S111503
Ran Wang
Discrete Mathematics homework03

01415 Computational Tools for Discrete Mathematics

Ran Wan
S111503
Matlab code is in collaboration with Carlos Monteserin Sanchez, s141560

S111503
Ran Wang
Discrete Mathematics homework03

Question 1. stream cipher:
A stream cipher consisting of a LFSR and a nonlinear filter function. According to exercise 2, a primitive polynomial $f(x) = x^6 + x^4 + x^3 + x + 1$ in $F_2[x]$ is given, which can be converted to the LFSR :

$$S_{n+6} = S_{n+4} + S_{n+3} + S_{n+1} + S_n \qquad n=0,1,\ldots$$

Since the polynomial is primitive, any non {0 0 0 0 0 0} initial state will produce a LRS with maximal period of $2^6-1=63$ (homogeneous).

The successive states for the initial vector is: (see code in Appendix 1)
0 1 1 0 1 0| 0 1 1 0 1 1 0 0 0 1 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 0

And according to exercise 3, the first 10 corresponding bits in keystream output is:
0 1 1 1 0 0 1 1 0 1

The nonlinear equations representing this stream cipher is constructed by following the nonlinear function from exercise 3. Since the initial state of the LFSR is secret, we represent it with x0,x1,x2,x3,x4,x5, and as the recurrence goes on, we have to introduce some new polynomials to represent the bits and plug these bits as variables into the nonlinear filter function to generate one bit each time, which yields the equations:

$a(x) = x0 + x1 + x3 + x4$ ~~(1.6)~~
$b(x) = x1 + x2 + x4 + x5$
$c(x) = x0 + x1 + x2 + x4 + x5$
$d(x) = x0 + x2 + x4 + x5$
$e(x) = x0 + x4 + x5$ ~~(1.7)~~
$f(x) = x0 + x3 + x4 + x5$
$F(S_1) = [(x3x4) + x4x5 + x3x5 + a(x)] \bmod 2 = 0$
$F(S2) =[x4x5 + (x4 + x5)a(x) + b(x)] \bmod 2 = 1$
$F(S3) =[x5a(x) + (x5 + a(x))b(x) + c(x)] \bmod 2 = 1$
$F(S4) =[a(x)b(x) + (a(x) + b(x))c(x) + d(x)] \bmod 2 = 1$
$F(S5) =[b(x)c(x) + (b(x) + c(x))d(x) + e(x)] \bmod 2 = 0$
$F(S6) =[c(x)d(x) + (c(x) + d(x))e(x) + g(x)] \bmod 2 = 0$

The idea now is calculate the keystream for the 63 possible state vectors1 and look for a coincidence with the 6 first digits of (1.3). We find only one coincidence :
[0 1 1 0 1 0]
It turns out that if we have access to the Stream cipher operations and keystream we can obtain the secret seed s0.

There are step by step hand calculations and Matlab code in the Appendix(1).

Question 2. Low order bits of LCGs.
(a). Suppose that $X_0 = 1$. Compute the 2-bit sequence Y0; Y1; Y2; : : : and its period.
A linear congruential sequence with parameters $(m; a; c; X_0)$ has the form:
$X_{n+1} = (aX_n + c) \bmod m$ for $n >=0$
And the following value is given:

S111503
Ran Wang
Discrete Mathematics homework03
a=1103515245; c=12345; m=2^{31}; x0=1
According to Theorem 1 (slide 13, Day08), we can prove that the linear congruential sequence with parameters (m; a; c;X_0) has period length m=2^{31}, because:

1. c is relatively prime to m; True, as c is odd, m is even and have only one prime divisor 2, so gcd(c,m)=1.
2. b = a - 1 is a multiple of p for every prime p dividing m; True, b is even.
3. b should be a multiple of 4. True, because m= 2^{29}* 2^2 , so b =0 mod 4.

So we can generate 2 random bits in iterations by using
X_{n+1} = (aX_n + c) mod m for n >=0
Y_n = X_n mod 2^2 for n >=0
The remainders can only be {0,1,2,3}, which in binary form are 00,01,10,11.
We use iteration 2^{15} = 32768, the first 10 computed linear congruential sequence X_n are:
1
1103527590
377401600
333417792
314102912
611429056
1995203584
18793472
1909564472
295447552
X_n has maximal period m, so the sequence contains all numbers from 0 to m − 1.

And Y_n are:
1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...
We observe that the majority of Y_n are 0, which means the corresponding X_n are multiple of 4. This seriously affect the randomness of the sequence.

