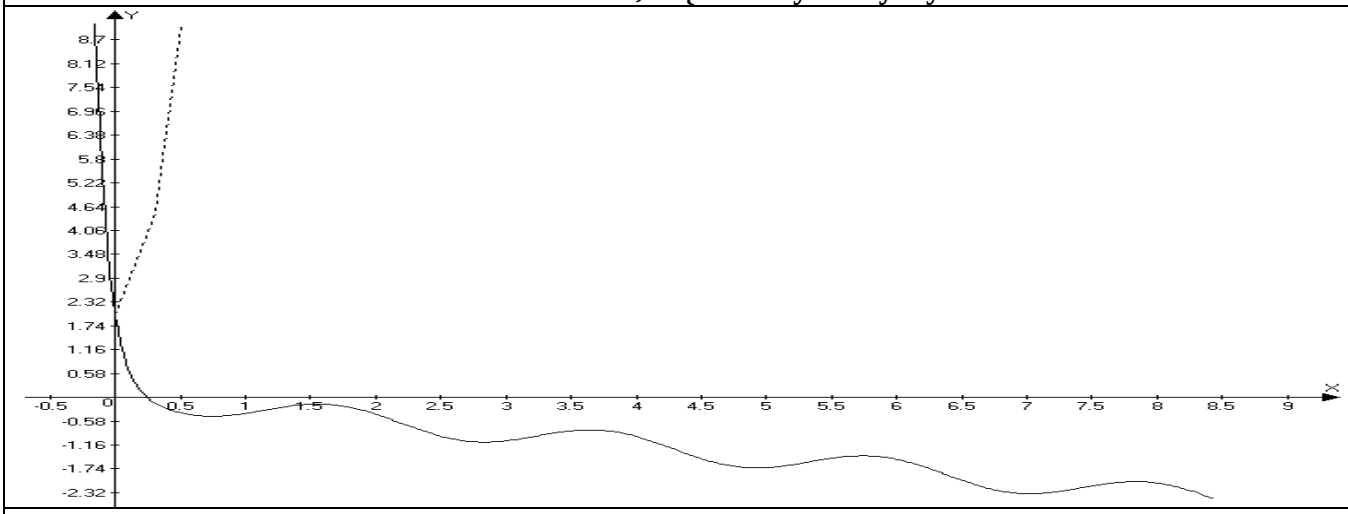
Rozwiązaliśmy równanie y' = - 10y - 3[x + sin(3x)] gdzie y(0)=2 i y=y(x) metodą przewidywań otrzymując równanie $y = 1.88e^{-10x} - 0.3x + 0.03 - (30/109)\sin(3x) + (9/109)\cos(3x)$
Funkcję wyznaczoną analitycznie zaznaczamy na wykresach linią ciągłą zaś wynik obliczeń numerycznych linią przerywaną.

H=0.3 xp=0 xk=2

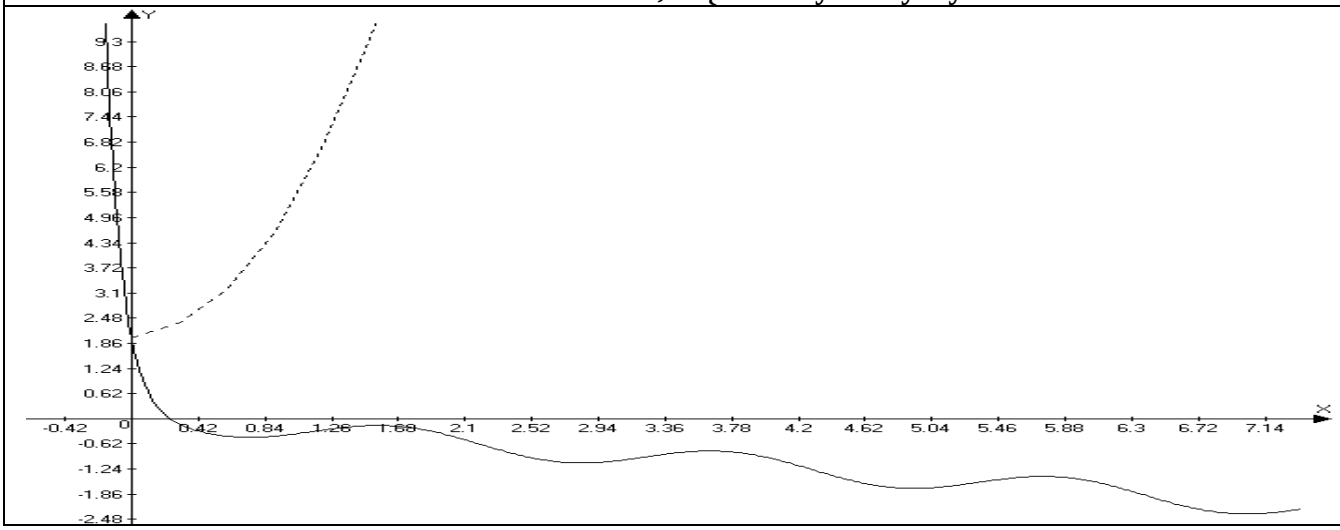
METODA EULERA, błąd maksymalny wyniósł: 240.017



