Problem 1 – Runge-Kutta Method (Fourth Order)

```cpp
// 04/16/2014 - ENGR 2450 - Meine, Joel
// Problem 25.21

// Runge-Kutta Method (Fourth Order)

#include <iostream>
#include <iomanip>
#include <math.h>
#include <vector>
using namespace std;

vector<double> T = {1950,1960,1970,1980,1990,2000}; // Year
vector<double> P = {2555,3040,3708,4454,5276,6079}; // Population (people in millions)
const int size = 6;

const double kgm = 0.026; // Maximum Growth Rate under Unlimited Conditions
const double pmax = 12000; // Carrying Capacity (people in millions)

double Population(double t,double p)
{
	double dpdt;
	p = pmax / (1-(1-(pmax/P[0]))*exp(-kgm*(t-T[0]))); // Population at Time
	dpdt = kgm*(1-(p/pmax))*p; // Growth Rate of Population with Time
	return(dpdt);
}

double Derivs(double x,double y)
{
	return(Population(x,y));
}

void RK4(double& x,double& y,double& h,double& ynew)
{
	double k1,k2,k3,k4;
	double ym,ye,slope;
	k1 = Derivs(x,y);
	ym = y + k1*(h/2);
	k2 = Derivs(x+(h/2),ym);
	ym = y + k2*(h/2);
	k3 = Derivs(x+(h/2),ym);
	ye = y + k3*h;
	k4 = Derivs(x+h,ye);
	slope = (k1 + 2*(k2 + k3) + k4)/6;
	ynew = y + slope*h;
	x = x + h;
}

void Integrator(double& x,double& y,double& h,double& xend)
{
	double ynew;
	do {
		if (xend - x < h) h = xend - x;
		RK4(x,y,h,ynew);
		y = ynew;
```

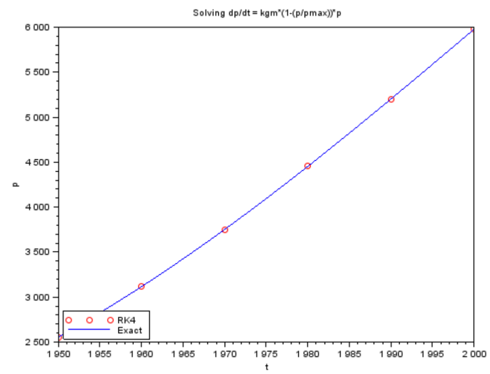
```
Chapter 25 - Problem 25.21
==================================================================
Growth Rate of Population with Time (millions of people per year)

dp/dt = kgm * (1-(p/pmax))*p

Population at Time (millions of people)

p = pmax / (1-(1-(pmax/p0))*exp(-kgm*(t-t0)))

Actual Population at Time (people in millions), pa
Initial Population at t0 (people in millions), p0 = 2555
Initial Year with p0, t0 = 1950
Maximum Growth Rate under Unlimited Conditions, kgm = 0.026
Carrying Capacity (people in millions), pmax = 12000
******************************************************************
 t     pa    p
------------------------------------------------------------------
 1950  2555       2555
 1960  3040  3116.6194
 1970  3708  3752.6407
 1980  4454  4453.3888
 1990  5276   5202.448
 2000  6079  5977.7003
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Press any key to continue . . .
```

```
	} while (x < xend);
}

int main()
{
	// Problem 25.21 - Runge-Kutta Method (Fourth Order)

	vector<double> xp,yp;
	double h;
	double x,xi,xf,xend,xout,dx,y;

	xi = T[0]; y = P[0]; xout = 10.0; xf = T.back(); dx = 0.1;

	x = xi;
	xp.push_back(x);
	yp.push_back(y);
	do {
		xend = x + xout;
		if (xend > xf) xend = xf;
		h = dx;
		Integrator(x,y,h,xend);
		xp.push_back(x);
		yp.push_back(y);
	} while (x < xf);

	std::cout << "Chapter 25 - Problem 25.21" << std::endl;
	std::cout << "===================================================================" << std::endl;
	std::cout << "Growth Rate of Population with Time (millions of people per year)" << std::endl;
	std::cout << std::endl;
	std::cout << "dp/dt = kgm * (1-(p/pmax))*p" << std::endl;
	std::cout << std::endl;
	std::cout << "Population at Time (millions of people)" << std::endl;
	std::cout << std::endl;
	std::cout << "p = pmax / (1-(1-(pmax/p0))*exp(-kgm*(t-t0)))" << std::endl;
	std::cout << std::endl;
	std::cout << "Actual Population at Time (people in millions), pa" << std::endl;
	std::cout << "Initial Population at t0 (people in millions), p0 = " << P[0] << std::endl;
	std::cout << "Initial Year with p0, t0 = " << T[0] << std::endl;
	std::cout << "Maximum Growth Rate under Unlimited Conditions, kgm = " << kgm << std::endl;
	std::cout << "Carrying Capacity (people in millions), pmax = " << pmax << std::endl;
	std::cout << "*******************************************************************" << std::endl;
	std::cout << " t     pa    p" << std::endl;
	std::cout << "-------------------------------------------------------------------" << std::endl;
	for (int i = 0; i < size; i++)
	{
		cout << setw(5) << xp[i];
		cout << setw(6) << P[i];
		cout << setw(11) << setprecision(8) << yp[i] << endl;
	}
	std::cout << "+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << std::endl;
	cout << "\n";

	system("pause");
	return 0;
}
```

