ОТЧЕТ

ОТВЕТЫ К ПРАКТИЧЕСКОМУ ЗАНЯТИЮ: Метод простой итерации

Выполнил:
Студент 3-го курса гр. 09-711
Лагунов Кирилл Игоревич

Упражнение 1

На каком шаге итерационного процесса была достигнута требуемая точность?

Ответ:
на 16-ом

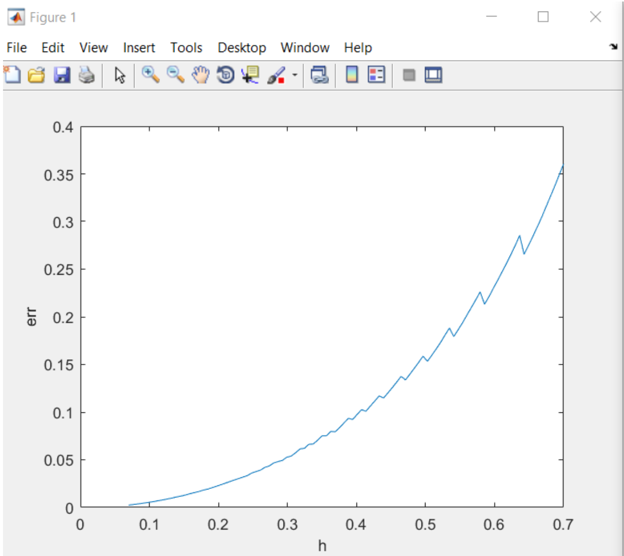
a	0
b	7
eps	1.0000e-03
er	0.0025
f	@(x)x*0+1
h	0.0700
iter	16
K	@(x,s)x*0+s*0+1
x	*1x101 double*
y	*1x101 double*
y_approx	*1x101 double*
y_exact	@(x)exp(x)

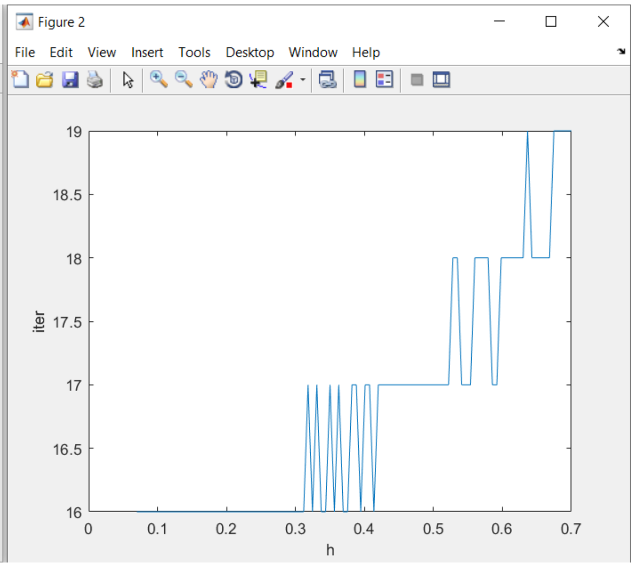
Упражнение 2

Исследуйте, как зависит от шага сетки h точность вычислений и скорость сходимости итерационного процесса.

Решение:

```
figure;
h = linspace(0.07,0.7,100); % сетка шага
n = size(h,2);
err = zeros(n,1);
iter = zeros(n,1);
for i = 1:1:n
x = a:h(i):b; %разбиение [a,b] при фиксированном размере шага
y_exact_in_nodes = y_exact(x);%точное значение в узлах сетки
[y_approx,it] = IterVolt(x,h(i),eps,f,K);%приближенное решение
iter(i) = it;
err(i) = norm((y_exact_in_nodes -
y_approx)/norm(y_exact_in_nodes,inf),inf);
%норма относительной ошибки
end
plot(h,err);
xlabel('h'); ylabel('err');
figure;
plot(h,iter);
xlabel('h'); ylabel('iter');
```





Исследуя точность и скорость вычисления от выбранного шага сетки h, выяснили: чем больше h, тем меньше точность (больше относительная ошибка), и в то же время скорость уменьшается (количество итераций увеличивается). При увеличении h количество итераций начинает расти экспоненциально (например, если шаг будет 1, то уже будет 23 итерации), что можно увидеть из правого графика.