METODA BROKEN LINES, błąd maksymalny wyniósł: 1156.39

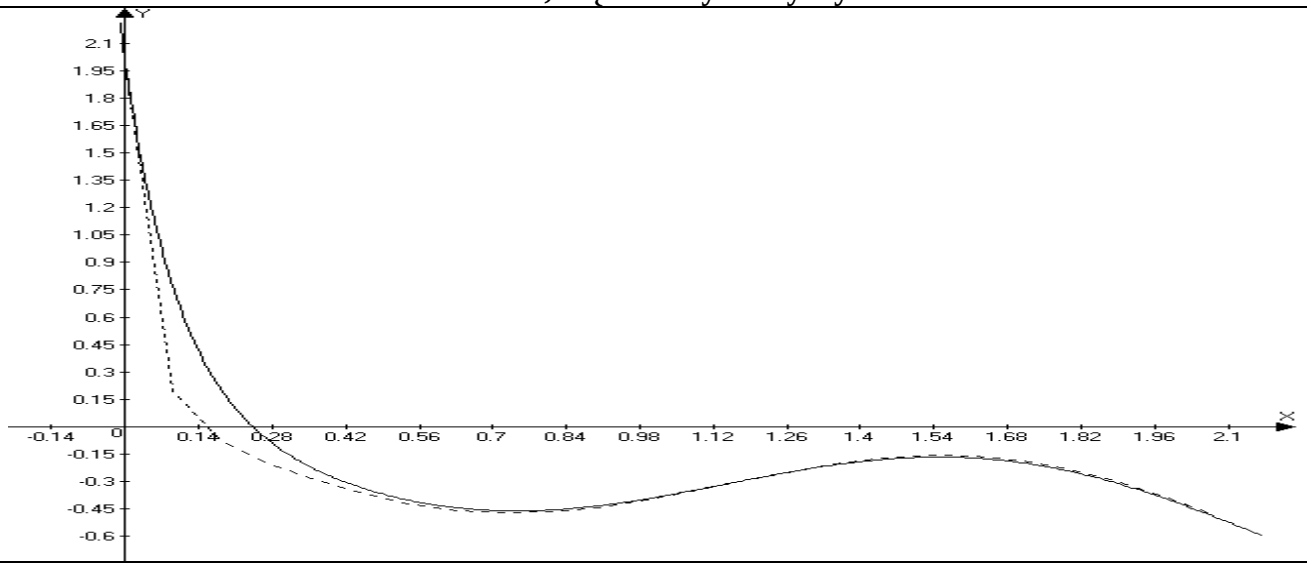


METODA RUNGEGO KUTTY, błąd maksymalny wyniósł: 17.9522

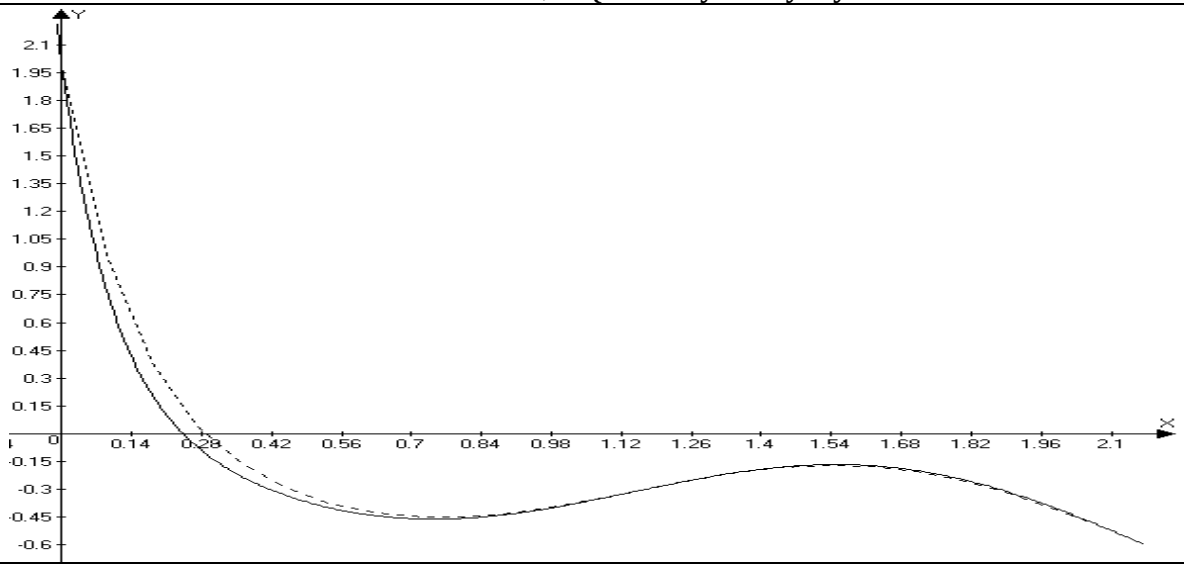


H=0.09 xp=0 xk=2

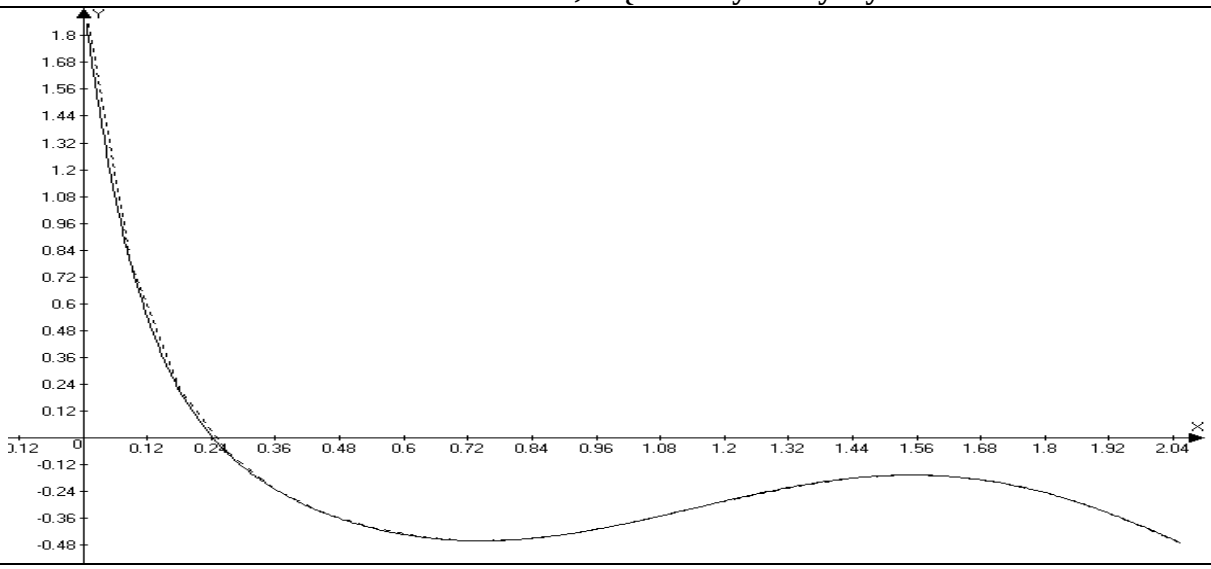
METODA EULERA, błąd maksymalny wyniósł: 0.570197



