The Continued Fraction Method for Factorization

All programs are attached in the end of the report.

Throughout the report B is the set of prime numbers smaller than 50.

Question 1:

We use the trial division to develop a test on whether N is B-smooth. We use the test to investigate the probability that a d-digit number is B-smooth. The table of corresponding probabilities is shown below.

From the table we see that as d increases, the probability decreases exponentially. Hence if $d \geq 10$, almost no d-digit number is B-smooth.

Question 2:

Consider the continued fraction expansion of $x = \sqrt{N}$. We show that each x_n can be written in the form $\dfrac{r + \sqrt{N}}{s}$ with r, s integers and $s | r^2 - N$.

WLOG, N is not a perfect square.

First note that by the continued fraction algorithm, $\sqrt{N} = \dfrac{P_{n-1}x_n + P_{n-2}}{Q_{n-1}x_n + Q_{n-2}}$. Rearrange we see that

$$x_n = \frac{r_n + \sqrt{N}}{s_n} \text{ for some } r_n, s_n \text{ to be determined.} \tag{1}$$

Table 1: Probability that a given d-digit number is B-smooth

$d=$	1	2	3	4	5	6
probability	1	0.888889	0.487778	0.214889	0.079733	0.025819

On the other hand,

$$x_n = a_n + \frac{1}{x_{n+1}} = a_n + \frac{s_{n+1}}{r_{n+1} + \sqrt{N}} \tag{2}$$

By equating 1 and 2 we see that

$$s_n(a_n r_{n+1} + s_{n+1}) - N - r_n r_{n+1} = \sqrt{N}(r_{n+1} + r_n - a_n s_n) \tag{3}$$

Since N is irrational, we have

$$s_n(a_n r_{n+1} + s_{n+1}) - N - r_n r_{n+1} = 0 \tag{4}$$

$$r_{n+1} + r_n - a_n s_n = 0 \tag{5}$$

Substitute 5 into 4 we have

$$N - r_{n+1}^2 = s_n s_{n+1} \tag{6}$$

We calculate x_n iteratively using 5 and 6 with $s_0 = 1$ and $r_0 = 0$ to find convergents for $\sqrt{N}$ without concerning the integer overflow issue.

We tabulate the partial quotients for $\sqrt{N}$ for $N \leq 50$ in Appendix A.

The table shows that the period of $\sqrt{N}$ is at most $\sqrt{N}$, and that the partial quotients are symmetric and the last term is always $\left\lfloor \sqrt{N} \right\rfloor$. Moreover by tabulating the values of r and s we see that $r \leq \left\lfloor \sqrt{N} \right\rfloor$ and that $s \leq \left\lfloor 2\sqrt{N} \right\rfloor$.

Question 3:

Now we consider the Pell equation

$$x^2 - Ny^2 = 1 \tag{7}$$

and the nagative Pell equation

$$x^2 - Ny^2 = -1 \tag{8}$$

We note in the first place two necessary conditions for the insolubility of the negative Pell equation 8.

1. If $4|N$, then by taking 8 mod 4 we have $x^2 = -1 \bmod 4$, which is not possible.
2. If N contains a prime factor $p = 3 \bmod 4$, then by taking 8 mod p we have $x^2 = -1 \bmod p$. But for such p, -1 is not a quadratic residue.

We also develop a test for $x^2 - Ny^2 = \pm 1$ with $x, y, N \leq 10^{15}$, we avoid the issue of integer overflow by exploiting the following fact:

For pairwise coprime integers n_i, there exists a unique solution x modulo M such that $x = k$ mod n_i, where M is the product of n_i's and $k = \pm 1$.

We choose our n_i's to be the prime numbers smaller than 200, so that $M \geq 10^{50}$, giving us enough magnitude to test whether 7 holds for $x, y, N \leq 10^{15}$.

Now we tabulate the quantities $P_n^2 - NQ_n^2$ for $N = 5, 13, 17$ to investigate the use of continued fractions to solve 7 and 8. The result is shown in Appendix A.