The track of output, the 1st output is 1, the 2nd is 2, the 450th is 1, the 516th is 1, the 522th is 1, and so forth:
1 1 2 2 450 1 516 1 522 1 832 1 864 1 951 1 1309 1 1565 1 1627 1 1772 1 1821 1 2554 1 ...
We don't see that Y_n is periodic when we use iteration 2^{15} = 32769 that is the maximum my computer can reach. But according to Theorem 1, its period should be 2^{31} . In this case, LCG uses the two least significant bits to produce random bit, lowest order of bit oscillates at each step, thus the bits produced are not equally random.

(b) Which period do you obtain in general? Compare this to using the 6-bit LFSR from classroom exercise 2 to generate the random bits.

S111503
Ran Wang
Discrete Mathematics homework03

Using the 6-bit LFSR from exercise 2 to generate the random bits gives 63 period, namely its maximal period sequence $2^6 - 1$. The constraint $X < m$ makes it only take the 2 lowest bits. So both this LFSR case and the LCG case in question (a) have a period as their maximal period.

(c) Can you propose a solution to x this problem with the LCG?

In general, lower-order bits of the LCG generated sequence have a far shorter period than the sequence to the big power of 2. LCG uses truncation technique cutting out the low-order bits to produce statistically better sequence. So instead of taking 2 lowest orders (mod 4), the solution could be to take two bits of sequence X from different positions among its 31 bits during each iteration, and then test the statistic of the produced sequence, we see that (figure 1) before 20th and after the 4th bits(the bits below the red line), the LCG generates a good random sequence.

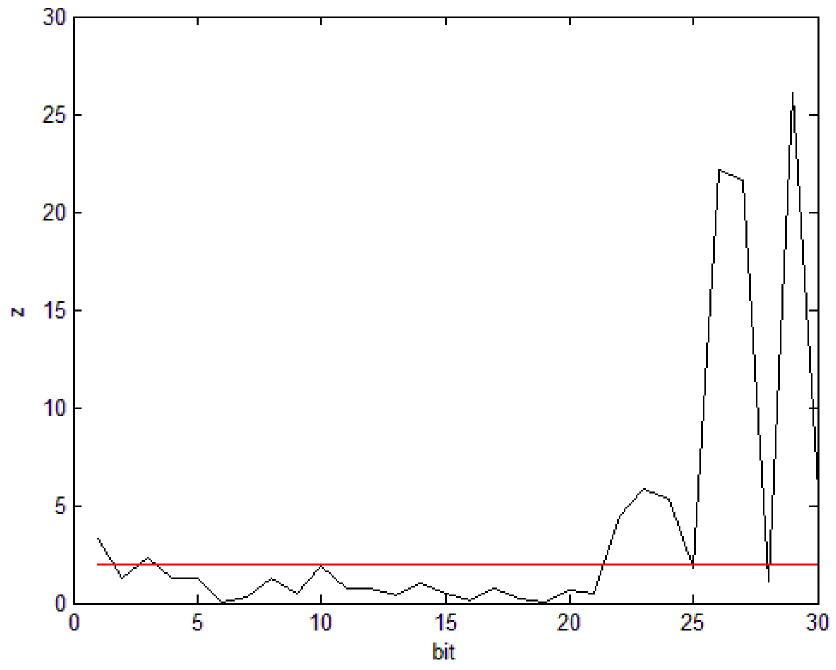


Figure 1, test of randomness, red line marks z = 1.96, the right most bit (30) is the lowest bit.

Appendix 1

```
% Question1

KS=[0 1 1 1 0 0 1 1 0 1 0];

% we implement the lhs of the system in lhsNLSYS.m, we can check if it is right
% by inserting the original initial state
s0=[0 1 1 0 1 0];
```

