UNIVERSITY OF LEICESTER

Computer Assignment 4

For Scientific Computing MA3012/MA7012

Gleb Vorobchuk

Leicester

December 2017

Task №1.

1. [10 marks] Given a vector of n data points $x = (x_0, x_1, ..., x_{n-1})^T$, the Discrete Fourier transform (DFT) is defined as

$$y_k = \sum_{l=0}^{n-1} x_l \omega_n^{kl}, \quad k = 0, \dots, n-1.$$

where $\omega_n = e^{-\frac{2\pi i}{n}}$ is the primitive n^{th} root of unity. Write a Matlab function `fastft.m` which implements the Fast Fourier Transform (FFT) algorithm for computing the DFT. Your function should have x as the input and $y = (y_0, y_1, ..., y_{n-1})^T$ as the output. Use the algorithm in Chapter 12 slide 16 of Heath's lecture notes as a template for your function. Note that in Heath's template n and ω are also specified as input variables. This is not necessary, since n can be determined from the length of the input vector x and ω can be calculated according to its definition with the given value of n. Test your function by calculating the DFT of the input vector $x = (0, 3, 3, 3, 1, -2, 1, -3)^T$. The result should be the same as that obtained with the Matlab intrinsic function `fft`.

```
function [y]= fastft(x)
n=length(x);
w=exp(((-2*pi)*1i)/n);
m=n/2;
y=zeros(n,1)
    if n == 1
        y(n)=x(n);
    else

        even=zeros(m,1);
        odd=zeros(m,1);

        for k =  0: (m-1)
            even(k+1)=x(2*k+2);
            odd(k+1)=x(2*k+1);
        end
        t=fastft(even);
        q=fastft(odd);
        for k=0 : (n-1)
           l=mod(k,(n/2))+1;
           y(k+1)=q(l)+(w.^k*t(l));
        end
    end
end
```

Task №2.

2. [20 marks]

a) Composite Simpson's rule with n intervals has the form

$$\int_a^b f(x)dx \approx \frac{h}{6}\left[f(a) + f(b) + 4\sum_{i=1}^{n} f\left(a + \left(i - \frac{1}{2}\right)h\right) + 2\sum_{i=1}^{n-1} f(a + ih)\right],$$

where $h = \frac{b-a}{n}$. Write a Matlab function with the header line `function S = simpson(f, a, b, n)` to compute this. Test your function on the following example:

```
>> format long
>> S = simpson(@(x)1./(1-sin(x)),0,pi/3,16)
S =
   2.732058440959821
```

```
function [S] = simpson(f, a, b, n)
h=(b-a)/n

for i=1:n
     s1(i)=f(a+(i-1/2)*h)
end
for i=1:n-1
     s2(i)=f(a+i*h)
end

S=(h/6) * (f(a) + f(b) + 4 * sum(s1) + 2 * sum(s2))

end
```

b) Write a Matlab function `gaussquad.m` which calculates approximate value of the integral $\int_a^b f(x)dx$ using the Gaussian Quadrature with up to 3 nodes. The function header should be `function G = gaussquad(f, a, b, k)`, where `k` can take values 1, 2, or 3. The nodes and weights can be found in the table on p. 112 of Pav's lecture notes (note that $k = n + 1$). You should get the following results for the test example:

```
>> G = gaussquad(@(x)1./(1-sin(x)),0,pi/3,1)
G =
   2.094395102393195
>> G = gaussquad(@(x)1./(1-sin(x)),0,pi/3,2)
G =
   2.647865728063311
>> G = gaussquad(@(x)1./(1-sin(x)),0,pi/3,3)
G =
   2.723239447875524
```

```
function [G] = gaussquad(f, a, b, k)
for i=1:k
    if k==1
        x(1)=0
        A(1)=2
    elseif k==2
        x(1)=-sqrt(1/3)
        x(2)=sqrt(1/3)
        A(i)=k-1
    elseif k==3
        x(1)=-sqrt(3/5);
        x(2)=0;
        x(3)=sqrt(3/5);
        A(1)=5/9;
        A(2)=8/9;
        A(3)=5/9;
    end
end
G=sum(A.*((b-a)/2) .* f(((b-a).*x+(b+a))/2))
end
```

c) Write a Matlab function with a header line `function G = compgaussquad(f, a, b, k, n)`, which uses the function from part b), or otherwise, to approximate the integral $\int_a^b f(x)dx$ using a *composite* Gaussian quadrature with n intervals. Test your function with the following example:

```
>> G = compgaussquad(@(x)1./(1-sin(x)),0,pi/3,3,16)
G =
   2.732050802516621
```

```
function CP = copmgaussquad(f,a,b,k,n)
iappx = 0;
x(1)=a;
for i=2:n
  x(i) = a + (b-a) * i/ n;
end
x
for i=1:n-1
iappx(i) = gaussquad(f,x(i),x(i+1),k);
end
CP=sum(iappx)
end
```

d) Use your functions from a) and c) to find approximate value of the integral $\int_0^{\pi/3} 1/(1-\sin x)\,dx$ using $n = 2^m$ intervals with $m = 2, 3, \ldots, 10$ and $k = 1, 2,$ and 3 in case of the composite Gaussian quadrature. Given that the exact value of the integral is $1 + \sqrt{3}$, plot the absolute error for each method as a function of $h = (b-a)/n$ on a log-log scale. From the obtained plots, determine the order of accuracy of each method, i.e., you should

