Project 997 PRNGs Part 7 LCGM Cubed Algorithms By Namir C. Shammas

1/ Introduction

This study looks at two versions of LCGM that use cubic expressions of the previous random numbers to calculate new random numbers.

2/ LCGM Cubed Algorithm (Version PSO)

The ascending-power LCGM Cubed method algorithm is defined as:

```
ix(1) = round(rand*seed,0);
ix(2) = a0+a1*ix(1) mod M
ix(3) = a0+a1*ix(1)+a2*ix(2)^2 mod M
for i=1 to maximum number of random numbers
  ix(4) = a0+a1*ix(1)+a2*ix(2)^2+a3*ix(3)^3 mod M
  x(i) = ix(4)/M
  ix(1:3) = ix(2:4)
end                                                    (2.1)
```

This section looks at optimizing the values of a0, a1, a2, and a3 used in equation 2.1. The array x() is the sought array of uniform random values in the range (0, 1]. The new random number x(i) obtains its value from the previous random integer values ix(1), ix(2) and ix(3). The random number generating loops keeps the newer three values of the integer array ix(). M is the modulus value.

The approache that estimates the best coefficients for the power method uses the following approach:

1. Optimize the penalty factor (see Appendix of Project 997 PRNGs Part 1) using particle swarm optimization algorithms. This optimization starts with a wide trust region and narrows it down.

2. Optimize the penalty using the narrowed optimum regions obtained in step 1. This step yields refined trust regions.
3. Perform a large run of generating random numbers using the refined trust regions from step 2. The calculations yield the statistics for the penalty factor. The upper confidence values for the mean penlty factor are the values we look at to determine the fitness of the algorithm.

The listing for do.m, which triggers the calculations for the first and second optimization phases is:

```
maxElems=10000;
maxIters=100;
lb=[100 11 11 11 11];
ub=[100000 1000000 1000000 1000000 1000000];
sFilename='res1.csv';
sSheetName='res1Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM=2^64-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
lb=[100 11 11 11 11];
ub=[100000 1000000 1000000 1000000 1000000];
sFilename='res2.csv';
sSheetName='res2Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM=2^54-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
lb=[100 11 11 11 11];
ub=[100000 1000000 1000000 1000000 1000000];
sFilename='res3.csv';
sSheetName='res3Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM=2^48-1;
bRound=true;
nDigits = 10;
```

```
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
lb=[100 11 11 11 11];
ub=[100000 1000000 1000000 1000000 1000000];
sFilename='res4.csv';
sSheetName='res4Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM=2^40-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
lb=[100 11 11 11 11];
ub=[100000 1000000 1000000 1000000 1000000];
sFilename='res5.csv';
sSheetName='res5Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM=2^32-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
lb=[100 11 11 11 11];
ub=[100000 1000000 1000000 1000000 1000000];
sFilename='res6.csv';
sSheetName='res6Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM1=2^24-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
lb=[100 11 11 11 11];
ub=[100000 1000000 1000000 1000000 1000000];
sFilename='res7.csv';
sSheetName='res7Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
```

```
xM1=2^16-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

% --------------------------------------------------------------------------

r=0.15;
rp = 1+r;
rm = 1 - r;
maxElems=10000;
maxIters=100;
c = [25464,754568,164418,284470,11];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res1b.csv';
sSheetName='res1Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM=2^64-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
c = [87326,610443,314599,11,223305];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res2b.csv';
sSheetName='res2Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM=2^54-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
c = [100000,639687,143059,78456,849312];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res3b.csv';
sSheetName='res3Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM=2^48-1;
bRound=true;
nDigits = 10;
```

```
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
c = [100000,11,332003,292469,567415];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res4b.csv';
sSheetName='res4Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM=2^40-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
c = [100000,598206,177932,86258,11];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res5b.csv';
sSheetName='res5Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM=2^32-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
c = [27845,126525,625719,383934,705626];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res6b.csv';
sSheetName='res6Sheet1.csv';
POSpopsize=250;
POSmaxIters=350;
xM1=2^24-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=100;
c = [45944,728307,244905,447851,254242];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res7b.csv';
sSheetName='res7Sheet1.csv';
```

```
POSpopsize=250;
POSmaxIters=350;
xM1=2^16-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

system('shutdown /s')
```

Listing 2.1. The listing of file do.m.

Listing 2.2 shows the source code for file doAll.m. The function doAll() optimizes the function rng997Gen1() using the Particle Swarm Optimization (PSO) method by calling function particleswarm(). The function do() calls function doAll() for the first and second optimization phases.

```
function
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)
global gmaxElems
global bestFactor
global M
global Xrnd
global doRounding
global numDigits

gmaxElems = maxElems;
M = xM;
doRounding = bRound;
numDigits = nDigits;
options =
optimoptions("particleswarm","SwarmSize",POSpopsize,"Display","off","MaxItera
tions",POSmaxIters,"FunctionTolerance",0.01);
resMat=zeros(maxIters,7);

for i=1:maxIters
  bestFactor = 1e+99;
%   xrnd = round(i/(maxIters+1),n);
  x = particleswarm(@rng997Gen1,length(lb),lb,ub,options);
  resMat(i,1) = bestFactor;
  resMat(i,2) = round(x(1),0);
  resMat(i,3) = round(x(2),0);
  resMat(i,4) = round(x(3),0);
  resMat(i,5) = round(x(4),0);
  resMat(i,6) = round(x(5),0);
  resMat(i,7) = Xrnd;

  fprintf("Itr: %d, Factur=%f, [", i, bestFactor);
  fprintf(" %d,", resMat(i,2:6));
  fprintf("%d]\n", Xrnd);
```

```
end

fprintf("\n\n");
resMat = sortrows(resMat,1);

beep on;
for i=1:3
  beep;
  pause(1);
end

fprintf("Entire result matrix written to file %s\n", sFilename)
fh = fopen(sFilename, "w");
fprintf(fh,"Factor,IX,A0,A1,A2,A3,Xrnd\n");
for i=1:maxIters
  fprintf(fh,"%f,", resMat(i,1));
  for j=2:6
      fprintf(fh,"%d,", resMat(i,j));
  end
  fprintf(fh,"%d\n", resMat(i,7));
end
fclose(fh);
% T1 = array2table(resMat);
% T1.Properties.VariableNames(1:7) = {"Factor" "IX," "A0,","A1", "A2" , "A3",
"Xrnd");
% writetable(T1, sFilename, "Sheet,%d\n" sSheetName);
% pause(10);
% c = cell(17,2);

fh = fopen(sSheetName, "w");
fprintf("c1,c2\n");
fprintf(fh,"Max Elements,%d\n", maxElems);
fprintf(fh,"Max Iters,%d\n", maxIters);
fprintf(fh,"Xrndlow,%d\n", lb(1));
fprintf(fh,"Xrndhi,%d\n", ub(1));
fprintf(fh,"A01low,%d\n", lb(2));
fprintf(fh,"A0hi,%d\n", ub(2));
fprintf(fh,"A1low,%d\n", lb(3));
fprintf(fh,"A1hi,%d\n", ub(3));
fprintf(fh,"A2low,%d\n", lb(4));
fprintf(fh,"A2hi,%d\n", ub(4));
fprintf(fh,"A3low,%d\n", lb(5));
fprintf(fh,"A3hi,%d\n", ub(5));
fprintf(fh,"PopSize,%d\n", POSpopsize);
fprintf(fh,"PopMaxIters,%d\n", POSmaxIters);
fprintf(fh,"M,%d\n", M);
if doRounding
  fprintf(fh,"Rounded?,1\n");
  fprintf(fh,"Rounded,%d\n", nDigits);
else
  fprintf(fh,"Rounded?,0\n");
  fprintf(fh,"Rounded,N/A\n");
end
```

```
fclose(fh);
% T2 = cell2table(c);
% writetable(T2, sFilename, "Sheet,%d\n" "Params");
fprintf("-----------------------------------------------------------\n\n");
end
```

Listing 2.2. The listing of file doAll.m.