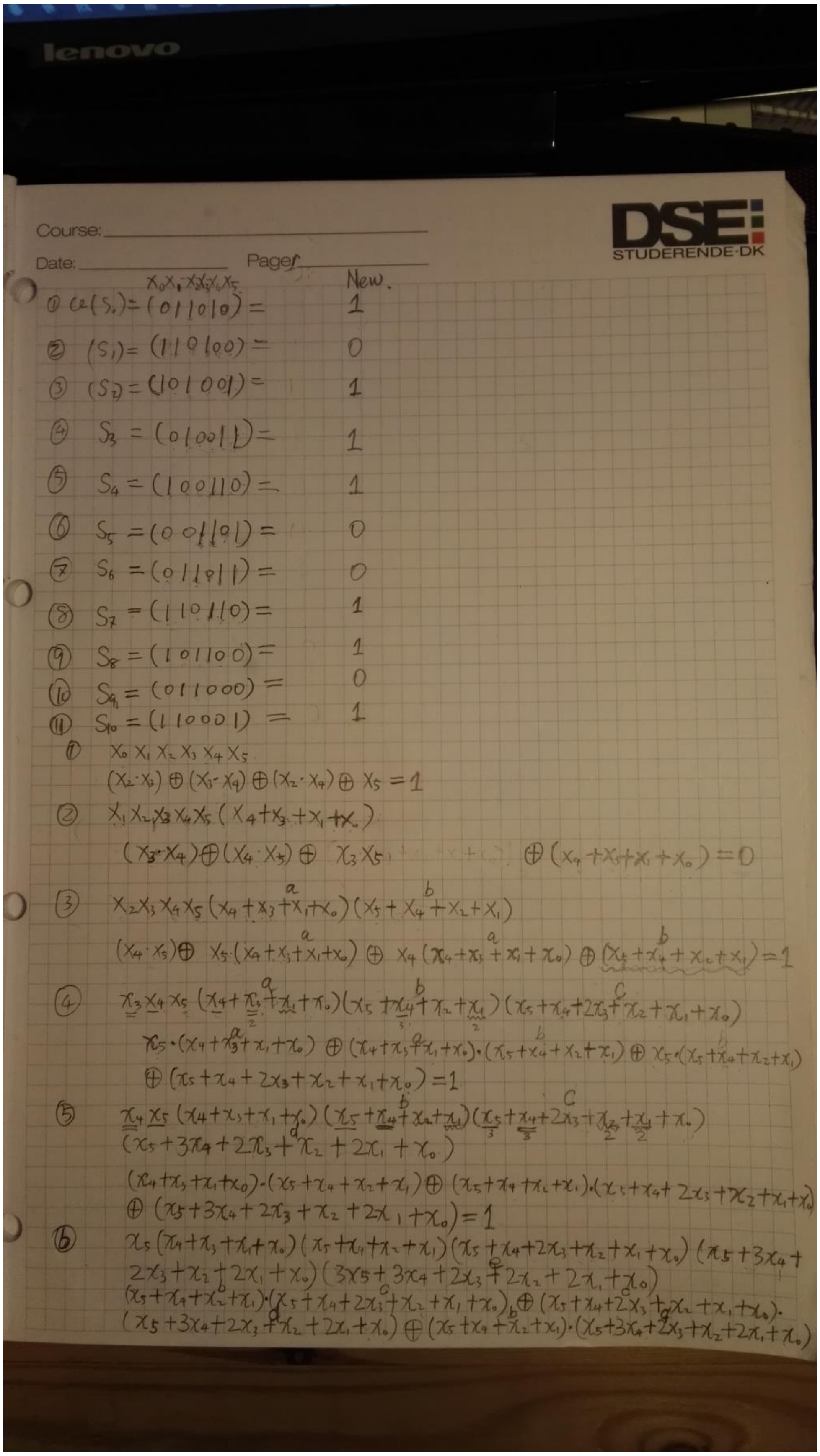
S111503
Ran Wang
Discrete Mathematics homework03

```
KS2=lhsNLSYS(s0);
% we can see KS2 = KS(1:6) :)
%Next thing is break the cipher we need to generate all the posible
%s0 and check which ones outcomes KS

% excluding [0 0 0 0 0 0] we can generate the 63 posibile s0's by using
% that the LRS is maximal period, and create all posible states.

%s0=[0 0 0 0 0 1]; %anyone is maximal
S=s0;
iterations=124

for i=0:iterations
    idx=i+1;
  next= S(idx) + S(idx+1) + S(idx+3)+ S(idx+4);
  next=mod(next,2);
  S=[S next];
end
%states
States=zeros(iterations,6);
KS3 =zeros(iterations,6);
coincidence=[];
cont=0;  cont1=0;
for i=1:iterations
 States(i,:)=S(1+cont:6+cont);
 KS3(i,:) = lhsNLSYS(States(i,:));
 % when i=1,the first 6 generated nonlinear
 cont=cont+1
 if (min(KS3(i,:)==KS(1:6))==1) %we have a coincidence
      cont1=cont1+1;
  %coincidence=[coincidence;States(i,:)];
  coincidence=[coincidence;States(i,:)]
 end

end

function eq = lhsNLSYS(x)

x0=x(1);x1=x(2);x2=x(3);
x3=x(4);x4=x(5);x5=x(6);

%auxiliar functions
a = x0 + x1 + x3 + x4;
b = x1 + x2 + x4 + x5;
c = x2 + x3 + x5 + a;
d = x3 + x4 + a + b;
e = x4 + x5 + b + c;
f = x5 + a + c + d;

%lhs of 6 equations

eq1 = x3 * x4 + x4*x5 + x3*x5 + a ; eq1 =mod(eq1,2);
eq2 = x4 * x5 + (x4+x5)*a + b; eq2 = mod(eq2,2);
eq3 = (x5+b) * a + x5*b + c; eq3=mod(eq3,2);
```