Упражнение 3

С помощью функции Iter_Volt.m найдите приближенное решение уравнения:

$$y(x) = x + \int_0^x (x - s)y(s)ds, \qquad x \in [0,2\pi]$$

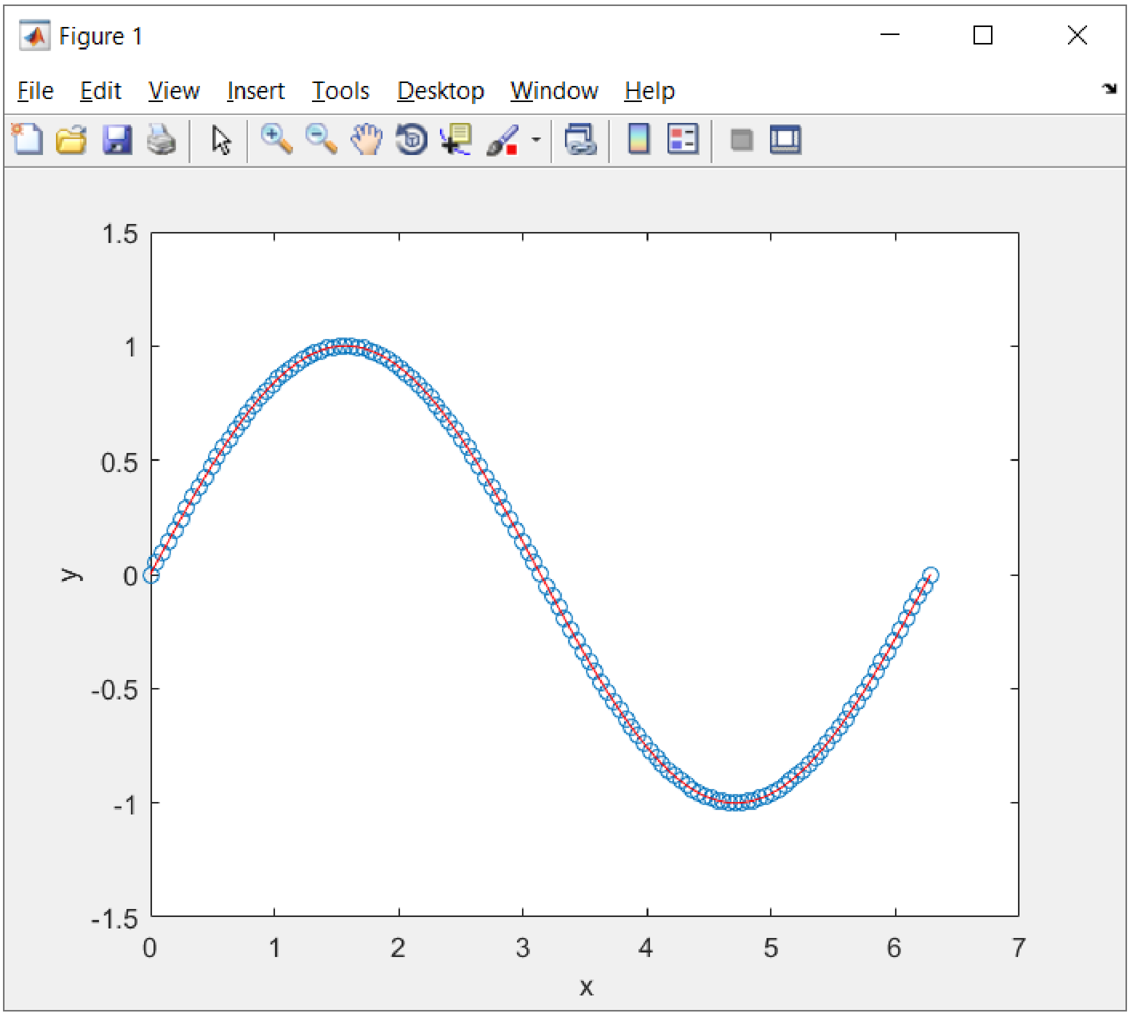
Его точное решение $y(x) = sinx$.

