Musterlösung

Aufgabe 1 und 2: [Newton-Verfahren]

```matlab
function NewtonNd
% NEWTONND Newton-Verfahren

% Aufgaben 1(a) und 1(b)
% ================

% Definition der Parameter
l1 = 8;
l2 = 10;
gamma = 3*pi/5;

% Definition der Funktion und ihrer Ableitung
f = @(alpha) (l2*cos(pi-gamma-alpha)./sin(pi-gamma-alpha).^2 - ...
    l1*cos(alpha)./sin(alpha).^2);

df = @(alpha) (l2*(sin(pi-gamma-alpha).^2+2*cos(pi-gamma-alpha).^2)./...
    sin(pi-gamma-alpha).^3 + ...
    l1*(sin(alpha).^2+2*cos(alpha).^2)./sin(alpha).^3);

% Startwert, Toleranz und maximale Anzahl an Iterationen
x0 = 0.1;
tol = 10^-10;
kmax = 100;

% Newton-Verfahren ausführen
angle = newton(f,df,x0,tol,kmax);

% Ergebnisse ausgeben
fprintf(1,'Die maximale Länge des Balkens beträgt: %03.1f
bei einem Wert des Winkels alpha von %2.1f Grad.
',...
    l2/sin(pi-gamma-angle(end))+l1/sin(angle(end)),angle(end)*180/pi);

figure, plotFL(angle(end),gamma,l1,l2);

% Aufgabe 1(c)
% ===========

% Definition der Parameter
l1 = 1;
l2 = l1;
gamma = pi/2;

% Definition der Funktion und ihrer Ableitung
f = @(alpha) (l2*cos(pi-gamma-alpha)./sin(pi-gamma-alpha).^2 - ...
11*cos(alpha)./sin(alpha).^2);
df = @(alpha) (l2*(sin(pi-gamma-alpha).^2+2*cos(pi-gamma-alpha).^2)./...
sin(pi-gamma-alpha).^3 + ...
11*(sin(alpha).^2+2*cos(alpha).^2)./sin(alpha).^3);

% Startwert, Toleranz und maximale Anzahl an Iterationen
x0 = 10^-7;
tol = 10^-10;
kmax = 100;

% Newton-Verfahren ausführen
angle = newton(f,df,x0,tol,kmax);

% exakte Lösung
aex = pi/4;

% Fehler berechnen
err = abs(angle-aex);

% Ordnung des Verfahrens bestimmen
p = log(err(2:end-1)./err(3:end))./log(err(1:end-2)./err(2:end-1));

% Ergebnisse ausgeben
fprintf(1,\nOrdnung:\n);
disp(num2str(transpose(p,'%1.1f\n')));

% Aufgabe 2
% =========

% Definition der Parameter
l1 = 2;
l2 = 1;

% Zielpunkt
P = [1.5; 2];

% Definition der Funktion und ihrer Ableitung
f = @(ab) [l1*cos(ab(1)) + l2*cos(ab(1)+ab(2)-pi) - P(1) ;
  l1*sin(ab(1)) + l2*sin(ab(1)+ab(2)-pi) - P(2)];
df = @(ab) [-l1*sin(ab(1))-l2*sin(ab(1)+ab(2)-pi),-l2*sin(ab(1)+ab(2)-pi),
  l1*cos(ab(1))+12*cos(ab(1)+ab(2)-pi), 12*cos(ab(1)+ab(2)-pi)];
x0v = [pi/2 0.5; pi/2 4];
tol = 10^-10;
kmax = 100;

figure
for i = 1:size(x0v,2)
    % Newton-Verfahren ausführen
    angle = newton(f,df,x0v(:,i),tol,kmax);
    fprintf(1,
        'Der Wert des Winkels alpha beträgt: %2.1f Grad,
        der Wert des Winkels beta beträgt: %2.1f Grad.
