МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М. В. Ломоносова

Факультет Вычислительной Математики и Кибернетики

Компьютерный практикум по курсу

«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 2

Численные методы решения дифферциальных уравнений

ОТЧЕТ

о выполнение задания

студента 205 учебной группы факультета ВМК МГУ

Швецова Дениса Андреевича

Москва. 2014 г.

Практическая работа № 2 (1)

Подвариант № 1

РЕШЕНИЕ ЗАДАЧИ КОШИ ДЛЯ ДИФФЕРИНЦИАЛЬНОГО УРАВНЕНИЯ ПЕРВОГО ПОРЯДКА ИЛИ СИСТЕМЫ ДИФФЕРИНЦАЛЬНЫХ УРАВНЕНИЙ ПЕРВОГО ПОРЯДКА.

Цель работы.

Освоить методы Рунге-Кутта второго и четвертого порядка точности, применяемые для численного решения задачи Коши для дифференциального уравнения (или системы дифференциальных уравнений) первого порядка.

Постановка задачи.

Рассмотрим обыкновенное дифференциальное уравнение первого порядка, разрешенное относительно производной и имеющее вид:

$$\frac{\partial y}{\partial x}=f(x,y), x_0<x, \qquad (1)$$

С дополнительным начальным условием, заданным в точке $x = x_0$:

$$y(x_0) = y_0 \qquad (2)$$

Предполагается, что правая часть уравнения (1) функция $f = f(x, y)$ такова, что гарантирует существование и единственность решения задачи Коши (1) – (2).

В том случае, если рассматривается не одно дифференциальное уравнение вида (1), а система обыкновенных дифференциальных уравнений первого порядка, разрешенных относительно производных неизвестных функций, то соответствующая задача имеет вид:

$$\frac{\partial y}{\partial x} = f(x,y), x_0 < x, \qquad (3)$$

где $y = (y_1, y_2, ..., y_n)^T$ вектор – столбец, а $f: R^{n+1} \to R^n$.
Дополнительные (начальные) условия задаются в точке x = x0:

$$y_1(x_0) = y_{10}, y(x_0) = y_{20}, \dots , y_n(x_0) = y_{n0} \qquad (4)$$

также предполагается, что правые части уравнений из (3) заданы так, что это гарантирует существование и единственность решения задачи Коши (3) – (4), но уже для системы обыкновенных дифференциальных уравнений первого порядка в форме, разрешенной относительно производных неизвестных функций.

Цели и задачи практической работы.

1) Решить задачу Коши (1)- (2) (или (3)-(4)) наиболее известными и широко используемыми на практике методами Рунге-Кутта второго и четвертого порядка точности, аппроксимировав дифференциальную задачу соответствующей разностной схемой(на равномерной сетке); полученное конечно- разностное уравнение (или

уравнения), представляющее фактически некоторую рекуррентную формулу, просчитать численно;

2) Найти численное решение задачи;

3) Найденное разностное решение сравнить с точным решением дифференциального уравнения (подобрать специальные тесты, где аналитические решения находятся в классе элементарных функций, при проверке можно использовать ресурсы on-line системы http://www.wolframalpha-ru.com).

Алгоритм решения.

Итак, пусть нам задана система обыкновенных дифференциальных уравнений первого порядка в форме, разрешенной относительно производных неизвестных функций.

$$\frac{\partial y}{\partial x} = f(x, y), x_0 < x,$$

где $y = (y_1, y_2, \ldots, y_n)^T$ вектор – столбец, а $f: R^{n+1} \to R^n$.

С дополнительными условиями точке $x = x_0$:

$$y_1(x_0) = y_{10}, y(x_0) = y_{20}, \ldots, y_n(x_0) = y_{n0}$$

Дискретизируем задачу, т.е. вводим сетку по переменной x:

$$h := \frac{b-a}{n}, \qquad i:=0 \ldots n$$

$$x_i = a + i \cdot h;$$

$$y_i := y(x_i);$$

1. **Метод Рунге-Кутта второго порядка точности.**

Формула рекуррентного соотношения метода Рунге-Кутта второго порядка точности имеет вид

$$y_{i+1} = y_i + \left[(1-\alpha) f(x_i, y_i) + \alpha f\left(x_i + \frac{h}{2\alpha}, y_i + f(x_i, y_i) \right) \right] h.$$

В общем случае *yi* понимаем как вектор $y = (y_1, y_2, \ldots, y_n)^T$, а $f: R^{n+1} \to R^n$.

Зная значения функции $y(x) = (y_1(x), y_2(x), \ldots, y_n(x))^T$ в точке $x = x_0$:

$$y_1(x_0) = y_{10}, y(x_0) = y_{20}, \ldots, y_n(x_0) = y_{n0}$$

находим значения функции y(x) в точках $x_i = a + i \cdot h$ $(i = 0 \ldots n)$.

2. **Метод Рунге-Кутта четвертого порядка точности.**

Формула рекуррентного соотношения метода Рунге-Кутта четвертого порядка точности имеет вид

$$y_{i+1} = y_i + \frac{1}{6}\left[k_1 + 2k_2 + 2k_3 + k_4 \right] h,$$

где

$$k_1 = f(x_i, y_i), \quad k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right), \; k_4 = f(x_i + h, y_i + hk_3)$$

Аналогично, как в предыдущем пункте, зная значения функции $y(x)$ в точке $x = x_0$:

$y_1(x_0) = y_{10}, y(x_0) = y_{20}, \dots, y_n(x_0) = y_{n0}$

находим значения функции y(x) в точках $x_i = a + i \cdot h$ $(i = 0 \dots n)$.

Описание программы

Программе на стандартном потоке ввода задаются :
x_0 — начальная точка
n — количество уравнений в системе
y_{01} y_{02} …. y_{0n} – n начальных условий
l — длина рассматриваемого отрезка
k — число равное числу точек в сетке
α - параметр

Программа выводит ответ в следующем виде
x 1-го порядка 2-го порядка аналитическая функция
x = …. y = …. | y = …. | y = ….
y2= … | y2 = … | y2 = ….
…...............................
yn = … | yn = …. | yn = …

Файл «Runge-Kutta.c» содержит код программы

Файл «functions.c» содержит функции задающие систему уравнений, и ее аналитический ответ. Этот файл формируется на основе заданной системы дифференциальных уравнений. И исполняемый файл перекомпилируется заново командой gcc Runge-Kutta.c -o Runge-Kutta.

Программа написана в соответствии с приведенным выше алгоритмом и содержит комментарии в достаточном для ее понимая количестве.