Решение:

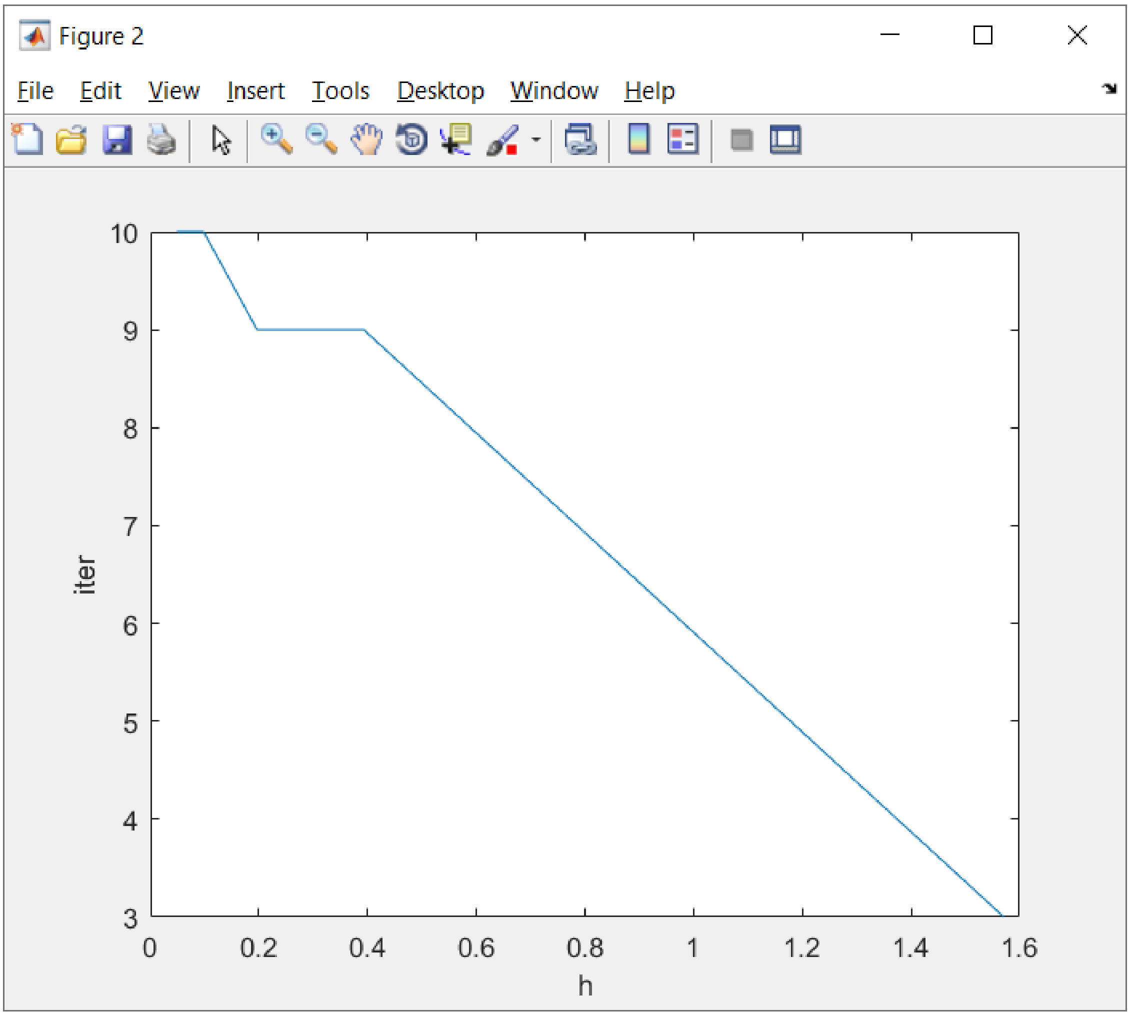
```
function [yk] = CalcInt(y,h,x,n,K,f)
yk = y;
for i = 1 : n
yk(i) = 0;
for j = 1 : i
yk(i) = yk(i) + 2*K(x(i),x(j))*y(j);
end
yk(i) = yk(i) - K(x(i),x(1))*y(1) - K(x(i),x(i))*y(i);
yk(i) = f(x(i)) + yk(i)*h/2;
end
end


function [yk,iter] = IterVolt(x,h,eps,f,K)
n = numel(x);
y = f(x);
yk = CalcInt(y,h,x,n,K,f);
iter = 0;
while norm(yk-y,inf)/norm(yk,inf) > eps
y = yk;
yk = CalcInt(y,h,x,n,K,f);
iter = iter + 1;
end
end

clear all
close all
clc
f = @(x) x; %f правой части
K = @(x,s) -x + s; %ядро
a = 0; %начало
b = 2*pi; %конец
h = pi/64; %шаг сетки
eps = 1e-03; %погрешность
y_exact = @(x) sin(x);
x = a : h : b;
[y_approx,iter] = IterVolt(x,h,eps,f,K);
y=y_exact(x);
plot(x,y,'o',x,y_approx,'r');
er = norm(y-y_approx,inf)/norm(y,inf);
xlabel('x');
ylabel('y');
```