Problems 2 & 3 – Runge-Kutta Method (Fourth Order) in SCILAB

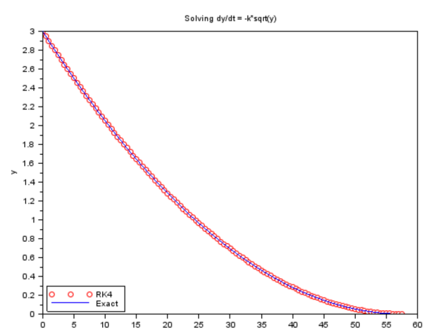
```
// 04/16/2014 - ENGR 2450 - Meine, Joel
// Problem 25.21

function [dpdt] = f(t,p)
    dpdt = kgm*(1-(p/pmax))*p;
endfunction;
t0 = 1950; p0 = 2555; tn = 2000; h = 10;
kgm = 0.026; pmax = 12000;
t = [t0:h:tn];
p = ode("rk",p0,t0,t,f);
disp("t   p");
disp("--------");
disp([t' p']);
function dpdt = fe(t)
    dpdt = pmax / (1-(1-(pmax/p0))*exp(-kgm*(t-t0)));
endfunction;
te = [t0:h/10:tn];
n = length(te);
for i = 1:n
    pe(i)=fe(te(i));
end;
plot(t,p,'ro',te,pe,'-b'); legend('RK4','Exact',3);
xtitle('Solving dp/dt = kgm*(1-(p/pmax))*p','t','p');
```

t	p
1950	2555
1960	3116.6194
1970	3752.6407
1980	4453.3888
1990	5202.448
2000	5977.7003




```
// 04/16/2014 - ENGR 2450 - Meine, Joel
// Problem 25.27

function [dydt] = f(t,y)
    dydt = -k*sqrt(y);
endfunction;
t0 = 0; y0 = 3; tmax = 57.5; h = 0.5;
k = 0.06;
t = [t0:h:tmax];
y = ode("rk",y0,t0,t,f);
disp("t   y");
disp("--------");
disp([t' y']);
function dydt = fe(t)
    dydt = ((k^2)/4)*((2/k)*sqrt(y0)-t)^2;
endfunction;
te = [t0:h/10:tmax];
n = length(te);
for i = 1:n
    ye(i)=fe(te(i));
end;
plot(t,y,'ro',te,ye,'-b');
legend('RK4','Exact',3);
xtitle('Solving dy/dt = -k*sqrt(y)','t','y');
```

t	y
0	3
5	2.5028848
10	2.0507695
15	1.6436543
20	1.281539
25	0.9644238
30	0.6923085
35	0.4651933
40	0.2830781
45	0.1459628
50	0.0538476
55	0.0067323
57.5	0.0000497



