Задание №5. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ КОШИ ДЛЯ СОДУ

Вариант 8

Цель задания: практическое освоение методов управления численным процессом с целью обеспечения заданной точности решения задачи Коши для СОДУ за приемлемое время, используя явные методы Рунге-Кутты (ЯМРК) с методами оценки полной и локальной погрешности.

Рассмотрим задачу Коши (1):

**Часть №1. Расчетные схемы метода Рунге-Кутты с постоянным шагом**

1.1. Используя условия порядка для 2-х-этапного ЯМРК 2-го порядка постройте расчетную схему второго порядка при значении параметра (параметр указан в Вашем варианте задания).

1.2. Постройте и программно реализуйте алгоритм решения задачи Коши (1) с заданной точностью () с постоянным шагом интегрирования и оценкой полной погрешности по методу Рунге. Параметры A, В указаны в Вашем варианте. Начальный шаг выбирать согласно алгоритму выбора начального шага (стр. 15 методического пособия).

Реализуем в Matlab.

Пользовательская функция (полный текст программы в Приложении):

% определяем значение правых частей системы ДУ

function F = f(t, X)

A = 1/12; B = 1/20;

k=2; %порядок системы

F = zeros(2,1); %вектор правых частей

F(1)=A\*X(2);

F(2)=-B\*X(1);

end

%Метод второго порядка

function [tn,YY] =RK2(A,B,h,y,c2)

b2 = 1/(2\*c2);

b1 = 1-b2;a21=c2;

t = A;

n=ceil((B-A)/h);

tn=A:h:B;%генерируем n+1 моментов времени

k=length(y);

yn = zeros(k,1);

yn(:,1) = y(:);

for i = 1:n

k1 = h\*f(t,y);

k2 = h\*f(t+c2\*h,y+a21\*k1);

y = y+b1\*k1+b2\*k2;

t = t + h;

yn(:,i+1) = y(:);

end

YY=yn;

end

(решение YY на отрезке [A,B] на сетке tn с шагом h и начальным условием y)

Часть программы определения начального шага:

clear all

close all

clc

c2=1/12;%параметр расчётной схемы

x0=0; xk=pi;

%%Часть №1. Расчетные схемы метода Рунге-Кутты с постоянным шагом

%выбор начального шага

s=2;%порядок точности метода

eps = 1e-4;

y0 = [pi/20 pi/12]';%начальные условия

delta=(1/max(abs([x0; xk])))^(s+1)+norm(f(x0,y0),"inf")^(s+1);

h=(eps/delta)^(1/(s+1));

n0G=ceil((xk-x0)/h);

h0G=(xk-x0)/n0G;%полученный начальный шаг

h0 = h0G;n0=n0G;

%сравниваем на 2-х разных размерах шага

[x1,y1]=RK2(x0,xk,h0/2,y0,c2);

[x2,y2]=RK2(x0,xk,h0,y0,c2);

k=length(y2);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

%оценка полной погрешности

E0=norm(ee/(2^s-1),"inf");

%если шаг слишком маленький

while E0<eps/(2^(s+1))

n0 = ceil(n0/2);

h0 = (xk-x0)/n0;

[x1,y1]=RK2(x0,xk,h0/2,y0,c2);

[x2,y2]=RK2(x0,xk,h0,y0,c2);

k=length(y2);

ee=zeros(1,k);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

E0=norm(ee/(2^s-1),"inf");

end

%если шаг слишком большой

while E0>eps

n0 = 2\*n0;

h0 = (xk-x0)/n0;

[x1,y1]=RK2(x0,xk,h0/2,y0,c2);

[x2,y2]=RK2(x0,xk,h0,y0,c2);

k=length(y2);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

E0=norm(ee/(2^s-1),"inf");

end

%строим графики

plot(x1,y1(1,:),'-\*',x1,y1(2,:),'-o')

grid on

legend('y1(x)','y2(x)','Location','best')

xlabel('x')

title('Решение системы с постоянным шагом (eps = 1e-4), RK2')

Полученный график решения с постоянным шагом :



**Часть №2. Расчетные схемы метода Рунге-Кутты с автоматическим выбором шага**

2.1. Реализуйте алгоритм решения задачи Коши (1) на базе построенной Вами (см. п. 1.1.) схемы 2-х-этапного ЯМРК 2-го порядка с автоматическим выбором шага с заданной максимально допустимой локальной погрешностью () и оценкой локальной погрешности по методу Рунге. Начальный шаг выбирать так же, как и в п. 1.2.