Правильность работы программы подтверждается системой тестов, которые были проверены с помощью специализированного программного обеспечения (http://www.wolframalpha.com).

Выводы.

Мной был изучен метод прогонки решения краевой задачи для дифференциального уравнения второго порядка.

Приложение 1

Текст программы

Файл «Runge-Kutta.c»

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "functions.c"

double *hq(double *q, double h, double tmp, double a, int cnt, double *ans)
{	/* реализация веторной операции ans = q + h * tmp / 2 * a, где ans и q — векторы */
	int i;
	for (i = 0; i < cnt; i++)
	{
		ans[i] = q[i] + h * tmp / (2 * a);
	}
	return ans;
}

double *iter(double *y0, double x0, double a, double h, double *y, int cnt)
{ // итерация метода Рунге-Кутта второго порядка
	int i;
	double tmp;
	double *hlp = malloc (cnt * sizeof(double));
	for (i = 0; i < cnt; i++)
	{
		tmp = func[i](x0, y0);
		y[i] = y0[i] + ((1 - a) * tmp + a * func[i](x0 + h / (2 * a),  hq(y0, h,tmp, a, cnt, hlp))) * h;
	}
	free(hlp);
	return y;
}

double *hn(double *a, int cnt, double h, double k, double *ans)
{	// реализация веторной операции ans = a + h * k / 2, где ans и a - векторы
	int i;
	for (i = 0; i < cnt; i++)
	{
		ans[i] = a[i] + h * k / 2;
	}
	return ans;
}

double *hm(double *a, int cnt, double h, double k, double *ans)
{
	// реализация веторной операции ans = a + h * k, где ans и a - векторы
	int i;
	for (i = 0; i < cnt; i++)
	{
		ans[i] = a[i] + h * k;
	}
```

```
		return ans;
	}

	double *iter2(double *y0, double x0, double h, double *y, int cnt)
	{ // итерация метода Рунге-Кутта четвертого порядка
		int i;
		double k1,k2,k3,k4;
		double *tmp = malloc (cnt * sizeof(double));
		for (i = 0; i < cnt; i++)
		{
			k1 = func[i](x0, y0);
			k2 = func[i](x0 + h/2, hn(y0, cnt, h, k1, tmp));
			k3 = func[i](x0 + h/2, hn(y0, cnt, h, k2, tmp));
			k4 = func[i](x0 + h, hm(y0, cnt, h, k3, tmp));
			y[i] = y0[i] + (k1 + 2 * k2 + 2 * k3 + k4) * h / 6;
		}
		free(tmp);
		return y;
	}

	void Runge_Kutta(double x, double *y0, int cnt, double l, double a, int n)
	{
		int i, j;
		double *y = malloc(cnt * sizeof(double)), h = l / n;
		double *u = malloc(cnt * sizeof(double));
		double *u0 = malloc(cnt * sizeof(double));
		memcpy(u0, y0, cnt * sizeof(double));
		printf("x = %5.2lf, y  = %9lf | y  = %9lf | y  = %9lf\n", x, y0[0], u0[0], proof[0](x));
		for (i = 1; i < cnt; i++)
			printf("          y%d = %9lf | y%d = %9lf | y%d = %9lf\n",i + 1, y0[i], i + 1,
u0[i], i + 1, proof[0](x));
		// y - для метода Рунге-Кутта второго порядка
		// u - для метода Рунге Кутта четвертого порядка
		for (i = 0; i < n; i++)
		{
			y = iter(y0, x, a, h, y, cnt); //итерация для второго порядка
			u = iter2(u0, x, h, u, cnt); //итерация для четвертого порядка
			x += h;
			printf("x = %5.2lf, ", x);
			printf("y  = %9lf | y  = %9lf | y  =  %9lf\n", y[0], u[0], proof[0](x));
			for (j = 1; j < cnt; j++)
			{
				printf("          y%d = %9lf | y%d = %9lf | y%d = %9lf\n", j+1, y[j],
j+1, u[j], j + 1, proof[0](x));
			}
			memcpy(y0, y, cnt * sizeof(double));
			memcpy(u0, u, cnt * sizeof(double));
		}
		free(y);
		free(u);
		free(u0);
	}
```

```
int main(void)
{
	double a;
	double x0, *y0, l;
	int cnt, n;
	// считываем все параметры
	scanf("%lf", &x0);
	scanf("%d", &cnt);
	y0 = malloc(cnt * sizeof(double));
	func = malloc(cnt * sizeof(double));
	proof = malloc(cnt * sizeof(double));
	int i;
	for (i = 0; i < cnt; i++)
	{
		scanf("%lf", &y0[i]);
	}
	scanf("%lf", &l);
	scanf("%d", &n);
	scanf("%lf", &a);
	deffunc(); // определяем функции
	printf("   x         1-го пордяка | 2-го порядка   | Аналит.реш.\n");
	Runge_Kutta(x0, y0, cnt, l, a, n);
	free(y0);
	free(func);
	return 0;
}
```

Приложение 2

Система тестов, подтверждающая правильность решения дифференциального уравнения первого порядка, разрешенного относительно производной

Прим. Для всех тестов α взят равным единице

I. Тесты из таблицы 1

1)

y'(x) = 3 — y — x , x0 = 0, n = 1, y0 = 0, l = 10, k = 35

Аналитечское решение : $y=4-x-4\,e^{-x}$

Файл «functions.c», соответствующий данному тесту:

```
#include <math.h>

typedef double(*func_t)(double , double *);
typedef double(*proof_t)(double );
func_t *func;
proof_t *proof;

double function(double x, double *y)
{
	return 3.0 - y[0] - x;
}

double function_proof(double x)
{
	return 4.0 - x - 4.0 * exp(-x);
}

void deffunc(void)
{
	func[0] = function;
	proof[0] = function_proof;
}
```

Ответ программы:

```
  x       ,   1-го пордяка | 2-го порядка   | Аналит.реш.
x =  0.00, y  =  0.000000 | y  =  0.000000 | y  =  0.000000
x =  0.20, y  =  0.520000 | y  =  0.525067 | y  =   0.525077
x =  0.40, y  =  0.910400 | y  =  0.918703 | y  =   0.918720
x =  0.60, y  =  1.194528 | y  =  1.204733 | y  =   1.204753
x =  0.80, y  =  1.391513 | y  =  1.402661 | y  =   1.402684
x =  1.00, y  =  1.517041 | y  =  1.528459 | y  =   1.528482
x =  1.20, y  =  1.583973 | y  =  1.595200 | y  =   1.595223
x =  1.40, y  =  1.602858 | y  =  1.613590 | y  =   1.613612
x =  1.60, y  =  1.582344 | y  =  1.592394 | y  =   1.592414
x =  1.80, y  =  1.529522 | y  =  1.538786 | y  =   1.538804
x =  2.00, y  =  1.450208 | y  =  1.458642 | y  =   1.458659
x =  2.20, y  =  1.349170 | y  =  1.356772 | y  =   1.356787
x =  2.40, y  =  1.230320 | y  =  1.237114 | y  =   1.237128
x =  2.60, y  =  1.096862 | y  =  1.102894 | y  =   1.102906
x =  2.80, y  =  0.951427 | y  =  0.956749 | y  =   0.956760
x =  3.00, y  =  0.796170 | y  =  0.800842 | y  =   0.800852
x =  3.20, y  =  0.632860 | y  =  0.636943 | y  =   0.636951
x =  3.40, y  =  0.462945 | y  =  0.466500 | y  =   0.466507
x =  3.60, y  =  0.287615 | y  =  0.290699 | y  =   0.290705
x =  3.80, y  =  0.107844 | y  =  0.110512 | y  =   0.110517
x =  4.00, y  = -0.075568 | y  = -0.073267 | y  =  -0.073263
x =  4.20, y  = -0.261966 | y  = -0.259986 | y  =  -0.259982
x =  4.40, y  = -0.450812 | y  = -0.449113 | y  =  -0.449109
x =  4.60, y  = -0.641666 | y  = -0.640210 | y  =  -0.640207
x =  4.80, y  = -0.834166 | y  = -0.832921 | y  =  -0.832919
x =  5.00, y  = -1.028016 | y  = -1.026954 | y  =  -1.026952
x =  5.20, y  = -1.222973 | y  = -1.222068 | y  =  -1.222066
x =  5.40, y  = -1.418838 | y  = -1.418068 | y  =  -1.418066
x =  5.60, y  = -1.615447 | y  = -1.614793 | y  =  -1.614791
x =  5.80, y  = -1.812667 | y  = -1.812111 | y  =  -1.812110
x =  6.00, y  = -2.010387 | y  = -2.009916 | y  =  -2.009915
x =  6.20, y  = -2.208517 | y  = -2.208119 | y  =  -2.208118
x =  6.40, y  = -2.406984 | y  = -2.406647 | y  =  -2.406646
x =  6.60, y  = -2.605727 | y  = -2.605442 | y  =  -2.605441
x =  6.80, y  = -2.804696 | y  = -2.804456 | y  =  -2.804455
x =  7.00, y  = -3.003851 | y  = -3.003648 | y  =  -3.003648
x =  7.20, y  = -3.203158 | y  = -3.202987 | y  =  -3.202986
x =  7.40, y  = -3.402589 | y  = -3.402445 | y  =  -3.402445
x =  7.60, y  = -3.602123 | y  = -3.602002 | y  =  -3.602002
x =  7.80, y  = -3.801741 | y  = -3.801639 | y  =  -3.801639
x =  8.00, y  = -4.001428 | y  = -4.001342 | y  =  -4.001342
x =  8.20, y  = -4.201171 | y  = -4.201099 | y  =  -4.201099
x =  8.40, y  = -4.400960 | y  = -4.400900 | y  =  -4.400899
x =  8.60, y  = -4.600787 | y  = -4.600737 | y  =  -4.600736
x =  8.80, y  = -4.800645 | y  = -4.800603 | y  =  -4.800603
x =  9.00, y  = -5.000529 | y  = -5.000494 | y  =  -5.000494
x =  9.20, y  = -5.200434 | y  = -5.200404 | y  =  -5.200404
x =  9.40, y  = -5.400356 | y  = -5.400331 | y  =  -5.400331
x =  9.60, y  = -5.600292 | y  = -5.600271 | y  =  -5.600271
x =  9.80, y  = -5.800239 | y  = -5.800222 | y  =  -5.800222
x = 10.00, y  = -6.000196 | y  = -6.000182 | y  =  -6.000182
```

2)

$y' = y - yx$, x0 = 0, n = 1, y0 = 5, l = 7, k = 35

Аналитечское решение : $5e^{\frac{-1}{2}x(-2+x)}$

Файл «functions.c», соответствующий данному тесту:

```
#include <math.h>

typedef double(*func_t)(double , double *);
typedef double(*proof_t)(double );

func_t *func;
proof_t *proof;