Listing 2.3 shows the listing of Reg997Gen1.m.

```
function factor = rng997Gen1(c)
%UNTITLED2 Summary of this function goes here
  global gmaxElems
  global Xrnd
  global M
  global doRounding
  global numDigits

  maxElems = gmaxElems;
  c = round(c, 0);
  ix = zeros(4,1);
  ix(1) = round(rand*c(1),0);
  Xrnd = ix(1);

  a0 = c(2);
  a1 = c(3);
  a2 = c(4);
  a3 = c(5);
  ix(2) = mod(a0+a1*ix(1),M);
  ix(3) = mod(a0+a1*ix(1)+a2*ix(2)^2,M);
  x=zeros(maxElems,1);
  for i=1:maxElems
    ix(4) = mod(a0+a1*ix(1)+a2*ix(2)^2+a3*ix(3)^3,M);
    x(i) = ix(4)/M;
    ix(1:3) = ix(2:4);
  end
  if doRounding, x = round(x,numDigits); end
  factor=calcFactor(x,false);
  if isnan(factor), factor=65535; end
end

function x = frac(x)
  x=mod(x,1);
end

function factor = calcFactor(x, bShowResults)
% Calculate the factor statistic for the array of random nnumbers x.

  if nargin < 2, bShowResults = false; end
  maxElems=length(x);
  meanx=mean(x);
  sdevx=std(x);
  % get the first 100 autocorrelation values
  acArr=autocorrArr(x,1,100);
  % calculate the chisquare for the 10-bin histogram
  numBins=10;
  expval=maxElems/numBins;
  [N1,ev1]=histcounts(x,numBins);
  chiSq10=sum((N1-expval).^2/expval);
  numBins=20;
  expval=maxElems/numBins;
  [N2,ev2]=histcounts(x,numBins);
  chiSq20=sum((N2-expval).^2/expval);
  numBins=20;
  [N3,ev3]=histcounts(acArr,numBins);
  ev3c=ev3(2:length(ev3));
  autoCorrSum = sum(dot(N3,abs(ev3c)));
  chsStat=chs(x);
  [Kplus,Kminus]=KStest(x);
  factor = 1000*(abs(meanx-0.5)+abs(sdevx-1/sqrt(12)))+100*(max(acArr)-
min(acArr))+100*autoCorrSum+chiSq10+chiSq20/2;
  factor = factor + 10*chsStat + 10*(Kplus + Kminus);
  if bShowResults
    fprintf('Mean = %g\nSdev = %g\n', meanx, sdevx);
    fprintf('Min = %g\nMax = %g\n', min(x), max(x));
    fprintf('Max lags = 100\n');
    fprintf('Auto correlation array\n');
    disp(acArr');
    fprintf('10-Bin Histogram\n');
    disp(N1); disp(ev1);
    fprintf('Chi-Sqr10 = %g\n', chiSq10);
    fprintf('20-Bin Histogram\n');
    disp(N2); disp(ev2);
    fprintf('Chi-Sqr20 = %g\n', chiSq20);
    fprintf('20-Bin Autocorrelation Histogram\n');
    disp(N3); disp(ev3);
    fprintf('Sum autocorrel product = %g\n', autoCorrSum);
    fprintf('Change of sign stat = %g\n', chsStat);
    fprintf('K+ = %g and K- = %g\n', Kplus, Kminus);
    fprintf('Factor = %g\n', factor);
  end
end

function acArr=autocorrArr(xdata,fromLag,toLag)

numLags=toLag-fromLag+1;
acArr=zeros(numLags,1);
j=1;
for i=fromLag:toLag
  acArr(j)=autocor(xdata,i);
  j=j+1;
end
end

function res = autocor(xdata,lag)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
maxElems=length(xdata);
res=corrcoef(xdata(1:maxElems-lag),xdata(lag+1:maxElems));
res=res(1,2);
end

function sumx=chs(x)
```

```
% Function CHS calculates the change of sign (between subsequent random
% numbers) moment. The function counts the number of consequtive positive
% abd negative changes of sign. The last nested loop calculates the
% statistic returned by this function. This value is the sum of:
%
%  sum = sum of difference(count,:) * count / difference(1,:)
%
% Keeping in mind that difference(1,:) is a good value that counts the
% sign flips that happpens one neighbor down. The values for
% difference(n,:) for n>1 are not desirable. The smaller, the better. The
% value difference(2,:) is the number of sign flips that occur
% two neighbors down. The value difference(3,:) is the number of sign flips
% that occur three neighbors down, and so on.

  n=length(x);
  nby2=fix(n/2);
  Diff=zeros(nby2,2);
  countPos=0;
  countNeg=0;
  s1=sign(x(2)-x(1));
  if s1>0
    bIsPos=true;
    countPos=1;
  else
    bIsPos=false;
    countNeg=1;
  end

  for i=3:n
    s2=sign(x(i)-x(i-1));
    % was positive and is still positive
    if s2>0 && bIsPos
      countPos=countPos+1;
    % was negative and is now positive
    elseif s2>0 && ~bIsPos
      bIsPos=true;
      countPos=1;
      Diff(countNeg,2)=Diff(countNeg,2)+1;
      countNeg=0;
    % was negative and is still negative
    elseif s2<0 && ~bIsPos
      countNeg=countNeg+1;
    % was positive is and is now negative
    elseif s2<0 && bIsPos
      bIsPos=false;
      countNeg=1;
      Diff(countPos,1)=Diff(countPos,1)+1;
      countPos=0;
    end
  end

  if s2>0
    if countPos>0, Diff(countPos,1)=Diff(countPos,1)+1; end
  else
    if countNeg>0, Diff(countNeg,2)=Diff(countNeg,2)+1; end
  end
  i=2:nby2;
  d=Diff(2:nby2,:);
  sumx=0;
  for j=1:2
    sumx = sumx + dot(d(:,j),i)/Diff(1,j);
  end
end

function [Kplus,Kminus]=KStest(x)
  x=sort(x);
  n=length(x);
  diffMaxPlus=-1e+99;
  diffMaxMinus=-1e+99;
  i=1;
  for xv=0.001:.001:1
    F=xv;
    while x(i)<=xv && i<n
      i=i+1;
    end
    Fn=1;
    if i<n, Fn=(i-1)/n; end
    diff=Fn-F;
    if diff>diffMaxPlus, diffMaxPlus=diff; end
    diff=-diff;
    if diff>diffMaxMinus, diffMaxMinus=diff; end
  end
  Kplus=sqrt(n)*diffMaxPlus;
  Kminus=sqrt(n)*diffMaxMinus;
end
```

Listing 2.3. The listing of file rng997Gen1.m.

