```
function y = FittingCountingDataMatlab()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Beispiel: Histogramme von Zählstatistiken mit Matlab fitten
%
% This Matlab-program was developed using "Matlab_R2014b" on a MacBook Pro
% running OS X 10.9.5
%
% written by T. Ihn, D-PHYS ETH Zurich, 7 Nov 2014
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% Daten aus einem csv-File einlesen
%
data = csvread('histdata.csv');
xi = data(:,1);
ci = data(:,2);

%
% Daten plotten
%

figure(1)
set(gca,...
    'FontName','Times New Roman',...
    'FontSize',24,...
    'XTick',0:0.25:1,...
    'Ytick',0:3,...
    'Color',[0.8 0.8 0.8],...
    'LineWidth',2,...
    'TickLength',[0.02 0.0250],...
    'XLim',[0 1],...
    'YLim',[0 3],...
    'Units','inches',...
    'Position',[1 0.8 5.77 3.57],...
    'Layer','Top')
bar(xi,ci,1);
axis([-11 11 0 60])
xlabel('x');
ylabel('counts');

% Funktion zur Berechnung des Erwartungswertes in einem Bin, wenn A und B
% gegeben sind
    function y = mu(A,B,x)
       y = (A*exp(-x.*x) + B)/2;
    end

% Funktion zur Berechnung der negativen log-Likelihood Funktion
    function y = L(parms,xi,ci)
       mui = mu(parms(1),parms(2),xi);
       y = -sum(ci.*log(mui) - mui);
    end

% Minimierung der negativen log-Likelihood Funktion

[parms,Lmin]=fminsearch(@(parms) L(parms,xi,ci),[100 20]);
```

```
A0 = parms(1);
B0 = parms(2);

% Plotte die negative log-Likelihood Funktion
AA = 75:35/51:110;
BB = 18:4.2/51:22.2;
LL = zeros(length(AA),length(BB));
[X,Y] = meshgrid(AA,BB);
for n=1:length(AA),
    for m=1:length(BB),
        LL(m,n) = 2*(L([AA(n) BB(m)],xi,ci) - Lmin);
    end;
end;

figure(2)
%set(gca,...
%    'FontName','Times New Roman',...
%    'FontSize',24,...
%    'XTick',0:0.25:1,...
%    'Ytick',0:3,...
%    'Color',[0.8 0.8 0.8],...
%    'LineWidth',2,...
%    'TickLength',[0.02 0.0250],...
%    'XLim',[0 1],...
%    'YLim',[0 3],...
%    'Units','inches',...
%    'Position',[1 0.8 5.77 3.57],...
%    'Layer','Top');
pcolor(X,Y,LL)
shading interp
hold on
xlabel('A')
ylabel('B')
title('-log Likelihood')
colorbar
[C,h] = contour(X,Y,LL,[1,2,3,0.25],'Linewidth',2,'LineColor','w');
clabel(C,h)
plot(A0,B0,'wo','MarkerFaceColor','w')
hold off

% We determine the standard error of the parameters from the contour lines
idx=find(C(1,:)==1);
CC = C(:,(idx+1):(idx+C(2,idx))); % This are the coordinates of the relevant contour line
sAp = max(CC(1,:))-A0; % the maximum in A-direction
sAm = A0-min(CC(1,:));
sBp = max(CC(2,:))-B0; % the maximum in B-direction
sBm = B0-min(CC(2,:));

fprintf('\nFit results:\n');
fprintf('A = %1.1f +%1.1f/-%1.1f\n',A0,sAp,sAm);
fprintf('B = %1.1f +%1.1f/-%1.1f\n',B0,sBp,sBm);

% Plot the results of the fit
figure(1)
hold on
x = -10.5:0.1:10.5;
plot(x,mu(A0,B0,x),'r','LineWidth',2);
hold off
```

```
% Determine the elements of the Hesse-Matrix numerically
h=1;
k=0.1;
d2LdA2 = (L([A0+h B0],xi,ci) - 2*L([A0 B0],xi,ci) + L([A0-h B0],xi,ci))/h/h;
d2LdB2 = (L([A0 B0+k],xi,ci) - 2*L([A0 B0],xi,ci) + L([A0 B0-k],xi,ci))/k/k;
d2LdAdB = (L([A0+h B0+k],xi,ci) - L([A0+h B0-k],xi,ci) - L([A0-h B0+k],xi,ci) + L([A0-h↙
B0-k],xi,ci))/4/h/k;
H = [d2LdA2 d2LdAdB; d2LdAdB d2LdB2];

% invert the Hesse matrix
cov = inv(H);
sigmaA = sqrt(cov(1,1));
sigmaB = sqrt(cov(2,2));
rho = cov(1,2)/sigmaA/sigmaB;
fprintf('\nFit results with errors from inverted Hesse-Matrix:\n');
fprintf('A = %1.1f +/-%1.1f\n',A0,sigmaA);
fprintf('B = %1.1f +/-%1.1f\n',B0,sigmaB);
fprintf('rho = %1.2f\n',rho);

end
```