Project 997 PRNGs Part 5
Power Method Algorithms
By
Namir C. Shammas

1/ Introduction

This study looks at a new PRNG algorithm that generates new random numbers by raising a previous random number to the power of a second previous one.

2/ Power Method Algorithms (Version PSO)

The power method algorithm is defined as:

```
  x(1) = round(seed1,numDigits);
  x(2) = round(seed2,numDigits);
  for i=3 to maximum number of random numbers
    x(i)= (a+x(i-1))^(b+x(i-2)) mod 1
  end
end                                                    (2.1)
```

This section looks at optimizing the values of *a* and *b* 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 two previous random values x(i-1) and x(i-2). Each of the last two random numbers are shifted by the values of *a* and *b*, respectively.

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:

```
fpi = pi - 3;

maxElems=10000;
maxIters=100;
lb=[0 0 10*fpi 1];
ub=[0.999999 0.999999 30*fpi 2];
sFilename='res1.xlsb';
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

maxElems=10000;
maxIters=100;
lb=[0 0 10*fpi 1];
ub=[0.999999 0.999999 50*fpi 5];
sFilename='res2.xlsb';
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

maxElems=10000;
maxIters=100;
lb=[0 0 10*fpi 1];
ub=[0.999999 0.999999 75*fpi 7];
sFilename='res3.xlsb';
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

maxElems=10000;
maxIters=100;
lb=[0 0 10*fpi 1];
ub=[0.999999 0.999999 100*fpi 10];
sFilename='res4.xlsb';
```



```
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

maxElems=10000;
maxIters=100;
lb=[0 0 10*fpi 1];
ub=[0.999999 0.999999 200*fpi 13];
sFilename='res5.xlsb';
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

maxElems=10000;
maxIters=100;
lb=[0 0 10*fpi 1];
ub=[0.999999 0.999999 250*fpi 17];
sFilename='res6.xlsb';
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
%
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

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

r = 0.15;
maxElems=10000;
maxIters=100;
x = [0.637057247,0.331621119,2.819143598,1.637757676];
lb=(1-r)*x;
ub=(1+r)*x;
sFilename='res1b.xlsb';
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

maxElems=10000;
maxIters=100;
x=[0,0.935849285,7.07963268,1.73136863];
lb=(1-r)*x;
ub=(1+r)*x;
sFilename='res2b.xlsb';
sSheetName='Sheet1';
```



```
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

%% SPECIAL
maxElems=10000;
maxIters=100;
x=[0.999999,0.456863931,3.643060647,2.694509227];
lb=(1-r)*x;
ub=(1+r)*x;
sFilename='res2b2.xlsb';
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)


maxElems=10000;
maxIters=100;
x=[0.676996312,0.03460785,9.036262607,5.047168771];
lb=(1-r)*x;
ub=(1+r)*x;
sFilename='res3b.xlsb';
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

maxElems=10000;
maxIters=100;
x=[0.455192345,0.334154966,2.238967977,10];
lb=(1-r)*x;
ub=(1+r)*x;
sFilename='res4b.xlsb';
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

maxElems=10000;
maxIters=100;
x=[0.201085442,0.01137226,13.90904759,2.167082095];
lb=(1-r)*x;
ub=(1+r)*x;
sFilename='res5b.xlsb';
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
```



```
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

maxElems=10000;
maxIters=100;
x=[0.589462435,0.661218353,31.65987909,5.548695472];
lb=(1-r)*x;
ub=(1+r)*x;
sFilename='res6b.xlsb';
sSheetName='Sheet1';
POSpopsize=250;
POSmaxIters=100;
nDigits = 10;
doAll(maxElems,maxIters,lb,ub,sFilename,sSheetName,POSpopsize,POSmaxIters,nDi
gits)

% 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,nDi
gits)
  global gmaxElems
  global numDigits
  global bestFactor

  gmaxElems = maxElems;
  numDigits = nDigits;