S111503
Ran Wang
Discrete Mathematics homework03

```
eq4 = a*b + (a+b)*c + d; eq4=mod(eq4,2);
eq5 = b * c + (b+c)*d + e; eq5=mod(eq5,2);
eq6 = c * d + (c+d)*e + f;  eq6=mod(eq6,2);

eq = [eq1 eq2 eq3 eq4 eq5 eq6];
```

S111503
Ran Wang
Discrete Mathematics homework03



Course:

Date: Page 1

	$x_0x_1x_2x_3x_4x_5$	New.
①	$e(S_0) = (011010) =$	1
②	$(S_1) = (110100) =$	0
③	$(S_2) = (101001) =$	1
④	$S_3 = (010011) =$	1
⑤	$S_4 = (100110) =$	1
⑥	$S_5 = (001101) =$	0
⑦	$S_6 = (011011) =$	0
⑧	$S_7 = (110110) =$	1
⑨	$S_8 = (101100) =$	1
⑩	$S_9 = (011000) =$	0
⑪	$S_{10} = (110001) =$	1

① $x_0\,x_1\,x_2\,x_3\,x_4\,x_5$

$(x_2 \cdot x_3) \oplus (x_3 \cdot x_4) \oplus (x_2 \cdot x_4) \oplus x_5 = 1$

② $x_1x_2x_3x_4x_5\,(x_4 + x_3 + x_1 + x_0)$

$(x_3 \cdot x_4) \oplus (x_4 \cdot x_5) \oplus x_3x_5 \oplus (x_4 + x_3 + x_1 + x_0) = 0$

③ $x_2x_3x_4x_5\,(x_4 + x_3 \overset{a}{+} x_1 + x_0)(x_5 + x_4 \overset{b}{+} x_2 + x_1)$

$(x_4 \cdot x_5) \oplus x_5 \cdot (x_4 + x_3 \overset{a}{+} x_1 + x_0) \oplus x_4 (x_4 + x_3 \overset{a}{+} x_1 + x_0) \oplus (x_5 + x_4 \overset{b}{+} x_2 + x_1) = 1$

④ $x_3x_4x_5\,(x_4 + x_3 \overset{a}{+} x_1 + x_0)(x_5 + x_4 \overset{b}{+} x_2 + x_1)(x_5 + x_4 + 2x_3 \overset{c}{+} x_2 + x_1 + x_0)$

$x_5 \cdot (x_4 + x_3 \overset{a}{+} x_1 + x_0) \oplus (x_4 + x_3 \overset{a}{+} x_1 + x_0) \cdot (x_5 + x_4 \overset{b}{+} x_2 + x_1) \oplus x_5 \cdot (x_5 + x_4 \overset{b}{+} x_2 + x_1)$

$\oplus (x_5 + x_4 + 2x_3 + x_2 + x_1 + x_0) = 1$

⑤ $x_4x_5\,(x_4 + x_3 + x_1 + x_0)(x_5 + x_4 \overset{b}{+} x_2 + x_1)(x_5 + x_4 + 2x_3 \overset{c}{+} x_2 + x_1 + x_0)$
$(x_5 + 3x_4 + 2x_3 \overset{d}{+} x_2 + 2x_1 + x_0)$

$(x_4 + x_3 + x_1 + x_0) \cdot (x_5 + x_4 + x_2 + x_1) \oplus (x_5 + x_4 + x_2 + x_1) \cdot (x_5 + x_4 + 2x_3 + x_2 + x_1 + x_0)$
$\oplus (x_5 + 3x_4 + 2x_3 + x_2 + 2x_1 + x_0) = 1$

⑥ $x_5 (x_4 + x_3 + x_1 + x_0)(x_5 + x_4 + x_2 + x_1)(x_5 + x_4 + 2x_3 + x_2 + x_1 + x_0)(x_5 + 3x_4 +$
$2x_3 + x_2 + 2x_1 + x_0)(3x_5 + 3x_4 + 2x_3 \overset{e}{+} 2x_2 + 2x_1 + x_0)$
$(x_5 + x_4 \overset{b}{+} x_2 + x_1) \cdot (x_5 + x_4 + 2x_3 \overset{c}{+} x_2 + x_1 + x_0) \oplus (x_5 + x_4 + 2x_3 \overset{c}{+} x_2 + x_1 + x_0) \cdot$
$(x_5 + 3x_4 + 2x_3 \overset{d}{+} x_2 + 2x_1 + x_0) \oplus (x_5 + x_4 + x_2 + x_1) \cdot (x_5 + 3x_4 + 2x_3 + x_2 + 2x_1 + x_0)$