Tables 2.1a and 2.1b show the results of the wide-trust region optimization.

	res1	res2	res3	res4
	Wide Range	Wide Range	Wide Range	Wide Range
Mean	81635.0541	77316.69005	112.177972	112.337923
Sdev	121.63083	445.5158084	2.26812131	2.37883775
Min	81371.369	76130.20812	105.082393	105.843282
Max	81927.3318	78833.9586	116.255757	116.910787
Range	555.962807	2703.750481	11.173364	11.067505
Count	100	100	100	100
Config	27.2623674	99.85803482	0.50837733	0.53319334
CI Upper	81662.3164	77416.54809	112.686349	112.871117
CI Lower	81607.7917	77216.83202	111.669594	111.80473
Factor	81371.369	76130.20812	105.082393	105.843282
IX	25464	87326	100000	100000
A0	754568	610443	639687	11
A1	164418	314599	143059	332003
A2	284470	11	78456	292469
A3	11	223305	849312	567415
Xrnd	26583	19275	20094	46316
Max Elements	10000	10000	10000	10000
Max Iters	100	100	100	100
Xrndlow	100	100	100	100

Xrndhi	100000	100000	100000	100000
A01low	11	11	11	11
A0hi	1000000	1000000	1000000	1000000
A1low	11	11	11	11
A1hi	1000000	1000000	1000000	1000000
A2low	11	11	11	11
A2hi	1000000	1000000	1000000	1000000
A3low	11	11	11	11
A3hi	1000000	1000000	1000000	1000000
PopSize	250	250	250	250
PopMaxIters	350	350	350	350
M	1.84E+19	1.80144E+16	2.8147E+14	1.0995E+12
Rounded?	1	1	1	1
Rounded	10	10	10	10

Table 2.1a. The results of the wide-trust region optimization.

	res5	res6	res7
	Wide Range	Wide Range	Wide Range
Mean	112.161783	112.0640227	112.157201
Sdev	2.27252377	2.708827233	2.31183383
Min	106.070723	102.687532	105.456473
Max	116.936164	117.15534	117.334279
Range	10.865441	14.467808	11.877806
Count	100	100	100
Config	0.5093641	0.607157275	0.51817506
CI Upper	112.671147	112.6711799	112.675376
CI Lower	111.652418	111.4568654	111.639026
Factor	106.070723	102.687532	105.456473
IX	100000	27845	45944
A0	598206	126525	728307
A1	177932	625719	244905
A2	86258	383934	447851
A3	11	705626	254242
Xrnd	22007	11907	20219
Max Elements	10000	10000	10000
Max Iters	100	100	100
Xrndlow	100	100	100
Xrndhi	100000	100000	100000
A01low	11	11	11
A0hi	1000000	1000000	1000000
A1low	11	11	11
A1hi	1000000	1000000	1000000
A2low	11	11	11
A2hi	1000000	1000000	1000000
A3low	11	11	11
A3hi	1000000	1000000	1000000
PopSize	250	250	250
PopMaxIters	350	350	350
M	4294967295	4294967295	4294967295
Rounded?	1	1	1
Rounded	10	10	10

Table 2.1b. The results of the wide-trust region optimization.

Tables 2.2a and 2.2b show the results of the narrow-trust region optimization.

	res1b	res2b	res3b	res4b
	Narrow Range	Narrow Range	Narrow Range	Narrow Range
Mean	81565.8372	76994.87718	112.1536634	111.9899625
Sdev	90.5848419	456.4185372	2.265219642	2.421662623
Min	81322.6984	75166.255	103.777367	103.340804
Max	81835.266	77668.22261	116.221927	116.118588
Range	512.567535	2501.967615	12.44456	12.777784
Count	100	100	100	100
Config	20.3037112	102.3017754	0.507726948	0.542792121
CI Upper	81586.1409	77097.17896	112.6613903	112.5327546
CI Lower	81545.5334	76892.57541	111.6459364	111.4471704
Factor	81322.6984	75166.255	103.777367	103.340804
IX	24782	91116	89438	103957
A0	698177	604332	656047	9
A1	151358	311643	152740	282203
A2	306517	9	69711	248599
A3	12	193971	819039	652527
Xrnd	13685	83509	36267	65285
Max Elements	10000	10000	10000	10000
Max Iters	100	100	100	100
Xrndlow	21644	74227	85000	85000
Xrndhi	29284	100425	115000	115000
A01low	641383	518877	543734	543734
A0hi	867753	702009	735640	735640
A1low	139755	267409	121600	121600
A1hi	189081	361789	164518	164518
A2low	241800	9	66688	66688
A2hi	327141	13	90224	90224
A3low	9	189809	721915	721915
A3hi	13	256801	976709	976709
PopSize	250	250	250	250
PopMaxIters	350	350	350	350
M	1.84E+19	1.80144E+16	2.81475E+14	2.81E+14
Rounded?	1	1	1	1
Rounded	10	10	10	10

Table 2.2a. The results of the narrow-trust region optimization.

	res5b	res6b	res7b
	Narrow Range	Narrow Range	Narrow Range
Mean	112.236503	111.767517	112.449532
Sdev	2.28377558	2.71789509	2.82236528
Min	106.719969	101.620224	100.787048
Max	116.782548	116.928214	116.986248
Range	10.062579	15.30799	16.1992
Count	100	100	100
Config	0.51188608	0.60918975	0.63260572
CI Upper	112.748389	112.376707	113.082138
CI Lower	111.724617	111.158328	111.816926

Factor	106.719969	101.620224	100.787048
IX	115000	26502	39390
A0	602849	143195	619061
A1	187018	566322	280235
A2	86608	393717	439815
A3	11	749697	268138
Xrnd	91931	16604	1870
Max Elements	10000	10000	10000
Max Iters	100	100	100
Xrndlow	85000	23668	39052
Xrndhi	115000	32022	52836
A01low	508475	107546	619061
A0hi	687937	145504	837553
A1low	151242	531861	208169
A1hi	204622	719577	281641
A2low	73319	326344	380673
A2hi	99197	441524	515029
A3low	9	599782	216106
A3hi	13	811470	292378
PopSize	250	250	250
PopMaxIters	350	350	350
M	4294967295	4294967295	4294967295
Rounded?	1	1	1
Rounded	10	10	10

Table 2.2b. The results of the narrow-trust region optimization.

Listing 2.4 shows the source code for file do2.m which performs the random-seed generation of one million sets of 10,000 random numbers for each tested version of the algorithm.

```
maxElems=10000;
maxIters=1000000;
c = [25464,754568,164418,284470,11];
sFilename='res1c.csv';
xM=2^64-1;
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [91116,604332,311643,9,193971];
sFilename='res2c.csv';
xM=2^54-1;
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [89438,656047,152740,69711,819039];
```

```
sFilename='res3c.csv';
xM=2^48-1;
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [103957,9,282203,248599,652527];
sFilename='res4c.csv';
xM=2^40-1;
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [115000,602849,187018,86608,11];
sFilename='res5c.csv';
xM=2^32-1;
bRound=true;
nDigits = 10;
doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [26502,143195,566322,393717,749697];
sFilename='res6c.csv';
xM1=2^24-1;
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [39390,619061,280235,439815,268138];
sFilename='res7c.csv';
xM1=2^16-1;
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

% ------------------------------------------------------------------------

system('shutdown /s')
```

Listing 2.4. The source code of file do2.m.

Listing 2.5 shows the source code for file doAll2.m.

```
function doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)
global gmaxElems
global M
global doRounding
global numDigits

gmaxElems = maxElems;
```

```
M = xM;
doRounding = bRound;
numDigits = nDigits;
resMat=zeros(maxIters,7);

for i=1:maxIters
  [factor,Xrnd] = rng997Gen2(c);
  resMat(i,1) = factor;
  resMat(i,2:6) = round(c,0);
  resMat(i,7) = Xrnd;

  fprintf("Itr: %d, Factor=%f, [", i, factor);
  fprintf(" %d,", resMat(i,2:6));
  fprintf("%d]\n", Xrnd);

end

fprintf("\n\n");
resMat = sortrows(resMat,1);

beep on;
for i=1:3
  beep;
  pause(1);
end

fprintf("Entire result matrix written to file %s\n", sFilename)
fh = fopen(sFilename, "w");
fprintf(fh,"Factor,IX,A0,A1,A2,A3,Xrnd\n");
for i=1:maxIters
  fprintf(fh,"%f,", resMat(i,1));
  for j=2:6
      fprintf(fh,"%d,", resMat(i,j));
  end
  fprintf(fh,"%d\n", resMat(i,7));
end
fclose(fh);


fprintf("-------------------------------------------------------\n\n");
end
```

Listing 2.5. The source code of file doAll2.m.

