**Θέμα 1: Μέθοδος Runge-Kutta**

**II) Συγγραφή κώδικα υλοποίησης**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <math.h>**

**#define MAX 1.0 /\* max for t \*/**

**#define MIN 0.0 /\* min for t \*/**

**int main(int argc, char \*argv[])**

**{**

**int N=10; //to plh8os tou t**

**double t, xa10t[N], fa10t, xb10t[N], fb10t, xc10t[N], xd10t[N];**

**double x\_euler\_a10[N], x\_euler\_b10[N], x\_euler\_c10[N], x\_euler\_d10[N];**

**double x\_opteuler\_a10[N], x\_opteuler\_b10[N], x\_opteuler\_c10[N], x\_opteuler\_d10[N];**

**double k1, k2, k3, k4;**

**int j;**

**double h, step;**

**h = (MAX-MIN)/N;**

**step = h/2;**

**//arxikes times**

**t = 0;**

**xa10t[0] = x\_euler\_a10[0] = x\_opteuler\_a10[0] = 1.0;**

**xb10t[0] = x\_euler\_b10[0] = x\_opteuler\_b10[0] = 1.0;**

**xc10t[0] = x\_euler\_c10[0] = x\_opteuler\_c10[0] = 0.5;**

**xd10t[0] = x\_euler\_d10[0] = x\_opteuler\_d10[0] = -1.0;**

**fa10t = exp(2\*t) + (t/2);**

**fb10t = - ((pow(t,2)+3)/(t-3));**

**//ektypwsh arxikwn timwn sta antistoixa arxeia**

**printf( "t(i)\t\t Euler\t\t opt.Euler\t Runge\t\t f(i)\n");**

**printf( "%f\t%f\t%f\t%f\t%f\n", t, x\_euler\_a10[0], x\_opteuler\_a10[0], xa10t[0], fa10t);**

**//to loop ypologismou**

**for (j=1; j<=N; j++)**

**{**

**t = MIN + j\*h;**

**//gia thn 1h synarthsh (erwthma II)**

**//times euler**

**k1 = 0.5 - t + (2\*x\_euler\_a10[j-1]);**

**x\_euler\_a10[j] = x\_euler\_a10[j-1] + (h\*k1);**

**//times veltiwmenou euler**

**k1 = 0.5 - t + (2\*x\_opteuler\_a10[j-1]);**

**k2 = 0.5 - (t+h) + (2\*(x\_opteuler\_a10[j-1]+(h\*k1)));**

**x\_opteuler\_a10[j] = x\_opteuler\_a10[j-1] + (h/2)\*(k1+k2);**

**//times runge-kutta**

**k1 = 0.5 - t + 2\*xa10t[j-1];**

**k2 = 0.5 - (t+step) + 2\*(xa10t[j-1]+step\*k1);**

**k3 = 0.5 - (t+step) + 2\*(xa10t[j-1]+step\*k2);**

**k4 = 0.5 - (t+h) + 2\*(xa10t[j-1]+h\*k3);**

**xa10t[j] = xa10t[j-1] + (h/6)\*(k1+2\*k2+2\*k3+k4);**

**//pragmatikes times**

**fa10t = exp(2\*t) + (t/2);**

**//ektypwsh timwn sto arxeio**

**printf( "%f\t%f\t%f\t%f\t%f\n",t, x\_euler\_a10[j], x\_opteuler\_a10[j], xa10t[j], fa10t);**

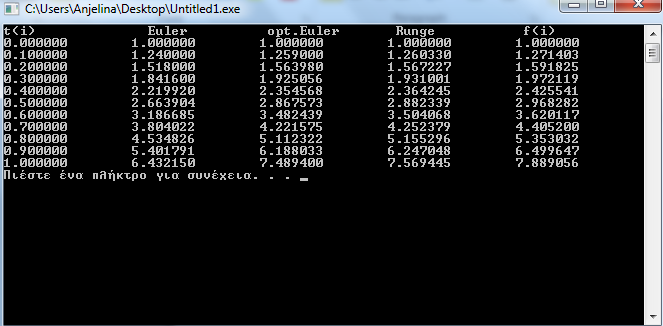
**}**

**system("PAUSE");**

**return 0;**

**}**

**Εμφάνιση αποτελεσμάτων σε κώδικα C**

****

**ΙΙΙ) Συγγραφή κώδικα υλοποίησης**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <math.h>**

**#define MAX 1.0 /\* max for t \*/**

**#define MIN 0.0 /\* min for t \*/**

**FILE \*output10a;**

**FILE \*output10alog;**

**int main(int argc, char \*argv[])**

**{**

**int N=10; //to plh8os tou t**

**double t, xa10t[N], fa10t;**

**double x\_euler\_a10[N];**

**double x\_opteuler\_a10[N];**

**double k1, k2, k3, k4;**

**int j;**

**double h, step;**

**h = (MAX-MIN)/N;**

**step = h/2;**

**//arxikes times**

**t = 0;**

**xa10t[0] = x\_euler\_a10[0] = x\_opteuler\_a10[0] = 1.0;**

**fa10t = exp(2\*t) + (t/2);**

**fb10t = - ((pow(t,2)+3)/(t-3));**

**output10a=fopen("aN10.dat", "w");**

**output10alog=fopen("aN10log.dat", "w");**

**//ektypwsh arxikwn timwn sta antistoixa arxeia**

**fprintf(output10a, "t(i)\t\t Euler\t\t opt.Euler\t Runge\t\t f(i)\n");**

**fprintf(output10a, "%f\t%f\t%f\t%f\t%f\n", t, x\_euler\_a10[0], x\_opteuler\_a10[0], xa10t[0], fa10t);**

**fprintf(output10alog, "t(i)\t\t Euler\t\t opt.Euler\t Runge\n"); fprintf(output10alog, "%f\t0.000000\t0.000000\t0.000000\n", t);**