double function(double x, double *y)
{return y[0] - y[0] * x;}


double function_proof(double x)
{return 5 * exp(-x*(-2 + x) / 2); }

void deffunc(void)
{
	func[0] = function;
	proof[0] = function_proof;
}
```

Ответ программы:

x	1-го пордяка	2-го порядка	Аналит.реш.
x = 0.00,	y = 5.000000	y = 5.000000	y = 5.000000
x = 0.20,	y = 5.990000	y = 5.986076	y = 5.986087
x = 0.40,	y = 6.895688	y = 6.885622	y = 6.885639
x = 0.60,	y = 7.626631	y = 7.609787	y = 7.609808
x = 0.80,	y = 8.102533	y = 8.080350	y = 8.080372
x = 1.00,	y = 8.267824	y = 8.243583	y = 8.243606
x = 1.20,	y = 8.102468	y = 8.080349	y = 8.080372
x = 1.40,	y = 7.626043	y = 7.609786	y = 7.609808
x = 1.60,	y = 6.893943	y = 6.885619	y = 6.885639
x = 1.80,	y = 5.986700	y = 5.986070	y = 5.986087
x = 2.00,	y = 4.995302	y = 4.999992	y = 5.000000
x = 2.20,	y = 4.006232	y = 4.012604	y = 4.012594
x = 2.40,	y = 3.089606	y = 3.093958	y = 3.093917
x = 2.60,	y = 2.292488	y = 2.292116	y = 2.292030
x = 2.80,	y = 1.637753	y = 1.631538	y = 1.631399
x = 3.00,	y = 1.127429	y = 1.115843	y = 1.115651
x = 3.20,	y = 0.748613	y = 0.733271	y = 0.733035
x = 3.40,	y = 0.480011	y = 0.463016	y = 0.462753
x = 3.60,	y = 0.297607	y = 0.280943	y = 0.280674
x = 3.80,	y = 0.178683	y = 0.163816	y = 0.163562
x = 4.00,	y = 0.104065	y = 0.091801	y = 0.091578
x = 4.20,	y = 0.058901	y = 0.049447	y = 0.049264
x = 4.40,	y = 0.032466	y = 0.025603	y = 0.025462
x = 4.60,	y = 0.017467	y = 0.012746	y = 0.012644
x = 4.80,	y = 0.009195	y = 0.006102	y = 0.006033
x = 5.00,	y = 0.004748	y = 0.002810	y = 0.002765
x = 5.20,	y = 0.002412	y = 0.001245	y = 0.001218
x = 5.40,	y = 0.001209	y = 0.000531	y = 0.000515
x = 5.60,	y = 0.000600	y = 0.000218	y = 0.000210
x = 5.80,	y = 0.000295	y = 0.000086	y = 0.000082
x = 6.00,	y = 0.000145	y = 0.000033	y = 0.000031
x = 6.20,	y = 0.000071	y = 0.000012	y = 0.000011
x = 6.40,	y = 0.000035	y = 0.000004	y = 0.000004
x = 6.60,	y = 0.000017	y = 0.000001	y = 0.000001
x = 6.80,	y = 0.000009	y = 0.000000	y = 0.000000
x = 7.00,	y = 0.000004	y = 0.000000	y = 0.000000

II Тесты проверенные с помощью http://www.wolframalpha.com.

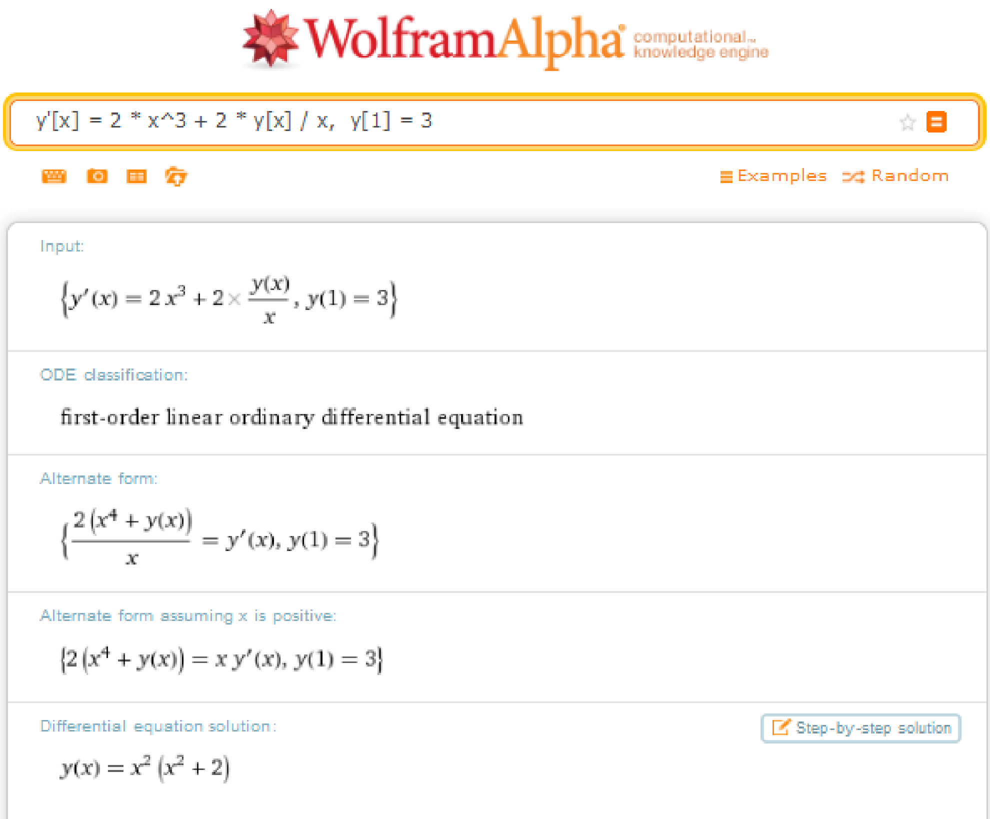
3)

$y' = 2x^3 + 2y / x$, x0 = 1, n = 1, y0 = 3, l = 7, n = 35

Файл «functions.c», соответствующий данному тесту:

```
#include <math.h>

typedef double(*func_t)(double , double *);
typedef double(*proof_t)(double );

func_t *func;
proof_t *proof;