Listing 2.6 shows the source code for file rng997Gen2.m.

```
function [factor,Xrnd] = rng997Gen2(c)
%UNTITLED2 Summary of this function goes here
  global gmaxElems
  global M
  global doRounding
  global numDigits

  maxElems = gmaxElems;
  c = round(c, 0);
  ix = zeros(4,1);
  ix(1) = round(rand*c(1),0);
  Xrnd = ix(1);

  a0 = c(2);
  a1 = c(3);
  a2 = c(4);
  a3 = c(5);
  ix(2) = mod(a0+a1*ix(1),M);
  ix(3) = mod(a0+a1*ix(1)+a2*ix(2)^2,M);
  x=zeros(maxElems,1);
  for i=1:maxElems
    ix(4) = mod(a0+a1*ix(1)+a2*ix(2)^2+a3*ix(3)^3,M);
    x(i) = ix(4)/M;
    ix(1:3) = ix(2:4);
  end
  if doRounding, x = round(x,numDigits); end
  factor=calcFactor(x,false);
  if isnan(factor), factor=65535; end

end

function x = frac(x)
  x=mod(x,1);
end

function factor = calcFactor(x, bShowResults)
% Calculate the factor statistic for the array of random nnumbers x.

  if nargin < 2, bShowResults = false; end
  maxElems=length(x);
  meanx=mean(x);
  sdevx=std(x);
  % get the first 100 autocorrelation values
  acArr=autocorrArr(x,1,100);
  % calculate the chisquare for the 10-bin histogram
  numBins=10;
  expval=maxElems/numBins;
  [N1,ev1]=histcounts(x,numBins);
  chiSq10=sum((N1-expval).^2/expval);
  numBins=20;
  expval=maxElems/numBins;
  [N2,ev2]=histcounts(x,numBins);
  chiSq20=sum((N2-expval).^2/expval);
  numBins=20;
  [N3,ev3]=histcounts(acArr,numBins);
  ev3c=ev3(2:length(ev3));
  autoCorrSum = sum(dot(N3,abs(ev3c)));
  chsStat=chs(x);
  [Kplus,Kminus]=KStest(x);
  factor = 1000*(abs(meanx-0.5)+abs(sdevx-1/sqrt(12)))+100*(max(acArr)-
min(acArr))+100*autoCorrSum+chiSq10+chiSq20/2;
  factor = factor + 10*chsStat + 10*(Kplus + Kminus);
  if bShowResults
    fprintf('Mean = %g\nSdev = %g\n', meanx, sdevx);
    fprintf('Min = %g\nMax = %g\n', min(x), max(x));
    fprintf('Max lags = 100\n');
    fprintf('Auto correlation array\n');
    disp(acArr');
    fprintf('10-Bin Histogram\n');
    disp(N1); disp(ev1);
    fprintf('Chi-Sqr10 = %g\n', chiSq10);
    fprintf('20-Bin Histogram\n');
    disp(N2); disp(ev2);
    fprintf('Chi-Sqr20 = %g\n', chiSq20);
    fprintf('20-Bin Autocorrelation Histogram\n');
    disp(N3); disp(ev3);
    fprintf('Sum autocorrel product = %g\n', autoCorrSum);
    fprintf('Change of sign stat = %g\n', chsStat);
    fprintf('K+ = %g and K- = %g\n', Kplus, Kminus);
    fprintf('Factor = %g\n', factor);
  end
end

function acArr=autocorrArr(xdata,fromLag,toLag)

numLags=toLag-fromLag+1;
acArr=zeros(numLags,1);
j=1;
for i=fromLag:toLag
  acArr(j)=autocor(xdata,i);
  j=j+1;
end
end

function res = autocor(xdata,lag)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
maxElems=length(xdata);
res=corrcoef(xdata(1:maxElems-lag),xdata(lag+1:maxElems));
res=res(1,2);
end

function sumx=chs(x)
% Function CHS calculates the change of sign (between subsequent random
% numbers) moment. The function counts the number of consequtive positive
% abd negative changes of sign. The last nested loop calculates the
% statistic returned by this function. This value is the sum of:
%
%  sum = sum of difference(count,:) * count / difference(1,:)
%
% Keeping in mind that difference(1,:) is a good value that counts the
% sign flips that happpens one neighbor down. The values for
% difference(n,:) for n>1 are not desirable. The smaller, the better. The
% value difference(2,:) is the number of sign flips that occur
% two neighbors down. The value difference(3,:) is the number of sign flips
% that occur three neighbors down, and so on.

  n=length(x);
  nby2=fix(n/2);
  Diff=zeros(nby2,2);
  countPos=0;
  countNeg=0;
  s1=sign(x(2)-x(1));
  if s1>0
    bIsPos=true;
    countPos=1;
  else
    bIsPos=false;
    countNeg=1;
  end

  for i=3:n
    s2=sign(x(i)-x(i-1));
    % was positive and is still positive
    if s2>0 && bIsPos
      countPos=countPos+1;
    % was negative and is now positive
    elseif s2>0 && ~bIsPos
      bIsPos=true;
      countPos=1;
      Diff(countNeg,2)=Diff(countNeg,2)+1;
      countNeg=0;
    % was negative and is still negative
    elseif s2<0 && ~bIsPos
      countNeg=countNeg+1;
    % was positive is and is now negative
    elseif s2<0 && bIsPos
      bIsPos=false;
      countNeg=1;
      Diff(countPos,1)=Diff(countPos,1)+1;
      countPos=0;
    end
  end

  if s2>0
    if countPos>0, Diff(countPos,1)=Diff(countPos,1)+1; end
  else
    if countNeg>0, Diff(countNeg,2)=Diff(countNeg,2)+1; end
  end

  i=2:nby2;
  d=Diff(2:nby2,:);
  sumx=0;
  for j=1:2
    sumx = sumx + dot(d(:,j),i)/Diff(1,j);
  end
end

function [Kplus,Kminus]=KStest(x)
  x=sort(x);
  n=length(x);
  diffMaxPlus=-1e+99;
  diffMaxMinus=-1e+99;
  i=1;
  for xv=0.001:.001:1
    F=xv;
    while x(i)<=xv && i<n
      i=i+1;
    end
    Fn=1;
    if i<n, Fn=(i-1)/n; end
    diff=Fn-F;
    if diff>diffMaxPlus, diffMaxPlus=diff; end
    diff=-diff;
    if diff>diffMaxMinus, diffMaxMinus=diff; end
  end
  Kplus=sqrt(n)*diffMaxPlus;
  Kminus=sqrt(n)*diffMaxMinus;
end
```

Listing 2.6. The source code of file rng997Gen2.m.

Tables 2.3a and 2.3b show the results of the penalty factor statistics for the different versions of the algorithm.

	res1c	res2c	res3c	res4c
	Random Seed	Random Seed	Random Seed	Random Seed
Mean	81365.6438	78566.7395	147.822052	147.791295
Sdev	38.2245686	2058.14194	11.7896165	11.7579757
Min	81282.6279	75135.4541	103.777367	103.340804
Max	81586.0334	83650.6705	214.063501	218.642275
Range	303.405561	8515.21647	110.286134	115.301471
Count	1000000	1000000	1000000	1000000
Config	0.08567665	4.61312496	0.02642528	0.02635436
CI Upper	81365.7295	78571.3526	147.848477	147.817649
CI Lower	81365.5581	78562.1264	147.795626	147.76494
Factor	81282.6279	75135.4541	103.777367	103.340804
IX	25464	91116	89438	103957
A0	754568	604332	656047	9
A1	164418	311643	152740	282203
A2	284470	9	69711	248599
A3	11	193971	819039	652527
Xrnd	5425	83438	3937	96000

Table 2.3a. The random-seed results for the penalty factor statistics.

	res5c	res6c	res7c
	Random Seed	Random Seed	Random Seed
Mean	147.736873	147.767284	147.752273
Sdev	11.7727362	11.8500221	11.7824902
Min	106.719969	101.620224	100.787048
Max	213.977883	220.426814	215.348181
Range	107.257914	118.80659	114.561133
Count	1000000	1000000	1000000
Config	0.02638744	0.02656067	0.02640931
CI Upper	147.763261	147.793845	147.778682
CI Lower	147.710486	147.740723	147.725863
Factor	106.719969	101.620224	100.787048
IX	115000	26502	39390
A0	602849	143195	619061
A1	187018	566322	280235
A2	86608	393717	439815
A3	11	749697	268138
Xrnd	19879	7450	4970

Table 2.3b. The random-seed results for the penalty factor statistics.