METODA BROKEN LINES, błąd maksymalny wyniósł: 0.281024

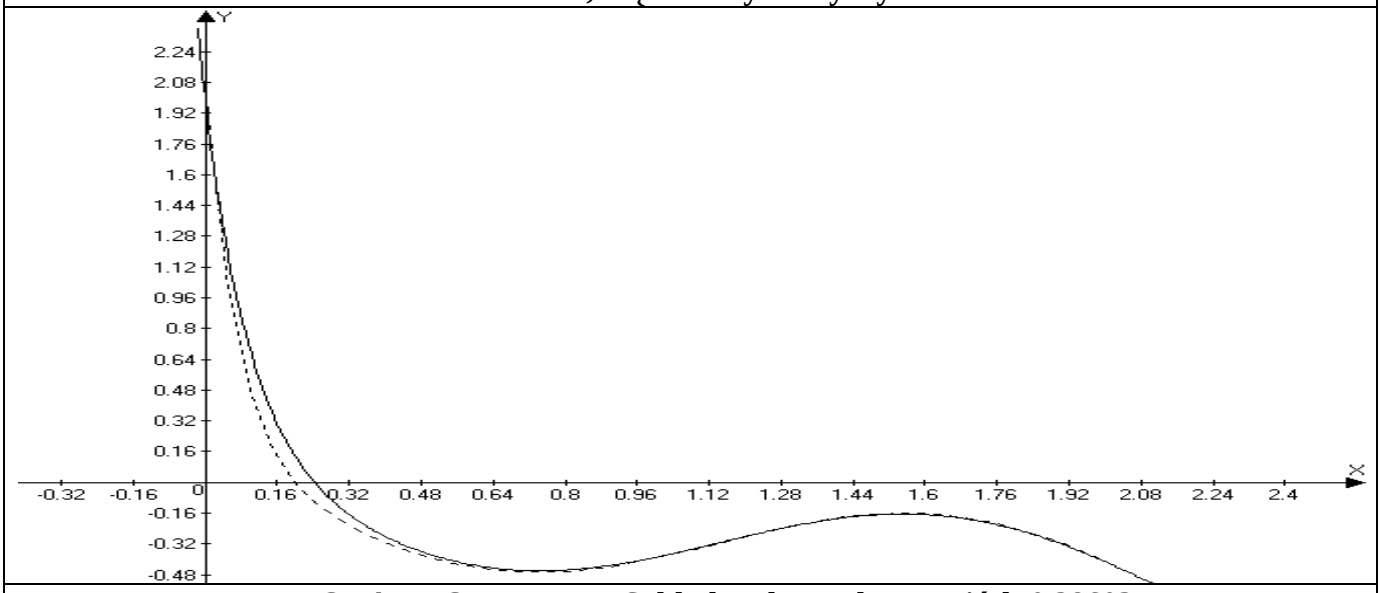


METODA RUNGEGO KUTTY, błąd maksymalny wyniósł: 0.286727