При данном h (129 элементов сетки) вышло 10 итераций



Интересно, что при увеличении шага, скорость падает линейно (хотя в предыдущем задании мы наблюдали другое). Зависит ли это от самой функции скорость итерационного процесса?

Упражнение 4

Напишите функцию, реализующую метод последовательных приближений, на основе квадратурной формулы Симпсона (12), с. 9. Решите с помощью этой функции уравнения из этого параграфа и сравните эффективность применения метода Симпсона с использованием метода трапеций.

Решение:

Не сделал

Упражнение 5

Напишите функцию, предназначенную для решения методом простой итерации нелинейных уравнений Вольтерра второго рода. Найдите приближенное решение уравнения

$$y(x) = \int_0^x \frac{1+y^2(x)}{1+s^2} ds, \; x \in [0,10]$$

Точное решение $y(x) = x$

Решение:

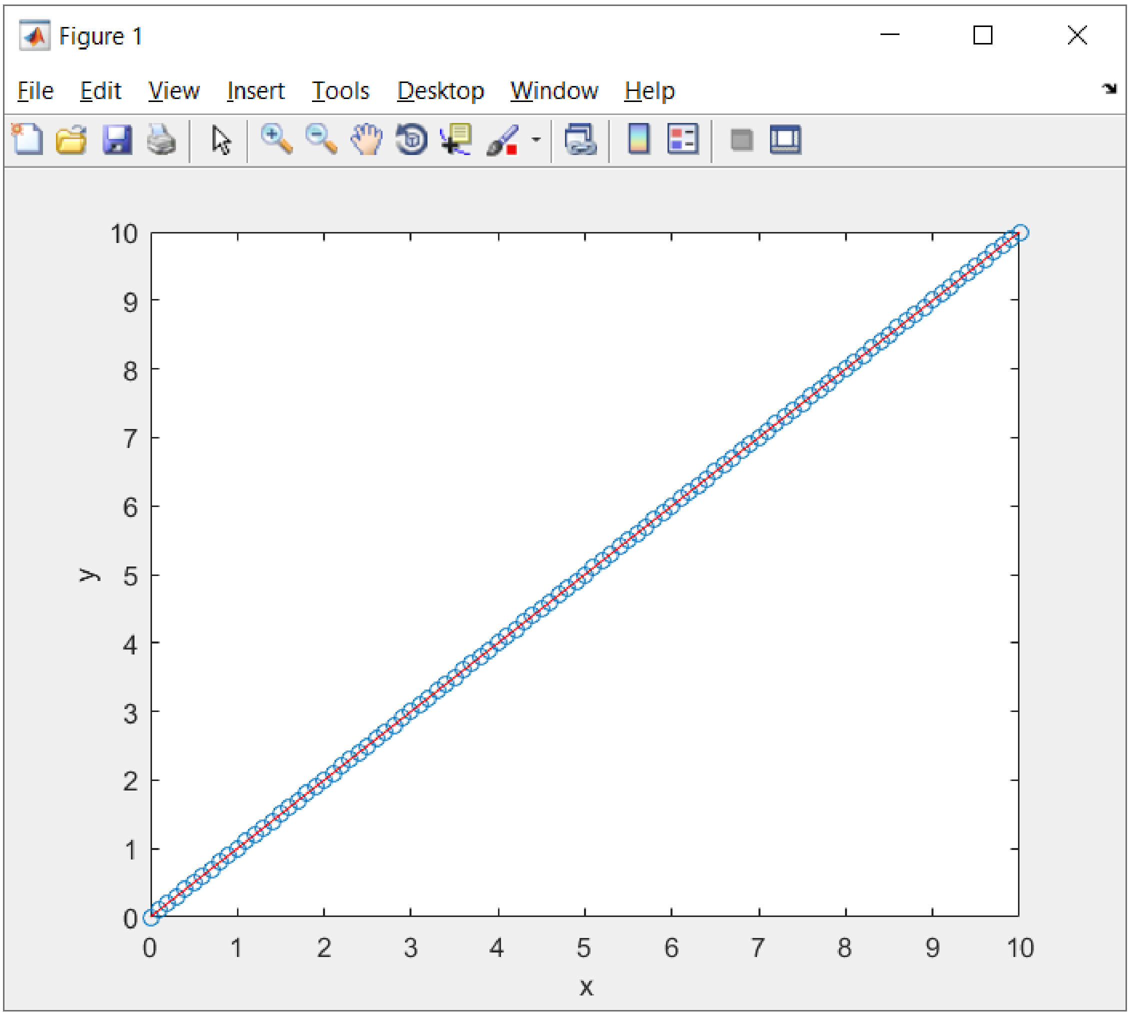
```