The column titled res5c in Table 2.3b has the lowest upper mean value. The best modified power method equation is:

```
M = 2^24-1
a0 = 602849
a1 = 187018
a2 = 86608
a3 = 11
ix(1) = round(rand* 115000,0);
ix(2) = a0+a1*ix(1) mod M
ix(3) = a0+a1*ix(1)+a2*ix(2)^2 mod M
for i=1 to maximum number of random numbers
  ix(4) = a0+a1*ix(1)+a2*ix(2)^2+a3*ix(3)^3 mod M
  x(i) = ix(4)/M
  ix(1:3) = ix(2:4)
end                                                        (2.2)
```

The value x(i) is the uniform random number generated in the range of 0 to 1(excluded) in each iteration.

3/ LCGM Cubed Algorithm (Version Ver2 PSO)

The descending-power LCGM Cubed algorithm is defined as:

```
ix(1) = round(rand*seed0);
ix(2) = a0+a1*ix(1) mod M
ix(3) = a0+a1*ix(1) ^2 +a2*ix(2) mod M
for i=1 to maximum number of random numbers
  ix(4) = a0+a1*ix(1) ^3 +a2*ix(2)^2+a3*ix(3) mod M
  x(i) = ix(4)/M
  ix(1:3) = ix(2:4)
end                                                        (3.1)
```

The difference between equations 2.1 and 3.1 is that the powers used to calculate ix(4) appear in a reversed order. Those in equation 2.1 have an ascending order. Those in equation 3.1 have a descending order.

This section looks at optimizing the values of a0, a1, a3, and a43 used in equation 3.1. The array x() is the sought array of uniform random values in the range (0, 1]. The new random number x(i) obtains its value from three previous random integer values ix(1), ix(2) and ix(3). The random number generating loops keeps the newer three values of the integer array ix(). M is the modulus value.

The approache that estimates the best coefficients for the power method uses the following approach:

1. Optimize the penalty factor (see Appendix of Project 997 PRNGs Part 1) using particle swarm optimization algorithms. This optimization starts with a wide trust region and narrows it down.
2. Optimize the penalty using the narrowed optimum regions obtained in step 1. This step yields refined trust regions.
3. Perform a large run of generating random numbers using the refined trust regions from step 2. The calculations yield the statistics for the penalty factor. The upper confidence values for the mean penlty factor are the values we look at to determine the fitness of the algorithm.

All the phases involve obtaining initial random values for the first k elements of arrays ix and iy. This is followed by applying equation set 3.1.

The listing for do.m, which triggers the calculations for the first and second optimization phases is:

```
maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=[1000 10000 10000 10000 10000];
sFilename='res1.csv';
sSheetName='res1Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^16-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=5*[1000 10000 10000 10000 10000];
sFilename='res2.csv';
sSheetName='res2Sheet1.csv';
```

```
POSpopsize=100;
POSmaxIters=150;
xM=2^24-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=[10000 100000 100000 100000 100000];
sFilename='res3.csv';
sSheetName='res3Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^32-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=5*[10000 100000 100000 100000 100000];
sFilename='res4.csv';
sSheetName='res4Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^40-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=[100000 1000000 1000000 1000000 1000000];
sFilename='res5.csv';
sSheetName='res5Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^48-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
```

```
ub=5*[100000 1000000 1000000 1000000 1000000];
sFilename='res6.csv';
sSheetName='res6Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^56-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=[1000000 10000000 10000000 10000000 10000000];
sFilename='res7.csv';
sSheetName='res7Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^64-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

%  ----------------------------------------------------------------------

r=0.15;
rp = 1+r;
rm = 1 - r;
maxElems=10000;
maxIters=30;
c = [269,1456,136,5715,3982]; % first is Xrnd
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res1b.csv';
sSheetName='res1bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^16-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
c = [19,37891,11421,42929,29737];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res2b.csv';
sSheetName='res2bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^24-1;
```

```
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
c = [432,26854,58862,67184,16596];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res3b.csv';
sSheetName='res3bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^32-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
c = [43234,430936,188672,151556,11];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res4b.csv';
sSheetName='res4bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^40-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
c = [24368,233091,99589,731258,440046];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res5b.csv';
sSheetName='res5bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^48-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
c = [181208,1457623,822718,488892,3959223];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res6b.csv';
sSheetName='res6bSheet1.csv';
```

```
POSpopsize=100;
POSmaxIters=150;
xM1=2^56-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
c = [467639,5289050,10000000,10000000,6997150];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res7b.csv';
sSheetName='res7bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM1=2^64-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

system('shutdown /s')

% -------------------------------------------------------------------------

function c = rotate(c)
  n=length(c);
  buff = c(n);
  for i=n:-1:2
    c(i) = c(i-1);
  end
  c(1) = buff;
end
```

Listing 3.1. The listing of file do.m.

Listing 3.2 shows the source code for file doAll.m. The function doAll() optimizes the function rng997Gen1() using the Particle Swarm Optimization (PSO) method by calling function particleswarm(). The function do() calls function doAll() for the first and second optimization phases.

```
maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=[1000 10000 10000 10000 10000];
sFilename='res1.csv';
sSheetName='res1Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^16-1;
bRound=true;
nDigits = 10;
```

```
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=5*[1000 10000 10000 10000 10000];
sFilename='res2.csv';
sSheetName='res2Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^24-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=[10000 100000 100000 100000 100000];
sFilename='res3.csv';
sSheetName='res3Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^32-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=5*[10000 100000 100000 100000 100000];
sFilename='res4.csv';
sSheetName='res4Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^40-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=[100000 1000000 1000000 1000000 1000000];
```

```
sFilename='res5.csv';
sSheetName='res5Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^48-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=5*[100000 1000000 1000000 1000000 1000000];
sFilename='res6.csv';
sSheetName='res6Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^56-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
lb=[100 11 11 11 11];
ub=[1000000 10000000 10000000 10000000 10000000];
sFilename='res7.csv';
sSheetName='res7Sheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^64-1;
bRound=true;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

%  ---------------------------------------------------------------------

r=0.15;
rp = 1+r;
rm = 1 - r;
maxElems=10000;
maxIters=30;
c = [269,1456,136,5715,3982]; % first is Xrnd
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res1b.csv';
sSheetName='res1bSheet1.csv';
```

```
POSpopsize=100;
POSmaxIters=150;
xM=2^16-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
c = [19,37891,11421,42929,29737];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res2b.csv';
sSheetName='res2bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^24-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
c = [432,26854,58862,67184,16596];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res3b.csv';
sSheetName='res3bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^32-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
c = [43234,430936,188672,151556,11];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res4b.csv';
sSheetName='res4bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^40-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)
```

```
maxElems=10000;
maxIters=30;
c = [24368,233091,99589,731258,440046];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res5b.csv';
sSheetName='res5bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM=2^48-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
c = [181208,1457623,822718,488892,3959223];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res6b.csv';
sSheetName='res6bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM1=2^56-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

maxElems=10000;
maxIters=30;
c = [467639,5289050,10000000,10000000,6997150];
lb=round(c*rm,0);
ub= round(c*rp,0);
sFilename='res7b.csv';
sSheetName='res7bSheet1.csv';
POSpopsize=100;
POSmaxIters=150;
xM1=2^64-1;
bRound=true;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,xM,
bRound,nDigits)

system('shutdown /s')

% -------------------------------------------------------------------------

function c = rotate(c)
  n=length(c);
  buff = c(n);
  for i=n:-1:2
```

```
    c(i) = c(i-1);
  end
  c(1) = buff;
end
```

Listing 3.2. The listing of file doAll.m.