Problems 4 & 5 – Runge-Kutta Method (Fourth Order) ~ System of Equations

```
// 04/16/2014 - ENGR 2450 - Meine, Joel
// Lorenz Equations (Pg. 816 | Chapra & Canale), Problem 28.19

// Runge-Kutta Method (Fourth Order) - System of Equations

#include <iostream>
#include <iomanip>
#include <math.h>
#include <vector>
using namespace std;

const double o = 10, b = 2.666667, r = 28;

vector<double> Lorenz(double t,vector<double> y)
{
	double dxdt,dydt,dzdt;
	dxdt = -o*y[0] + o*y[1];
	dydt = r*y[0] - y[1] - y[0]*y[2];
	dzdt = -b*y[2] + y[0]*y[1];
	vector<double> F = {dxdt,dydt,dzdt};
	return(F);
}

const double f = 60, L = 30, E = 1.25e8, I = 0.05;

vector<double> Sailboat(double z,vector<double> y)
{
	double dy1dz,dy2dz;
	dy1dz = y[1];
	dy2dz = (f/(2*E*I))*pow(L-z,2);
	vector<double> F = {dy1dz,dy2dz};
	return(F);
}

vector<double> Derivs(double x,vector<double> y,int p)
{
	if (p == 1)
	{
		return(Lorenz(x,y));
	}
	if (p == 2)
	{
		return(Sailboat(x,y));
	}
}

void RK4(double& x,vector<double>& y,int& n,double& h,int& p)
{
	vector<double> k1,k2,k3,k4;
	vector<double> ym1,ym2,ye,slope;
	k1 = Derivs(x,y,p);
	for (int i = 0; i < n; i++)
	{
		ym1.push_back(y[i] + k1[i]*(h/2.0));
```

```
Lorenz Equations (Pg. 816 | Chapra & Canale)
==================================================
dx/dt = -o*x + o*y

dy/dt = r*x - y - x*z

dz/dt = -b*z + x*y

o = 10; b = 2.66667; r = 28

Atmospheric Fluid Motion, x
Temperature Variation (Horizontal Axis), y
Temperature Variation (Vertical Axis), z
Time, t
**************************************************
 t   x          y          z
--------------------------------------------------
  0        5          5        5
  2  -8.2168    -11.905   20.604
  4  -7.7511    -11.674   19.225
  6  -5.0959    -7.8971   16.432
  8  -1.5965    -1.0298   19.972
 10   3.9774      6.222   20.469
 12   4.1309     5.1608   19.501
 14   3.0388     2.9783   20.585
 16   5.3927   0.005294   30.114
 18   4.4553     7.5438   14.798
 20   11.372    0.82944   39.307
++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
Problem 28.19
==================================================
dy1/dz = y2

dy2/dz = (f/(2*E*I))*pow(L-z,2)

Wind Force, f = 60
Modulus of Elasticity, E = 1.25e+008
Mast Length, L = 30
Moment of Inertia, I = 0.05

Deflection of Mast (Horizontal Axis), y
Deflection of Mast (Vertical Axis), z
**************************************************
 z    y2         dy2/dz
--------------------------------------------------
  0        0         0
  5  0.04825    0.0182
 10    0.172    0.0304
 15  0.34425    0.0378
 20    0.544    0.0416
 25  0.75625     0.043
 30    0.972    0.0432
++++++++++++++++++++++++++++++++++++++++++++++++++

Press any key to continue . . .
```