From the table we see that the continued fraction is an ideal method for solving the Pell equation and, when N is not divisible by 4 or contains prime factor congruent to 3 mod 4, the negative Pell equation and usually gives the solution with only a few iterations.

We use the continued fraction method to solve the Pell equation for $1 \leq N \leq 100$ and $500 \leq N \leq 550$ and tabulate the results in Appendix A.

We see that for $N = 509$, 521, and 526 the Pell equation's solution is too large to be found by the program.

Question 4:

Given $x^2 = y^2$ mod N we have $N|(x + y)(x - y)$. Hence $\gcd(N, x + y)$ and $\gcd(N, x - y)$ are two possible non-trivial factors of N. Note also that the Euclid algorithm is a fast method: At each iteration, the algorithm will reduce the divident or the divisor by at least a half, which gives an $O(\log N)$ complexity.

If N is an odd composite with $N = pq$, where $\gcd(p, q) = 1$, then $x = \dfrac{p+q}{2}$ and $y = \dfrac{p-q}{2}$ are two instances for which $x^2 = y^2$ mod N.

Modular Multiplication with Large Numbers:

Before proceeding to other questions, it's necessary to tackle the issue of integer overflow when we consider the expression of the form ab mod N, where a, b and N are limited to 10^{10}. Here we show one way to circumvent the issue by decomposing a and b.

To begin with, we use the standard division algorithm to decompose $a = 10^7 a_1 + a_2$ and $b = 10^7 b_1 + b_2$, where a_2 and b_2 are small than 10^7. Moreover by the upper bound of a and b we also have a_1 and b_1 are smaller than 10^3.

Now we do the modular multiplication with standard properties of modular arithmetics. We compute

$$\begin{aligned}
ab \bmod N &= (10^7 a_1 + a_2)(10^7 b_1 + b_2) \bmod N \\
&= 10^{14} a_1 b_1 + 10^7 b_2 a_1 + 10^7 a_2 b1 + a_2 b_2 \bmod N \\
&= ((10^{14} \bmod N) a_1 \bmod N) b_1 \bmod N \qquad (9) \\
&+ (10^7 b_2 \bmod N) a_1 \bmod N \\
&+ (10^7 a_2 \bmod N) b_1 \bmod N \\
&+ a_2 b_2 \bmod N \qquad (10)
\end{aligned}$$

Note that we completely avoid the issue of integer overflow in this way because, for example, in 9, $10^{14} \bmod N$ evaluates to a number smaller 10^{10}, and hence $((10^{14} \bmod N)a_1 \bmod N)$ can be done with the exact arithmetic, which multiplied with b_1 can also be done exactly modular N.

Question 5:

We modify our programs to tabulate $P_n \bmod N$ and $P_n^2 \bmod N$. The program is capable for $N \leq 10^{10}$. The results of the program for N= 1449774329, 3333999913 and 7686335197 are given in Appendix A.

We note that $P_n^2 \bmod N$ is usually small, which is more likely to be B-smooth. Hence the continued fractions of $\sqrt{N}$ is a good source for factorization.

Question 6:

We use Gaussian elimination to solve the Equation $A\mathbf{v} = 0$ over $\mathbb{F}_2$. We put the matrix A in its row echelon form and use back substitution to find a non-trivial solution when it it exists.

For example, when

$$T = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \qquad (11)$$

the program reduces T to its row echelon form

$$T_{ref} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{12}$$

and generates a non-trivial solution to $T\mathbf{v} = 0$: $\mathbf{v} = (0,0,0,1,1,1)^T$ is such a solution.

Question 7:

From Question 6 and tabulated values for $P_n^2 \bmod N$, which are highly likely to be B-smooth, we see that the continued fraction is a good way of finding a sequence of integers $\{x_n\}_{n=1}^{K}$ such that $x_n^2 = y_n \bmod N$, where y_n is B-smooth for all n. If we find a non-empty subset $A \subseteq [1, K]$ such that $\prod_{n \in A} y_n$ is a perfect square, we have a choice of (x, y) with $x^2 = y^2 \bmod N$ as in Question 4.