Пользовательская функция для автоматического выбора шага:

%Метод второго поряка с автоматическим выбором шага

%hh - i-ый шаг

%nf-число вычислений правой части

function [tn,YY,r,hh,nf] =RK2auto(A,B,h,y,ro,c2)

k = 1;%счётчик числа узлов

t = A; tn(1)=A; YY(:,1) = y;

s=2;

%(считаем что вся система - это один раз подсчёт правой части)

nf = 0;

while (t<=B)

[t1,y1] = RK2(t,t+h,h,y,c2);%считаем на шаге h

[t2,y2] = RK2(t,t+h,h/2,y,c2);%считаем на шаге h/2

nf = nf + 6;%всего за один вызов посчитали 2\*число точек х

%оценка локальной погрешности

roi = max(abs(y2(:,3)-(y1(:,2))))/(1-2^(-s));%на шаге h

if roi <= ro %данный шаг нас устраивает

hh(k)=h;t=t+h;tn(k+1)=t;

YY(:,k+1)=y1(:,2);y=y1(:,2);

if roi< ro/(2^(s+1))

h=2\*h; %можно увеличить в 2 раза после

end

r(k)= roi;

else

r(k)= roi/(2^s);

if roi > ro\*(2^s)%точность неподходящая

%шаг уменьшаем вдвое

h=h/2;hh(k)=h;

t=t+h;tn(k+1)=t;

YY(:,k+1)=y2(:,2);y=y2(:,2);

else %точность неподходящая, но можно использовать y2

hh(k)=h;t=t+h;tn(k+1)=t;

YY(:,k+1)=y2(:,3);y=y2(:,3);

h=h/2;

end

end

k=k+1;

end

end

Расчёт по схеме с автоматическим выбором шага:

%%

%%Часть №2. Расчетные схемы метода Рунге-Кутты с автоматическим выбором шага

ro=1e-5;%локальная погрешность

%выбор начального шага

eps = ro;

delta=(1/max(abs([x0; xk])))^(s+1)+norm(f(x0,y0),"inf")^(s+1);

h=(eps/delta)^(1/(s+1));

n0G=ceil((xk-x0)/h);

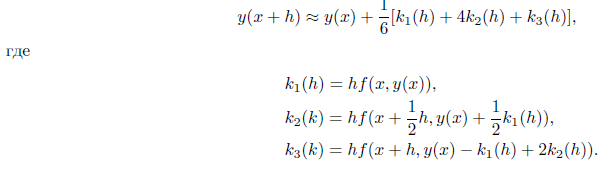
h0G=(xk-x0)/n0G;%полученный начальный шаг

[x3,y3,r,hh,nf]=RK2auto(x0,xk,h0G,y0,ro,c2);

**Часть №3. Анализ эффективности методов Рунге-Кутты**

3.1. Реализуйте алгоритмы решения задачи Коши с постоянным и автоматическим выбором шага на базе одной из классических расчетных схем интегрирования 3 или 4 порядка точности: формулы (36), (37), (39), (40).