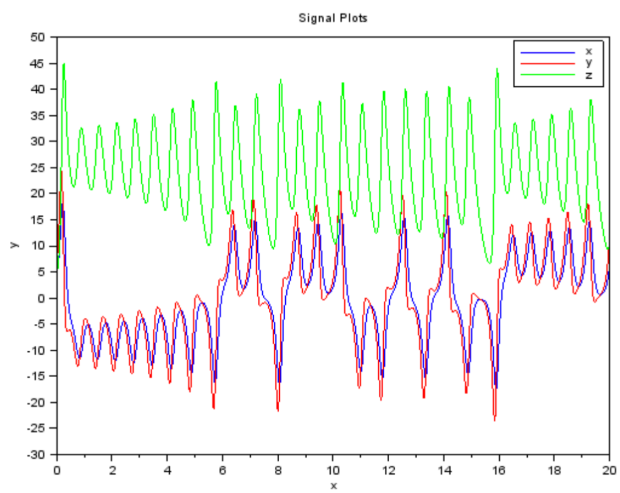
```
        }
        k2 = Derivs(x+(h/2.0),ym1,p);
        for (int i = 0; i < n; i++)
        {

                ym2.push_back(y[i] + k2[i]*(h/2.0));
        }
        k3 = Derivs(x+(h/2.0),ym2,p);
        for (int i = 0; i < n; i++)
        {
                ye.push_back(y[i] + k3[i]*h);
        }
        k4 = Derivs(x+h,ye,p);
        for (int i = 0; i < n; i++)
        {
                slope.push_back((k1[i] + 2.0*(k2[i] + k3[i]) + k4[i])/6.0);
                y[i] = y[i] + slope[i]*h;
        }
        x = x + h;
}

void Integrator(double& x,vector<double>& y,int& n,double& h,double& xend,int p)
{
        do {
                if (xend - x < h) h = xend - x;
                RK4(x,y,n,h,p);
        } while (x < xend);
}

int main()
{
        // Lorenz Equations (Pg. 816 | Chapra & Canale)

        vector<double> xp1;
        vector< vector<double> > yp1;
        vector<double> y1,yi1;
        int n1;
        double x1,xi1,xf1,dx1,h1,xout1,xend1;

        yi1 = {5,5,5}; n1 = 3; xi1 = 0; xf1 = 20; dx1 = 0.1; xout1 = 2.0;
        const int size1 = 11;

        x1 = xi1;
        xp1.push_back(x1);
        for (int i = 0; i < n1; i++)
        {
                vector<double> ypw1;
                ypw1.push_back(yi1[i]);
                yp1.push_back(ypw1);
                y1.push_back(yi1[i]);
        }
        do {
                xend1 = x1 + xout1;
                if (xend1 > xf1) xend1 = xf1;
                h1 = dx1;
                Integrator(x1,y1,n1,h1,xend1,1);
                xp1.push_back(x1);
```

```
        for (int i = 0; i < n1; i++)
        {
            yp1[i].push_back(y1[i]);
        }
    } while (x1 < xf1);

    std::cout << "Lorenz Equations (Pg. 816 | Chapra & Canale)" << std::endl;
    std::cout << "=================================================" << std::endl;
    std::cout << "dx/dt = -o*x + o*y" << std::endl;
    std::cout << std::endl;
    std::cout << "dy/dt = r*x - y - x*z" << std::endl;
    std::cout << std::endl;
    std::cout << "dz/dt = -b*z + x*y" << std::endl;
    std::cout << std::endl;
    std::cout << "o = " << o << "; b = " << b << "; r = " << r <<std::endl;
    std::cout << std::endl;
    std::cout << "Atmospheric Fluid Motion, x" << std::endl;
    std::cout << "Temperature Variation (Horizontal Axis), y" << std::endl;
    std::cout << "Temperature Variation (Vertical Axis), z" << std::endl;
    std::cout << "Time, t" << std::endl;
    std::cout << "*************************************************" << std::endl;
    std::cout << " t   x         y          z" << std::endl;
    std::cout << "-------------------------------------------------" << std::endl;
    for (int i = 0; i < size1; i++)
    {
        cout << setw(3) << xp1[i];
        cout << setw(9) << setprecision(5) << yp1[0][i];
        cout << setw(11) << setprecision(5) << yp1[1][i];
        cout << setw(9) << setprecision(5) << yp1[2][i] << endl;
    }
    std::cout << "+++++++++++++++++++++++++++++++++++++++++++++++++" << std::endl;
    cout << "\n";

    // Problem 28.19

    vector<double> xp2;
    vector< vector<double> > yp2;
    vector<double> y2,yi2;
    int n2;
    double x2,xi2,xf2,dx2,h2,xout2,xend2;

    yi2 = {0,0}; n2 = 2; xi2 = 0; xf2 = L; dx2 = 0.5; xout2 = 5.0;
    const int size2 = 7;

    x2 = xi2;
    xp2.push_back(x2);
    for (int i = 0; i < n2; i++)
    {
        vector<double> ypw2;
        ypw2.push_back(yi2[i]);
        yp2.push_back(ypw2);
        y2.push_back(yi2[i]);
    }
    do {
        xend2 = x2 + xout2;
        if (xend2 > xf2) xend2 = xf2;
        h2 = dx2;
```