The subset A is found by considering cases where the following quantity is a perfect square.

$$\prod_{n \in A} y_n = \prod_{n=1}^{K} y_n^{a_n}, \text{where } a_n = 0 \text{ or } 1. \tag{13}$$

13 is a square if and only if the prime factors of 13 are congruent to 0 mod 2. The combination of y_n's for which the equivalent condition holds can be calculated by the program in Question 6.

We implement the continued fraction method for factorization.

To improve the performance of our method for very large values of N, we could take larger base B, so that the obtained sequence of P_n^2 is more likely to be B-smooth, and hence reduce the number of convergents required to factorize N.

Appendix A: Tables

Table 2: Partial Quotients for $\sqrt{N}$ for $N \leq 50$

N	Partial Quotients						
1	1						
2	1	2					
3	1	1	2				
4	2						
5	2	4					
6	2	2	4				
7	2	1	1	1	4		
8	2	1	4				
9	3						
10	3	6					
11	3	3	6				
12	3	2	6				
13	3	1	1	1	1	6	
14	3	1	2	1	6		
15	3	1	6				
16	4						
17	4	8					
18	4	4	8				
19	4	2	1	3	1	2	8
20	4	2	8				
21	4	1	1	2	1	1	8
22	4	1	2	4	2	1	8
23	4	1	3	1	8		
24	4	1	8				
25	5						
26	5	10					
27	5	5	10				
28	5	3	2	3	10		

N	Partial Quotients												
29	5	2	1	1	2	10							
30	5	2	10										
31	5	1	1	3	5	3	1	1	10				
32	5	1	1	1	10								
33	5	1	2	1	10								
34	5	1	4	1	10								
35	5	1	10										
36	6												
37	6	12											
38	6	6	12										
39	6	4	12										
40	6	3	12										
41	6	2	2	12									
42	6	2	12										
43	6	1	1	3	1	5	1	3	1	1	12		
44	6	1	1	1	2	1	1	1	12				
45	6	1	2	2	2	1	12						
46	6	1	3	1	1	2	6	2	1	1	3	1	12
47	6	1	5	1	12								
48	6	1	12										
49	7												
50	7	14											

Table 3: Values of $P_n^2 - NQ_n^2$ for $N = 5, 13, 17$

N	$P_n^2 - NQ_n^2$							
5	1	1	-1	1	-1	1	-1	1
13	1	3	-3	4	-1	4	-3	3
17	1	1	-1	1	-1	1	-1	1

Table 4: Solutions to the Pell equation for $1 \leq N \leq 100$

n			n			n			n		
1	1	0	26	51	10	51	50	7	76	57799	6630
2	3	2	27	26	5	52	649	90	77	351	40
3	2	1	28	127	24	53	66249	9100	78	53	6
4	1	0	29	9801	1820	54	485	66	79	80	9
5	9	4	30	11	2	55	89	12	80	9	1
6	5	2	31	1520	273	56	15	2	81	1	0
7	8	3	32	17	3	57	151	20	82	163	18
8	3	1	33	23	4	58	19603	2574	83	82	9
9	1	0	34	35	6	59	530	69	84	55	6
10	19	6	35	6	1	60	31	4	85	285769	30996
11	10	3	36	1	0	61	1766319049	226153980	86	10405	1122
12	7	2	37	73	12	62	63	8	87	28	3
13	649	180	38	37	6	63	8	1	88	197	21
14	15	4	39	25	4	64	1	0	89	500001	53000
15	4	1	40	19	3	65	129	16	90	19	2
16	1	0	41	2049	320	66	65	8	91	1574	165
17	33	8	42	13	2	67	48842	5967	92	1151	120
18	17	4	43	3482	531	68	33	4	93	12151	1260
19	170	39	44	199	30	69	7775	936	94	2143295	221064
20	9	2	45	161	24	70	251	30	95	39	4
21	55	12	46	24335	3588	71	3480	413	96	49	5
22	197	42	47	48	7	72	17	2	97	62809633	6377352
23	24	5	48	7	1	73	2281249	267000	98	99	10
24	5	1	49	1	0	74	3699	430	99	10	1
25	1	0	50	99	14	75	26	3	100	1	0