**//to loop ypologismou**

**for (j=1; j<=N; j++)**

**{**

**t = MIN + j\*h;**

**//gia thn 1h synarthsh (erwthma II)**

**//times euler**

**k1 = 0.5 - t + (2\*x\_euler\_a10[j-1]);**

**x\_euler\_a10[j] = x\_euler\_a10[j-1] + (h\*k1);**

**//times veltiwmenou euler**

**k1 = 0.5 - t + (2\*x\_opteuler\_a10[j-1]);**

**k2 = 0.5 - (t+h) + (2\*(x\_opteuler\_a10[j-1]+(h\*k1)));**

**x\_opteuler\_a10[j] = x\_opteuler\_a10[j-1] + (h/2)\*(k1+k2);**

**//times runge-kutta**

**k1 = 0.5 - t + 2\*xa10t[j-1];**

**k2 = 0.5 - (t+step) + 2\*(xa10t[j-1]+step\*k1);**

**k3 = 0.5 - (t+step) + 2\*(xa10t[j-1]+step\*k2);**

**k4 = 0.5 - (t+h) + 2\*(xa10t[j-1]+h\*k3);**

**xa10t[j] = xa10t[j-1] + (h/6)\*(k1+2\*k2+2\*k3+k4);**

**//pragmatikes times**

**fa10t = exp(2\*t) + (t/2);**

**//ektypwsh timwn sto arxeio**

**fprintf(output10a, "%f\t%f\t%f\t%f\t%f\n",t, x\_euler\_a10[j], x\_opteuler\_a10[j], xa10t[j], fa10t);**

**//gia to posostiaio la8os**

**fprintf(output10alog, "%f\t%f\t%f\t%f\n",t, (100\*fabs(x\_euler\_a10[j]-fa10t))/fa10t, (100\*fabs(x\_opteuler\_a10[j]-fa10t))/fa10t, (100\*fabs(xa10t[j]-fa10t))/fa10t);**

**}**

**fclose(output10a);**

**fclose(output10alog);**

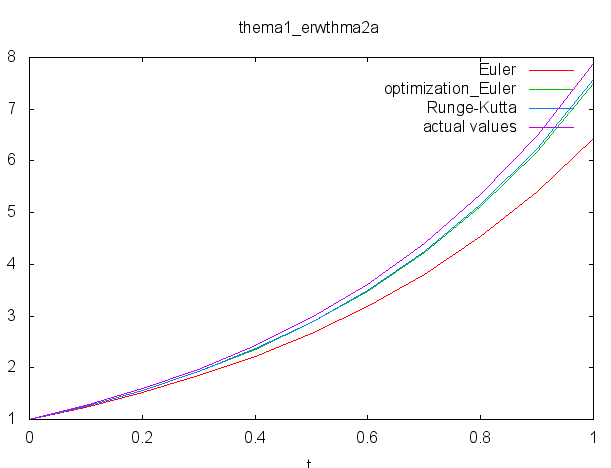
**system("PAUSE");**

**return 0;**

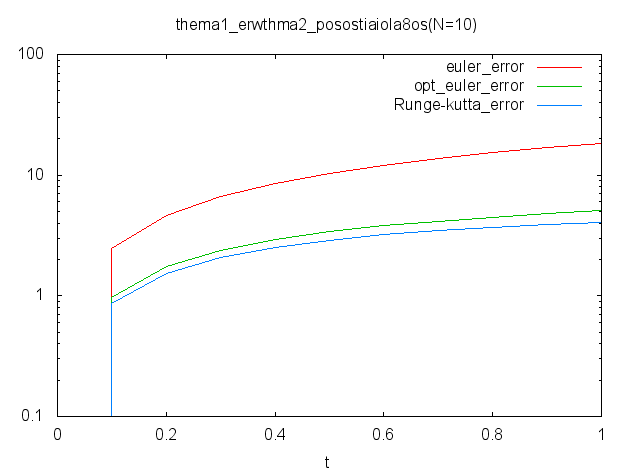
**}**

**Έτσι δημιουργούμε το αρχείο aN10 και το aN10log για το ποσοστιαίο λάθος.**

**Η γραφική παράσταση των στηλών 1-2, 1-3, 1-4, 1-5 είναι:**

****

**η γραφική παράσταση των ποσοστιαίων σχετικών λαθών για τους τύπους Euler, βελτιωmένου Euler, Runge-Kutta ως προς t είναι:**

****

**ΙV) Συγγραφή κώδικα υλοποίησης**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define MAX 1.0 /\* max for t \*/

#define MIN 0.0 /\* min for t \*/

FILE , \*output20a ;

FILE \*output20alog;

int main(int argc, char \*argv[])

{

//h idia diadikasia gia N=20

N=20;

double xa20t[N]];

double x\_euler\_a20[N];

double x\_opteuler\_a20[N];

h = (MAX-MIN)/N;

t = 0;

xa20t[0] = x\_euler\_a20[0] = x\_opteuler\_a20[0] = 1.0;

fa20t = exp(2\*t) + (t/2);

fb20t = - ((pow(t,2)+3)/(t-3));

output20a=fopen("aN20.dat", "w");

output20alog=fopen("aN20log.dat", "w");

fprintf(output20a, "t(i)\t\t Euler\t\t opt.Euler\t Runge\t\t f(i)\n");

fprintf(output20a, "%f\t%f\t%f\t%f\t%f\n", t, x\_euler\_a20[0], x\_opteuler\_a20[0], xa20t[0], fa20t);

fprintf(output20alog, "t(i)\t\t Euler\t\t opt.Euler\t Runge\n");

fprintf(output20alog, "%f\t0.000000\t0.000000\t0.000000\n", t);

for (j=1; j<=N; j++)

{ t = MIN + j\*h;

k1 = 0.5 - t + (2\*x\_euler\_a20[j-1]);

x\_euler\_a20[j] = x\_euler\_a20[j-1] + (h\*k1);

k1 = 0.5 - t + (2\*x\_opteuler\_a20[j-1]);

k2 = 0.5 - (t+h) + (2\*(x\_opteuler\_a20[j-1]+(h\*k1)));

x\_opteuler\_a20[j] = x\_opteuler\_a20[j-1] + (h/2)\*(k1+k2);

k1 = 0.5 - t + 2\*xa20t[j-1];

k2 = 0.5 - (t+step) + 2\*(xa20t[j-1]+step\*k1);

k3 = 0.5 - (t+step) + 2\*(xa20t[j-1]+step\*k2);

k4 = 0.5 - (t+h) + 2\*(xa20t[j-1]+h\*k3);

xa20t[j] = xa20t[j-1] + (h/6)\*(k1+2\*k2+2\*k3+k4);

fa20t = exp(2\*t) + (t/2);

fprintf(output20a, "%f\t%f\t%f\t%f\t%f\n",t, x\_euler\_a20[j], x\_opteuler\_a20[j], xa20t[j], fa20t);

//gia to posostiaio la8os

fprintf(output20alog, "%f\t%f\t%f\t%f\n",t, (100\*fabs(x\_euler\_a20[j]-fa20t))/fa20t, (100\*fabs(x\_opteuler\_a20[j]-fa20t))/fa20t, (100\*fabs(xa20t[j]-fa20t))/fa20t);

}

fclose(output20a);