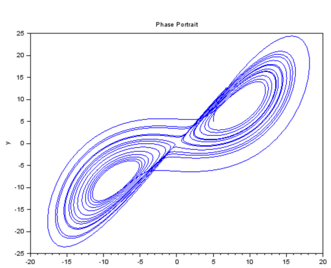
```
            Integrator(x2,y2,n2,h2,xend2,2);
            xp2.push_back(x2);
            for (int i = 0; i < n2; i++)
            {
                   yp2[i].push_back(y2[i]);
            }
      } while (x2 < xf2);

      std::cout << "Problem 28.19" << std::endl;
      std::cout << "==================================================" << std::endl;
      std::cout << "dy1/dz = y2" << std::endl;
      std::cout << std::endl;
      std::cout << "dy2/dz = (f/(2*E*I))*pow(L-z,2)" << std::endl;
      std::cout << std::endl;
      std::cout << "Wind Force, f = " << f << std::endl;
      std::cout << "Modulus of Elasticity, E = " << E << std::endl;
      std::cout << "Mast Length, L = " << L << std::endl;
      std::cout << "Moment of Inertia, I = " << I << std::endl;
      std::cout << std::endl;
      std::cout << "Deflection of Mast (Horizontal Axis), y" << std::endl;
      std::cout << "Deflection of Mast (Vertical Axis), z" << std::endl;
      std::cout << "**************************************************" << std::endl;
      std::cout << " z    y2        dy2/dz" << std::endl;
      std::cout << "--------------------------------------------------" << std::endl;
      for (int i = 0; i < size2; i++)
      {
            cout << setw(3) << xp2[i];
            cout << setw(10) << setprecision(5) << yp2[0][i];
            cout << setw(9) << setprecision(5) << yp2[1][i] << endl;
      }
      std::cout << "++++++++++++++++++++++++++++++++++++++++++++++++++" << std::endl;
      cout << "\n";

      system("pause");
      return 0;
}
```

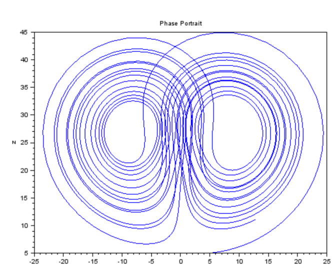
Problems 6 & 7 – Runge-Kutta Method (Fourth Order) ~ System of Equations in SCILAB