double function(double x, double *y)
{return 2.0 * x* x * x + 2.0 * y[0] / x;;}


double function_proof(double x)
{return (2.0 + x * x) * x * x; }

void deffunc(void)
{
	func[0] = function;
	proof[0] = function_proof;
}
```

Ответ http://www.wolframalpha.com



Ответ программы

x	1-го порядка	2-го порядка	Аналит.реш.
x = 1.00,	y = 3.000000	y = 3.000000	y = 3.000000
x = 1.20,	y = 4.914218	y = 4.953242	y = 4.953600
x = 1.40,	y = 7.663435	y = 7.760827	y = 7.761600
x = 1.60,	y = 11.495305	y = 11.672352	y = 11.673600
x = 1.80,	y = 16.696132	y = 16.975815	y = 16.977600
x = 2.00,	y = 23.590818	y = 23.997615	y = 24.000000
x = 2.20,	y = 32.542818	y = 33.102552	y = 33.105600
x = 4.60,	y = 485.277683	y = 490.049636	y = 490.065600
x = 4.80,	y = 571.559556	y = 576.904146	y = 576.921600
x = 5.00,	y = 669.026743	y = 674.980993	y = 675.000000
x = 5.20,	y = 778.619520	y = 785.220976	y = 785.241600
x = 5.40,	y = 901.316586	y = 908.603296	y = 908.625600
x = 5.60,	y = 1038.135058	y = 1046.145552	y = 1046.169600
x = 5.80,	y = 1190.130469	y = 1198.903745	y = 1198.929600
x = 6.00,	y = 1358.396771	y = 1367.972274	y = 1368.000000

<table>
<tr><td>x = 2.40, y = 43.954115 | y = 44.693825 | y = 44.697600
x = 2.60, y = 58.265196 | y = 59.213035 | y = 59.217600
x = 2.80, y = 75.955039 | y = 77.140181 | y = 77.145600
x = 3.00, y = 97.541093 | y = 98.993664 | y = 99.000000
x = 3.20, y = 123.579278 | y = 125.330283 | y = 125.337600
x = 3.40, y = 154.663967 | y = 156.745239 | y = 156.753600
x = 3.60, y = 191.427984 | y = 193.872130 | y = 193.881600
x = 3.80, y = 234.542597 | y = 237.382959 | y = 237.393600
x = 4.00, y = 284.717515 | y = 287.988123 | y = 288.000000
x = 4.20, y = 342.700880 | y = 346.436424 | y = 346.449600
x = 4.40, y = 409.279268 | y = 413.515062 | y = 413.529600</td>
<td>x = 6.20, y = 1544.066329 | y = 1554.483940 | y = 1554.513600
x = 6.40, y = 1748.309924 | y = 1759.609942 | y = 1759.641600
x = 6.60, y = 1972.336749 | y = 1984.559881 | y = 1984.593600
x = 6.80, y = 2217.394411 | y = 2230.581756 | y = 2230.617600
x = 7.00, y = 2484.768926 | y = 2498.961967 | y = 2499.000000
x = 7.20, y = 2775.784725 | y = 2791.025315 | y = 2791.065600
x = 7.40, y = 3091.804645 | y = 3108.135000 | y = 3108.177600
x = 7.60, y = 3434.229938 | y = 3451.692621 | y = 3451.737600
x = 7.80, y = 3804.500260 | y = 3823.138179 | y = 3823.185600
x = 8.00, y = 4204.093681 | y = 4223.950073 | y = 4224.000000</td></tr>
</table>

III Тесты из таблицы 2

4)

$$u' = \sin(x + u) — 1.1\, v, \; x0 = 0, \; n = 2, \; y10 = 0.5, \; y20 = 1, \; l = 10, \; k = 50$$
$$v' = 2.5\, u — (x + v)^2$$

http://www.wolframalpha.com Не посчитал аналитечское решение системы.

Ответ программы :

<table>
<tr><td>x 1-го пордяка | 2-го порядка
x = 1.00, y = 0.500000 | y = 0.500000
y2 = 1.000000 | y2 = 1.000000
x = 1.29, y = 0.475560 | y = 0.474264
y2 = 0.201531 | y2 = 0.196437
x = 1.57, y = 0.636353 | y = 0.641458
y2 = -0.192353 | y2 = -0.190641
x = 1.86, y = 0.823187 | y = 0.841170
y2 = -0.393270 | y2 = -0.385497
x = 2.14, y = 0.962188 | y = 0.992823
y2 = -0.540442 | y2 = -0.520790
x = 2.43, y = 1.047939 | y = 1.083766
y2 = -0.717037 | y2 = -0.685000
x = 2.71, y = 1.104887 | y = 1.139064
y2 = -0.936302 | y2 = -0.898349
x = 3.00, y = 1.161209 | y = 1.190787
y2 = -1.180355 | y2 = -1.142380
x = 3.29, y = 1.243209 | y = 1.267851
y2 = -1.428559 | y2 = -1.392910
x = 3.57, y = 1.380331 | y = 1.400597
y2 = -1.663224 | y2 = -1.629760
x = 3.86, y = 1.614457 | y = 1.631040
y2 = -1.866991 | y2 = -1.834192
x = 4.14, y = 2.009911 | y = 2.022660
y2 = -2.019201 | y2 = -1.984025
x = 4.43, y = 2.638108 | y = 2.639820
y2 = -2.095557 | y2 = -2.052054
x = 4.71, y = 3.440708 | y = 3.421671
y2 = -2.085661 | y2 = -2.024293
x = 5.00, y = 4.126111 | y = 4.124041
y2 = -2.049474 | y2 = -1.959111
x = 5.29, y = 4.596721 | y = 4.618121
y2 = -2.093590 | y2 = -1.970805
x = 5.57, y = 4.953434 | y = 4.970887
y2 = -2.219476 | y2 = -2.083133
x = 5.86, y = 5.297286 | y = 5.293056
y2 = -2.388991 | y2 = -2.250157
x = 6.14, y = 5.728821 | y = 5.690815
y2 = -2.573564 | y2 = -2.428568</td>
<td>x = 6.43, y = 6.371694 | y = 6.281101
y2 = -2.751315 | y2 = -2.584224
x = 6.71, y = 7.303282 | y = 7.135593
y2 = -2.908398 | y2 = -2.686082
x = 7.00, y = 8.261316 | y = 8.071968
y2 = -3.061124 | y2 = -2.728514
x = 7.29, y = 8.985185 | y = 8.818149
y2 = -3.262139 | y2 = -2.774431
x = 7.57, y = 9.613411 | y = 9.384166
y2 = -3.527060 | y2 = -2.878096
x = 7.86, y = 10.336529 | y = 9.923797
y2 = -3.853992 | y2 = -3.028375
x = 8.14, y = 11.386390 | y = 10.616403
y2 = -4.264465 | y2 = -3.195854
x = 8.43, y = 12.773512 | y = 11.604023
y2 = -4.839063 | y2 = -3.357314
x = 8.71, y = 13.962382 | y = 12.717160
y2 = -5.671329 | y2 = -3.523218
x = 9.00, y = 15.207901 | y = 13.636989
y2 = -6.486011 | y2 = -3.761273
x = 9.29, y = 16.996988 | y = 14.430277
y2 = -7.139836 | y2 = -4.136156
x = 9.57, y = 19.007814 | y = 15.387720
y2 = -8.078588 | y2 = -4.751068
x = 9.86, y = 20.877111 | y = 16.814532
y2 = -8.652784 | y2 = -5.908085
x = 10.14, y = 23.388230 | y = 18.500994
y2 = -9.675121 | y2 = -7.922444
x = 10.43, y = 25.716112 | y = 20.437983
y2 = -9.821068 | y2 = -9.015620
x = 10.71, y = 28.359413 | y = 23.014021
y2 = -12.830758 | y2 = -9.993156
x = 11.00, y = 31.569231 | y = 25.585325
y2 = -1.931601 | y2 = -11.454748</td></tr>
</table>

ПРАКТИЧЕСКАЯ РАБОТА №2 (2)

Цель работы.

Освоить метод прогонки решения краевой задачи для дифференциального уравнения второго порядка.

Постановка задачи.

Рассмотрим линейное дифференциальное уравнение