    ',angle(1,end)*180/pi,angle(2,end)*180/pi);
    plotRA(angle(1,end),angle(2,end),l1,l2,P);
end
quit;
end % of function NewtonNd

function [] = plotFL(alpha,gamma,l1,l2)
% PLOTFL Flur mit eingepasstem Balken plotten
% PLOTFL(alpha,gamma,l1,l2) plottet den Balken mit der durch den
% Winkel alpha gegebenen Länge in dem durch die Parameter l1, l2
% und gamma definierten Flurabschnitt
%
% INPUT : alpha - Winkel zwischen Balken und Flurwand
%         gamma - Winkel zwischen beiden Flurteilen
%         l1   - Breite des rechten Flurteils
%         l2   - Breite des linken Flurteils
%
% Kontrolle auf Gültigkeit der Parameter
if ~rem(gamma,pi)
    error('Ungültiger Wert für gamma!');
end
%
% Negative Winkel korrigieren
a = 0.5*pi*(alpha<0) + rem(alpha,0.5*pi);
g =  pi*(gamma<0) + rem(gamma,pi);
%
% Eckpunkte des Flurs berechnen
e1 = [cos(a) ; sin(a) ];
e2 = [cos(2*pi-g); sin(2*pi-g)];
P1 = l1/sin(a)   * e1;
P2 = l2/sin(pi-g-a)*(-e1);
P3 = -1/sin(pi-g) * [e2 eye(2,1)]*[l1; l2];
\[ c = \max\left(\text{ceil}\left(\frac{P2(2)}{(e2(2) \cdot \min(l1, l2))}\right), \text{ceil}\left(\frac{P1(1)}{\min(l1, l2)}\right)\right); \]
\[ P4 = 12 \cdot \left[ c \cdot e2 - \frac{1}{\sin(\pi - g)} \cdot \text{eye}(2, 1) \right] \cdot \text{ones}(2, 1); \]

\% Funktionswerte für die Darstellung der Winkel
\[ PA = 0.5 \cdot \min(l1, l2) \cdot \left[ \cos(0:0.01:a) ; \sin(0:0.01:a) \right]; \]
\[ PG = 0.7 \cdot \min(l1, l2) \cdot \left[ \cos(2\pi:-0.01:2\pi-g) ; \sin(2\pi:-0.01:2\pi-g) \right]; \]

\% neues Plot-Fenster anlegen
hold on;
\% Balken einzeichnen
plot([P2(1) P1(1)], [P2(2) P1(2)], 'r');
\% unteren Flurteil einzeichnen
plot(c*[l2*e2(1) 0 l1], c*[l2*e2(2) 0 0]);
\% oberen Flurteil einzeichnen
plot([P4(1) P3(1) c*l1], [P4(2) l1 l1]);
\% Winkel einzeichnen und beschriften
plot(PA(1,:), PA(2,:), 'k');
text(0.25*min(l1,l2)*cos(a/2),0.25*min(l1,l2)*sin(a/2), '\alpha');
plot(PG(1,:), PG(2,:), 'k');
text(0.25*min(l1,l2)*cos(2*pi-g/2),0.25*min(l1,l2)*sin(2*pi-g/2), '\gamma');
\% Titel festlegen
title('Eingepasster Balken');
\% sichtbaren Bereich anpassen
xlim([floor(min(P3(1),P4(1)))-1 c*l1+1]);
ylim([floor(P4(2))-1 l1+1]);
end \% of function plotFL

function [] = plotRA(alpha,beta,l1,l2,P)
\% PLOTRA Roboterarm plotten
\% PLOTRA(alpha,beta,l1,l2,P) plottet den durch die Winkel alpha und
\% beta und Gliederlängen l1 und l2 definierten Roboterarm mit
\% Zielpunkt P
\%
\% INPUT :  alpha - Winkel des ersten Gliedes
\% beta - Winkel des zweiten Gliedes