function [yk] = Nonlin_CalcInt(y,h,x,n,K,f)
yk = y;
for i = 1 : n
yk(i) = 0;
for j = 1 : i
yk(i) = yk(i) + 2*K(x(i),x(j),y(j));
end
yk(i) = yk(i) - K(x(i),x(1),y(1)) - K(x(i),x(i),y(i));
yk(i) = f(x(i)) + yk(i)*h/2;
end
end
```

```
function [yk,iter] = Nonlin_IterVolt(x,h,eps,f,K)
n = numel(x);
y = zeros(n,1);
for i=1:n
y(i)= f(x(i));
yk = Nonlin_CalcInt(y,h,x,n,K,f);
iter = 0;
while norm(yk-y,inf)/norm(yk,inf) > eps
y = yk;
yk = Nonlin_CalcInt(y,h,x,n,K,f);
iter = iter + 1;
end
end
```

```
clear all
close all
clc
f = @(x) x0; %f правой части
K = @(x,s,y) x*0+(1+y*y)/(1+s*s); %Ядро
a = 0; %начало
b = 10; %конец участка
h = 0.1; %шаг
eps = 1e-03; %погрешность
y_exact = @(x) x;
x = a : h : b;
[y_approx,iter] = Nonlin_IterVolt(x,h,eps,f,K);
y=y_exact(x);
plot(x,y,'o',x,y_approx,'r');
```

```
xlabel('x');
ylabel('y');
```

Получим:



Всё сошлось