$$y'' + p(x)\cdot y' + q(x)\cdot y = f(x) \qquad (1)$$

на интервале [a,b] с дополнительными условиями в граничных точках

$$\begin{aligned} &ct1\cdot y(a) + ct2\cdot y'(a) = ct \\ &dt1\cdot y(b) + dt2\cdot y'(b) = dt \end{aligned} \qquad (2)$$

Цели и задачи практической работы.

3. Решить краевую задачу (1)–(2) методом конечных разностей, аппроксимировав ее разностной схемой второго порядка точности (на равномерной сетке); полученную систему конечно-разностных уравнений решить методом прогонки;

4. Найти разностное решение задачи;

5. Найденное разностное решение сравнить с точным решением дифференциального уравнения (подобрать специальные тесты, где аналитические решения находятся в классе элементарных функций, при можно использовать ресурсы on-line системы http://www.wolframalpha-ru.com).

Алгоритм решения.

1. Дискретизируем задачу, т.е. вводим сетку по переменной x:

$$h := \frac{b-a}{n}, \qquad i:=0 \ldots n$$
$$x_i = a + i\cdot h;$$
$$y_i := y(x_i);$$

2. Заменяем исходное уравнение (1) конечно-разностным во внутренних узлах:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + p_i\cdot\frac{y_{i+1} - y_{i-1}}{2h} + q_i\cdot y_i = f_i$$

Это уравнение приводим к каноническому трехдиагональному виду

$$a_i\cdot y_{i-1} - b_i\cdot y_i + c_i\cdot y_{i+1} = d_i \ , \qquad (3)$$

где

$$a_i = \frac{1}{h^2} - \frac{p_i}{2h}, \quad b_i = \frac{2}{h^2} - q_i, \quad c_i = \frac{1}{h^2} + \frac{p_i}{2h}, \quad d_i = -f_i$$

3. Линейную систему (3) решаем методом прогонки. Ищем в виде

$$y_{j-1} := y_j \cdot \xi_j + \eta_j$$

$$j = n, n - 1, ...2 \qquad (4)$$

где ξ_j , η_j - прогоночные коэффициенты, которые необходимо предварительно вычислить. Решение реализуется в два этапа.

3.1. Прямой ход − от левого края заданного интервала [a,b] до правого в узлах сетки вычисляются ξ_j , η_j , $j = 1, . . .,n$

3.2. Обратный ход − от правого края до левого по формуле (4) в тех же узлах находится искомое решение.

Прямой ход реализуется рекуррентными формулами для прогоночных коэффициентов, которые получаются из (3) и (4) при k=1..n-1:

$$\begin{bmatrix} \xi_{k+1} \\ \eta_{k+1} \end{bmatrix} := \begin{bmatrix} \dfrac{c_k}{b_k - a_k \cdot \xi_k} \\ \dfrac{\eta_k \cdot a_k - d_k}{b_k - a_k \cdot \xi_k} \end{bmatrix},$$

Начальные значения коэффициентов для этих рекуррентных формул можно найти из (4) и левого краевого условия (2):

$$\xi_1 := \frac{-ct2}{ct1 \cdot h - ct2} \qquad \eta_1 := \frac{ct \cdot h}{ct1 \cdot h - ct2}$$

После того как получены прогоночные коэффициенты, можно приступать к обратному ходу по формуле (4), но предварительно необходимо найти значение y_n . Из (4) и правого краевого условия (2) получаем

$$y_n = \frac{dt2 \cdot \eta_n + dt \cdot h}{dt2 \cdot (1 - \xi_n) + dt1 \cdot h},$$

а, теперь, собственно обратный ход - искомую функцию находим по рекуррентной формуле

$$j := n .. 1$$

$$y_{j-1} := y_j \cdot \xi_j + \eta_j$$

Описание программы

Программе на стандартный поток ввода задаются:

Коэффициенты ct1, ct2, ct, dt1, dt2, dt

a — координата правого конца, b — координата левого конца

Программа выводит ответ в виде

….

y(...) = ….. {….}

…..

где в фигурных скобочках указано значения аналитического решения уравнения при том же аргументе.

Файл «sweep.c» Содержит исходный код программы

Файл «functions2.c» содержит функции задающие систему уравнений, и ее аналитический ответ. Этот файл формируется на основе заданной системы дифференциальных уравнений. И исполняемый файл перекомпилируется заново командой gcc sweep.c -o sweep.

Программа написана в соответствии с приведенным выше алгоритмом и содержит комментарии в достаточном для ее понимая количестве.

Правильность работы программы подтверждается системой тестов, которые были проверены с помощью специализированного программного обеспечения (http://www.wolframalpha.com).

Выводы.

Мной был изучен метод прогонки решения краевой задачи для дифференциального уравнения второго порядка.Файл «functions.c», соответствующий данному тесту:

Приложение 3

Текст прграммы

Файл «sweep.c»

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "progfunc.c"

void solut(double ct1, double ct2, double ct, double dt1, double dt2, double dt, double start, double end, int n)
{
	double *s, *w, *y;
	s = malloc((n + 1) * sizeof(double));
	w = malloc((n + 1) * sizeof(double));
	y = malloc((n + 1) * sizeof(double));
	double h = (end - start) / n, x = start;
	int i;
	double c, b, a;
	//находим начальные значения прогоночных коэффициентов
	s[1] = -ct2 / (ct1 * h - ct2);
	w[1] = ct * h / (ct1 * h - ct2);
	for (i = 1; i < n; i++)
	{// Прямой ход метода прогонки. Вычисляются прогоночные коэффициенты
		x += h;
		a = (1.0 / (h * h)) - (p(x) / (2.0 * h));
		b = (2.0 / (h * h)) - q(x);
		c = (1.0 / (h * h)) + (p(x) / (2.0 * h));
		s[i + 1] = c / (b - a * s[i]);
		w[i + 1] = (w[i] * a - f(x)) / (b - a * s[i]);
	}

	y[n] = (dt2 * w[n] + dt * h) / (dt2 * (1.0 - s[n]) + dt1 * h);
	for (i = n ; i > 0; i--)
	{
		// Обратный ход метода прогонки. По прогоночным коэффициентам,
используя
		//рекурентное соотношение находится вектор y
		y[i - 1] = y[i] * s[i] + w[i];
	}
	x = start;
	for (i = 0;  i <= n; i++)
	{
		printf("y(%lf) = %lf {%lf}\n", x, y[i], proof(x));
		x += h;
	}
}

int main (void)
{
	double ct1, ct2, ct, dt1, dt2, dt, a, b;
	int n;
	// считываем параметры
```

```
    scanf("%lf %lf %lf", &ct1, &ct2, &ct);
    scanf("%lf %lf %lf", &dt1, &dt2, &dt);
    scanf("%lf %lf %d", &a, &b, &n);
    deffunc(); // определяем функции
    solut(ct1, ct2, ct, dt1, dt2, dt, a, b, n);
    return 0;
}
```

