Aufgabe 1

Schreiben Sie ein script-file, welches auffordert, zwei Matrizen A und B einzugeben. Das file soll sicherstellen, dass Zeilen- und Spaltenzahl beider Maztrizen übereinstimmen.
Andernfalls ist eine entsprechende Fehlermeldung auszugeben.
Es ist eine Matrix C zu erstellen, dessen Elemente C_{ij} berechnet werden nach der Vorschrift

$C_{ij} = 1+a_{ij}\,b_{ij}$ für $a_{ij} < b_{ij}$ und $C_{ij} = -(1/2)a_{ij}\,b_{ij}$ für $a_{ij} >= b_{ij}$

Das Ergebnis C ist mit einer Meldung (etwa: "Die Ausgabematrix C lautet:…. ") auszugeben.
Beispiel:

$$A=\begin{pmatrix}2 & 1 & 6\\ 3 & -4 & 0\\ 1 & 2 & -5\end{pmatrix} \; und\; B=\begin{pmatrix}5 & 1 & 7\\ 2 & -1 & -2\\ 0 & 3 & 8\end{pmatrix} \; ergibt\; C=\begin{pmatrix}11 & -0{,}5 & 43\\ -3 & 5 & 0\\ 0 & 7 & -39\end{pmatrix}$$

Aufgabe 2

Gegeben ist das nichtlineare Gleichungssystem

$$f1(x,y)=8\mathrm{xy}-x^2e^x=0$$

$$f2(x,y)=y+x^2y-1=0$$

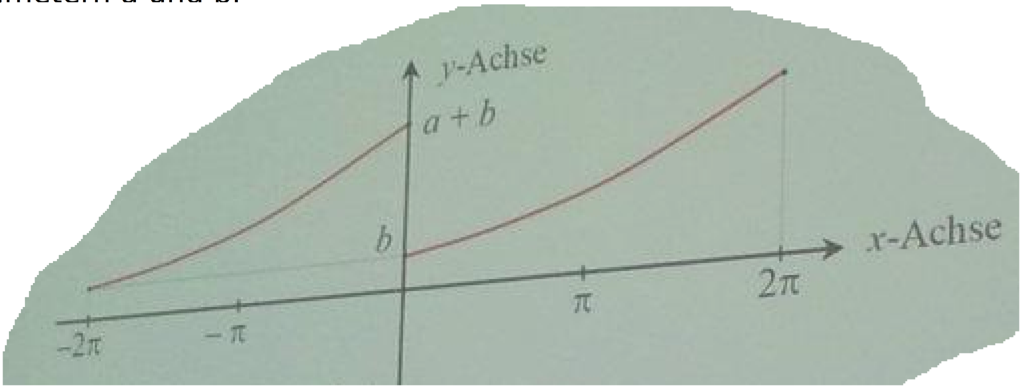
Mit Hilfe des Newton-Raphson-Verfahren soll seine einzige Lösung mit $x > 0$ gefunden werden.

a) Hierzu schreibe man ein function-file (NewtonRaphson) zur Implementierung einer 2-dimensionalen Newton-Raphson-Iteration. Dabei soll die Iterationsfolge mittels einer for-Schleife erzeugt werden.
Gewünschte Eingabegrößen:
Startwerte x_0, y_0, Maximalzahl zulässiger Iterationen (k_max), Konvergenzschranke ($\mathcal{E}$) für das Konvergenzgesteuerte Abbruchkriterium sowie die Funktionen f_1,f_2 und ihre benötigten partiellen Ableitungen.
Gewünschte Ausgabegrößen:
Die Näherungslösung x,y und die Anzahl Interationen bis zum Abbruch des Verfahrens.

b) Das function-file NewtonRaphson soll von einem script-file aufgerufen werden, welches außer f_1,f_2 und den partiellen Ableitungen auch die Werte $x_0 = 2.3$, $y_0 = 0.9$, k_max = 20, Ɛ=10^-8 zur Verfügung stellt.
Die Ergebnisse x,y und die Anzahl benötigter Iterationen sollte Ausgegeben werden (x,y mit jeweils 7 Nachkommastellen).