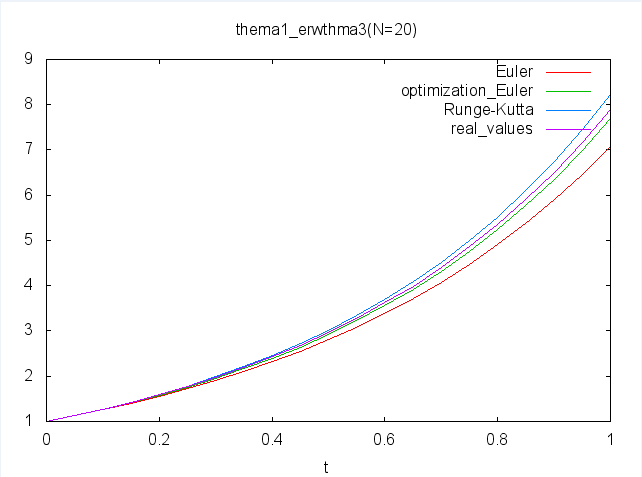
fclose(output20alog);

system("PAUSE");

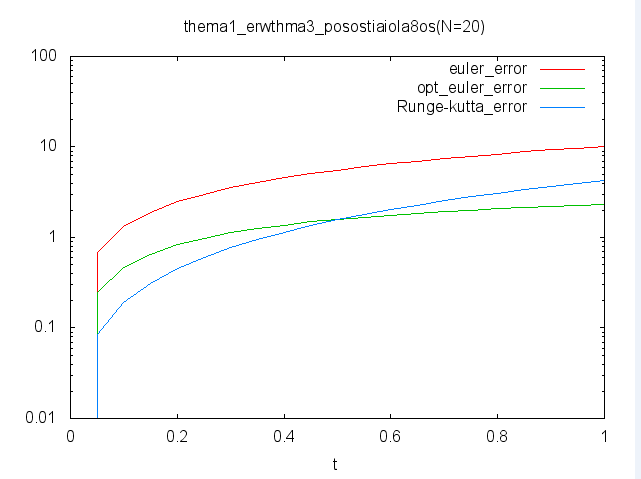
return 0;}

**Έτσι δημιουργούμε το αρχείο aN20 και το aN20log για το ποσοστιαίο λάθος.**

**Η γραφική παράσταση των στηλών 1-2, 1-3, 1-4, 1-5 είναι:**

****

**η γραφική παράσταση των ποσοστιαίων σχετικών λαθών για τους τύπους Euler, βελτιωmένου Euler, Runge-Kutta ως προς t είναι:**

****

**V) Συγγραφή κώδικα υλοποίησης**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define MAX 1.0 /\* max for t \*/

#define MIN 0.0 /\* min for t \*/

FILE \*output10b, \*output20b;

FILE \*output10blog, \*output20blog;

int main(int argc, char \*argv[])

{

int N=10; //to plh8os tou t

double t, xb10t[N], fb10[t];

double x\_euler\_b10[N];

double x\_opteuler\_b10[N];

double k1, k2, k3, k4;

int j;

double h, step;

h = (MAX-MIN)/N;

step = h/2;

//arxikes times

t = 0;

xb10t[0] = x\_euler\_b10[0] = x\_opteuler\_b10[0] = 1.0;

fa10t = exp(2\*t) + (t/2);

fb10t = - ((pow(t,2)+3)/(t-3));

output10b=fopen("bN10.dat", "w");

output10blog=fopen("bN10log.dat", "w");

fprintf(output10b, "t(i)\t\t Euler\t\t opt.Euler\t Runge\t\t f(i)\n");

fprintf(output10b, "%f\t%f\t%f\t%f\t%f\n", t, x\_euler\_b10[0], x\_opteuler\_b10[0], xb10t[0], fb10t);

fprintf(output10blog, "t(i)\t\t Euler\t\t opt.Euler\t Runge\n");

fprintf(output10blog, "%f\t0.000000\t0.000000\t0.000000\n", t);

//to loop ypologismou

for (j=1; j<=N; j++)

{

t = MIN + j\*h;

//omoia gia thn 2h synarthsh (erwthma V)

k1 = (pow(x\_euler\_b10[j-1],2)+(2\*t\*x\_euler\_b10[j-1]))/(3 + pow(t,2));

x\_euler\_b10[j] = x\_euler\_b10[j-1] + h\*k1;

k1 = (pow(x\_opteuler\_b10[j-1],2)+(2\*t\*x\_opteuler\_b10[j-1]))/(3 + pow(t,2));

k2 = (pow(x\_opteuler\_b10[j-1]+(h\*k1),2)+(2\*(t+h)\*(x\_opteuler\_b10[j-1]+(h\*k1))))/(3 + pow(t+h,2));

x\_opteuler\_b10[j] = x\_opteuler\_b10[j-1] + (h/2)\*(k1+k2);

k1 = (pow(xb10t[j-1],2)+(2\*t\*xb10t[j-1]))/(3+pow(t,2));

k2 = (pow(xb10t[j-1]+(step\*k1),2)+(2\*(t+step)\*(xb10t[j-1]+(step\*k2))))/(3+pow(t+step,2));

k3 = (pow(xb10t[j-1]+(step\*k2),2)+(2\*(t+step)\*(xb10t[j-1]+(step\*k2))))/(3+pow(t+step,2));

k4 = (pow(xb10t[j-1]+(h\*k3),2)+(2\*(t+h)\*(xb10t[j-1]+(h\*k3))))/(3+pow(t+h,2));

xb10t[j] = xb10t[j-1] + (h/6)\*(k1+2\*k2+2\*k3+k4);

fb10t = - ((pow(t,2)+3)/(t-3));

fprintf(output10b, "%f\t%f\t%f\t%f\t%f\n",t, x\_euler\_b10[j], x\_opteuler\_b10[j], xb10t[j], fb10t);

//gia to posostiaio la8os

fprintf(output10blog, "%f\t%f\t%f\t%f\n",t, (100\*fabs(x\_euler\_b10[j]-fb10t))/fb10t, (100\*fabs(x\_opteuler\_b10[j]-fb10t))/fb10t, (100\*fabs(xb10t[j]-fb10t))/fb10t); }

fclose(output10b);

fclose(output10blog);