Listing 3.3 shows the listing of Reg997Gen1.m.

```
function factor = rng997Gen1(c)
%UNTITLED2 Summary of this function goes here
  global gmaxElems
  global M
  global Xrnd
  global doRounding
  global numDigits
  global bestFactor

  maxElems = gmaxElems;
  c = round(c, 0);
  ix = zeros(4,1);
  ix(1) = round(rand*c(1),0);
  Xrnd = ix(1);

  a0 = c(2);
  a1 = c(3);
  a2 = c(4);
  a3 = c(5);
  ix(2) = mod(a0+a1*ix(1),M);
  ix(3) = mod(a0+a1*ix(1)^2+a2*ix(2),M);
  x=zeros(maxElems,1);
  for i=1:maxElems
    ix(4) = mod(a0+a1*ix(1)^3+a2*ix(2)^2+a3*ix(3),M);
    x(i) = ix(4)/M;
    ix(1:3) = ix(2:4);
  end
  if doRounding, x = round(x,numDigits); end
  factor=calcFactor(x,false);
  if isnan(factor), factor=65535; end
  if bestFactor > factor
      bestFactor = factor;
  end
end

function x = frac(x)
  x=mod(x,1);
end

function factor = calcFactor(x, bShowResults)
% Calculate the factor statistic for the array of random nnumbers x.

  if nargin < 2, bShowResults = false; end
  maxElems=length(x);
  meanx=mean(x);
  sdevx=std(x);
  % get the first 100 autocorrelation values
  acArr=autocorrArr(x,1,100);
  % calculate the chisquare for the 10-bin histogram
  numBins=10;
  expval=maxElems/numBins;
  [N1,ev1]=histcounts(x,numBins);
  chiSq10=sum((N1-expval).^2/expval);
  numBins=20;
  expval=maxElems/numBins;
  [N2,ev2]=histcounts(x,numBins);
  chiSq20=sum((N2-expval).^2/expval);
  numBins=20;
  [N3,ev3]=histcounts(acArr,numBins);
  ev3c=ev3(2:length(ev3));
  autoCorrSum = sum(dot(N3,abs(ev3c)));
  chsStat=chs(x);
  [Kplus,Kminus]=KStest(x);
  factor = 1000*(abs(meanx-0.5)+abs(sdevx-1/sqrt(12)))+100*(max(acArr)-
min(acArr))+100*autoCorrSum+chiSq10+chiSq20/2;
  factor = factor + 10*chsStat + 10*(Kplus + Kminus);
  if bShowResults
    fprintf('Mean = %g\nSdev = %g\n', meanx, sdevx);
    fprintf('Min = %g\nMax = %g\n', min(x), max(x));
    fprintf('Max lags = 100\n');
    fprintf('Auto correlation array\n');
    disp(acArr');
    fprintf('10-Bin Histogram\n');
    disp(N1); disp(ev1);
    fprintf('Chi-Sqr10 = %g\n', chiSq10);
    fprintf('20-Bin Histogram\n');
    disp(N2); disp(ev2);
    fprintf('Chi-Sqr20 = %g\n', chiSq20);
    fprintf('20-Bin Autocorrelation Histogram\n');
    disp(N3); disp(ev3);
    fprintf('Sum autocorrel product = %g\n', autoCorrSum);
    fprintf('Change of sign stat = %g\n', chsStat);
    fprintf('K+ = %g and K- = %g\n', Kplus, Kminus);
    fprintf('Factor = %g\n', factor);
  end
end

function acArr=autocorrArr(xdata,fromLag,toLag)

numLags=toLag-fromLag+1;
acArr=zeros(numLags,1);
j=1;
for i=fromLag:toLag
  acArr(j)=autocor(xdata,i);
  j=j+1;
end
end

function res = autocor(xdata,lag)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
maxElems=length(xdata);
res=corrcoef(xdata(1:maxElems-lag),xdata(lag+1:maxElems));
res=res(1,2);
```

```
end

function sumx=chs(x)
% Function CHS calculates the change of sign (between subsequent random
% numbers) moment. The function counts the number of consequtive positive
% abd negative changes of sign. The last nested loop calculates the
% statistic returned by this function. This value is the sum of:
%
%  sum = sum of difference(count,:) * count / difference(1,:)
%
% Keeping in mind that difference(1,:) is a good value that counts the
% sign flips that happpens one neighbor down. The values for
% difference(n,:) for n>1 are not desirable. The smaller, the better. The
% value difference(2,:) is the number of sign flips that occur
% two neighbors down. The value difference(3,:) is the number of sign flips
% that occur three neighbors down, and so on.

  n=length(x);
  nby2=fix(n/2);
  Diff=zeros(nby2,2);
  countPos=0;
  countNeg=0;
  s1=sign(x(2)-x(1));
  if s1>0
    bIsPos=true;
    countPos=1;
  else
    bIsPos=false;
    countNeg=1;
  end

  for i=3:n
    s2=sign(x(i)-x(i-1));
    % was positive and is still positive
    if s2>0 && bIsPos
      countPos=countPos+1;
    % was negative and is now positive
    elseif s2>0 && ~bIsPos
      bIsPos=true;
      countPos=1;
      Diff(countNeg,2)=Diff(countNeg,2)+1;
      countNeg=0;
    % was negative and is still negative
    elseif s2<0 && ~bIsPos
      countNeg=countNeg+1;
    % was positive is and is now negative
    elseif s2<0 && bIsPos
      bIsPos=false;
      countNeg=1;
      Diff(countPos,1)=Diff(countPos,1)+1;
      countPos=0;
    end
  end

  if s2>0
    if countPos>0, Diff(countPos,1)=Diff(countPos,1)+1; end
  else
```

```
    if countNeg>0, Diff(countNeg,2)=Diff(countNeg,2)+1; end
  end

  i=2:nby2;
  d=Diff(2:nby2,:);
  sumx=0;
  for j=1:2
    sumx = sumx + dot(d(:,j),i)/Diff(1,j);
  end
end

function [Kplus,Kminus]=KStest(x)
  x=sort(x);
  n=length(x);
  diffMaxPlus=-1e+99;
  diffMaxMinus=-1e+99;
  i=1;
  for xv=0.001:.001:1
    F=xv;
    while x(i)<=xv && i<n
      i=i+1;
    end
    Fn=1;
    if i<n, Fn=(i-1)/n; end
    diff=Fn-F;
    if diff>diffMaxPlus, diffMaxPlus=diff; end
    diff=-diff;
    if diff>diffMaxMinus, diffMaxMinus=diff; end
  end
  Kplus=sqrt(n)*diffMaxPlus;
  Kminus=sqrt(n)*diffMaxMinus;
end
```

Listing 3.3. The listing of file rng997Gen1.m.

