%hf3. egyenletek

%1 linear equations:

A=[3 6 4; 1 5 0; 0 7 7];

b=[1; 2; 3];

rank(A);

x=A\b

%x=inv(A)\*b

error=abs(A\*x-b)

x =

 -0.5824

 0.5165

 -0.0879

error =

 1.0e-015 \*

 0.1110

 0

 0

%2 numerical integration:

analytical=-24\*exp(-5/3)+9;

L=0:5/10000:5;

Y=(L.\*exp(-L/3));

T=trapz(L,Y)

errorT=abs(analytical-T)

T =

 4.4670

errorT =

 2.3457e-008

F=@(x1) x1.\*exp(-x1/3);

Q=quad(F, 0, 5)

errorQ=abs(analytical-Q)

Q =

 4.4670

errorQ =

 2.0131e-009

%3 Computing the invers:

M=[1 2; 3 4];

IM=inv(M)

MxIM=M\*IM

IM =

 -2.0000 1.0000

 1.5000 -0.5000

MxIM =

 1.0000 0

 0.0000 1.0000

%4 Fitting polinomials:

ndata=load('randomData.mat');

x=ndata.x;

y=ndata.y;

[p, s, m]=polyfit(x, y, 1);

y1=polyval(p, x, s, m);

[p, s, m]=polyfit(x, y, 2);

y2=polyval(p, x, s, m);

[p, s, m]=polyfit(x, y, 3);

y3=polyval(p, x, s, m);

[p, s, m]=polyfit(x, y, 4);

y4=polyval(p, x, s, m);

[p, s, m]=polyfit(x, y, 5);

y5=polyval(p, x, s, m);

figure;

plot(x, y, 'ko', 'MarkerSize', 3, 'MarkerFaceColor', 'k');

hold on;

plot(x, y1, 'g', x, y2, 'b', x, y3, 'c', x, y4, 'm', x, y5, 'r');

xlabel('X');

ylabel('Y');

title('Polynomial fits to noisy data');

legend('Data', 'Order 1', 'Order 2', 'Order 3', 'Order 4', 'Order 5');



%5. Hodgkin-huxley model of the neuron:

function dnmhVdt=nmhV(t, nmhV)

%nmhV(1)=n;

%nmhV(2)=m;

%nmhV(3)=h;

%nmhV(4)=V;

%dnmhVdt(1)=dndt;

%dnmhVdt(2)=dmdt;

%dnmhVdt(3)=dhdt;

%dnmhVdt(4)=dVdt;

c=1;

gk=36;

gna=120;

gl=0.3;

ek=-72;

ena=55;

el=-49.4;

dnmhVdt=zeros(4, 1);

dnmhVdt(1)=(1-nmhV(1))\*alphan(nmhV(4))-nmhV(1)\*betan(nmhV(4));

dnmhVdt(2)=(1-nmhV(2))\*alpham(nmhV(4))-nmhV(2)\*betam(nmhV(4));

dnmhVdt(3)=(1-nmhV(3))\*alphah(nmhV(4))-nmhV(3)\*betah(nmhV(4));

dnmhVdt(4)=(-1/c)\*(gk\*(nmhV(1)^4)\*(nmhV(4)-ek)+gna\*(nmhV(2)^3)\*nmhV(3)\*(nmhV(4)-ena)+gl\*(nmhV(4)-el));

end

%5. HH solver:

figure,

[t,y]=ode45('nmhV', [0 20], [0.5 0.5 0.5 -60]);

plot(t, y(:,4), 'b');

xlabel('Time (ms)');

ylabel('Transmembrane Voltage (mV)');

title('Approaching Steady State');



figure;

ySS=y(size(y, 1),:);

ySSnew=ySS;

for i=1:10

 ySSnew(1,4)=ySS(1,4)+i;

 [t,y]=ode45('nmhV', [0 20], ySSnew);

if max(y(:,4))>0

 plot(t, y(:,4), 'r');

else

 plot(t, y(:,4), 'k');

end

hold on;

end

xlabel('Time (ms)');

ylabel('Transmembrane Voltage (mV)');

title('Threshold Behavior');



%6.a. julia sets: escape velocity

function n=escapeVelocity(z0, c, N)

%zn=zn-1^2+c

z=zeros(1, N);

z(1, 1)=z0;

n=0;

if abs((z0^2)+c)<=2

 a=2;

 nagyobb=0;

 while a<=N && nagyobb==0;

 z(1,a)=(z(1,a-1)^2)+c;

 if abs(z(1,a))>2

 nagyobb=1;

 n=a;

 end

 a=a+1;

 end

 if nagyobb==0

 n=N;

 end

end

end

%6.b. julia sets:

function M=julia(zMax, c, N)

m=500;

[Re,Im]=meshgrid(linspace(-zMax, zMax, m),linspace(-zMax, zMax, m));

Z=Re+1i\*Im;

M=zeros(m);

for a=1:m

 for b=1:m

M(a,b)=escapeVelocity(Z(a,b),c,N);

 end

end

imagesc(-zMax:zMax, -zMax:zMax, atan(0.1\*M));

xlabel('Re(Z)');

ylabel('Im(Z)');

axis xy;

end

%julia (1, -.297491+i\*0.641051, 250);



%julia (.5, -.297491+i\*0.641051, 250);