//h idia diadikasia gia N=20

N=20;

double xb20t[N], fb20t;

double x\_euler\_b20[N];

double x\_opteuler\_b20[N];

h = (MAX-MIN)/N;

t = 0;

xb20t[0] = x\_euler\_b20[0] = x\_opteuler\_b20[0] = 1.0;

fa20t = exp(2\*t) + (t/2);

fb20t = - ((pow(t,2)+3)/(t-3));

output20b=fopen("bN20.dat", "w");

output20blog=fopen("bN20log.dat", "w");

fprintf(output20b, "t(i)\t\t Euler\t\t opt.Euler\t Runge\t\t f(i)\n");

fprintf(output20b, "%f\t%f\t%f\t%f\t%f\n", t, x\_euler\_b20[0], x\_opteuler\_b20[0], xb20t[0], fb20t);

fprintf(output20blog, "t(i)\t\t Euler\t\t opt.Euler\t Runge\n");

fprintf(output20blog, "%f\t0.000000\t0.000000\t0.000000\n", t);

for (j=1; j<=N; j++)

{

t = MIN + j\*h;

k1 = (pow(x\_euler\_b20[j-1],2)+(2\*t\*x\_euler\_b20[j-1]))/(3 + pow(t,2));

x\_euler\_b20[j] = x\_euler\_b20[j-1] + h\*k1;

k1 = (pow(x\_opteuler\_b20[j-1],2)+(2\*t\*x\_opteuler\_b20[j-1]))/(3 + pow(t,2));

k2 = (pow(x\_opteuler\_b20[j-1]+(h\*k1),2)+(2\*(t+h)\*(x\_opteuler\_b20[j-1]+(h\*k1))))/(3 + pow(t+h,2));

x\_opteuler\_b20[j] = x\_opteuler\_b20[j-1] + (h/2)\*(k1+k2);

k1 = (pow(xb20t[j-1],2)+(2\*t\*xb20t[j-1]))/(3+pow(t,2));

k2 = (pow(xb20t[j-1]+(step\*k1),2)+(2\*(t+step)\*(xb20t[j-1]+(step\*k2))))/(3+pow(t+step,2));

k3 = (pow(xb20t[j-1]+(step\*k2),2)+(2\*(t+step)\*(xb20t[j-1]+(step\*k2))))/(3+pow(t+step,2));

k4 = (pow(xb20t[j-1]+(h\*k3),2)+(2\*(t+h)\*(xb20t[j-1]+(h\*k3))))/(3+pow(t+h,2));

xb20t[j] = xb20t[j-1] + (h/6)\*(k1+2\*k2+2\*k3+k4);

fb20t = - ((pow(t,2)+3)/(t-3));

fprintf(output20b, "%f\t%f\t%f\t%f\t%f\n",t, x\_euler\_b20[j], x\_opteuler\_b20[j], xb20t[j], fb20t);

//gia to posostiaio la8os

fprintf(output20blog, "%f\t%f\t%f\t%f\n",t, (100\*fabs(x\_euler\_b20[j]-fb20t))/fb20t, (100\*fabs(x\_opteuler\_b20[j]-fa20t))/fb20t, (100\*fabs(xb20t[j]-fb20t))/fb20t); }

fclose(output20a);

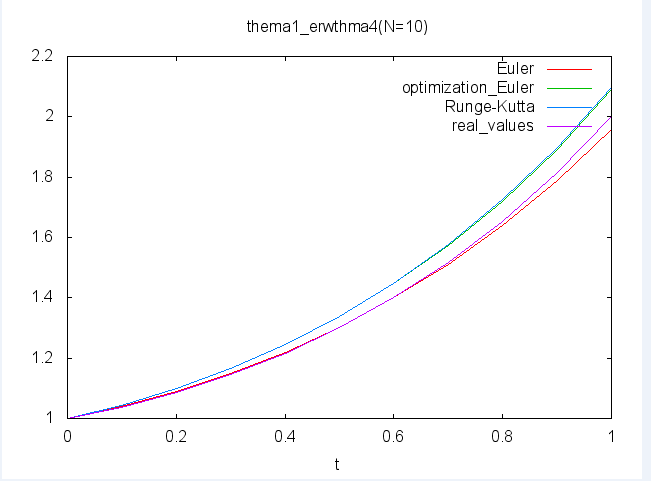
fclose(output20blog);

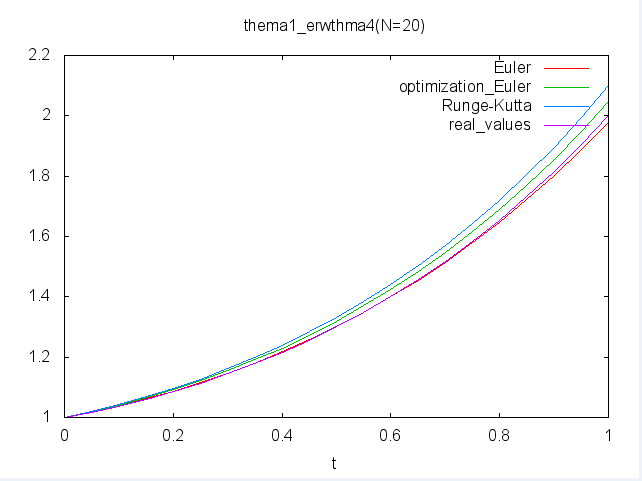
system("PAUSE");

return 0;}

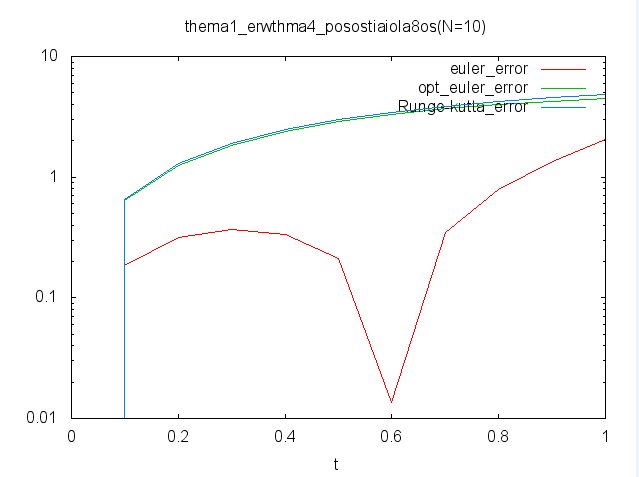
**Έτσι δημιουργούμε το αρχείο bN10- bN20 και το bN10log- bN20log για το ποσοστιαίο λάθος.**