Aufgabe 3

Gegeben sei die 2(pi) periodische Funktion f im folgenden Bild mit **wählbaren** Parametern a und b.



$$f(x) = a \tan\left(\frac{x}{8}\right) + b, 0 < x < 2\pi$$

Die Fourierkoeffizienten von f errechnen sich für k=0,1,2,3,... bekanntlich zu

$$ak = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos(kx)\,dx = \frac{1}{\pi}\int_0^{2\pi}\left(a\tan\left(\frac{x}{8}\right)+b\right)\cos(kx)\,dx$$

$$bk = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(kx)\,dx = \frac{1}{\pi}\int_0^{2\pi}\left(a\tan\left(\frac{x}{8}\right)+b\right)\sin(kx)\,dx$$

Schreiben Sie ein function-file, welches nach Erhalt der Eingabegrößen a, b und n die Fourierapproximation f_n (in blauer Farbe) zusammen mit f selbst (in roter Farbe) in einer Graphik im Bereich von -2(pi) bis 2(pi) darstellt.
Berechnen Sie mit MATLAB die Fourierkoeffizienten a_k und b_k mit dem Trapezverfahren.
Die Grafik soll (neben den Graphen von f und f_n) folgendes enthalten:

(a) Die x- und die y-Achse mit gleichlautender Beschriftung unmittelbar an den Achsen wie im obigen Bild . Je nach Wahl von a und b soll sich die y-Achse in ihrer erforderlichen Länge sowie die Lage der Beschriftung “y-Achse“ automatisch im Sinne des obigen Bildes anpassen. Farbe: schwarz.
(b) Einen Titel mit Angabe von Signalwert *a*, Signalwert *b* sowie Ordnung *n* (Ordnung der Approximation)
(c) Ein Label für die x-Achse unterhalb des Anzeigefensters t Eintrag “x“.
(d) Ein Label für die y-Achse links des Anzeigefensters mit Eintrag “f_n(x) und f(x)“.
(e) Den Eintrag “Gibbsphänomen“ rechts des Koordinatenpunktes (0,a+b).
(f) Eine “Legende“ im ersten Quadranten, welche die Linienfarbe von f und f_n anzeigt.

Testen Sie zur Kontrolle die Ausgabe für unterschiedliche a, b sowie positive ganze Zahlen n.
Erzeugen Sie die Graphik für a=3, b=-2 und n=50 und speichern Sie diese ab.

Aufgabe 4

Gegeben ist die Zielfunktion
$f(x,y) = -\sin x + \cos y - \sin(x+y)$

Sie besitzt im Gebiet *0 < x < π, 0 < y < π* genau eine Minimalstelle. Diese soll wahlweise mit Hilfe des **Gradientenverfahrens** oder des **Hooke-Jeeves-Verfahrens** berechnet werden: ebenso soll der dazugehörige Minimalwert von f angegeben werden. Hierzu schreibe man ein script-file zur Implementierung der Steepest-Descent-Methode oder des Hooke-Jeeves-Algorithmus für den Fall n=2 Einflussgrößen und die gegebene Zielfunktion f. Die Anzahl Iterationen soll stets auf 100 begrenzt werden. Das jeweilige Abbruchkriterium soll mit $\mathcal{E}$ = 10^-8 verwendet werden. Im Fall des Gradientenverfahrens wähle man als Obergrenze des Suchintervalls für die lineare ID-Suche den festen Wert b=3. Bei Wahl des Hooke-Jeeves-Verfahrens verwende man als

Anfangsschrittweite h= 0.2 . Als Startpunkt soll bei beiden Verfahren von $X0 = \left(\frac{3}{2}, 1\right)$ ausgegangen werden. Die Anzahl benötigter Iterationen , die

Minimalstelle x_{min} und der Minimalstelle y_{min} der Zielfunktion sollen formatiert ausgegeben werden (x_{min}, y_{min} und f_{min} mit jeweils 7 Nachkommastellen)

Lösung:

Aufgabe 1

```
A=input('A eingeben: ');
[n,m]=size(A);
B=input('B eingeben: ');
[o,p]=size(B);
C=zeros(n,k);

if(n~=p) %n ungleich b
    error('Zeilenzahl von A muss gleich Spaltenzahl von B sein!');
elseif n>10
    error('Matrix ist zu groß!')
end

for i=1:n
    for j=1:p
        if A(i,j)<B(i,j)
            C(i,j)=1+A(i,j)*B(i,j);
        else
            C(i,j)=-0.5*A(i,j)*B(i,j);
        end
        %for k=1:m
        %    C(i,j)=C(i,j)+A(i,k)+B(k,j); %wird benötigt, wenn andere
Regeln oder keine gelten
        %end
    end
end
C
```