Table 5: Solutions to the Pell equation for $500 \leq N \leq 550$

n			n		
500	930249	41602	526	1	0
501	11242731902975	502288218432	527	528	23
502	3832352837	171046278	528	23	1
503	24648	1099	529	1	0
504	449	20	530	1059	46
505	809	36	531	530	23
506	45	2	532	2588599	112230
507	1351	60	533	74859849	3242540
508	44757606858751	1985797689600	534	3678725	159194
509	1	0	535	1618804	69987
510	271	12	536	145925	6303
511	4188548960	185290497	537	192349463	8300492
512	665857	29427	538	9536081203	411129654
513	13771351	608020	539	3970	171
514	4625	204	540	119071	5124
515	17406	767	541	1	0
516	16855	742	542	4293183	184408
517	590968985399	25990786260	543	669337	28724
518	2367	104	544	2449	105
519	14851876	651925	545	1961	84
520	6499	285	546	701	30
521	1	0	547	160177601264642	6848699678673
522	19603	858	548	6083073	259856
523	81810300626	3577314675	549	1766319049	75384660
524	225144199	9835470	550	30580901	1303974

Table 6: P_n mod N for some values of N

N	P_n mod N							
1449774329	38075	38076	380759	1561112	3502983	8567078	12070061	286178481
3333999913	57740	57741	230963	288704	18419315	18708019	55835353	465390843
7686335197	87671	87672	263015	350687	6926068	48833163	153425557	509109834

Table 7: P_n^2 mod N for some values of N

N	P_n^2 mod N							
1449774329	-68704	7447	-16819	29495	-22367	52459	-3160	29137
3333999913	-92313	23168	-91239	1791	-77568	37273	-12648	83849
7686335197	-130956	44387	-126548	8817	-23853	50516	-52251	6503