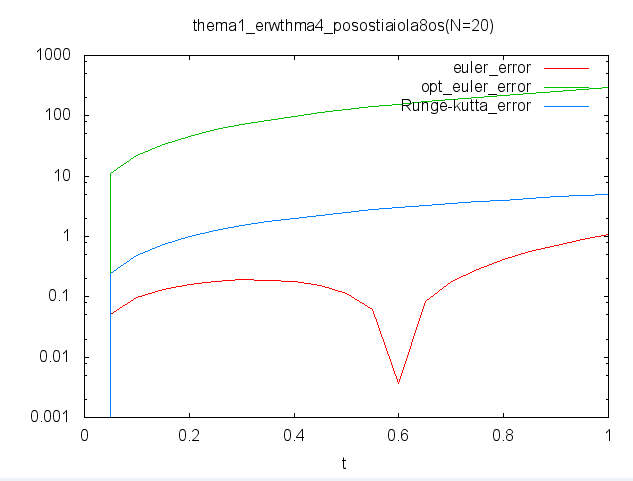
**Η γραφική παράσταση των στηλών 1-2, 1-3, 1-4, 1-5 είναι:**

****

****

**η γραφική παράσταση των ποσοστιαίων σχετικών λαθών για τους τύπους Euler, βελτιωmένου Euler, Runge-Kutta ως προς t είναι:**

****

****

**VI) Συγγραφή κώδικα υλοποίησης**

**//omoia gia thn 3h synarthsh (erwthma VI)**

k1 = pow(t,2) + pow(x\_euler\_c10[j-1],2);

x\_euler\_c10[j] = x\_euler\_c10[j-1] + h\*k1;

k1 = pow(t,2) + pow(x\_opteuler\_c10[j-1],2);

k2 = pow(t+h,2) + pow(x\_opteuler\_c10[j-1]+(h\*k1),2);

x\_opteuler\_c10[j] = x\_opteuler\_c10[j-1] + (h/2)\*(k1+k2);

k1 = pow(t,2) + pow(xc10t[j-1],2);

k2 = pow(t+step,2) + pow(xc10t[j-1]+(step\*k1),2);

k3 = pow(t+step,2) + pow(xc10t[j-1]+(step\*k2),2);

k4 = pow(t+h,2) + pow(xc10t[j-1]+(h\*k3),2);

xc10t[j] = xc10t[j-1] + (h/6)\*(k1+2\*k2+2\*k3+k4);

fprintf(output10c, "%f\t%f\t%f\t%f\n",t, x\_euler\_c10[j], x\_opteuler\_c10[j], xc10t[j]);

**//omoia gia thn 4h synarthsh (erwthma VI)**

k1 = (pow(t,2) - pow(x\_euler\_d10[j-1],2))\*sin(x\_euler\_d10[j-1]);

x\_euler\_d10[j] = x\_euler\_d10[j-1] + h\*k1;

k1 = (pow(t,2) - pow(x\_opteuler\_d10[j-1],2))\*sin(x\_opteuler\_d10[j-1]);

k2 = (pow(t+h,2) - pow(x\_opteuler\_d10[j-1]+(h\*k1),2))\*sin(x\_opteuler\_d10[j-1]+(h\*k1));

x\_opteuler\_d10[j] = x\_opteuler\_d10[j-1] + (h/2)\*(k1+k2);

k1 = (pow(t,2) - pow(xd10t[j-1],2))\*sin(xd10t[j-1]);

k2 = (pow(t+step,2) - pow(xd10t[j-1]+(step\*k1),2))\*sin(xd10t[j-1]);

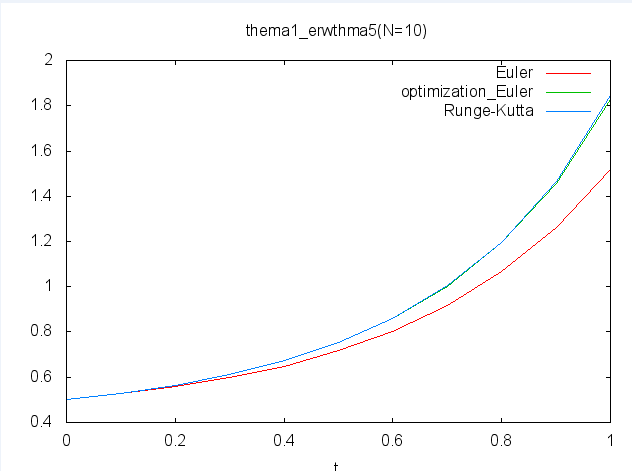
k3 = (pow(t+step,2) - pow(xd10t[j-1]+(step\*k2),2))\*sin(xd10t[j-1]);

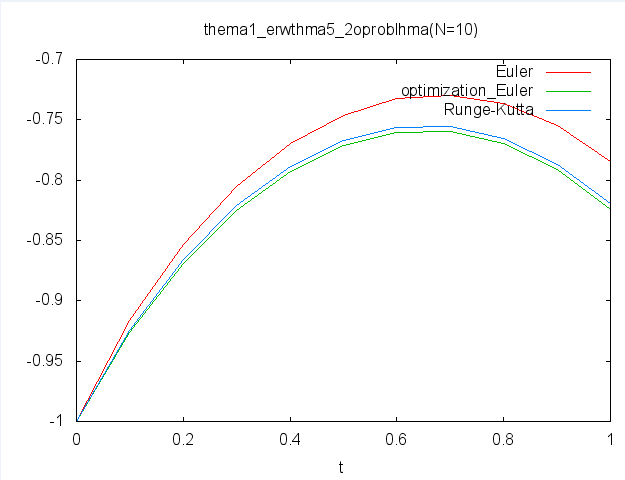
k4 = (pow(t+h,2) - pow(xd10t[j-1]+(h\*k3),2))\*sin(xd10t[j-1]);

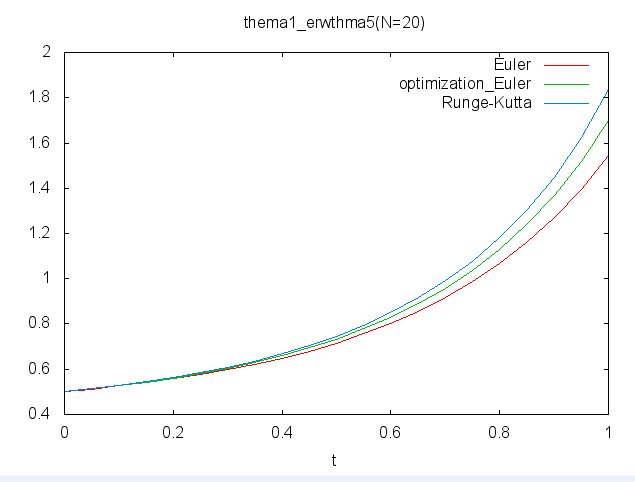
xd10t[j] = xd10t[j-1] + (h/6)\*(k1+2\*k2+2\*k3+k4);