Будем использовать схему 3-го порядка:



Скрипт в MATLAB:

%Метод третьего поряка

function [tn,YY] =RK3(A,B,h,y)

t = A;

n=ceil((B-A)/h);

tn=A:h:B;%генерруем n+1 моментов времени

k=length(y);

yn = zeros(k,1);

yn(:,1) = y(:);

for i = 1:n

k1 = h\*f(t,y);

k2 = h\*f(t+h/2,y+k1/2);

k3 = h\*f(t+h,y-k1+2\*k2);

y = y+(k1+4\*k2+k3)/6;

t = t + h;

yn(:,i+1) = y(:);

end

YY=yn;

end

%Метод третьего порядка с автоматическим выбором шага

%hh - i-ый шаг

%nf-число вычислений правой части

function [tn,YY,r,hh,nf] =RK3auto(A,B,h,y,ro)

k = 1;%счётчик числа узлов

t = A; tn(1)=A; YY(:,1) = y;

s=3;%порядок точности

%(считаем что вся система - это один раз подсчёт правой части)

nf = 0;

while (t<=B)

[t1,y1] = RK3(t,t+h,h,y);%считаем на шаге h

[t2,y2] = RK3(t,t+h,h/2,y);%считаем на шаге h/2

nf = nf + 9;%всего за один вызов посчитали 3\*число точек х

%оценка локальной погрешности

roi = max(abs(y2(:,3)-(y1(:,2))))/(1-2^(-s));%на шаге h

if roi <= ro %данный шаг нас устраивает

hh(k)=h;t=t+h;tn(k+1)=t;

YY(:,k+1)=y1(:,2);y=y1(:,2);

if roi< ro/(2^(s+1))

h=2\*h; %можно увеличить в 2 раза после

end

r(k)= roi;

else

r(k)= roi/(2^s);

if roi > ro\*(2^s)%точность неподходящая

%шаг уменьшаем вдвое

h=h/2;hh(k)=h;

t=t+h;tn(k+1)=t;

YY(:,k+1)=y2(:,2);y=y2(:,2);

else %точность неподходящая, но можно использовать y2

hh(k)=h;t=t+h;tn(k+1)=t;

YY(:,k+1)=y2(:,3);y=y2(:,3);

h=h/2;

end

end

k=k+1;

end

end

%%------------Метод Рунге-Кутта 3-го порядка-----------

%выбор начального шага

s=3;%порядок точности метода

delta=(1/max(abs([x0; xk])))^(s+1)+norm(f(x0,y0),"inf")^(s+1);

h=(eps/delta)^(1/(s+1));

n0G=ceil((xk-x0)/h);

h0G=(xk-x0)/n0G;%полученный начальный шаг

h0 = h0G;n0=n0G;

%сравниваем на 2-х разных размерах шага

[x1,y1]=RK3(x0,xk,h0/2,y0);

[x2,y2]=RK3(x0,xk,h0,y0);

k=length(y2);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

%оценка полной погрешности

E0=norm(ee/(2^s-1),"inf");

%если шаг слишком маленький

while E0<eps/(2^(s+1))

n0 = ceil(n0/2);

h0 = (xk-x0)/n0;

[x1,y1]=RK3(x0,xk,h0/2,y0);

[x2,y2]=RK3(x0,xk,h0,y0);

k=length(y2);

ee=zeros(1,k);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

E0=norm(ee/(2^s-1),"inf");

end

%если шаг слишком большой

while E0>eps

n0 = 2\*n0;

h0 = (xk-x0)/n0;

[x1,y1]=RK3(x0,xk,h0/2,y0);

[x2,y2]=RK3(x0,xk,h0,y0);

k=length(y2);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

E0=norm(ee/(2^s-1),"inf");

end

%строим графики

plot(x1,y1(1,:),'-\*',x1,y1(2,:),'-o')

grid on

legend('y1(x)','y2(x)','Location','best')

xlabel('x')

title('Решение системы с постоянным шагом (eps = 1e-4), RK3')

Полученный график решения с постоянным шагом :



3.2. Для выбранного метода (схемы оппонента) из п. 3.1 и реализованного Вами в п.1.2 2-х этапного ЯМРК 2-го порядка для решения задачи Коши (1) с постоянным шагом определите величину шага интегрирования h, обеспечивающего вычисление приближенного решения с заданной точностью (). Постройте графики зависимости истинной полной погрешности от значения независимой переменной х при интегрировании с полученным шагом h.