Source Code

```
function L = trialFactorize(n)
L = [];
if n<0
n = -n;
L = [-1];
end

[a,b] = TrialDivFactorize(n);
if a == 1
L = [L,n];
else
t = n/b;
L = [L,b];
while t>1
[a,b] = TrialDivFactorize(t);
if a == 1
L = [L,t];
t = 1;
else
L = [L,b];
t = t/b;
end
end
end
```

```

function [x,f] = TrialDivFactorize(n)
x = 1;
k = 2;
f = n;
while k≤sqrt(n)
r = Mod(n,k); % Mod stands for modulo
if r == 0
f = k;
x = 0; % x is composite with a factor = k
break
end
k = k + 1;
end
```

```
function p = productMod(a,b,N,k)% k = 10^14 mod N
[a1,a2] = decompose(a);% n = 10^7 * a + b
[b1,b2] = decompose(b);


f = Mod(k*a1,N);%first term
```

```
f = Mod(f*b1,N);

s = Mod(10^7*b2,N);%second term
s = Mod(s*a1,N);


t = Mod(10^7*a2,N);%third term
t = Mod(t*b1,N);

fo = Mod(a2*b2,N);%fourth term

p = Mod(f+s+t+fo,N);
end
```

```

function x = powerMod(a,b,N) %% this method aims to compute ...
    a^b mod N, accelarated by binary operations
k = Mod(10^14,N);
b2 = base2(b);
x = 1;
A = [Mod(a,N)];
for i = 1:(length(b2)-1)
A = [productMod(A(1),A(1),N,k),A]; %A = [a^2^m,a^2^m-1 ... ...
    a^2, a] mod N
end

for i = 1:length(b2)
if b2(i) == 1
x = productMod(A(i),x,N,k);
else
x = x;
end
end
end
```

```
function [pn,pn_sqr] = pnModN(N)
k = Mod(10^14,N);
root_N = sqrt(N);
a0 = floor(root_N);
pn = zeros(1,10);
pn_sqr = zeros(1,10);
if (root_N - a0) ≠ 0
R(1) = a0;
S(1) = N - a0^2;
A(1) = floor(nthConvergent(N,R(1),S(1)));
P(1) = a0*A(1)+1;
Q(1) = A(1);
X = [a0];
i = 2;
while 1
```

```
if i < 10
r = A(i-1)*S(i-1) - R(i-1);
s = (N-r^2)/S(i-1);
x = nthConvergent(N,r,s);
R(i) = r;
S(i) = s;
X = [X,x];
A(i) = floor(x);
if i == 2
P(i) = A(i)*P(i-1)+a0;
Q(i) = A(i)*Q(i-1)+1;
else
P(i) = A(i)*P(i-1)+P(i-2);
Q(i) = A(i)*Q(i-1)+Q(i-2);
end
else
break
end
i = i + 1;
end
P = [a0,P];
Q = [1,Q];
end
for i = 1:length(P)
a = Mod(P(i),N);
if a>N/2
pn(i) = a-N;
else
pn(i) = a;
end

b = productMod(P(i),P(i),N,k);
if b>N/2%%%%%we find minimal positive b mod N.
pn_sqr(i) = b-N;
else
pn_sqr(i) = b;
end
end
```

```
function L = PellTabulate(N)
root_N = sqrt(N);
a0 = floor(root_N);
L = [];
if (root_N - a0) ≠ 0
R(1) = a0;
S(1) = N - a0^2;
A(1) = floor(nthConvergent(N,R(1),S(1)));
P(1) = a0*A(1)+1;
Q(1) = A(1);
X = [a0];
L = [a0^2-N];
L = [L,P(1)^2-N*Q(1)^2];
```

```
i = 2;
while 1
if i < 8
r = A(i-1)*S(i-1) - R(i-1);
s = (N-r^2)/S(i-1);
x = nthConvergent(N,r,s);
R(i) = r;
S(i) = s;
X = [X,x];
A(i) = floor(x);
if i == 2
P(i) = A(i)*P(i-1)+a0;
Q(i) = A(i)*Q(i-1)+1;
else
P(i) = A(i)*P(i-1)+P(i-2);
Q(i) = A(i)*Q(i-1)+Q(i-2);
end
L = [L,P(i)^2-N*Q(i)^2];
else
break
end
i = i + 1;
end
P = [a0,P];
Q = [1,Q];

end
end
```

```
function flag = PellEqnTest(x,y,N)%L is the list of primes ...
    smaller than 200
flag = 1;
L = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, ...
    47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, ...
    107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, ...
    167, 173, 179, 181, 191, 193, 197, 199];
for i = 1:46
a = Mod(x,L(i));
a = Mod(a^2,L(i));
b = Mod(y,L(i));
b = Mod(b^2,L(i));
c = Mod(N,L(i));
if Mod(a-b*c,L(i)) ≠ Mod(1,L(i))
flag = 0;
break
end
end
end
```

```
function T = PellEqnSolver(N)
```

```
root_N = sqrt(N);
a0 = floor(root_N);
threshold = 500;
T = [1,0];
if PellEqnTest(a0,1,N)== 1%N=2
T = [a0,1];
i = threshold + 1;
end
if (root_N - a0) ≠ 0