Tables 3.1a and 3.1b show the results of the wide-trust region optimization.

	res1	res2	res3	res4
	Wide Range	Wide Range	Wide Range	Wide Range
Mean	114.317013	113.7338428	114.60363	114.9292865
Sdev	2.78439167	2.69911576	2.74356001	1.919133897
Min	108.656993	106.120012	108.711506	111.148654
Max	118.360758	117.389336	118.699493	117.625809
Range	9.703765	11.269324	9.987987	6.477155
Count	30	30	30	30
Config	1.1394351	1.1045383	1.12272588	0.785352346
CI Upper	115.456448	114.8383811	115.726356	115.7146389
CI Lower	113.177577	112.6293045	113.480904	114.1439342
Max Elements	10000	10000	10000	10000
Max Iters	30	30	30	30
Xrndlow	100	100	100	100
Xrndhi	1000	5000	10000	50000
A01low	11	11	11	11
A0hi	10000	50000	100000	500000
A1low	11	11	11	11

A1hi	10000	50000	100000	500000
A2low	11	11	11	11
A2hi	10000	50000	100000	500000
A3low	11	11	11	11
A3hi	10000	50000	100000	500000
PopSize	100	100	100	100
PopMaxIters	150	150	150	150
M	65535	16777215	4294967295	1.09951E+12
Rounded?	1	1	1	1
Rounded	10	10	10	10

Table 3.1a. The results of the wide-trust region optimization.

	res5	res6	res7
	Wide Range	**Wide Range**	**Wide Range**
Mean	114.1516545	133739.4574	110189.668
Sdev	2.606089667	10198.94026	27394.6116
Min	106.358872	102176.7162	65535
Max	118.544638	156563.7393	127451.449
Range	12.185766	54387.02313	61916.4491
Count	30	30	30
Config	1.06646995	4173.633568	11210.4854
CI Upper	115.2181245	137913.0909	121400.153
CI Lower	113.0851845	129565.8238	98979.1822
Max Elements	10000	10000	10000
Max Iters	30	30	30
Xrndlow	100	100	100
Xrndhi	100000	500000	1000000
A01low	11	11	11
A0hi	1000000	5000000	10000000
A1low	11	11	11
A1hi	1000000	5000000	10000000
A2low	11	11	11
A2hi	1000000	5000000	10000000
A3low	11	11	11
A3hi	1000000	5000000	10000000
PopSize	100	100	100
PopMaxIters	150	150	150
M	2.81475E+14	7.20576E+16	1.84E+19
Rounded?	1	1	1
Rounded	10	10	10

Table 3.1b. The results of the wide-trust region optimization.

Table 3.2a and 3.2b show the results of the narrow-trust region optimization.

	res1b	res2b	res3b	res4b
	Narrow Range	**Narrow Range**	**Narrow Range**	**Narrow Range**
Mean	113.471868	114.6504073	114.2827493	113.82751
Sdev	2.27932646	2.738373956	2.609556701	2.616003242
Min	109.927226	109.115436	109.093243	108.403647
Max	118.125687	120.53568	119.630191	120.240222
Range	8.198461	11.420244	10.536948	11.836575
Count	30	30	30	30

Config	0.93275116	1.120603628	1.067888738	1.070526806
CI Upper	114.404619	115.7710109	115.350638	114.8980368
CI Lower	112.539117	113.5298036	113.2148605	112.7569832
Max Elements	10000	10000	10000	10000
Max Iters	30	30	30	30
Xrndlow	229	16	367	36749
Xrndhi	309	22	497	49719
A01low	1238	32207	22826	366296
A0hi	1674	43575	30882	495576
A1low	116	9708	50033	160371
A1hi	156	13134	67691	216973
A2low	4858	36490	57106	128823
A2hi	6572	49368	77262	174289
A3low	3385	25276	14107	9
A3hi	4579	34198	19085	13
PopSize	100	100	100	100
PopMaxIters	150	150	150	150
M	65535	16777215	4294967295	1.09951E+12
Rounded?	1	1	1	1
Rounded	10	10	10	10

Table 3.2a. The results of the narrow-trust region optimization.

	res5b	res6b	res7b
	Narrow Range	Narrow Range	Narrow Range
Mean	114.1191284	82612.13836	125893.7933
Sdev	2.399901682	2697.307691	8.925302637
Min	108.298396	79277.06128	125882.2712
Max	119.134508	91764.04781	125917.9114
Range	10.836112	12486.98654	35.640261
Count	30	30	30
Config	0.982093233	1103.798398	3.652432678
CI Upper	115.1012216	83715.93676	125897.4457
CI Lower	113.1370351	81508.33996	125890.1408
Max Elements	10000	10000	10000
Max Iters	30	30	30
Xrndlow	20713	64117	48
Xrndhi	28023	86747	66
A01low	198127	74028	9
A0hi	268055	100156	13
A1low	84651	9	9
A1hi	114527	13	13
A2low	621569	9	2325566
A2hi	840947	13	3146354
A3low	374039	175120	1096172
A3hi	506053	236928	1483056
PopSize			
PopMaxIters			
M			
Rounded?			
Rounded			

Table 3.2b. The results of the narrow-trust region optimization.

Listing 3.4 shows the source code for file do2.m which performs the random-seed generation of one million sets of 10,000 random numbers for each tested version of the algorithm.

```
maxElems=10000;
maxIters=1000000;
c = [122,1576,127,5040,4510];
sFilename='res1c.csv';
xM=2^16-1;
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [11,34876,9754,45847,29574];
sFilename='res2c.csv';
xM=2^24-1;
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [116,24270,67691,73136,19085];
sFilename='res3c.csv';
xM=2^32-1;
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [24954,461360,165181,147147,10];
sFilename='res4c.csv';
xM=2^40-1;
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [14789,204314,89152,748267,471935];
sFilename='res5c.csv';
xM=2^48-1;
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [8531,1272183,946126,562226,3683294];
sFilename='res6c.csv';
xM=2^56-1;
```

```
bRound=true;
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

maxElems=10000;
maxIters=1000000;
c = [254661,6025044,8947555,8500000,6187994];
sFilename='res7c.csv';
xM=2^64-1;
bRound=true;
nDigits = 10;
doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)

% --------------------------------------------------------------------------

system('shutdown /s')
```

Listing 3.4. The source code of file do2.m.

Listing 3.5 shows the source code for file doAll2.m.

```
function doAll2(maxElems,maxIters,c,sFilename,xM,bRound,nDigits)
global gmaxElems
global M
global doRounding
global numDigits

gmaxElems = maxElems;
M = xM;
doRounding = bRound;
numDigits = nDigits;
resMat=zeros(maxIters,7);

for i=1:maxIters
  [factor,Xrnd] = rng997Gen2(c);
  resMat(i,1) = factor;
  resMat(i,2:6) = round(c,0);
  resMat(i,7) = Xrnd;

  fprintf("Itr: %d, Factor=%f, [", i, factor);
  fprintf(" %d,", resMat(i,2:6));
  fprintf("%d]\n", Xrnd);

end

fprintf("\n\n");
resMat = sortrows(resMat,1);

beep on;
for i=1:3
  beep;
  pause(1);
end

fprintf("Entire result matrix written to file %s\n", sFilename)
fh = fopen(sFilename, "w");
fprintf(fh,"Factor,IX,A0,A1,A2,A3,Xrnd\n");
```

```
for i=1:maxIters
  fprintf(fh,"%f,", resMat(i,1));
  for j=2:6
      fprintf(fh,"%d,", resMat(i,j));
  end
  fprintf(fh,"%d\n", resMat(i,7));
end
fclose(fh);
% T1 = array2table(resMat);
% T1.Properties.VariableNames(1:7) = {"Factor" "IX," "A0,","A1", "A2" , "A3",
"Xrnd");
% writetable(T1, sFilename, "Sheet,%d\n" sSheetName);
% pause(10);
% c = cell(17,2);


fprintf("------------------------------------------------------\n\n");
end
```

Listing 3.5. The source code of file doAll2.m.