\% l1 - Länge des ersten Gliedes
\% l2 - Länge des zweiten Gliedes
\% P - Zielpunkt des Robotersarms in \( R^2 \)
\%
\% Kontrolle auf Gültigkeit des Zielpunktes
if l1+l2 < norm(P,2) \| norm(P,2) < abs(l2-11)
    error('NewtonNd:plotRA', 'Ungültiger Zielpunkt');
end

\% Negative Winkel korrighieren
a = 2*pi*(alpha<0) + rem(alpha,2*pi);
b = 2*pi*(beta <0) + rem(beta ,2*pi);
% Positionen der Gelenke berechnen
P1 = l1*[cos(a) ; sin(a)];
P2 = P1 + l2*[cos(a+b-pi); sin(a+b-pi)];

% Funktionswerte für die Darstellung der Winkel
c1 = 0.25+rand(1)/7; c2 = 0.25+rand(1)/7; c3 = 3/7;
PA = c1*l1*[cos(0:0.01:a) ; sin(0:0.01:a)];
PB = c2*l2*[cos(a-pi:0.01:a+b-pi); sin(a-pi:0.01:a+b-pi)];

% neues Plot-Fenster anlegen
hold on;
% Roboterarm einzeichnen
plot([0 P1(1) P2(1)],[0 P1(2) P2(2)],'r');
% Gelenke einzeichnen
plot([0 P1(1)],[0 P1(2)],'o');
% Zielpunkt einzeichnen und beschriften
plot(P1,P2,'*'); text(1.025*P1,1.025*P2,'P');
% Winkel einzeichnen und beschriften
plot([0 c1*l1],zeros(1,2),'k'); plot(PA(1,:),PA(2,:),'k');
text(c1*c3*l1*cos(a/2),c1*c3*l1*sin(a/2),'\alpha');
plot(P1+PB(1,:),P1(2)+PB(2,:),'k');
text(P1(1)+c2*c3*l2*cos(a-pi+b/2),P1(2)+c2*c3*l2*sin(a-pi+b/2),'\beta');
% Titel festlegen
title('Roboterarm');
% sichtbaren Bereich anpassen
xl = min([0 PA(1,:) P1(1)+PB(1,:) P2(1)]); 
xr = max([0.25*l1 P1(1)+PB(1,:) P2(1)]);
yl = min([0 P1(2) P1(2)+PB(2,:) P2(2)]);
yu = max([P1(2) P2(2) PA(2,:) P1(2)+PB(2,:)]);
xlim([xl-0.1 xr+0.1]); ylim([yl-0.1 yu+0.1]);
end % of function plotRA
function x = newton(f, df, x0, tol, kmax)

% NEWTON Newton-Verfahren
% NEWTON(f, df, x0, tol, kmax) berechnet ausgehend von x0 mit maximal kmax
% Iterationen eine Näherung der Nullstelle der durch f definierten Funktion
%
% INPUT: f - reellwertige Funktion mit f : R ^ {m} -> R ^ {m}
% df - zugehörige Ableitung
% x0 - Startwert in R ^ {m}
% tol - Abbruchtoleranz
% kmax - maximale Anzahl an Iterationen
%
% OUTPUT: x - reellwertige Matrix in (maximal) R ^ {m x kmax+1},
% die sämtliche Iterationsschritte des Newton-Verfahrens enthält

% Startwerte initialisieren
k = 1;
if k > kmax
    warning('NewtonNd:newton', 'Maximale Anzahl Iterationen erreicht');
    return;
end

% Wert der Ableitung bestimmen
dfv = df(x(:,k));
if abs(det(dfv)) < tol
    warning('NewtonNd:newton', 'Ableitung zu klein');
    return;
end

% Newton-Schritt
x(:,k+1) = x(:,k) - dfv\f(x(:,k));
k = k + 1;

% Rückgabe sämtlicher durchgeführter Iterationsschritte
x = x(:,1:k);
end % of function newton