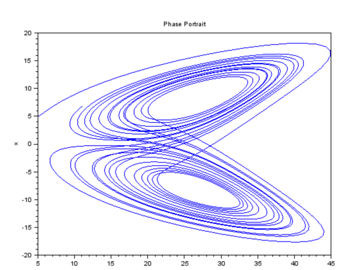
```
// 04/16/2014 - ENGR 2450 - Meine, Joel
// Lorenz Equations (Pg. 816 | Chapra & Canale)

function [Z] = F(t,Y)
    Z(1) = -o*Y(1) + o*Y(2);
    Z(2) = r*Y(1) - Y(2) - Y(1)*Y(3);
    Z(3) = -b*Y(3) + Y(1)*Y(2);
endfunction
o = 10; b = 2.666667; r = 28;
dt = 0.5; // 0.5 for table values; 0.01 for plots
t = [0.0:dt:20.0];
t0 = 0;
Y0 = [5;5;5];
Y = ode("rk",Y0,t0,t,F);
disp("t   x   y   z");
disp([t' Y']);
y1 = Y(1,:); y2 = Y(2,:); y3 = Y(3,:);
scf(); plot(t,y1,'b-',t,y2,'r-',t,y3,'g-');
legend('x','y','z',1);
xtitle('Signal Plots','x','y','z');
scf(); plot(y1,y2);
xtitle('Phase Portrait','x','y');
scf(); plot(y2,y3);
xtitle('Phase Portrait','y','z');
scf(); plot(y3,y1);
xtitle('Phase Portrait','z','x');
```



t	x	y	z
0	5	5	5
0.5	-3.7814106	-6.3169311	23.647634
1	-7.0906465	-4.1386816	29.061623
1.5	-11.595625	-10.143753	32.548029
2	-9.1334883	-12.695516	22.463797
2.5	-4.769982	-6.0499785	20.033587
3	-5.0081004	-2.5352284	26.615529
3.5	-12.125545	-8.7253588	35.088112
4	-8.1282379	-12.686092	18.56868
4.5	-2.3818654	-2.7159269	18.774727
5	-7.6106433	-0.5349684	33.467966
5.5	-3.686293	-6.7968469	10.088964
6	1.0242741	3.3178066	22.77626
6.5	10.854489	4.497227	36.107139
7	6.1728431	10.650915	13.895474
7.5	0.0907112	-0.8721182	19.676757
8	-14.232407	-21.597105	25.62721
8.5	4.9661488	7.8260625	16.949726
9	2.7805677	1.4514892	23.063958
9.5	13.263849	8.3341082	37.837017
10	2.1147599	3.7251072	11.395483
10.5	1.5377615	-3.2983107	27.516312
11	-14.092675	-13.880206	34.713982
11.5	-2.801965	-4.704838	13.16033
12	-1.7012279	1.9638365	26.076456
12.5	12.909464	19.531438	24.169917
13	-1.9550934	-3.2937689	15.004878
13.5	-4.4209484	1.7909131	30.334953
14	7.7547255	13.784561	13.699074
14.5	-1.6475387	-2.9407104	18.197664
15	-7.6171841	-0.1606193	33.787152
15.5	-0.7267126	-1.3565902	8.8388018
16	-7.3987413	4.4236376	36.789302
16.5	12.085237	12.815898	30.873117
17	7.2777553	10.622188	19.635627
17.5	3.8369271	4.0400133	21.13478
18	6.8700656	2.157853	30.694365
18.5	12.98711	16.074606	29.346399
19	3.5027921	5.5218223	15.163712
19.5	2.4027939	-0.5091207	25.361272
20	6.9793893	12.966543	11.232319







```
// 04/16/2014 - ENGR 2450 - Meine, Joel
// Problem 28.47

function [Z] = F(t,Y)
    Z(1) = Y(2);
    Z(2) = (Fo*sin(w*t)-a*abs(Y(2))*Y(2)-k*Y(1))/m;
endfunction
m = 2; a = 5; k = 6; Fo = 2.5; w = 0.5;
dt = 0.25;
t = [0.0:dt:15.0];
t0 = 0;
Y0 = [1;0];
Y = ode("rk",Y0,t0,t,F);
disp("t   x   dx/dt");
disp([t' Y']);
y1 = Y(1,:); y2 = Y(2,:);
scf(); plot(t,y1,'b-',t,y2,'r-');
legend('x','dx/dt',1);
xtitle('Signal Plots','x','dx/dt');
scf(); plot(y1,y2);
xtitle('Phase Portrait','x','dx/dt');
```