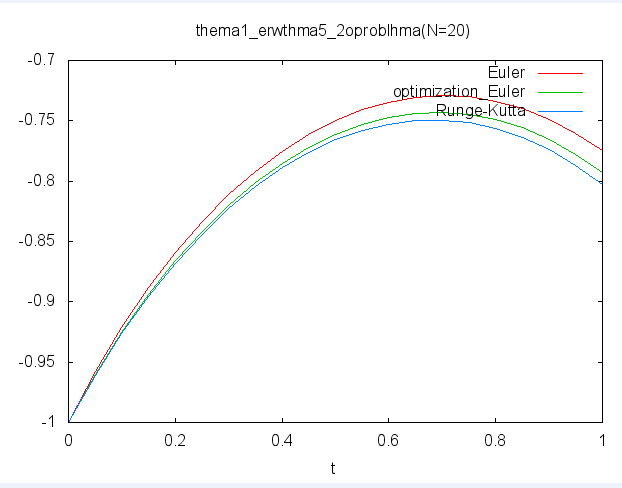
fprintf(output10d, "%f\t%f\t%f\t%f\n",t, x\_euler\_d10[j], x\_opteuler\_d10[j], xd10t[j]);

**οι γραφικές παραστάσεις:**

****

****

****

****

**Θέμα 2: Μέθοδος Runge-Kutta-Fehlberg**

**I) Συγγραφή κώδικα υλοποίησης**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define MAX 2.56 /\* max for t \*/

#define MIN 1.0 /\* min for t \*/

int main(int argc, char \*argv[])

{

int N=10;

double t, h, xRKF10t[N], x\_RKF\_10t[N], f10t, xRK10t[N];

double x\_euler10[N], x\_opteuler10[N];

double k1, k2, k3, k4, k5, k6;

int j;

h = (MAX-MIN)/N;

//arxikes times

t = MIN;

xRKF10t[0] = x\_RKF\_10t[0] = xRK10t[0] = x\_euler10[0] = x\_opteuler10[0] = 2.0;

f10t = 1 + t + tan(t-1);

for (j=1; j<=N; j++)

{ t = MIN + j\*h;

//oi times gia Euler

k1 = 2 + pow(x\_euler10[j-1]-t-1,2);

x\_euler10[j] = x\_euler10[j-1] + h\*k1;

//oi times gia veltistopoihmeno Euler

k1 = 2 + pow(x\_opteuler10[j-1]-t-1,2);

k2 = 2 + pow((x\_opteuler10[j-1]+(h\*k1))-(t+h)-1,2);

x\_opteuler10[j] = x\_opteuler10[j-1] + (h/2)\*(k1+k2);

//oi times gia runge-kutta-fehleberg

k1 = 2 + pow(xRKF10t[j-1]-t-1,2);

k2 = 2 + pow((xRKF10t[j-1]+((h/4.0)\*k1))-(t+(h/4.0))-1.0,2);

k3 = 2 + pow((xRKF10t[j-1]+((3.0/32.0)\*h\*k1)+((9.0/32.0)\*h\*k2))-(t+(3.0/8.0)\*h)-1.0,2);

k4 = 2 + pow((xRKF10t[j-1]+((1932.0/2197.0)\*h\*k1)-((7200.0/2197.0)\*h\*k2)+((7296.0/2197.0)\*h\*k3))-(t+(12.0/13.0)\*h)-1.0,2);

k5 = 2 + pow((xRKF10t[j-1]+((439.0/216.0)\*h\*k1)-(8\*h\*k2)+((3680.0/513.0)\*h\*k3)-((845.0/4104.0)\*h\*k4))-(t+h)-1.0,2);

k6 = 2 + pow((xRKF10t[j-1]-((8.0/27.0)\*h\*k1)+(2\*h\*k2)-((3544.0/2565.0)\*h\*k3)-((1859.0/4104.0)\*h\*k4)-((11.0/40.0)\*h\*k5))-(t+(h/2))-1.0,2);

//o typos 2

x\_RKF\_10t[j] = xRKF10t[j-1] +((25.0/216.0)\*h\*k1)+((1408.0/2565.0)\*h\*k3)+((2197.0/4104.0)\*h\*k4)-((h/5)\*k5);

//o typos 3

xRKF10t[j] = xRKF10t[j-1] +((16.0/135.0)\*h\*k1)+((6656.0/12825.0)\*h\*k3)+((28561.0/56430.0)\*h\*k4)-((9.0/50.0)\*h\*k5)+((2.0/55.0)\*h\*k6);

//oi pragmatikes times

f10t = 1 + t + tan(t-1);

//oi rimes gia runge kutta

k1 = 2 + pow(xRK10t[j-1]-t-1,2);

k2 = 2 + pow((xRK10t[j-1]+((h/2.0)\*k1))-(t+(h/2.0))-1,2);

k3 = 2 + pow((xRK10t[j-1]+((h/2.0)\*k2))-(t+(h/2.0))-1,2);

k4 = 2 + pow((xRK10t[j-1]+(h\*k3))-(t+h)-1,2);

xRK10t[j] = xRK10t[j-1] + (h/6.0)\*(k1 + (2\*k2) + (2\*k3) + k4);

printf( "%f\t%f\t%f\t%f\t%f\t%f\n", t, xRK10t[j], xRKF10t[j], x\_RKF\_10t[j], f10t);

} system("PAUSE"); return 0;}

**II) Συγγραφή κώδικα υλοποίησης**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define MAX 2.56 /\* max for t \*/

#define MIN 1.0 /\* min for t \*/

FILE \*output10a;

int main(int argc, char \*argv[])

{

int N=10;

double t, h, xRKF10t[N], x\_RKF\_10t[N], f10t, xRK10t[N], error;

double x\_euler10[N], x\_opteuler10[N];

double k1, k2, k3, k4, k5, k6;

int j;

h = (MAX-MIN)/N;

//arxikes times

t = MIN;

xRKF10t[0] = x\_RKF\_10t[0] = xRK10t[0] = x\_euler10[0] = x\_opteuler10[0] = 2.0;

f10t = 1 + t + tan(t-1);

output10a=fopen("aN10.dat", "w");

fprintf(output10a, "t(i)\t\t RK\t\t RKF \t\t RKF\_\t\t error\t\t f(i)\n");

fprintf(output10a, "%f\t%f\t%f\t%f\t%f\t%f\n", t, xRK10t[0], xRKF10t[0], x\_RKF\_10t[0], xRKF10t[0]-x\_RKF\_10t[0], f10t);

for (j=1; j<=N; j++)