Aufgabe 2

Erst Ableitungen kreieren:
```
f1=8*y*x^2*exp(x) %Einfach im Command window so eingeben
```

```
f2=y+x^2.*y-1     %Bedenke, fals nötig, Punktoperator hinzuzufügen

syms x y
f11=diff(f1,x)    %Bedenke, wenn du die Funktionen übernimmst,
f12=diff(f1,y)    %musst du wieder @(x,y) hinzufügen

syms x y
f21=diff(f2,x)
f22=diff(f2,y)
```

mit for-Schleife:

```
function [x,y,n]=NewtonRahpson(f1,f2,f11,f12,f21,f22,epsi,k_Max,x0,y0)
x=x0;
y=y0;

for n=1:k_Max
    x_vorher=x;
    y_vorher=y;
    J=[f11(x,y) f12(x,y);f21(x,y) f22(x,y)];% Matrix bilden 2x2 von den
                                            % jeweils abgeleiteten
                                            % Funtionen
    b=[-f1(x,y);-f2(x,y)];                  % Spaltenvector bilden von den Inversen
der
                                            % jeweiligen Funktionen
    h=J\b; % Linksdivision
    x=x+h(1);
    y=y+h(2);
if abs(x-x_vorher)+abs(y-y_vorher)<=epsi
    break
end
end

%Für die Teilaufgabe a)
%fprintf('Anzahl benötigter Iterationen = %u\n',n);
%fprintf('Lösung x = %.7f\n',x);
%fprintf('Lösung y = %.7f\n',y);
```

mit while-Schleife:

```
function
[x,y,k]=NewtonRaphson_Klausur_a_while(f1,f2,f11,f12,f21,f22,epsi,k_Max,x0,y0)
k=0; % anfangswert von k setzen
x_vorher=x0+1; y_vorher=y0+1; % mindestens 1 Durchlauf garantieren
% bei while-schleife aufpassen dass ">=epsi" ist,
% wobei bei der for-schleife ist es anderes rum, sprich "<=epsi"
while (k<=k_max) && (abs(x-x_vorher)+abs(y-y_vorher)>=epsi)

    x_vorher=x; y_vorher=y;
    J=[f11(x,y), f12(x,y); f21(x,y), f22(x,y)]; % Matrix bilden 2x2 von den
                                            % jeweils abgeleiteten
                                            % Funtionen
    b=[-f1(x,y); -f2(x,y)]; % Spaltenvector bilden von den Inversen der
                            % jeweiligen Funktionen
    h=J\b; % Linksdivision

    x=x+h(1);
    y=y+h(2);

    k=k+1;
end

%Für die Teilaufgabe a)
% fprintf('\nIterationen = %u\n', k)
% fprintf('Lösung X = %.7f\n', x)
% fprintf('Lösung Y = %.7f\n', y)
```

Script-file

```
f1=@(x,y)8*x*y-x^2*exp(x);
f2=@(x,y)y+x^2.*y-1;
```

```
f11=@(x,y)8*y - x^2.*exp(x) - 2*x*exp(x);
f12=@(x,y)8*x;
f21=@(x,y)2*x*y;
f22=@(x,y)x^2 + 1;

epsi=1e-8;
k_Max=20;
x0=2.3;
y0=0.9;
[x,y,n]=NewtonRaphson(f1,f2,f11,f12,f21,f22,epsi,k_Max,x0,y0);
disp(['Startwert x0: ', num2str(x0),' y0: ', num2str(y0)]);
disp(['Lösung x: ', num2str(x,'%.7f'), ' y: ', num2str(y,'%.7f'), ' Anzahl
Iterationen n: ',int2str(n)]);
%num2str(x,'%.7f') -> zahl wird mit  Nachkommastellen ausgegeben
%man kann auswählen, ob man hier ausgibt oder im function-file
```