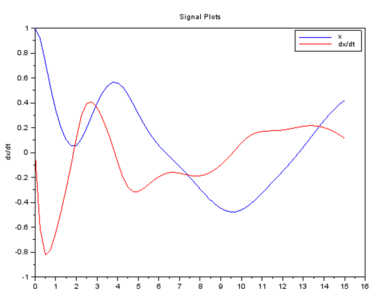
$$m\frac{d^2x}{dt^2}+a\left|\frac{dx}{dt}\right|\frac{dx}{dt}+kx=F_o\sin(\omega t)$$

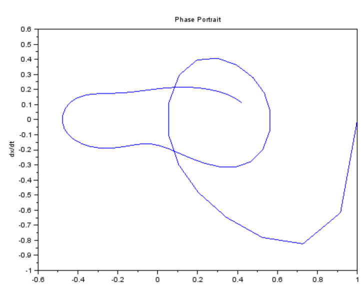
$$x=y_1 \qquad \frac{dx}{dt}=y_1'=y_2$$

$$my_2'+a|y_2|y_2+ky_1=F_o\sin(\omega t)$$

$$y_2'=\frac{F_o\sin(\omega t)-a|y_2|y_2-ky_1}{m}$$

$$y_1'=y_2$$





t	x	dx/dt
0	1	0
0.25	0.9154604	-0.6166714
0.5	0.7277995	-0.8234252
0.75	0.5241681	-0.7814252
1	0.3443730	-0.6485298
1.25	0.2026036	-0.4820589
1.5	0.1047191	-0.2986876
1.75	0.0544165	-0.1012159
2	0.0555627	0.1123009
2.25	0.1081797	0.2971271
2.5	0.1966896	0.3951189
2.75	0.2985625	0.4077322
3	0.3956909	0.3617355
3.25	0.4764842	0.2800503
3.5	0.5338876	0.1761942
3.75	0.5632914	0.0567133
4	0.5611603	-0.0751275
4.25	0.5265626	-0.1968859
4.5	0.4661165	-0.2787639
4.75	0.3911007	-0.3141686
5	0.3120706	-0.3132178
5.25	0.2363239	-0.2901899
5.5	0.1677952	-0.2572100
5.75	0.1078092	-0.2230526
6	0.0558665	-0.1936491
6.25	0.0102790	-0.1726452
6.5	-0.0312944	-0.1616342
6.75	-0.0713415	-0.1602004
7	-0.1120069	-0.1660669
7.25	-0.1546814	-0.1755937
7.5	-0.1997620	-0.1846352
7.75	-0.2466461	-0.1894800
8	-0.2939305	-0.1875105
8.25	-0.3397195	-0.1773972
8.5	-0.3819292	-0.1589106
8.75	-0.4185188	-0.1325642
9	-0.4476323	-0.0992748
9.25	-0.4676678	-0.0601262
9.5	-0.4773013	-0.0162483
9.75	-0.4755040	0.0309496
10	-0.4619349	0.0768392
10.25	-0.4376947	0.1155002
10.5	-0.4050750	0.1435889
10.75	-0.3668033	0.1609207
11	-0.3253325	0.1696744
11.25	-0.2824083	0.1731202
11.5	-0.2389400	0.1745300
11.75	-0.1950957	0.1764743
12	-0.1505291	0.1804641
12.25	-0.1046604	0.1868576
12.5	-0.0569533	0.1949904
12.75	-0.0071341	0.2034867
13	0.0446803	0.2106703
13.25	0.0979588	0.2149603
13.5	0.1518169	0.2151444
13.75	0.2051265	0.2104894
14	0.2566337	0.2007102
14.25	0.3050586	0.1858619
14.5	0.3491643	0.1662135
14.75	0.3877967	0.1421434
15	0.4199023	0.1140695