Скрипты, реализующие анализ полученных решений приведены в Приложении, здесь и далее приведём лишь результирующие графики:





3.3. Для схемы оппонента из п. 3.1 и реализованного Вами в п.2.1 2-х этапного ЯМРК 2-го порядка для решения задачи Коши (1) с автоматическим выбором шага интегрирования постройте:

3.3.1 графики зависимости величины шага интегрирования от значения независимой переменной х;





3.3.2 графики зависимости отношения истинной локальной погрешности к полученной оценке локальной погрешности от значения независимой переменной х (см. формулу (91)).





3.3.3 графики зависимости количества вычислений правой части системы от заданной точности (например, ).





На основании полученных результатов сделайте вывод о надежности и экономичности реализованных алгоритмов.

**Выводы**

1. Для методов с постоянным шагом для достижения точности : для построенного нами метода второго порядка достаточно шага , а для метода Рунге-Кутта 3-го порядка шага . При этом погрешность решения возрастает на интервале .
2. При решении с автоматическим шагом мы начинаем с рекомендованного шага и далее для нашей системы шаг уменьшается в двое на каждом шаге сетки, стабилизируясь на величине для метода второго порядка и на величине 1.4 для метода третьего порядка.
3. Графики зависимости отношения истинной локальной погрешности к полученной оценке локальной погрешности от значения независимой переменной х показывают, что отношение не превосходит 1, т.е. истинная погрешность не превосходит её оценки по правилу Рунге.
4. Количество вычислений правой части системы для метода 2-го порядка возрастает с ростом требуемой точности сильнее, чем для метода 3-го порядка. Так при точности первый требует 54 вычисления, а второй 45. При точности эффективность метода 3-го порядка возрастает: 1632 вычислений против 378 для методов 2-го и 3-го порядка соответственно.

Приложение

clear all

close all

clc

c2=1/12;%параметр расчётной схемы

x0=0; xk=pi;

%%Часть №1. Расчетные схемы метода Рунге-Кутты с постоянным шагом

%выбор начального шага

s=2;%порядок точности метода

eps = 1e-4;

y0 = [pi/20 pi/12]';%начальные условия

delta=(1/max(abs([x0; xk])))^(s+1)+norm(f(x0,y0),"inf")^(s+1);

h=(eps/delta)^(1/(s+1));

n0G=ceil((xk-x0)/h);

h0G=(xk-x0)/n0G;%полученный начальный шаг

h0 = h0G;n0=n0G;

%сравниваем на 2-х разных размерах шага

[x1,y1]=RK2(x0,xk,h0/2,y0,c2);

[x2,y2]=RK2(x0,xk,h0,y0,c2);

k=length(y2);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

%оценка полной погрешности

E0=norm(ee/(2^s-1),"inf");

%если шаг слишком маленький

while E0<eps/(2^(s+1))

n0 = ceil(n0/2);

h0 = (xk-x0)/n0;

[x1,y1]=RK2(x0,xk,h0/2,y0,c2);

[x2,y2]=RK2(x0,xk,h0,y0,c2);

k=length(y2);

ee=zeros(1,k);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

E0=norm(ee/(2^s-1),"inf");

end

%если шаг слишком большой

while E0>eps

n0 = 2\*n0;

h0 = (xk-x0)/n0;

[x1,y1]=RK2(x0,xk,h0/2,y0,c2);

[x2,y2]=RK2(x0,xk,h0,y0,c2);

k=length(y2);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

E0=norm(ee/(2^s-1),"inf");

end

%строим графики

plot(x1,y1(1,:),'-\*',x1,y1(2,:),'-o')

grid on

legend('y1(x)','y2(x)','Location','best')

xlabel('x')

title('Решение системы с постоянным шагом (eps = 1e-4), RK2')

%%

%%Часть №2. Расчетные схемы метода Рунге-Кутты с автоматическим выбором шага

ro=1e-5;%локальная погрешность

%выбор начального шага

eps = ro;

delta=(1/max(abs([x0; xk])))^(s+1)+norm(f(x0,y0),"inf")^(s+1);

h=(eps/delta)^(1/(s+1));

n0G=ceil((xk-x0)/h);

h0G=(xk-x0)/n0G;%полученный начальный шаг

[x3,y3,r,hh,nf]=RK2auto(x0,xk,h0G,y0,ro,c2);