  bResetResMat = true;
  % output file exists?
  if isfile(sFilename)
    resMat = readmatrix(sFilename,"Sheet", sSheetName,"Range","A2:E251");
    [nrows,~] = size(resMat);
    if nrows > 0
      for nStart=1:nrows
        if resMat(nStart,1)>1e+99, break; end
      end
      if nStart <= nrows, bResetResMat = false; end
    end
  end

  if bResetResMat
    resMat=2e+99+zeros(maxIters,5);
    nStart = 1;
  end
```



```
  options =
optimoptions('particleswarm','SwarmSize',POSpopsize,'Display','off','MaxItera
tions',POSmaxIters,'FunctionTolerance',0.01);
  for i=nStart:maxIters
    bestFactor = 1e+99;
    x = particleswarm(@rng997Gen1,length(lb),lb,ub,options);
    x = round(x, nDigits);
    resMat(i,1) = bestFactor;
    resMat(i,2:5) = x;

    fprintf("itr = %i, Factor = %f, X= [", i , bestFactor);
    fprintf("%f, ", resMat(i,2:5));
    fprintf("]\n");

    resMat = sortrows(resMat,1);

    T1 = array2table(resMat);
    T1.Properties.VariableNames(1:5) = {'Factor', 'Xrnd1', 'Xrnd2', 'A',
'B'};
    writetable(T1, sFilename, "Sheet", sSheetName);
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)
T1 = array2table(resMat);
    T1.Properties.VariableNames(1:5) = {'Factor', 'Xrnd1', 'Xrnd2','A', 'B'};
writetable(T1, sFilename, "Sheet", sSheetName);
c = cell(13,2);
c(1,1:2) = {'Max Elements', maxElems};
c(2,1:2) = {'Max Iters', maxIters};
c(3,1:2) = {'Xrnd1low', lb(1)};
c(4,1:2) = {'Xrnd1hi', ub(1)};
c(5,1:2) = {'Xrnd2low', lb(2)};
c(6,1:2) = {'Xrnd2hi', ub(2)};
c(7,1:2) = {'Alow', lb(3)};
c(8,1:2) = {'Ahi', ub(3)};
c(9,1:2) = {'Blow', lb(4)};
c(10,1:2) = {'Bhi', ub(4)};
c(11,1:2) = {'PopSize', POSpopsize};
c(12,1:2) = {'PopMaxIters', POSmaxIters};
c(13,1:2) = {'Rounded', nDigits};

T2 = cell2table(c);
writetable(T2, sFilename, "Sheet", "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 numDigits
  global bestFactor

  maxElems = gmaxElems;
%   rng('shuffle','twister');
  x=zeros(maxElems,1);
  x(1) = round(c(1),numDigits);
  x(2) = round(c(2),numDigits);
  a = c(3);
  b = c(4);
  for i=3:maxElems
    x(i)= round(mod((a+x(i-1))^(b+x(i-2)),1), numDigits);
  end
  factor=calcFactor(x,false);
  if isnan(factor), factor=65535; end
  if factor < bestFactor
      bestFactor = factor;
      bestX = x;
  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
	Wide Range	Wide Range	Wide Range
Mean	112.490012	112.611152	112.026735
Sdev	2.37552758	2.48666761	2.54781865
Min	106.615512	105.773631	105.041068
Max	117.16476	117.023231	117.760491
Range	10.5492481	11.2495994	12.7194233
Count	100	100	100
Conf Int	0.5324514	0.55736236	0.57106877
Upper CI	113.022464	113.168514	112.597804
Lower CI	111.957561	112.05379	111.455666
Xrnd1	0.63705725	0	0.67699631
Xrnd2	0.33162112	0.93584928	0.03460785
A	2.8191436	7.07963268	9.03626261
B	1.63775768	1.73136863	5.04716877
Max Elements	10000	10000	10000
Max Iters	100	100	100
Xrnd1low	0	0	0
Xrnd1hi	0.999999	0.999999	0.999999
Xrnd2low	0	0	0
Xrnd2hi	0.999999	0.999999	0.999999
Alow	1.41592654	1.41592654	1.41592654
Ahi	4.24777961	7.07963268	10.619449
Blow	1	1	1
Bhi	2	5	7
PopSize	250	250	250
PopMaxIters	100	100	100
Rounded	10	10	10

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

	res4	res5	res6

	Wide Range	Wide Range	Wide Range
Mean	112.537693	112.482995	112.349234
Sdev	2.30461538	2.80482136	2.51858343
Min	104.958512	104.554524	102.921788
Max	117.035853	118.964232	117.57261
Range	12.0773407	14.4097078	14.6508215
Count	100	100	100
Conf Int	0.51655712	0.62867343	0.56451598
Upper CI	113.05425	113.111669	112.91375
Lower CI	112.021136	111.854322	111.784718
Xrnd1	0.45519234	0.20108544	0.58946244
Xrnd2	0.33415497	0.01137226	0.66121835