Приложение 4

Система тестов подтверждающая решение дифференциального уравнения второго порядка.

I. Тесты подтвержденные с помощью http://www.wolframalpha.com.

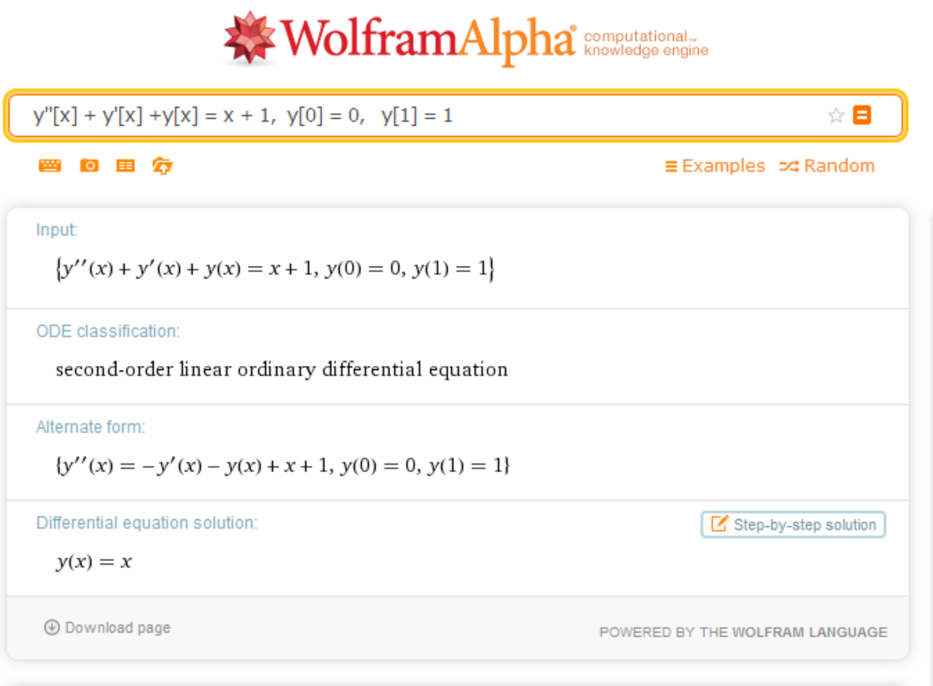
1)

y’’ + y’+y = x + 1
с дополнительными условиями:
y(0) = 0;
y(1) = 1;   n = 100

Файл «functions2.c», соответствующий данному тесту:

```
#include <math.h>
typedef double (*func_t) (double x);
func_t p, q, f, proof;
double pz(double x)
{return 2;}
double qz(double x)
{return 2;}
double fz(double x)
{return x* exp(-x);}
double pr(double x)
{return exp(-x) * (x - sin(x));}
void deffunc(void)
{       p = pz;
        q = qz;
        f = fz;
        proof = pr;}
```

Ответ http://www.wolframalpha.com.



Ответ программы

y(0.510000) = 0.013104 {0.013104} y(0.520000) = 0.013745 {0.013745} y(0.530000) = 0.014401 {0.014401} y(0.540000) = 0.015072 {0.015072} y(0.550000) = 0.015758 {0.015758} y(0.560000) = 0.016458 {0.016459} y(0.570000) = 0.017173 {0.017174} y(0.580000) = 0.017903 {0.017903} y(0.590000) = 0.018646 {0.018647} y(0.600000) = 0.019404 {0.019405} y(0.610000) = 0.020175 {0.020176} y(0.620000) = 0.020960 {0.020961} y(0.630000) = 0.021759 {0.021759} y(0.640000) = 0.022570 {0.022571} y(0.650000) = 0.023394 {0.023395} y(0.660000) = 0.024231 {0.024232}	y(0.670000) = 0.025080 {0.025081} y(0.680000) = 0.025942 {0.025942} y(0.690000) = 0.026815 {0.026816} y(0.700000) = 0.027700 {0.027701} y(0.710000) = 0.028597 {0.028597} y(0.720000) = 0.029504 {0.029505} y(0.730000) = 0.030423 {0.030423} y(0.740000) = 0.031352 {0.031352} y(0.750000) = 0.032291 {0.032292} y(0.760000) = 0.033241 {0.033241} y(0.770000) = 0.034200 {0.034200} y(0.780000) = 0.035169 {0.035169} y(0.790000) = 0.036147 {0.036147} y(0.800000) = 0.037134 {0.037134} y(0.810000) = 0.038130 {0.038130} y(0.820000) = 0.039134 {0.039134}	y(0.830000) = 0.040146 {0.040146} y(0.840000) = 0.041166 {0.041167} y(0.850000) = 0.042194 {0.042194} y(0.860000) = 0.043229 {0.043229} y(0.870000) = 0.044271 {0.044271} y(0.880000) = 0.045319 {0.045320} y(0.890000) = 0.046374 {0.046375} y(0.900000) = 0.047435 {0.047436} y(0.910000) = 0.048502 {0.048503} y(0.920000) = 0.049575 {0.049575} y(0.930000) = 0.050653 {0.050653} y(0.940000) = 0.051735 {0.051735} y(0.950000) = 0.052823 {0.052823} y(0.960000) = 0.053914 {0.053915} y(0.970000) = 0.055010 {0.055010} y(0.980000) = 0.056110 {0.056110} y(0.990000) = 0.057213 {0.057213} y(1.000000) = 0.058320 {0.058320}

II Тесты, проверенные вручную

2)

y'' — y = 2x
С дополнительными условиями
y(0) = 0, y(1) = -1. n = 50

Аналитечское решение: $y=\frac{sh(x)}{sh(1)}-2x$

Файл «functions2.c», соответствующий данному тесту:

```
#include <math.h>

typedef double (*func_t) (double x);
func_t p, q, f, proof;

double p2(double x)
{return 0;}

double q2(double x)
{return -1;}

double f2(double x)
{return 2 * x;}

double pr2(double x)
{return  (sinh(x)/sinh(1)) - 2 * x;}

void deffunc(void)
{
	p = p2;
	q = q2;
	f = f2;
	proof = pr2;
}
```