%%

%%Часть №3. Анализ эффективности методов Рунге-Кутты

%3.2. для постоянного шага

ytoch = tochno(x1);%точное решение

%истинная полная погрешность

yrazn=abs(ytoch-y1);

figure

plot(x1,yrazn(1,:),'-\*',x1,yrazn(2,:),'-o')

grid on

legend('y1(x)','y2(x)','Location','best')

xlabel('x')

ylabel('истинная полная погрешность')

title(strcat('Постоянный шаг h = ',num2str(h0/2),', e=1e-4, RK2'))

%3.3. для автоматического шага

ytoch = tochno(x3);%точное решение

%истинная полная погрешность

yrazn=abs(ytoch-y3);

%график зависимости величины шага интегрирования от значения независимой переменной х

figure

plot(x3(2:length(x3)),hh,'-\*')

xlabel('x')

ylabel('Шаг интегрирования')

title('Зависимость шага интегрирования от х (авт. шаг, ro=1e-5),RK2')

grid on

%графики зависимости отношения истинной локальной погрешности

%к полученной оценке локальной погрешности от значения независимой переменной х

kk=length(x3)-1;

for i=1:kk

otn(i)=r(i)/max(yrazn(:,i+1));

end

figure

plot(x3(2:kk+1),otn,'-\*')

xlabel('x')

title('Отношение истинной локальной погрешности к её оценке (авт. шаг, ro=1e-5, RK2)')

grid on

%график зависимости количества вычислений правой части системы от заданной точности ε

nmax = 10;

i=1:nmax; epsi=10.^(-i);

for i=1:nmax

delta=(1/max(abs([x0; xk])))^(s+1)+norm(f(x0,y0),"inf")^(s+1);

h=(epsi(i)/delta)^(1/(s+1));

n0G=ceil((xk-x0)/h);

h0G=(xk-x0)/n0G;%полученный начальный шаг

[x3,y3,r,hh,nf]=RK2auto(x0,xk,h0G,y0,epsi(i),c2);

nfi(i)=nf;

end

figure

plot(1:nmax,nfi,'-\*')

xlabel('i')

ylabel('Количество вычислений правой части системы')

title('Число вычислений в зависимости от погрешности e = 1/10^i, RK2')

grid on

disp(nfi)

%%------------Метод Рунге-Кутта 3-го порядка-----------

%выбор начального шага

s=3;%порядок точности метода

delta=(1/max(abs([x0; xk])))^(s+1)+norm(f(x0,y0),"inf")^(s+1);

h=(eps/delta)^(1/(s+1));

n0G=ceil((xk-x0)/h);

h0G=(xk-x0)/n0G;%полученный начальный шаг

h0 = h0G;n0=n0G;

%сравниваем на 2-х разных размерах шага

[x1,y1]=RK3(x0,xk,h0/2,y0);

[x2,y2]=RK3(x0,xk,h0,y0);

k=length(y2);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

%оценка полной погрешности

E0=norm(ee/(2^s-1),"inf");

%если шаг слишком маленький

while E0<eps/(2^(s+1))

n0 = ceil(n0/2);

h0 = (xk-x0)/n0;

[x1,y1]=RK3(x0,xk,h0/2,y0);

[x2,y2]=RK3(x0,xk,h0,y0);

k=length(y2);

ee=zeros(1,k);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

E0=norm(ee/(2^s-1),"inf");

end

%если шаг слишком большой

while E0>eps

n0 = 2\*n0;

h0 = (xk-x0)/n0;

[x1,y1]=RK3(x0,xk,h0/2,y0);

