```
mureps = mean(reps);
mureps2 = mean(reps2);
mureps3 = mean(reps3);
mureps4 = mean(reps4);
%*************************************************************************
% get the estimate of the standard error

sehat1 = sqrt((n-1)/n *sum((reps - mureps). ^ 2));
sehat2 = sqrt((n-1)/n *sum((reps2 - mureps2). ^ 2));
sehat3 = sqrt((n-1)/n *sum((reps3 - mureps3). ^ 2));
sehat4 = sqrt((n-1)/n *sum((reps4 - mureps4). ^ 2));
%*************************************************************************

sehatJack = [sehat1 sehat2 sehat3 sehat4];
return
```

Algorithm 4: Probability-probability plots

The Matlab code below plots the Probability-probability plots of the convolved CDF given the estimated parameter vector $[L, U, a, \text{sigma}]$ obtained by algorithm 2 and dataset.

The Matlab code of Algorithm 4

```
function ppplots(paramsEst,data)

% ppplots(paramsEst,data) returns the Probability-probability plots

z = sort(data);
n = length(z);

L = paramsEst(1);
U = paramsEst(2);
a = paramsEst(3);
```

```
sigma = paramsEst(4);
%*************************************************************************

for i = 1 : n
intepart(i) = quad(@CDFfuntrail,L,U,[],[],z(i),a,sigma);
constant = a/(L^(-a)- U^(-a));
emp(i) = (i./n);
end
%*************************************************************************
k = 0:0.5:1;
G2 = emp';
G1 = (constant.*intepart)';
plot(G2,G1,'+',k,k,'r','LineWidth',2)
legend('probabilities','straight line');
xlabel('Empirical CDFs')
ylabel('Theoretical CDFs')
return
%*************************************************************************

function y = CDFfuntrail(P,z,a,sigma)

% integral part of the CDF function
%*************************************************************************
N = length(P);
for j = 1:N;
x = P(j);
k = (z - x)./(sqrt(2).*sigma);
CDFnorm = 0.5.*(1 + erf(k));
v = (x.^(-a-1)).*CDFnorm;
u(j) = x;
y(j) = v;
end
return
```

Algorithm 5: convolved CDF graph and K-S test statistic

The Matlab code below calculates the K-S test statistic and plots the graph of the convolved CDF, given the estimated parameter vector $[L, U, a, \text{sigma}]$ obtained by algorithm 2 and dataset.

The Matlab code of Algorithm 5

```
function [F1] = CDFParetoNorm(paramsEst,data);
%************************************************************************
z = sort(data(:));
n = length(z);
%************************************************************************
% estimates
L = paramsEst(1);
U = paramsEst(2);
a = paramsEst(3);
sigma = paramsEst(4);
%************************************************************************
% Tolerance parameter for quadrature algorithm (quad or lobbato-quadl)
tolr = 1.e-6;
%************************************************************************
% for loop
for i = 1 : n
D(i) =(quad(@CDFfuntrail,L,U,tolr,[],z(i),a,sigma));
jj = find(z <= z(i));
empir2(i) = length (jj)./n;
end
%************************************************************************
constant = a/(L^(-a)- U^(-a));
G = (constant.*D);
%************************************************************************
F = G';
emp2 = empir2';
%************************************************************************
plot(z,F,'b-',z,emp2,'-r','LineWidth',2)
```

```
legend('Theoretical CDF','Empirical CDF');
xlabel('observe random variables')
ylabel('Probabilities')
%**************************************************************************
F1 = [F emp2];
kol = max(abs(emp2 - F))        % kolmogorov-smirnov value
return
```

Algorithm 6: Bootstrapping critical values of the K-S test statistic

The Matlab code below returns the critical values of the K-S test statistic given the estimated parameter vector $[L, U, a, \text{sigma}]$ obtained by algorithm 2, dataset and number of bootstrap replicates, nboot.

The Matlab code of Algorithm 6


```
function [KSvaleus paramsEst1] = KSbootscriticalV(data,nboot,paramsEst)

z = sort(data)'; % input data
n = length(z);
%**************************************************************************
for i=1:nboot,
a = ceil(n*rand(n,1));
zt = z(a);

paramsEst0 = Maxlikelihood(zt,paramsEst);
ztt = CDFParetoNorm(paramsEst0,zt);

D = max(abs(ztt(:,2) - ztt(:,1)));
Dx = abs(ztt(:,2) - ztt(:,1));
jj = find(Dx == D);
xx = length(jj);
%**************************************************************************

parL(i) = paramsEst0(:,1);
```

```
parU(i) = paramsEst0(:,2);
para(i) = paramsEst0(:,3);
pars(i) = paramsEst0(:,4);
end;
%*************************************************************************

KSvaleus = sort(mx11)';
paramsEstL = sort(parL)';
paramsEstU = sort(parU)';
paramsEsta = sort(para)';
paramsEsts = sort(pars)';
paramsEst1 = [paramsEstL paramsEstU paramsEsta paramsEsts];
return
```