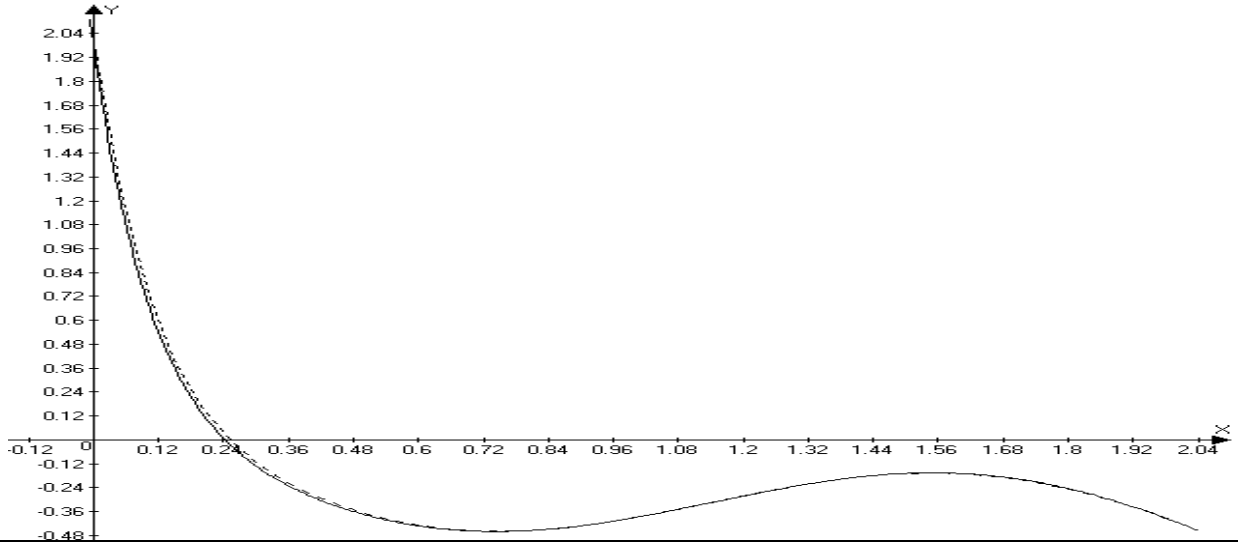


H=0.05 xp=0 xk=2

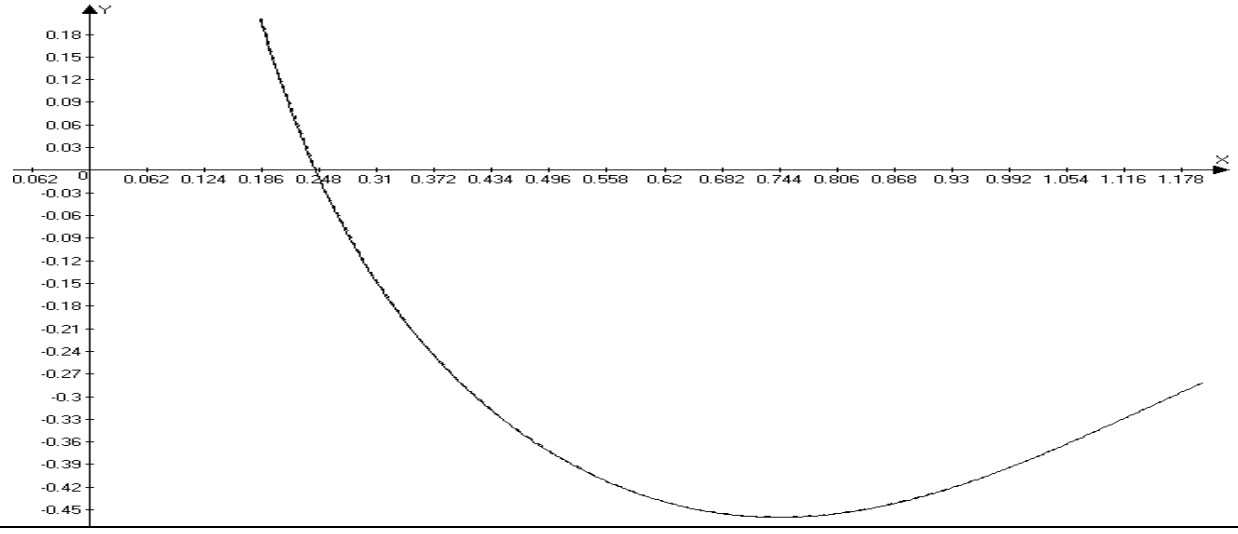
METODA EULERA, błąd maksymalny wyniósł: 0.29703