[x2,y2]=RK3(x0,xk,h0,y0);

k=length(y2);

for i=2:k

ee(i)=max(abs(y2(:,i)-y1(:,2\*i-1)));

end

E0=norm(ee/(2^s-1),"inf");

end

%строим графики

figure

plot(x1,y1(1,:),'-\*',x1,y1(2,:),'-o')

grid on

legend('y1(x)','y2(x)','Location','best')

xlabel('x')

title('Решение системы с постоянным шагом (eps = 1e-4), RK3')

%3.2. для постоянного шага

ytoch = tochno(x1);%точное решение

%истинная полная погрешность

yrazn=abs(ytoch-y1);

figure

plot(x1,yrazn(1,:),'-\*',x1,yrazn(2,:),'-o')

grid on

legend('y1(x)','y2(x)','Location','best')

xlabel('x')

ylabel('истинная полная погрешность')

title(strcat('Постоянный шаг h = ',num2str(h0/2),', e=1e-4, RK3'))

%3.3. для автоматического шага

ro=1e-5;%локальная погрешность

%выбор начального шага

eps = ro;

delta=(1/max(abs([x0; xk])))^(s+1)+norm(f(x0,y0),"inf")^(s+1);

h=(eps/delta)^(1/(s+1));

n0G=ceil((xk-x0)/h);

h0G=(xk-x0)/n0G;%полученный начальный шаг

[x3,y3,r,hh,nf]=RK3auto(x0,xk,h0G,y0,ro);

ytoch = tochno(x3);%точное решение

%истинная полная погрешность

yrazn=abs(ytoch-y3);

%график зависимости величины шага интегрирования от значения независимой переменной х

figure

plot(x3(2:length(x3)),hh,'-\*')

xlabel('x')

ylabel('Шаг интегрирования')

title('Зависимость шага интегрирования от х (авт. шаг, ro=1e-5),RK3')

grid on

%графики зависимости отношения истинной локальной погрешности

%к полученной оценке локальной погрешности от значения независимой переменной х

kk=length(x3)-1;

otn=zeros(1,kk);

for i=1:kk

otn(i)=r(i)/max(yrazn(:,i+1));

end

figure

plot(x3(2:kk+1),otn,'-\*')

xlabel('x')

title('Отношение истинной локальной погрешности к её оценке (авт. шаг, ro=1e-5, RK3)')

grid on

%график зависимости количества вычислений правой части системы от заданной точности ε

nmax = 10;

i=1:nmax; epsi=10.^(-i);

for i=1:nmax

delta=(1/max(abs([x0; xk])))^(s+1)+norm(f(x0,y0),"inf")^(s+1);

h=(epsi(i)/delta)^(1/(s+1));

n0G=ceil((xk-x0)/h);

h0G=(xk-x0)/n0G;%полученный начальный шаг

[x3,y3,r,hh,nf]=RK3auto(x0,xk,h0G,y0,epsi(i));

nfi(i)=nf;

end

figure

plot(1:nmax,nfi,'-\*')

xlabel('i')

ylabel('Количество вычислений правой части системы')

title('Число вычислений в зависимости от погрешности e = 1/10^i, RK3')

grid on

%% Пользовательские функции

%Точное решение

function F = tochno(t)

A = 1/12; B = 1/20;

c=sqrt(A\*B);

F(1,:)=B\*pi\*cos(c\*t)+A\*pi\*sin(c\*t)\*sqrt(A/B);

F(2,:)=A\*pi\*cos(c\*t)-B\*pi\*sin(c\*t)\*sqrt(B/A);

end

% определяем значение правых частей системы ДУ

function F = f(t, X)

A = 1/12; B = 1/20;

k=2; %порядок системы

F = zeros(2,1); %вектор правых частей

F(1)=A\*X(2);