observe that the error is $O(h^p)$ and determine p. Comment on the observed order of accuracy of the four methods compared to the theoretical prediction.

```
function [P]=plotgausserror()
f=@(x)1./(1-sin(x));
a=0; b=pi/3;
m=(2:10);
k=(1:3);
EXV=1+sqrt(3);
for i=1:length(m)
    n(i)=2^m(i);
    h(i)=(b-a)/n(i);
end
for i=1:length(k)
    for j=1:length(n)
        CP(j,i) = copmgaussquad(f,a,b,k(i),n(j));
        SP(j) = simpson(f, a, b, n(j));
        ERR1(i,j)=abs(EXV-CP(j,i));
        ERR2(j)=abs(EXV-SP(j));

    end
end
for i=1:length(k)
    for j=1:length(n)-1
        p1(i,j)=(log(ERR1(i,j))-log(ERR1(i,j+1)))/(log((b-a)/n(j))-
log((b-a)/n(j+1)));
        p2(j)=(log(ERR2(j))-log(ERR2(j+1)))/(log((b-a)/n(j))-log((b-a)/
n(j+1)));
    end
end
p1%compgaussquad k=1
p2%simpson
P1=min(p1)
P2=min(p2)
loglog(h,ERR1(1,:),h,ERR1(2,:),h,ERR1(3,:),h,ERR2);
xlabel('h');ylabel('Error')
legend('compgaussquad k=1','compgaussquad k=2','compgaussquad
k=3','simpsons')
end
```

3. [10 marks] Consider the initial value problem $du/dt = u(1 - u^2)$, $u(0) = 0.1$.

a) Write a Matlab function `flowmap.m` which calculates the flow map of this initial value problem. *Hint:* The function should output the value of the exact solution $u(t)$ of this IVP.

b) Write a Matlab program which computes the local and global errors of Euler's method along the solution from $t = 0$ to $t = 5$. Plot the local and global errors as functions of t for the numerical solutions with step sizes $h = 0.2$ and $h = 0.1$. Comment on how the local and global errors scale with the step size.

c) Same as b), but for the explicit midpoint method (see bottom of p. 10 of my lecture notes on Numerical Solution of ODEs).

File flowmap.m

```
function [U] = flowmap(u0,t)
U=u0*exp(t)/(sqrt(-
u0^2+(1+u0^2*exp(2*t))))
end
```

File exact.m

```
function [UE,UE2,t,t2]=exact(u0)
t=0:0.1:5
t2=0:0.2:5
UE(1)=u0
UE2(1)=u0

for k=2:length(t)
    UE(k)=flowmap(u0,0.1)
end
for k=2:length(t2)
    UE2(k)=flowmap(u0,0.2)
end
end
```

File euler.m

```
function U=euler(f,a,b,u0,N)
h=(b-a)/N;
U=zeros(1,N);
U(1)=u0;
for j=1:N
    U(j+1)=U(j)+h*f(U(j));
end
end
```

File err.m

```
function [LER,GER] = err(E,T,h)

LER(1)=0.1-E(1);
for i=2:length(T)
LER(i)=flowmap(E(i-1),h)-E(i);
end



for i=1:length(T)
GER(i)=flowmap(0.1,T(i))-E(i);
end
GER=GER'
end
```

File ploterror.m

```
function [P]=ploterror(u0)

[UE,UE2,t,t2]=exact(u0);


E=euler(@(u)u*(1-u^2),0,5,u0,50);
E2=euler(@(u)u*(1-u^2),0,5,u0,25);

[LER,GER] = err(E,t,0.1);
[LER2,GER2] = err(E2,t2,0.2);

figure(1)
xlabel('t'), ylabel('Error');
plot(t,GER,t,LER);
title('Local and Global Error h=0.1');
legend('Global error','Local error');

figure(2);
xlabel('t'), ylabel('Error');
plot(t2,GER2,t2,LER2);
legend('Global error','Local error');
title('Local and Global Error h=0.2')



end
```

File midpoint.m

```
function UM=midpoint(f,a,b,u0,N)
h=(b-a)/N;
UM=zeros(1,N);
UM(1)=u0;
for j=1:N
    UU(j)=UM(j)+h/2*f(UM(j))
    UM(j+1)=UM(j)+h*f(UU(j))
end
end
```

File ploterror_midpoint.m

```
function [P]=ploterror_midpoint(u0)

[UE,UE2,t,t2]=exact(u0);


E=midpoint(@(u)u*(1-u^2),0,5,u0,50);
E2=midpoint(@(u)u*(1-u^2),0,5,u0,25);

[LER,GER] = err(E,UE,t);
[LER2,GER2] = err(E2,UE2,t2);

figure(1)
xlabel('t'), ylabel('Error');
plot(t,LER,t,GER);
title('Local and Global Error h=0.1');
legend('local error','global error');

figure(2);
xlabel('t'), ylabel('Error');
plot(t2,LER2,t2,GER2);
legend('local error','global error');
title('Local and Global Error h=0.2')



end
```