METODA BROKEN LINES, błąd maksymalny wyniósł: 0.28603

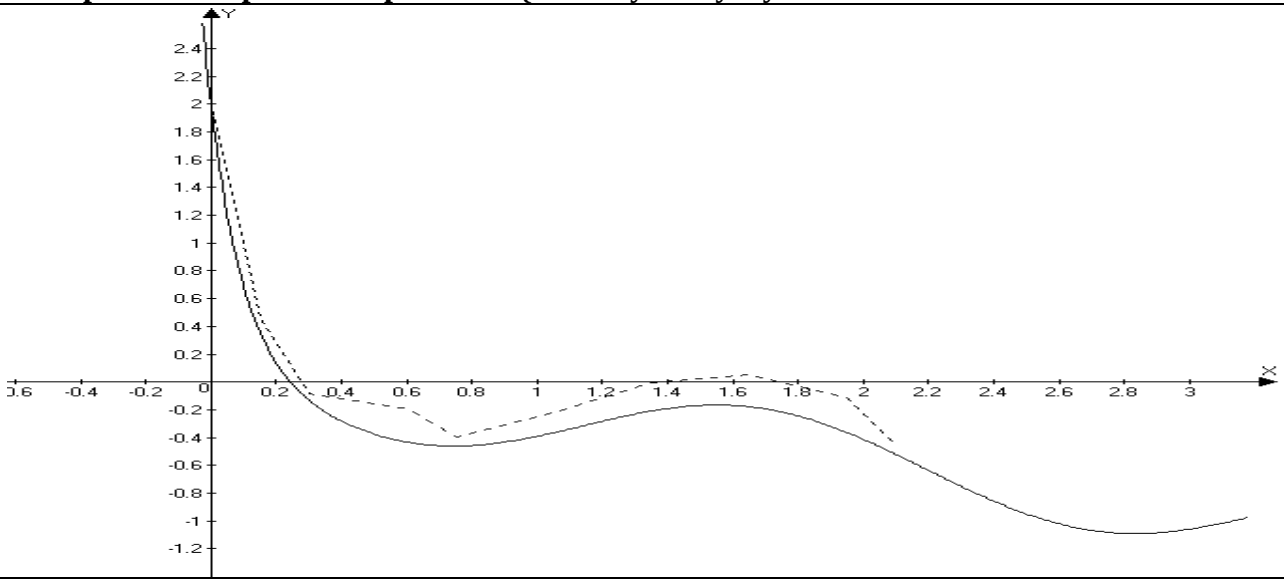


METODA RUNGEGO KUTTY, błąd maksymalny wyniósł: 0.0286912