F(2)=-B\*X(1);

end

%Метод второго порядка

function [tn,YY] =RK2(A,B,h,y,c2)

b2 = 1/(2\*c2);

b1 = 1-b2;a21=c2;

t = A;

n=ceil((B-A)/h);

tn=A:h:B;%генерируем n+1 моментов времени

k=length(y);

yn = zeros(k,1);

yn(:,1) = y(:);

for i = 1:n

k1 = h\*f(t,y);

k2 = h\*f(t+c2\*h,y+a21\*k1);

y = y+b1\*k1+b2\*k2;

t = t + h;

yn(:,i+1) = y(:);

end

YY=yn;

end

%Метод второго порядка с автоматическим выбором шага

%hh - i-ый шаг

%nf-число вычислений правой части

function [tn,YY,r,hh,nf] =RK2auto(A,B,h,y,ro,c2)

k = 1;%счётчик числа узлов

t = A; tn(1)=A; YY(:,1) = y;

s=2;

%(считаем что вся система - это один раз подсчёт правой части)

nf = 0;

while (t<=B)

[t1,y1] = RK2(t,t+h,h,y,c2);%считаем на шаге h

[t2,y2] = RK2(t,t+h,h/2,y,c2);%считаем на шаге h/2

nf = nf + 6;%всего за один вызов посчитали 2\*число точек х

%оценка локальной погрешности

roi = max(abs(y2(:,3)-(y1(:,2))))/(1-2^(-s));%на шаге h

if roi <= ro %данный шаг нас устраивает

hh(k)=h;t=t+h;tn(k+1)=t;

YY(:,k+1)=y1(:,2);y=y1(:,2);

if roi< ro/(2^(s+1))

h=2\*h; %можно увеличить в 2 раза после

end

r(k)= roi;

else

r(k)= roi/(2^s);

if roi > ro\*(2^s)%точность неподходящая

%шаг уменьшаем вдвое

h=h/2;hh(k)=h;

t=t+h;tn(k+1)=t;

YY(:,k+1)=y2(:,2);y=y2(:,2);

else %точность неподходящая, но можно использовать y2

hh(k)=h;t=t+h;tn(k+1)=t;

YY(:,k+1)=y2(:,3);y=y2(:,3);

h=h/2;

end

end

k=k+1;

end

end

%Метод третьего поряка

function [tn,YY] =RK3(A,B,h,y)

t = A;

n=ceil((B-A)/h);

tn=A:h:B;%генерируем n+1 моментов времени

k=length(y);

yn = zeros(k,1);

yn(:,1) = y(:);

for i = 1:n

k1 = h\*f(t,y);

k2 = h\*f(t+h/2,y+k1/2);

k3 = h\*f(t+h,y-k1+2\*k2);

y = y+(k1+4\*k2+k3)/6;

t = t + h;

yn(:,i+1) = y(:);

end

YY=yn;

end

%Метод третьего поряка с автоматическим выбором шага

%hh - i-ый шаг

%nf-число вычислений правой части

function [tn,YY,r,hh,nf] =RK3auto(A,B,h,y,ro)

k = 1;%счётчик числа узлов

t = A; tn(1)=A; YY(:,1) = y;

s=3;%порядок точности

%(считаем что вся система - это один раз подсчёт правой части)

nf = 0;

while (t<=B)

[t1,y1] = RK3(t,t+h,h,y);%считаем на шаге h

[t2,y2] = RK3(t,t+h,h/2,y);%считаем на шаге h/2

nf = nf + 9;%всего за один вызов посчитали 3\*число точек х

%оценка локальной погрешности

roi = max(abs(y2(:,3)-(y1(:,2))))/(1-2^(-s));%на шаге h

if roi <= ro %данный шаг нас устраивает

hh(k)=h;t=t+h;tn(k+1)=t;

YY(:,k+1)=y1(:,2);y=y1(:,2);

if roi< ro/(2^(s+1))

h=2\*h; %можно увеличить в 2 раза после

end

r(k)= roi;

else

r(k)= roi/(2^s);

if roi > ro\*(2^s)%точность неподходящая

%шаг уменьшаем вдвое

h=h/2;hh(k)=h;

t=t+h;tn(k+1)=t;

YY(:,k+1)=y2(:,2);y=y2(:,2);

else %точность неподходящая, но можно использовать y2

hh(k)=h;t=t+h;tn(k+1)=t;

YY(:,k+1)=y2(:,3);y=y2(:,3);

h=h/2;

end

end

k=k+1;

end

end