{

t = MIN + j\*h;

//oi times gia Euler

k1 = 2 + pow(x\_euler10[j-1]-t-1,2);

x\_euler10[j] = x\_euler10[j-1] + h\*k1;

//oi times gia veltistopoihmeno Euler

k1 = 2 + pow(x\_opteuler10[j-1]-t-1,2);

k2 = 2 + pow((x\_opteuler10[j-1]+(h\*k1))-(t+h)-1,2);

x\_opteuler10[j] = x\_opteuler10[j-1] + (h/2)\*(k1+k2);

//oi times gia runge-kutta-fehleberg

k1 = 2 + pow(xRKF10t[j-1]-t-1,2);

k2 = 2 + pow((xRKF10t[j-1]+((h/4.0)\*k1))-(t+(h/4.0))-1.0,2);

k3 = 2 + pow((xRKF10t[j-1]+((3.0/32.0)\*h\*k1)+((9.0/32.0)\*h\*k2))-(t+(3.0/8.0)\*h)-1.0,2);

k4 = 2 + pow((xRKF10t[j-1]+((1932.0/2197.0)\*h\*k1)-((7200.0/2197.0)\*h\*k2)+((7296.0/2197.0)\*h\*k3))-(t+(12.0/13.0)\*h)-1.0,2);

k5 = 2 + pow((xRKF10t[j-1]+((439.0/216.0)\*h\*k1)-(8\*h\*k2)+((3680.0/513.0)\*h\*k3)-((845.0/4104.0)\*h\*k4))-(t+h)-1.0,2);

k6 = 2 + pow((xRKF10t[j-1]-((8.0/27.0)\*h\*k1)+(2\*h\*k2)-((3544.0/2565.0)\*h\*k3)-((1859.0/4104.0)\*h\*k4)-((11.0/40.0)\*h\*k5))-(t+(h/2))-1.0,2);

//o typos 2

x\_RKF\_10t[j] = xRKF10t[j-1] +((25.0/216.0)\*h\*k1)+((1408.0/2565.0)\*h\*k3)+((2197.0/4104.0)\*h\*k4)-((h/5)\*k5);

//o typos 3

xRKF10t[j] = xRKF10t[j-1] +((16.0/135.0)\*h\*k1)+((6656.0/12825.0)\*h\*k3)+((28561.0/56430.0)\*h\*k4)-((9.0/50.0)\*h\*k5)+((2.0/55.0)\*h\*k6);

//o typos 4

error = fabs(x\_RKF\_10t[j]-xRKF10t[j]);

//oi pragmatikes times

f10t = 1 + t + tan(t-1);

//oi rimes gia runge kutta

k1 = 2 + pow(xRK10t[j-1]-t-1,2);

k2 = 2 + pow((xRK10t[j-1]+((h/2.0)\*k1))-(t+(h/2.0))-1,2);

k3 = 2 + pow((xRK10t[j-1]+((h/2.0)\*k2))-(t+(h/2.0))-1,2);

k4 = 2 + pow((xRK10t[j-1]+(h\*k3))-(t+h)-1,2);

xRK10t[j] = xRK10t[j-1] + (h/6.0)\*(k1 + (2\*k2) + (2\*k3) + k4);

fprintf(output10a, "%f\t%f\t%f\t%f\t%f\t%f\n", t, xRK10t[j], xRKF10t[j], x\_RKF\_10t[j], error, f10t);

}

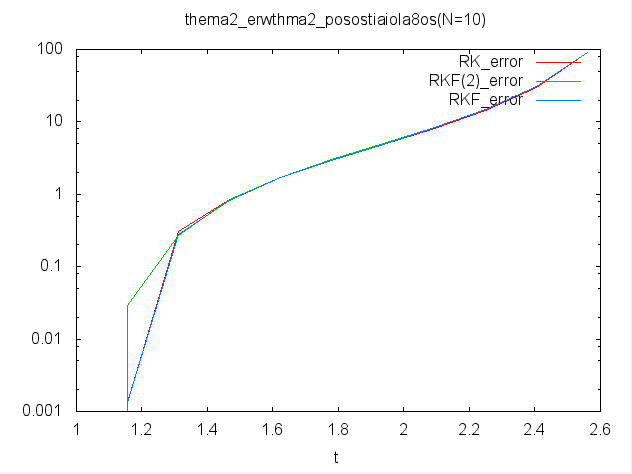
fclose(output10a);

system("PAUSE");

return 0;}

**Έτσι δημιουργούμε το αρχείο aN10.**

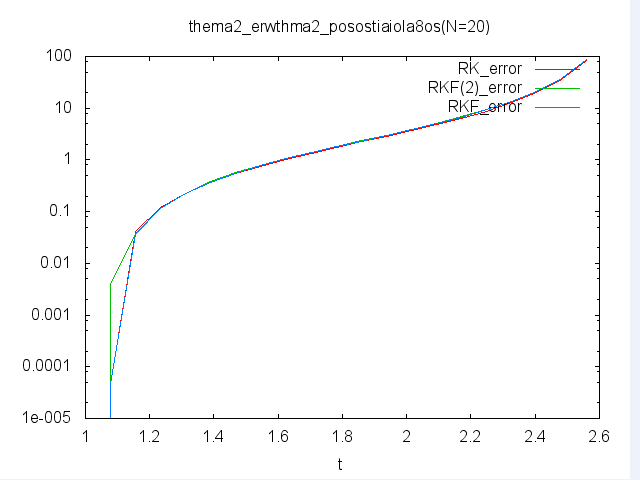
**η γραφική παράσταση των ποσοστιαίων σχετικών λαθών για τους τύπους Runge-Kutta, Runge-Kutta-Fehlberg τύπος (2), Runge-Kutta-Fehlberg τύπος (3), ως προς t σε ένα σχήμα χρησιmοποιώντας λογαριθμική κλίμακα στον άξονα-y που παριστάνει το ποσοστιαίο λάθος.**