Aufgabe 3

```
%% Simpson-Verfahren
function ak=F_koeffizient_ak(a,b,k)
h=2*pi/1000;
%Nenner muss gerade sein!! (hier 1000)
x=0:h:2*pi; y=(1/pi)*(a*tan(x./8)+b).*cos(k*x); %Punktoperation beachten!
%Zunaechst wird S=y1+2y2+2y3+...+2ym+ym+1 berechnet:
S=2*sum(y)-y(1)-y(end);
%Zu S nun noch 2y2,2y4,2y6,...,2ym hinzuaddieren.
%Ergebnis dann: S = y1 + 4y2 + 2y3 + 4y4 + 2y5 +...+ 4ym + ym+1
for j=2:2:1000
S=S+2*y(j);
end
Integral=S*h/3;
if k==0
ak=Integral/2;
else
ak=Integral;
end

function bk=F_koeffizient_bk(a,b,k)
h=2*pi/1000;
%Nenner muss gerade sein!! (hier 1000)
x=0:h:2*pi; y=(1/pi)*(a*tan(x./8)+b).*sin(k*x); %Punktoperation beachten!
S=2*sum(y)-y(1)-y(end);
%Vorgehensweise wie bei ak
for j=2:2:1000
S=S+2*y(j);
end
bk=S*h/3;

function Fourierapproximation_Klausur(a,b,n)
x=-2*pi:0.002:2*pi; % Bereich [-2*pi, 2*pi] festlegen
fn=zeros(size(x));  % fn vorinitialisieren

% fn als Summe berechnen
for k=0:n

    fn=fn+F_koeffizient_ak(a,b,k)*cos(k*x)
+F_koeffizient_bk(a,b,k)*sin(k*x);

end
plot(x,fn,'blue')
hold on
```

```
xAchse1=[-2*pi-1, 2*pi+1]; xAchse2=[0,0];

yAchse1=[0,0];
ymax=max(abs(b), abs(a+2*b));
yAchse2=[-ymax,ymax];

x=-2*pi:0.002:0; plot(x, a*tan((x+2*pi)./8)+b, 'red-')
x=0:0.002:2*pi; plot(x, a*tan(x./8)+b, 'red')

plot (xAchse1, xAchse2, 'black:')
plot (yAchse1, yAchse2, 'black:')

title(['Fourierapproximation eines tangensialen Signals der Ordnung ',...
    num2str(n),' mit Öffnungsbreite ',num2str(a), ...
    ' und y-Achsen-Abschnitt ',num2str(b)])
xlabel('X')
ylabel('f(x) und fn(x)')
text(2*pi+1.2,0,'x-Achse')
text(b/15, a+1.85*b, 'y-Achse')
text(b/15, a+b, 'Gibbsphaenomen')

text(-2*pi,-b/4,'-2Pi')
text(-pi,-b/4,'-Pi')
text(pi,-b/4,'Pi')
text(2*pi,-b/4,'2Pi')

legend('fn(x)','f(x)',1), legend('boxon')

hold off
```