A	2.23896798	13.9090476	31.6598791
B	10	2.16708209	5.54869547
Max Elements	10000	10000	10000
Max Iters	100	100	100
Xrnd1low	0	0	0
Xrnd1hi	0.999999	0.999999	0.999999
Xrnd2low	0	0	0
Xrnd2hi	0.999999	0.999999	0.999999
Alow	1.41592654	1.41592654	1.41592654
Ahi	14.1592654	28.3185307	35.3981634
Blow	1	1	1
Bhi	10	13	17
PopSize	250	250	250
PopMaxIters	100	100	100
Rounded	10	10	10

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

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

	res1b	res2b	res2b2	res3b
	Narrow Range	Narrow Range	Narrow Range	Narrow Range
Mean	111.769826	112.285782	112.273255	112.1121
Sdev	3.09686431	2.48249541	2.313912	2.34414936
Min	102.128054	103.009792	104.377288	104.171572
Max	115.793058	117.232464	115.879002	117.180585
Range	13.6650037	14.2226722	11.5017138	13.0090127
Count	100	100	100	100
Conf Int	0.69413201	0.5564272	0.51864087	0.52541828
Upper CI	112.463958	112.842209	112.791896	112.637518
Lower CI	111.075694	111.729355	111.754615	111.586681
Xrnd1	0.73261583	0	0.89840083	0.69298258
Xrnd2	0.3751077	0.79547189	0.43720202	0.03811965
A	3.15006659	7.02283262	3.67326277	7.7773443
B	1.49188412	1.7077561	2.47945677	5.0062391
Max Elements	10000	10000	10000	10000

Max Iters	100	100	100	100
Xrnd1low	0.54149866	0	0.84999915	0.57544687
Xrnd1hi	0.73261583	0	1.14999885	0.77854576
Xrnd2low	0.28187795	0.79547189	0.38833434	0.02941667
Xrnd2hi	0.38136429	1.07622668	0.52539352	0.03979903
Alow	2.39627206	6.01768778	3.09660155	7.68082322
Ahi	3.24201514	8.14157758	4.18951974	10.391702
Blow	1.39209402	1.47166334	2.29033284	4.29009346
Bhi	1.88342133	1.99107392	3.09868561	5.80424409
PopSize	250	250	250	250
PopMaxIters	100	100	100	100
Rounded	10	10	10	10

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

	res4b	res5b	res6b
	Narrow Range	Narrow Range	Narrow Range
Mean	112.34063	111.959281	112.152283
Sdev	2.10773283	2.81518346	2.34337301
Min	105.670297	102.963422	106.715688
Max	116.532166	117.691144	116.203769
Range	10.8618688	14.7277213	9.48808141
Count	100	100	100
Conf Int	0.47242781	0.63099599	0.52524426
Upper CI	112.813058	112.590277	112.677527
Lower CI	111.868202	111.328285	111.627039
Xrnd1	0.4609445	0.19427483	0.50253722
Xrnd2	0.29611665	0.00991394	0.7003228
A	2.57481317	14.3245802	35.955811
B	8.5	2.05239463	6.11810893
Max Elements	10000	10000	10000
Max Iters	100	100	100
Xrnd1low	0.38691349	0.17092263	0.50104307
Xrnd1hi	0.5234712	0.23124826	0.6778818
Xrnd2low	0.28403172	0.00966642	0.5620356
Xrnd2hi	0.38427821	0.0130781	0.76040111
Alow	1.90312278	11.8226905	26.9108972
Ahi	2.57481317	15.9954047	36.408861
Blow	8.5	1.84201978	4.71639115
Bhi	11.5	2.49214441	6.38099979
PopSize	250	250	250
PopMaxIters	100	100	100
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=[0.732615834,0.375107697,3.150066592,1.491884123];
sFilename='res1c.xlsb';
sSheetName='Sheet1';
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,sSheetName,nDigits)

maxElems=10000;
maxIters=1000000;
c=[0,0.795471892,7.022832617,1.707756098];
sFilename='res2c.xlsb';
sSheetName='Sheet1';
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,sSheetName,nDigits)

maxElems=10000;
maxIters=1000000;
c=[0.898400826,0.437202017,3.673262768,2.479456774];
sFilename='res2c2.xlsb';
sSheetName='Sheet1';
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,sSheetName,nDigits)

maxElems=10000;
maxIters=1000000;
c=[0.692982576,0.038119647,7.777344303,5.006239098];
sFilename='res3c.xlsb';
sSheetName='Sheet1';
nDigits = 10;
% doAll2(maxElems,maxIters,c,sFilename,sSheetName,nDigits)

maxElems=10000;
maxIters=1000000;
c=[0.460944497,0.296116647,2.574813174,8.5];
sFilename='res4c.xlsb';