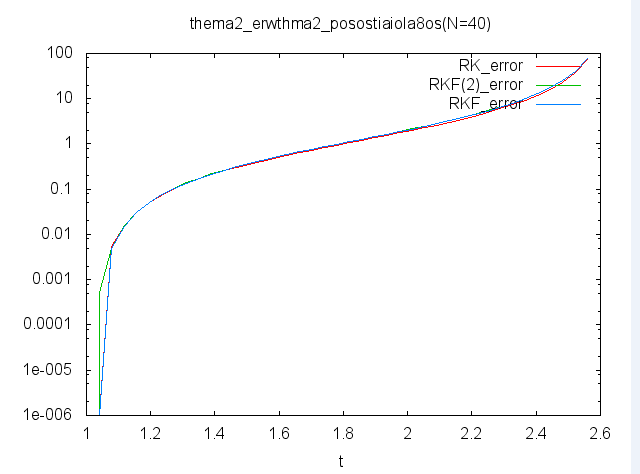


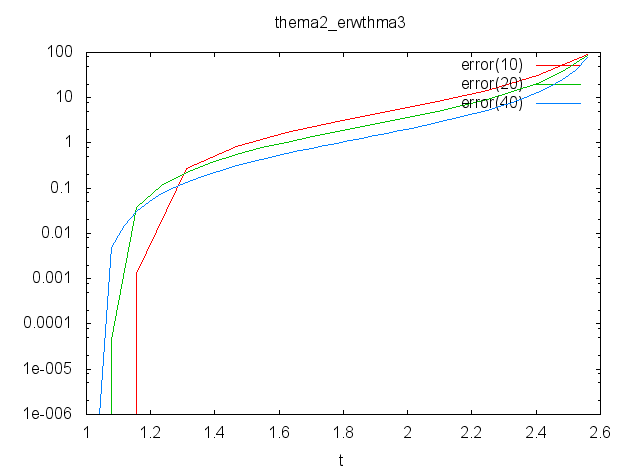
**III)**

**Τα αποτελέσματα είναι μέσα στο φάκελο καθώς και ο κώδικας**

**Παρακάτω φαίνονται οι γραφικές:**

****





**Θέμα 3: Προσαρμοστική μέθοδος Runge-Kutta-Fehlberg**

**I) Συγγραφή κώδικα υλοποίησης**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

FILE \*outputa;

FILE \*outputb;

FILE \*outputaerror, \*outputberror;

typedef struct

{ double x;

double x\_;

double error;

}values;

values rkf45(double h, double t);

int main(int argc, char \*argv[])

{

double a,b;

double emin,emax;

double h,t, f;

int N,j;

values value\_final;

a = 1.0;

b = 2.56;

N = 10;

h = (b-a)/N;

emax = pow(10,-6);

emin = pow(10,-8);

t = a;

value\_final.x = 2.0;

value\_final.error = 0.0;

f = 1 + t + tan(t-1);

outputa=fopen("aN.dat", "w");

fprintf(outputa, "t(i)\t\t x(i)\t\t e(i) \t\t f(i)\n");

fprintf(outputa, "%f\t%f\t%f\t%f\n", t, value\_final.x, value\_final.error, f);

outputaerror=fopen("aNerror.dat", "w");

fprintf(outputaerror, "t(i)\t\terror\n");

fprintf(outputaerror, "%f\t0.000000\n", t);

for(j=0; j<N; j++)

{

if ((emin<=value\_final.error) && (value\_final.error<=emax))

{

t = t + h;

value\_final = rkf45(h,t);

f = 1 + t + tan(t-1);

}

if(value\_final.error<emin) {

h = 2\*h;

t = t +h;

value\_final = rkf45(h,t);

f = 1 + t + tan(t-1);

}

if(value\_final.error>emax)

{

h = h/2;

t = t + h;

value\_final = rkf45(h/2,t);

f = 1 + t + tan(t-1);

}

fprintf(outputa, "%f\t%f\t%f\t%f\n", t, value\_final.x, value\_final.error, f);

fprintf(outputaerror, "%f\t%f\n", t, (100\*fabs(value\_final.x-f))/f);

}

fclose(outputa);

fclose(outputaerror);

system("PAUSE");

return 0;

}

values rkf45(double h, double t)

{

values value;

double k1,k2,k3,k4,k5,k6;

value.x = 2.0;

value.error = 0.0;

k1 = 2 + pow(value.x-t-1,2);

k2 = 2 + pow((value.x+((h/4.0)\*k1))-(t+(h/4.0))-1.0,2);

k3 = 2 + pow((value.x+((3.0/32.0)\*h\*k1)+((9.0/32.0)\*h\*k2))-(t+(3.0/8.0)\*h)-1.0,2);

k4 = 2 + pow((value.x+((1932.0/2197.0)\*h\*k1)-((7200.0/2197.0)\*h\*k2)+((7296.0/2197.0)\*h\*k3))-(t+(12.0/13.0)\*h)-1.0,2);

k5 = 2 + pow((value.x+((439.0/216.0)\*h\*k1)-(8\*h\*k2)+((3680.0/513.0)\*h\*k3)-((845.0/4104.0)\*h\*k4))-(t+h)-1.0,2);

k6 = 2 + pow((value.x-((8.0/27.0)\*h\*k1)+(2\*h\*k2)-((3544.0/2565.0)\*h\*k3)-((1859.0/4104.0)\*h\*k4)-((11.0/40.0)\*h\*k5))-(t+(h/2))-1.0,2);

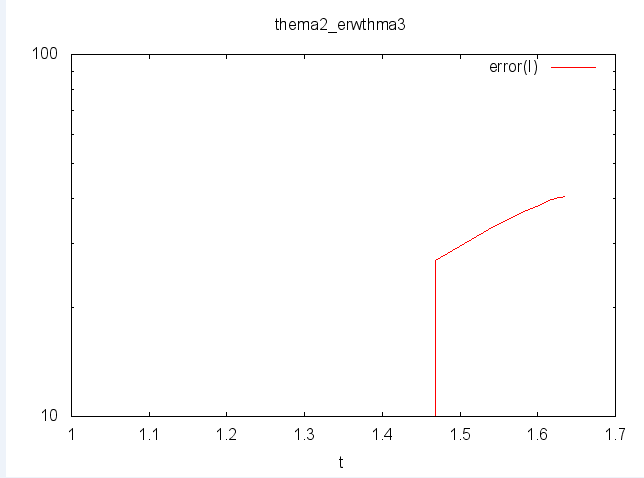
value.x\_ = value.x +((25.0/216.0)\*h\*k1)+((1408.0/2565.0)\*h\*k3)+((2197.0/4104.0)\*h\*k4)-((h/5)\*k5);

value.x = value.x +((16.0/135.0)\*h\*k1)+((6656.0/12825.0)\*h\*k3)+((28561.0/56430.0)\*h\*k4)-((9.0/50.0)\*h\*k5)+((2.0/55.0)\*h\*k6);

value.error = fabs(value.x\_-value.x);

return value;}

**Παρακάτω φαίνονται οι γραφικές:**



**II) Συγγραφή κώδικα υλοποίησης**