S111503
Ran Wang
Discrete Mathematics homework03

Appendix 2

```
%% question 2. (a)
a=1103515245;
c=12345;
m=2^31;
iter=2^15;
%initial state of LCG
x0=1; %integer in [0,m-1]

%Arrays initialization
X=x0;
Y=mod(X,4);
BitSeq = m4tobinar(Y);
here=[];
for i=1:iter
%   if(mod(i,m/10)==0)
%      disp(horzcat(num2str(round(i/m*100)),'% done'));
%   end
  xcurrent=X(end);
  newX = mod(a*xcurrent+c,m);
  newY = mod(newX,4);
  newBitSeq=m4tobinar(newY);
  %update arrays
  X=[X newX];
  Y = [Y newY];
  BitSeq=[BitSeq newBitSeq];

  if newY ~= 0
   here=[here i newY]; % numbers nonmultiple of 4
  end

end


periodY = seqperiod(Y)





function [b]=m4tobinar(a)

if (a==0)
    b = [0 0];
  elseif (a==1)
    b = [0 1];
  elseif (a==2)
    b = [1 0];
  elseif(a == 3)
    b = [1 1];
end
```

S111503
Ran Wang
Discrete Mathematics homework03

```
%Question 2.(b)
Last2bits=[];    Y=[];
  for i = 1:124
     newLast2bits = States(i,end-1:end);
      Last2bits=[Last2bits newLast2bits];
      newY = bin2dec(num2str(newLast2bits));
      Y=[Y newY];
  end
periodS = seqperiod(Y)



% Question2.(c)
a=1103515245;
c=12345;
m=2^31;
%initial state of LCG
x0=1;%integer in [0,m-1]
iter=2^10;

ini_bit_vec=1:30;
u=zeros(length(ini_bit_vec),1); z_value=u;
for ii=1:length(ini_bit_vec)
 disp(ii)
 ini_bit=ini_bit_vec(ii);
 %Arrays initialization
  X=x0;
  newBitSeq=dec2bin(X,31); %pass X to binary
  BitSeq=newBitSeq(ini_bit:ini_bit+1); %and extract 2 bits

  for i=1:iter
   xcurrent=X(end);
   newX = mod(a*xcurrent+c,m);
   newBitSeq=dec2bin(newX,31);
   newBitSeq=newBitSeq(ini_bit:ini_bit+1); % extract 2 bits
   %update arrays
   X=[X newX];
   BitSeq=[BitSeq newBitSeq];  %length 2050
  end
%Perform run_test
[u(ii),z_value(ii)]=run_test(BitSeq);


end
% Y = str2double(X)
periodY = seqperiod(X)
figure(1)
plot(ini_bit_vec,z_value,'-k')
hold on
plot(ini_bit_vec,1.96*ones(1,length(ini_bit_vec)),'-r')
xlabel('bit'), ylabel('z')

function [u,z_value]=run_test(s)
% run_test, performes a statistical test on on sequence of bits
```

S111503
Ran Wang
Discrete Mathematics homework03

```
% the run is defined as the number of changes from 0 to 1 or 1 to 0 in the
% bit sequence
%
%
% INPUT PARAMETERS
%
% s : mx1 vector with the sequence of bits
%
% OUTPUT PARAMETERS
% u :: number of runs
% z_value  : statistical value of interest z<1.94 pass the test

m=length(s); %The number of elements of the sequence

% Lets count the number of zeros and ones
n_1=0; %inizialization
n_0=0;
for i=1:m;
    if(str2num(s(i))==1)
     n_1=n_1+1;
    else
     n_0=n_0+1;
    end
end
% u, number of changes in the sequence.
u=1;
for i=1:m-1
    if (str2num(s(i))~=str2num(s(i+1)))
        u=u+1;
    end
end

% Now is time to check the stats of this u number. U is expected to be a
% Gaussian Random distribution with mean and variance:
mean_u=((2*n_1*n_0)/m )+ 1;
var_u=(mean_u-1)*(mean_u-2)/(m-1);


% We acept that our sequence is Gaussian and random if z_value is lower
% than 1.96

z_value=abs(u-mean_u)/sqrt(var_u);
end
```