Ответ программы:

y(0.000000) = 0.000000 {0.000000}	y(0.340000) = -0.385080 {-0.385081}	y(0.700000) = -0.754506 {-0.754507}
y(0.020000) = -0.022980 {-0.022981}	y(0.360000) = -0.407008 {-0.407010}	y(0.720000) = -0.773014 {-0.773016}
y(0.040000) = -0.045954 {-0.045954}	y(0.380000) = -0.428811 {-0.428813}	y(0.740000) = -0.791256 {-0.791258}
y(0.060000) = -0.068914 {-0.068914}	y(0.400000) = -0.450482 {-0.450483}	y(0.760000) = -0.809222 {-0.809224}
y(0.080000) = -0.091854 {-0.091854}	y(0.420000) = -0.472013 {-0.472014}	y(0.780000) = -0.826904 {-0.826906}
y(0.100000) = -0.114766 {-0.114766}	y(0.440000) = -0.493396 {-0.493398}	y(0.800000) = -0.844293 {-0.844295}
y(0.120000) = -0.137644 {-0.137645}	y(0.460000) = -0.514625 {-0.514627}	y(0.820000) = -0.861380 {-0.861381}
y(0.140000) = -0.160481 {-0.160482}	y(0.480000) = -0.535692 {-0.535694}	y(0.840000) = -0.878155 {-0.878156}
y(0.160000) = -0.183271 {-0.183271}	y(0.500000) = -0.556589 {-0.556591}	y(0.860000) = -0.894609 {-0.894610}
y(0.180000) = -0.206006 {-0.206006}	y(0.520000) = -0.577309 {-0.577310}	y(0.880000) = -0.910733 {-0.910734}
y(0.200000) = -0.228679 {-0.228680}	y(0.540000) = -0.597843 {-0.597845}	y(0.900000) = -0.926517 {-0.926518}
y(0.220000) = -0.251283 {-0.251284}	y(0.560000) = -0.618185 {-0.618187}	y(0.920000) = -0.941953 {-0.941953}
y(0.240000) = -0.273812 {-0.273813}	y(0.580000) = -0.638326 {-0.638328}	y(0.940000) = -0.957028 {-0.957029}
y(0.260000) = -0.296259 {-0.296260}	y(0.600000) = -0.658258 {-0.658260}	y(0.960000) = -0.971735 {-0.971735}
y(0.280000) = -0.318616 {-0.318617}	y(0.620000) = -0.677974 {-0.677976}	y(0.980000) = -0.986062 {-0.986062}
y(0.300000) = -0.340877 {-0.340878}	y(0.640000) = -0.697465 {-0.697466}	y(1.000000) = -1.000000 {-1.000000}
y(0.320000) = -0.363034 {-0.363035}	y(0.660000) = -0.716722 {-0.716724}	y(0.680000) = -0.735739 {-0.735741}

III Тесты заданные в подварианте

3)

$$y'' + 2x^2\, y' + y = x$$

С дополнительными условиями

2y(0.5) — y'(0.5) = 1

y(0.8) = 3,

n = 100

http://www.wolframalpha.com. Не справился с этим уравнением

Файл «functions2.c», соответствующий данному тесту:

```
#include <math.h>

typedef double (*func_t) (double x);
func_t p, q, f, proof;

double p1(double x)
{return 2 * x * x;}

double q1(double x)
{return 1;}

double f1(double x)
{return x;}

void deffunc(void)
{
	p = p1;
	q = q1;
	f = f1;
}
```

Ответ программы :

y(0.500000) = 2.173946	y(0.602000) = 2.497250	y(0.701000) = 2.770934
y(0.503000) = 2.183990	y(0.605000) = 2.506155	y(0.704000) = 2.778552
y(0.506000) = 2.194003	y(0.608000) = 2.515024	y(0.707000) = 2.786127
y(0.509000) = 2.203986	y(0.611000) = 2.523855	y(0.710000) = 2.793662
y(0.512000) = 2.213938	y(0.614000) = 2.532650	y(0.713000) = 2.801155
y(0.515000) = 2.223859	y(0.617000) = 2.541407	y(0.716000) = 2.808607
y(0.518000) = 2.233749	y(0.620000) = 2.550128	y(0.719000) = 2.816016
y(0.521000) = 2.243608	y(0.623000) = 2.558810	y(0.722000) = 2.823384
y(0.524000) = 2.253435	y(0.626000) = 2.567456	y(0.725000) = 2.830711
y(0.527000) = 2.263230	y(0.629000) = 2.576063	y(0.728000) = 2.837995
y(0.530000) = 2.272993	y(0.632000) = 2.584633	y(0.731000) = 2.845237
y(0.533000) = 2.282725	y(0.635000) = 2.593164	y(0.734000) = 2.852437
y(0.536000) = 2.292424	y(0.638000) = 2.601657	y(0.737000) = 2.859594
y(0.539000) = 2.302090	y(0.641000) = 2.610112	y(0.740000) = 2.866709
y(0.542000) = 2.311724	y(0.644000) = 2.618529	y(0.743000) = 2.873782
y(0.545000) = 2.321325	y(0.647000) = 2.626906	y(0.746000) = 2.880813
y(0.548000) = 2.330893	y(0.650000) = 2.635245	y(0.749000) = 2.887800
y(0.551000) = 2.340427	y(0.653000) = 2.643545	y(0.752000) = 2.894745
y(0.554000) = 2.349929	y(0.656000) = 2.651806	y(0.755000) = 2.901647
y(0.557000) = 2.359396	y(0.659000) = 2.660027	y(0.758000) = 2.908507
y(0.560000) = 2.368830	y(0.662000) = 2.668210	y(0.761000) = 2.915323
y(0.563000) = 2.378230	y(0.665000) = 2.676352	y(0.764000) = 2.922096
y(0.566000) = 2.387595	y(0.668000) = 2.684455	y(0.767000) = 2.928827
y(0.569000) = 2.396926	y(0.671000) = 2.692519	y(0.770000) = 2.935514
y(0.572000) = 2.406223	y(0.674000) = 2.700542	y(0.773000) = 2.942158
y(0.575000) = 2.415485	y(0.677000) = 2.708525	y(0.776000) = 2.948759
y(0.578000) = 2.424712	y(0.680000) = 2.716468	y(0.779000) = 2.955316
y(0.581000) = 2.433904	y(0.683000) = 2.724371	y(0.782000) = 2.961830
y(0.584000) = 2.443060	y(0.686000) = 2.732233	y(0.785000) = 2.968301
y(0.587000) = 2.452182	y(0.689000) = 2.740055	y(0.788000) = 2.974728
y(0.590000) = 2.461267	y(0.692000) = 2.747836	y(0.791000) = 2.981111
y(0.593000) = 2.470317	y(0.695000) = 2.755577	y(0.794000) = 2.987451
y(0.596000) = 2.479331	y(0.698000) = 2.763276	y(0.797000) = 2.993747
y(0.599000) = 2.488309		y(0.800000) = 3.000000