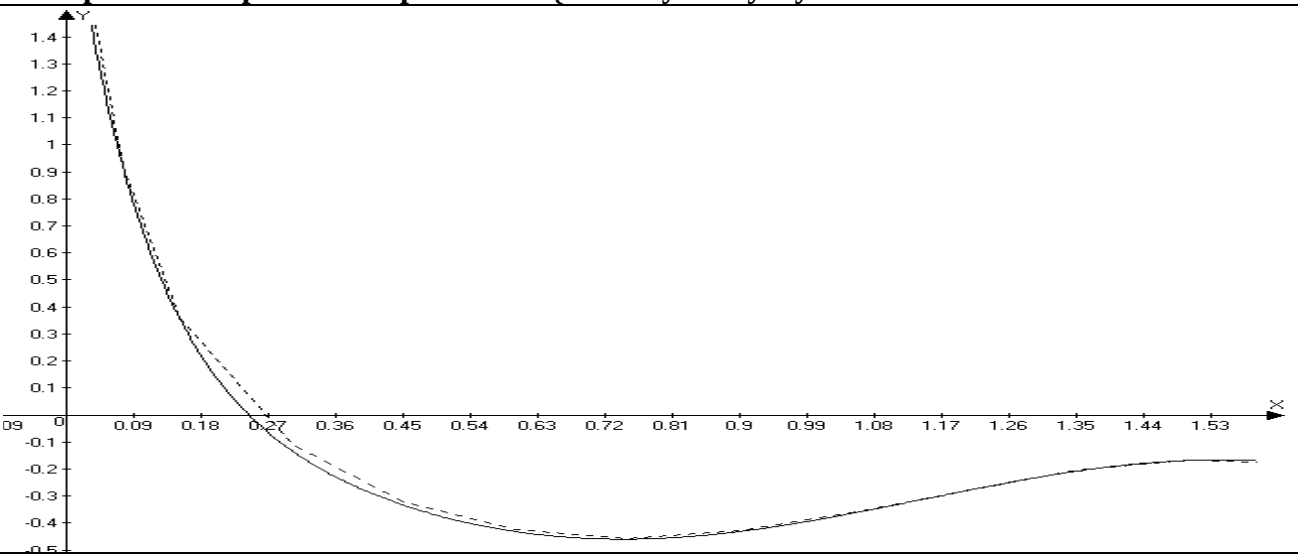


METODA RUNGEGO KUTTY – ZMIENNY KROK

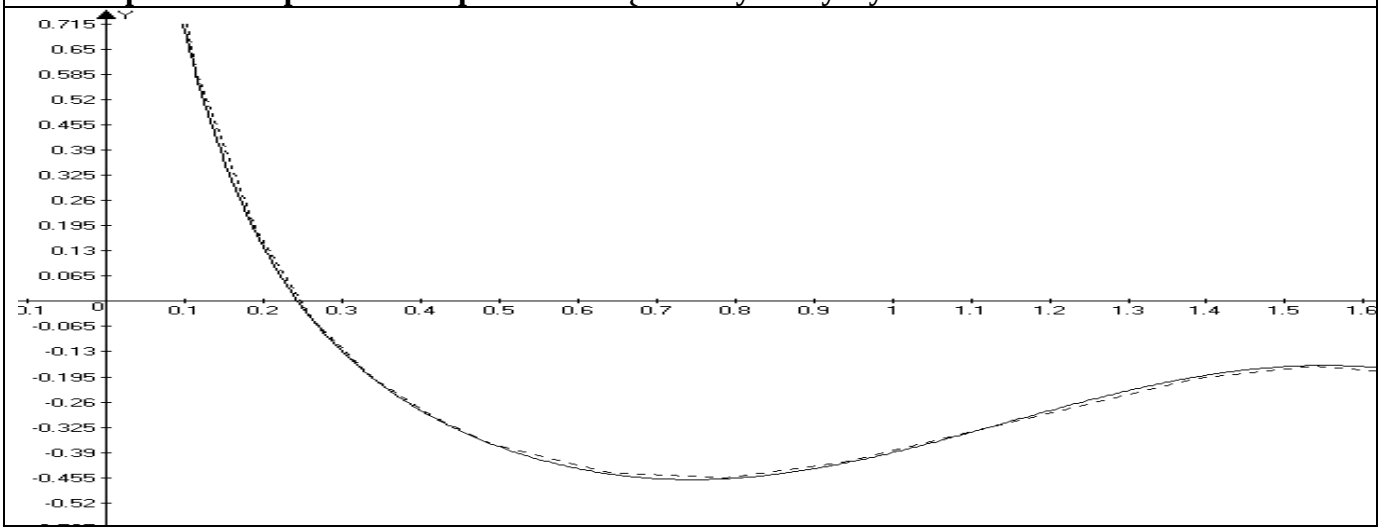
H=0.3 xp=0 xk=2 eps1=0.01 eps=0.1 błąd maksymalny wyniósł: 0.51680