sSheetName='Sheet1';
nDigits = 10;
doAll2(maxElems,maxIters,c,sFilename,sSheetName,nDigits)

maxElems=10000;
maxIters=1000000;
c=[0.194274833,0.009913935,14.32458021,2.052394632];
sFilename='res5c.xlsb';
sSheetName='Sheet1';
nDigits = 10;
doAll2(maxElems,maxIters,c,sFilename,sSheetName,nDigits)

maxElems=10000;
maxIters=1000000;
c=[0.502537219,0.700322795,35.95581097,6.118108926];
sFilename='res6c.xlsb';
sSheetName='Sheet1';
nDigits = 10;
doAll2(maxElems,maxIters,c,sFilename,sSheetName,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,sSheetName,nDigits)
  global gmaxElems
  global numDigits

  gmaxElems = maxElems;
  numDigits = nDigits;
  bResetResMat = true;
  % output file exists?
  if isfile(sFilename)
    resMat = readmatrix(sFilename,"Sheet", sSheetName,"Range","A2:E251");
    [nrows,~] = size(resMat);
    if nrows > 0
      for nStart=1:nrows
        if resMat(nStart,1)>1e+99, break; end
      end
      if nStart <= nrows, bResetResMat = false; end
    end
  end

  if bResetResMat
    resMat=2e+99+zeros(maxIters,5);
    nStart = 1;
  end
  rng('shuffle','twister');
  c0 = c;
  for i=1:maxIters
    c(1) = round(c0(1)*rand, nDigits);
    c(2) = round(c0(2)*rand, nDigits);
    factor = rng997Gen2(c);
    resMat(i,1) = factor;
    resMat(i,2:5) = c;

    fprintf("itr = %i, Factor = %f, X= [", i , factor);
    fprintf("%f, ", resMat(i,2:5));
    fprintf("]\n");
  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)
T1 = array2table(resMat);
    T1.Properties.VariableNames(1:5) = {'Factor', 'Xrnd1', 'Xrnd2','A', 'B'};
writetable(T1, sFilename, "Sheet", sSheetName);

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 = rng997Gen2(c)
%UNTITLED2 Summary of this function goes here
  global gmaxElems
  global numDigits

  maxElems = gmaxElems;
%   rng('shuffle','twister');
  x=zeros(maxElems,1);
  x(1) = round(c(1),numDigits);
  x(2) = round(c(2),numDigits);
  a = c(3);
  b = c(4);
  for i=3:maxElems
    x(i)= round(mod((a+x(i-1))^(b+x(i-2)),1), 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	res2c2	res3c
	Random Seed	Random Seed	Random Seed	Random Seed
Mean	147.904033	147.787421	147.758832	147.757349
Sdev	11.8364159	11.7680366	11.7714331	11.7857344
Min	105.075218	103.382045	105.916181	106.157884
Max	223.003481	230.702177	227.267077	224.584741
Range	117.928263	127.320132	121.350896	118.426858
Count	1000000	1000000	1000000	1000000
Conf Int	0.02653017	0.02637691	0.02638452	0.02641658
Upper CI	147.930563	147.813798	147.785217	147.783765
Lower CI	147.877503	147.761044	147.732448	147.730932
Xrnd1	0.07023735	0	0.85079835	0.05247702
Xrnd2	0.17066908	0.62394632	0.22196813	0.01271283
A	3.15006659	7.02283262	3.67326277	7.7773443
B	1.49188412	1.7077561	2.47945677	5.0062391
Max Elements	10000	10000	10000	10000
Max Iters	1000000	1000000	1000000	1000000

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

	res4c	res5c	res6c
	Random Seed	Random Seed	Random Seed
Mean	147.765576	147.754236	147.731201
Sdev	11.7742304	11.7635829	20.2159284
Min	104.849541	103.767218	105.380176
Max	236.352359	220.719418	15416.1494
Range	131.502817	116.9522	15310.7693
Count	1000000	1000000	1000000
Conf Int	0.02639079	0.02636693	0.04531204
Upper CI	147.791967	147.780603	147.776513
Lower CI	147.739185	147.727869	147.685889
Xrnd1	0.0866711	0.1901164	0.48966879
Xrnd2	0.23849025	0.00081765	0.13448617
A	2.57481317	14.3245802	35.955811

B	8.5	2.05239463	6.11810893
Max Elements	10000	10000	10000
Max Iters	1000000	1000000	1000000

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

The column titled res6c in Table 2.3b has the lowest upper mean value. The best modified power method equation is:

```
  x(1) = round(0.4896687856,numDigits);
  x(2) = round(0.1344861661,numDigits);
  for i=3 to maximum number of random numbers
    x(i)= round((35.95581097+x(i-1))^(6.118108926+x(i-2)) mod 1, numDigits);
  end
end                                                                    (2.2)
```

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

3/PROLOG USING OTHER OPTIMIZATION METHODS

This section briefly presents final results to equations similar to 2.2 using different optimization Methods.

Using one of my own implementations of the PSO algorithm I get the best result as:

```
  x(1) = round(0.4983864824     ,numDigits);
  x(2) = round(0.1864367385,numDigits);
  for i=3 to maximum number of random numbers
    x(i)= round((18.85824905+x(i-1))^(6.589921696+x(i-2)) mod 1, numDigits);
  end
end                                                                    (3.1)
```

Using simple search method, I get the best result as:

```
  x(1) = round(0.0429471828,numDigits);
  x(2) = round(0.5705651308,numDigits);
  for i=3 to maximum number of random numbers
    x(i)= round((6.941015832+x(i-1))^( 3.376832468+x(i-2)) mod 1, numDigits);
  end
```



```
end                                                                    (3.2)
```

Using the Social Media Optimization (SMO) algorithm I get the best result as:

```
  x(1) = round(0.0838327111,numDigits);
  x(2) = round(0.1861985732,numDigits);
  for i=3 to maximum number of random numbers
    x(i)= round((8.482811324+x(i-1))^(2.571380759+x(i-2)) mod 1, numDigits);
  end
end                                                                    (3.2)
```

DOCUMENT HISTORY

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