Listing 3.6 shows the source code for file rng997Gen2.m.

```
function [factor,Xrnd] = rng997Gen2(c)
%UNTITLED2 Summary of this function goes here
  global gmaxElems
  global M
  global doRounding
  global numDigits

  maxElems = gmaxElems;
  c = round(c, 0);
  ix = zeros(4,1);
  ix(1) = round(rand*c(1),0);
  Xrnd = ix(1);

  a0 = c(2);
  a1 = c(3);
  a2 = c(4);
  a3 = c(5);
  ix(2) = mod(a0+a1*ix(1),M);
  ix(3) = mod(a0+a1*ix(1)^2+a2*ix(2),M);
  x=zeros(maxElems,1);
  for i=1:maxElems
    ix(4) = mod(a0+a1*ix(1)^3+a2*ix(2)^2+a3*ix(3),M);
    x(i) = ix(4)/M;
    ix(1:3) = ix(2:4);
  end
  if doRounding, x = round(x,numDigits); end
  factor=calcFactor(x,false);
  if isnan(factor), factor=65535; end
end

function x = frac(x)
  x=mod(x,1);
end
```

```
function factor = calcFactor(x, bShowResults)
% Calculate the factor statistic for the array of random nnumbers x.

  if nargin < 2, bShowResults = false; end
  maxElems=length(x);
  meanx=mean(x);
  sdevx=std(x);
  % get the first 100 autocorrelation values
  acArr=autocorrArr(x,1,100);
  % calculate the chisquare for the 10-bin histogram
  numBins=10;
  expval=maxElems/numBins;
  [N1,ev1]=histcounts(x,numBins);
  chiSq10=sum((N1-expval).^2/expval);
  numBins=20;
  expval=maxElems/numBins;
  [N2,ev2]=histcounts(x,numBins);
  chiSq20=sum((N2-expval).^2/expval);
  numBins=20;
  [N3,ev3]=histcounts(acArr,numBins);
  ev3c=ev3(2:length(ev3));
  autoCorrSum = sum(dot(N3,abs(ev3c)));
  chsStat=chs(x);
  [Kplus,Kminus]=KStest(x);
  factor = 1000*(abs(meanx-0.5)+abs(sdevx-1/sqrt(12)))+100*(max(acArr)-
min(acArr))+100*autoCorrSum+chiSq10+chiSq20/2;
  factor = factor + 10*chsStat + 10*(Kplus + Kminus);
  if bShowResults
    fprintf('Mean = %g\nSdev = %g\n', meanx, sdevx);
    fprintf('Min = %g\nMax = %g\n', min(x), max(x));
    fprintf('Max lags = 100\n');
    fprintf('Auto correlation array\n');
    disp(acArr');
    fprintf('10-Bin Histogram\n');
    disp(N1); disp(ev1);
    fprintf('Chi-Sqr10 = %g\n', chiSq10);
    fprintf('20-Bin Histogram\n');
    disp(N2); disp(ev2);
    fprintf('Chi-Sqr20 = %g\n', chiSq20);
    fprintf('20-Bin Autocorrelation Histogram\n');
    disp(N3); disp(ev3);
    fprintf('Sum autocorrel product = %g\n', autoCorrSum);
    fprintf('Change of sign stat = %g\n', chsStat);
    fprintf('K+ = %g and K- = %g\n', Kplus, Kminus);
    fprintf('Factor = %g\n', factor);
  end
end

function acArr=autocorrArr(xdata,fromLag,toLag)

numLags=toLag-fromLag+1;
acArr=zeros(numLags,1);
j=1;
for i=fromLag:toLag
  acArr(j)=autocor(xdata,i);
  j=j+1;
end
```

```
end

function res = autocor(xdata,lag)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
maxElems=length(xdata);
res=corrcoef(xdata(1:maxElems-lag),xdata(lag+1:maxElems));
res=res(1,2);
end

function sumx=chs(x)
% Function CHS calculates the change of sign (between subsequent random
% numbers) moment. The function counts the number of consequtive positive
% abd negative changes of sign. The last nested loop calculates the
% statistic returned by this function. This value is the sum of:
%
%  sum = sum of difference(count,:) * count / difference(1,:)
%
% Keeping in mind that difference(1,:) is a good value that counts the
% sign flips that happpens one neighbor down. The values for
% difference(n,:) for n>1 are not desirable. The smaller, the better. The
% value difference(2,:) is the number of sign flips that occur
% two neighbors down. The value difference(3,:) is the number of sign flips
% that occur three neighbors down, and so on.

  n=length(x);
  nby2=fix(n/2);
  Diff=zeros(nby2,2);
  countPos=0;
  countNeg=0;
  s1=sign(x(2)-x(1));
  if s1>0
    bIsPos=true;
    countPos=1;
  else
    bIsPos=false;
    countNeg=1;
  end

  for i=3:n
    s2=sign(x(i)-x(i-1));
    % was positive and is still positive
    if s2>0 && bIsPos
      countPos=countPos+1;
    % was negative and is now positive
    elseif s2>0 && ~bIsPos
      bIsPos=true;
      countPos=1;
      Diff(countNeg,2)=Diff(countNeg,2)+1;
      countNeg=0;
    % was negative and is still negative
    elseif s2<0 && ~bIsPos
      countNeg=countNeg+1;
    % was positive is and is now negative
    elseif s2<0 && bIsPos
      bIsPos=false;
      countNeg=1;
      Diff(countPos,1)=Diff(countPos,1)+1;
      countPos=0;
    end
  end

  if s2>0
    if countPos>0, Diff(countPos,1)=Diff(countPos,1)+1; end
  else
    if countNeg>0, Diff(countNeg,2)=Diff(countNeg,2)+1; end
  end

  i=2:nby2;
  d=Diff(2:nby2,:);
  sumx=0;
  for j=1:2
    sumx = sumx + dot(d(:,j),i)/Diff(1,j);
  end
end

function [Kplus,Kminus]=KStest(x)
  x=sort(x);
  n=length(x);
  diffMaxPlus=-1e+99;
  diffMaxMinus=-1e+99;
  i=1;
  for xv=0.001:.001:1
    F=xv;
    while x(i)<=xv && i<n
      i=i+1;
    end
    Fn=1;
    if i<n, Fn=(i-1)/n; end
    diff=Fn-F;
    if diff>diffMaxPlus, diffMaxPlus=diff; end
    diff=-diff;
    if diff>diffMaxMinus, diffMaxMinus=diff; end
  end
  Kplus=sqrt(n)*diffMaxPlus;
  Kminus=sqrt(n)*diffMaxMinus;
end
```

Listing 3.6. The source code of file rng997Gen2.m.

Tables 3.3a and 3.3b show the results of the penalty factor statistics for the different versions of the algorithm.

	res1c	res2c	res3c	res4c
	Random Seed	Random Seed	Random Seed	Random Seed
Mean	145.606095	144.450418	147.628115	147.753821
Sdev	11.2664655	13.3173712	11.6545544	11.6881105
Min	109.927226	130.324851	121.761742	110.840027
Max	185.542601	177.400176	174.791853	201.133513
Range	75.615375	47.075325	53.030111	90.293486
Count	1000000	1000000	1000000	1000000
Conf	0.02525269	0.02984959	0.02612255	0.02619776
CI Upper	145.631348	144.480268	147.654238	147.780019

CI Lower	145.580842	144.420569	147.601992	147.727623

Table 3.3a. The random-seed results for the penalty factor statistics.

	res5c	res6c	res7c
	Random Seed	Random Seed	Random Seed
Mean	147.9012	Failed!	Failed
Sdev	11.8870851		
Min	108.298396		
Max	201.418133		
Range	93.119737		
Count	1000000		
Conf	0.02664375		
CI Upper	147.927844		
CI Lower	147.874556		

Table 3.3b. The random-seed results for the penalty factor statistics.

The column titled res2c in Table 3.3a has the lowest upper mean value. The best modified power method equation is:

```
M = 2^24-1
a0 = 34876
a1 = 9754
a2 = 45847
a3 = 29574
 ix(1) = round(rand*11, 0);
 ix(2) = a0+a1*ix(1) mod M
 for i=1 to maximum number of random numbers
   ix(3) = a0+a1*ix(1)^2 +a2*ix(2) mod M
   x(i) = ix(3)/M
   ix(1:2) = ix(2:3)
 end                                                    (3.2)
```

The value x(i) is the uniform random number generated in the range of 0 to 1 (excluded) in each iteration.

DOCUMENT HISTORY

Date	***Version***	***Comments***
2/10/2023	1.00.00	Initial release.