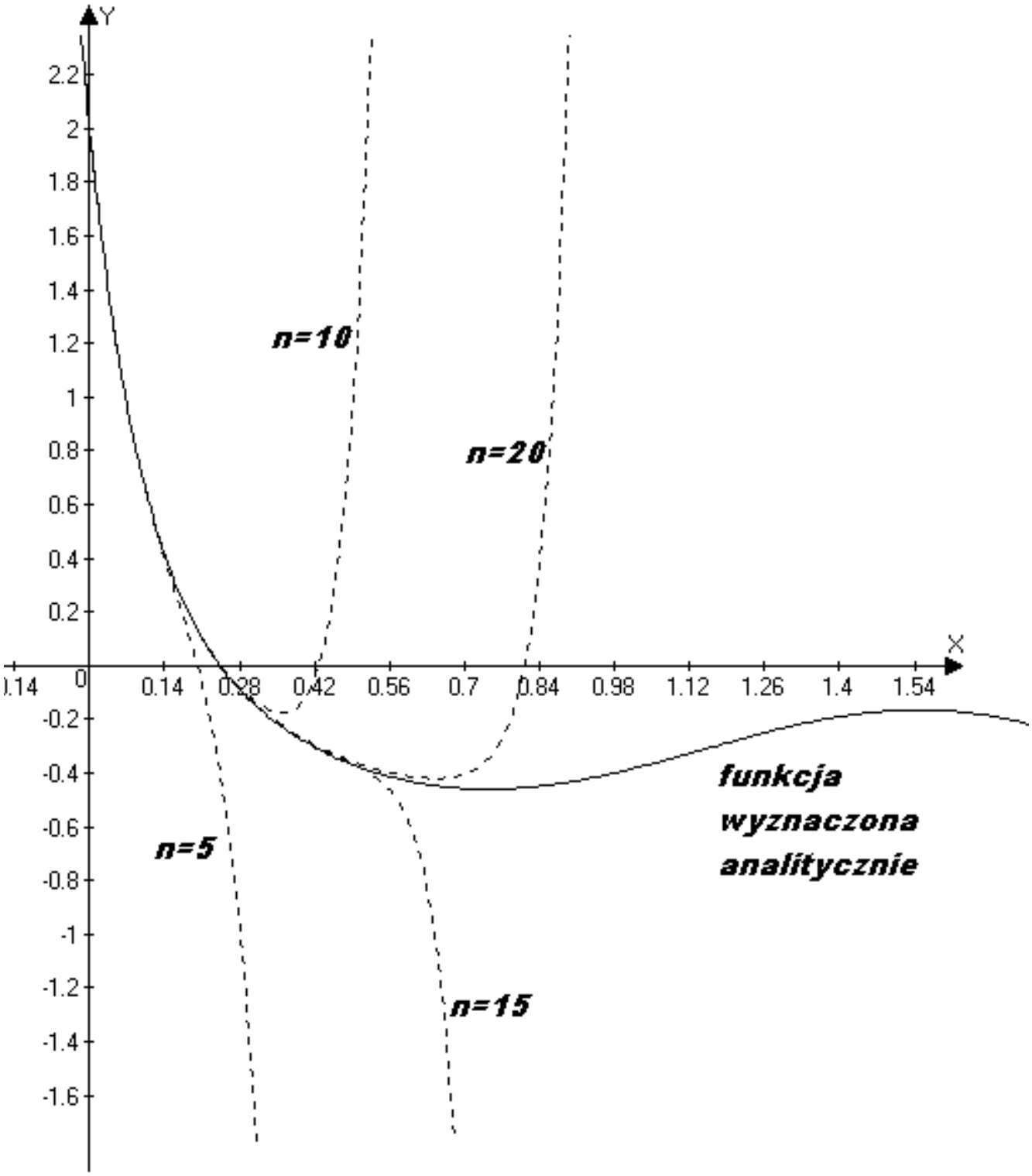
H=0.3 xp=0 xk=2 eps1=0.001 eps=00.1 błąd maksymalny wyniósł: 0.285



H=0.3 xp=0 xk=2 eps1=0.0001 eps=0.001 błąd maksymalny wyniósł: 0.28424



Metoda szeregów Tylora:
Rozwiązaliśmy najpierw 4 pierwsze pochodne po czym zauważyliśmy zależność rekurencyjną co pomogło nam w napisaniu programu. Sporządziliśmy wykres na którym widać wykres funkcji wyznaczony analitycznie (linia ciągła) oraz kolejne przybliżenia szeregu tylora dla n=(5,10,15,20) pochodnych. Jak widać na wykresach liczba kroków zwiększa zasięg działania metody tylora.



Program 1 (metoda Eulera, broken lines, rungego kutty)

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

float fxy(float x,float y)
 {
  float f;
  f=-10*y-3*(x+sin(3*x));
  return f;
 }

float fa(float x)
 {
  float f;
  f=0.03-0.3*x+1.887*exp(-10*x)-(30/109)*sin(3*x)+(9/109)*(cos(3*x));
  return f;
 }


int main()
 {
  int i,n;
  double x,y,h,x0,xk,y0,dy,k1,k2,k3,k4,eps1,eps2,eps;
  double yy,kk1,kk2,kk3,kk4,hh,xx,yyy,bmax,b;

  cout<<"Podaj warunek poczatkowy y0  >>> "; cin>>y0;
  cout<<"Podaj   dlugosc   kroku      >>> "; cin>>h;
  cout<<"Podaj argument poczatkowy x0 >>> "; cin>>x0;
  cout<<"Podaj argument koncowy x_max >>> "; cin>>xk;
  cout<<endl;
  cout<<"Podaj wartosc eps1 dla zmiennego kroku >>> "; cin>>eps1;
  cout<<"Podaj wartosc eps2 dla zmiennego kroku >>> "; cin>>eps2;
  bmax=0;

//metoda eulera
  cout<<endl<<"Metoda Eulera : "<<endl;
  y=y0;
  i=0;
  do
  {
   x=x0+i*h;
   b=fabs(fa(x)-y);
   if(b>bmax) bmax=b;
   cout<<"   "<<x<<"         "<<y<<endl;
   dy=h*fxy(x,y);
   y=y+dy;
   i++;
  }while(x<xk);
  cout<<endl<<"Maksymalny blad >>> "<<bmax;
```