Aufgabe 4

```
function [x_min,f_min,k]=Aufgabe4_HJV(n,xo,h,Eps,k_max,f)
E=eye(n);
for k=1:k_max
    %--------------------------------------------
    hold on;
    plot(xo(1),xo(2),'black.');
    if k<=5 %k<=8 beim zweiten Beispiel
        text(xo(1)+0.05,xo(2)+0.05,int2str(k-1));
    end
    hold off;
    %--------------------------------------------
    z=xo;fo=f(z);
    for j=1:n
        zm=z-h*E(j,:);fm=f(zm);
        zp=z+h*E(j,:);fp=f(zp);
        if fm<=min(fo,fp)
           z=zm;fo=fm;
        end
        if fp<=min(fo,fm)
           z=zp;fo=fp;
        end
    end
    if z==xo
        h=h/2;
        if h<=Eps
            break
        end
    else
        x=z;z=x+(x-xo);fz=f(z);
        if fz<fo
```

```
            xo=z;fo=fz;
        else
            xo=x;
        end
    end
end
x_min=xo;f_min=fo;


%Hauptscript-file, welches auf das function-file Hooke_Jeeves_Verfahren
%zugreift.
[x,y]=meshgrid(-3:0.01:3,-3:0.01:3);
z=-sin(x)+cos(y)-sin(x+y);
C=contour(x,y,z,20);
clabel(C,'manual')
xo=[3/2, 1];h=0.2;k_max=100;Eps=1e-8;
f=@(x)-sin(x(1))+cos(x(2))-sin(x(1)+x(2));

[x_min,f_min,k]=Aufgabe4_HJV(2,xo,h,Eps,k_max,f);

fprintf('Das Minimum liegt bei x_min = (%.4f,%.4f)\n',x_min);
fprintf('Der Minimalwert von f beträgt f_min = %.4f\n',f_min);
fprintf('Es wurden %u Iterationen benötigt',k);

%% Aufgabe 4 Hooke-Jeeves-Verfahren

[x,y]=meshgrid(-4:0.1:4,-4:0.1:4);
z=-sin(x)+cos(y)-sin(x+y);
C=contour(x,y,z,20);
clabel(C,'manual')

xo=[3/2, 1];h=0.2;k_max=100;Eps=1e-8;
f=@(x)-sin(x(1))+cos(x(2))-sin(x(1)+x(2));
n=2;

E=eye(n);

for k=1:k_max
    %---------------------------------------------
    hold on;
    plot(xo(1),xo(2),'black.');
    if k<=5 %k<=8 beim zweiten Beispiel
        text(xo(1)+0.05,xo(2)+0.05,int2str(k-1));
    end
    hold off;
    %---------------------------------------------
    z=xo;fo=f(z);
    for j=1:n
        zm=z-h*E(j,:);fm=f(zm);
        zp=z+h*E(j,:);fp=f(zp);
        if fm<=min(fo,fp)
           z=zm;fo=fm;
        end
        if fp<=min(fo,fm)
           z=zp;fo=fp;
        end
    end
    if z==xo
        h=h/2;
        if h<=Eps
            break
        end
```

```
    else
        x=z;z=x+(x-xo);fz=f(z);
        if fz<fo
            xo=z;fo=fz;
        else
            xo=x;
        end
    end
end
x_min=xo;f_min=fo;

fprintf('Das Minimum liegt bei x_min = (%.7f,%.7f)\n',x_min);
fprintf('Der Minimalwert von f beträgt f_min = %.7f\n',f_min);
fprintf('Es wurden %u Iterationen benötigt\n',k);

%% Aufgabe 4 Hooke-Jeeves-Verfahren

[x,y]=meshgrid(-4:0.1:4,-4:0.1:4);
z=-sin(x)+cos(y)-sin(x+y);
C=contour(x,y,z,20);
clabel(C,'manual')

xo=[3/2, 1];h=0.2;k_max=100;Eps=1e-8;
f=@(x)-sin(x(1))+cos(x(2))-sin(x(1)+x(2));
n=2;

E=eye(n);

k=0;

while (k<=k_max) && (h>=Eps)
    %--------------------------------------------
    hold on;
    plot(xo(1),xo(2),'black.');
    if k<=5 %k<=8 beim zweiten Beispiel
        text(xo(1)+0.05,xo(2)+0.05,int2str(k-1));
    end
    hold off;
    %--------------------------------------------
    z=xo;fo=f(z);
    for j=1:n
        zm=z-h*E(j,:);fm=f(zm);
        zp=z+h*E(j,:);fp=f(zp);
        if fm<=min(fo,fp)
           z=zm;fo=fm;
        end
        if fp<=min(fo,fm)
           z=zp;fo=fp;
        end
    end
    if z==xo
        h=h/2;
    else
        x=z;z=x+(x-xo);fz=f(z);
        if fz<fo
            xo=z;fo=fz;
        else
            xo=x;
        end
    end

    k=k+1;
```

```
end
x_min=xo;f_min=fo;

fprintf('Das Minimum liegt bei x_min = (%.7f,%.7f)\n',x_min);
fprintf('Der Minimalwert von f beträgt f_min = %.7f\n',f_min);
fprintf('Es wurden %u Iterationen benötigt\n',k);

[x,y]=meshgrid(-4:0.1:7,-4:0.1:4);
z=-sin(x)+cos(y)-sin(x+y);

subplot(1,2,2)
surf(x,y,z);

subplot(1,2,1)
C=contour(x,y,z,20);
clabel(C,'manual')

xo=[3/2, 1];h=0.2;k_max=100;Eps=1e-8;
f=@(x)-sin(x(1))+cos(x(2))-sin(x(1)+x(2));
n=2;

E=eye(n);
for k=1:k_max
    %--------------------------------------------
    hold on;
    plot(xo(1),xo(2),'black.');
    if k<=5 %k<=8 beim zweiten Beispiel
        text(xo(1)+0.05,xo(2)+0.05,int2str(k-1));
    end
    hold off;
    %--------------------------------------------
    z=xo;fo=f(z);
    for j=1:n
        zm=z-h*E(j,:);fm=f(zm);
        zp=z+h*E(j,:);fp=f(zp);
        if fm<=min(fo,fp)
           z=zm;fo=fm;
        end
        if fp<=min(fo,fm)
           z=zp;fo=fp;
        end
    end
    if z==xo
        h=h/2;
        if h<=Eps
            break
        end
    else
        x=z;z=x+(x-xo);fz=f(z);
        if fz<fo
            xo=z;fo=fz;
        else
            xo=x;
        end
    end
end
x_min=xo;f_min=fo;

fprintf('Das Minimum liegt bei x_min = (%.4f,%.4f)\n',x_min);
fprintf('Der Minimalwert von f beträgt f_min = %.4f\n',f_min);
fprintf('Es wurden %u Iterationen benötigt\n',k);
```