```
//metoda broken lines
  bmax=0;
  cout<<endl<<"Metoda broken lines : "<<endl;
  y=y0;
  i=0;
  do
  {
   x=x0+i*h;
   b=fabs(fa(x)-y);
   if(b>bmax) bmax=b;
   cout<<"  "<<x<<"       "<<y<<endl;
   y=y+h*fxy(x+h/2.0,y+h*(1/2.0)*fxy(x,y));
   i++;
  }while(x<xk);
  cout<<endl<<"Maksymalny blad >>> "<<bmax;

//metoda Rungego Kutty
  bmax=0;
  cout<<endl<<"Metoda Rungego Kutty : "<<endl;
  y=y0;
  i=0;
  do
  {
   x=x0+i*h;
   b=fabs(fa(x)-y);
   if(b>bmax) bmax=b;
   cout<<"   "<<x<<"        "<<y<<endl;
   k1=h*fxy(x,y);
   k2=h*fxy(x+h/2.0,y+k1/2.0);
   k3=h*fxy(x+h/2.0,y+k2/2.0);
   k4=h*fxy(x+h,y+k3);
   dy=(k1+2*k2+2*k3+k4)/6.0;
   y=y+dy;
   i++;
  }while(x<xk);
  cout<<endl<<"Maksymalny blad >>> "<<bmax;

//metoda Rungego Kutty (zmienny krok)
  bmax=0;
  cout<<endl<<"Metoda Rungego Kutty (zmienny krok) : "<<endl;
  y=y0;
  x=x0;
  xx=x0;
  i=0;
  do
  {
   b=fabs(fa(x)-y);
   if(b>bmax) bmax=b;
   cout<<i<<" >>> x = "<<x<<"     y = "<<y<<endl;
   i++;
   x=x+h;
   yy=y;
//---------------------------------
   k1=h*fxy(x,y);
   k2=h*fxy(x+h/2.0,y+k1/2.0);
   k3=h*fxy(x+h/2.0,y+k2/2.0);
   k4=h*fxy(x+h,y+k3);
   dy=(k1+2*k2+2*k3+k4)/6.0;
   y=y+dy;
```

```
//---------------------------------
   hh=h/2.0;
   xx=xx+hh;

   kk1=hh*fxy(xx,yy);
   kk2=hh*fxy(xx+hh/2.0,yy+kk1/2.0);
   kk3=hh*fxy(xx+hh/2.0,yy+kk2/2.0);
   kk4=hh*fxy(xx+hh,yy+kk3);
   dy=(kk1+2*kk2+2*kk3+kk4)/6.0;
   yy=yy+dy;
   xx=xx+hh;

   kk1=hh*fxy(xx,yy);
   kk2=hh*fxy(xx+hh/2.0,yy+kk1/2.0);
   kk3=hh*fxy(xx+hh/2.0,yy+kk2/2.0);
   kk4=hh*fxy(xx+hh,yy+kk3);
   dy=(kk1+2*kk2+2*kk3+kk4)/6.0;
   yy=yy+dy;
//---------------------------------
   eps=fabs(yy-y);
   if(eps<eps1) h=h*2.0;
   if(eps>eps2) { x=x-h; h=h/2.0; i--;}
  }while(x<xk && i<10);
  cout<<endl<<"Maksymalny blad >>> "<<bmax;
  getch();
 }
```

Program 2 (Metoda szeregów Tylora)

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

double silnia(int x)
 {
  double s=1;
  int i;

  for(i=1;i<=x;i++)
   {
    s=s*i;
   }
  return s;
 }


int main()
 {
  int i,n,ile,k;
  double *tab,yx,il,ix;

  cout<<"ile pochodnych liczyc (>4 )>>> "; cin>>ile;
  tab=(double*)malloc((ile+5)*sizeof(double));
  n=ile;

  tab[0]=2;
  tab[1]=-20;
  tab[2]=188;
  tab[3]=-1880;
  tab[4]=18881; il=81;

  for(i=5;i<n;i=i)
  {
   k=i;
   tab[k]=-10*tab[k-1]-3*il*sin(0); k++; il=il*3;
   tab[k]=-10*tab[k-1]-3*il*cos(0); k++; il=il*3;
   tab[k]=-10*tab[k-1]+3*il*sin(0); k++; il=il*3;
   tab[k]=-10*tab[k-1]+3*il*cos(0); k++; il=il*3;
   i=k;
  }

  cout<<"wyniki"<<endl;
  for(i=0;i<n;i++)
  {
   cout<<setprecision(50)<<tab[i]<<endl;
  }
  cout<<"y = ";
  for(i=0;i<n;i++)
  {
   ix=tab[i]/silnia(i);
   cout<<ix<<"x^"<<i<<"+";
  }
  getch();
}
```