R(1) = a0;
S(1) = N - a0^2;
A(1) = floor(nthConvergent(N,R(1),S(1)));
P(1) = a0*A(1)+1;
Q(1) = A(1);
X = [a0];
i = 2;
if PellEqnTest(P(1),Q(1),N) == 1
T = [P(1),Q(1)];
i = threshold + 1;
end
while 1
if i < threshold
r = A(i-1)*S(i-1) - R(i-1);
s = (N-r^2)/S(i-1);
x = nthConvergent(N,r,s);
R(i) = r;
S(i) = s;
X = [X,x];
A(i) = floor(x);
if i == 2
P(i) = A(i)*P(i-1)+a0;
Q(i) = A(i)*Q(i-1)+1;
else
P(i) = A(i)*P(i-1)+P(i-2);
Q(i) = A(i)*Q(i-1)+Q(i-2);
end
if PellEqnTest(P(i),Q(i),N) == 1
T = [P(i),Q(i)];
break
end
else
break
end
i = i + 1;

end
end
end
```

```

function x = nthConvergent(N,r,s)
x = (sqrt(N)+r)/s;
```

```
end
```

```
function flag = negPellEqnTest(x,y,N)%L is the list of ...
    primes smaller than 200
flag = 1;
L = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, ...
    47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, ...
    107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, ...
    167, 173, 179, 181, 191, 193, 197, 199];
for i = 1:46
a = Mod(x,L(i));
a = Mod(a^2,L(i));
b = Mod(y,L(i));
b = Mod(b^2,L(i));
c = Mod(N,L(i));
if Mod(a-b*c,L(i)) ≠ Mod(-1,L(i))
flag = 0;
break
end
end
end
```

```
function x = multiplicity(p,L)
x = 0;
for i = 1:length(L)
if p == L(i)
x = x + 1;
end
end
```

```
function r = Mod(n,k) % n mod k
r = (n - k*floor(n/k));
end
```

```

function B = matrixMod2(A)
B = A - 2*floor(A./2);
end
```

```
function [v] = F2LinearSolver(A)
r = length(A(:,1));
c = length(A(1,:));
REF = F2GaussElimination(A);
v = zeros(c,1);
column = 0;
```

```
for i = 1:c
if all(REF(:,i)==0) == 1
column = i;
v(i) = 1;
end
end

if column == 0 %no zero pivot
s = 0;
for i = 1:r

if s == 0
for j = 1:c-1
if REF(i,j)≠0
for k = j+1:c
if REF(i,k)≠0
row = i;
column = k;
s = 1;
break
end
end
end
end
else
break
end
end

v(row) = 1;
v(column) = 1;
for i = 1:r
if REF(i,column) == 1
v(i) = 1;
end
end
end
end
```

```
function [f,B] = F2GaussEliminationStep(A,k,r,c)%we start ...
    the F2gaussEliminiation from (k,k) index
A = matrixMod2(A);
f = 1;
if k == min(r,c)%we have reached the end of F2 gauss ...
    eliminiation.
if A(k,k) == 1
for i = 1:k-1
if A(i,k) == 1
A(i,:) = matrixMod2(A(i,:) - A(k,:));
end
end
end
```

```
f = 0;
elseif k == 1
swapflag = 0;%swapflag indicates whether there exists an ...
    element A(i,k)!= 0
for i = k:r
if A(i,k) == 1
temp_row = A(i,:);
A(i,:) = A(k,:);
A(k,:) = temp_row;
swapflag = 1;
break
end
end
if swapflag == 1
for i = k+1:r
if A(i,k) == 1
A(i,:) = matrixMod2(A(k,:) - A(i,:));
end
end
end
else
swapflag = 0;%swapflag indicates whether there exists an ...
    element A(i,k)!= 0
for i = k:r
if A(i,k) == 1
temp_row = A(i,:);
A(i,:) = A(k,:);
A(k,:) = temp_row;
swapflag = 1;
break
end
end
if swapflag == 1
for i = k+1:r
if A(i,k) == 1
A(i,:) = matrixMod2(A(k,:) - A(i,:));
end
end

%%we eliminiate nonzero elements above the kth row.
for i = 1:k-1
if A(i,k) == 1
A(i,:) = matrixMod2(A(i,:) - A(k,:));
end
end
end
end
B = A;
end
```

```

function B = F2GaussElimination(A)
```

```
r = length(A(:,1));
c = length(A(1,:));
k_max = min(r,c);
B = A;
f = 1;
for k = 1:k_max

if f == 1
[f,B] = F2GaussEliminationStep(B,k,r,c);
else
break
end
end
end
```

```
function [a,b] = decompose(n) % n = 10^7 * a + b
a = floor(n/10^7);
b = n - 10^7*a;
end
```

```

function continuedFractionFactor(N)

k = Mod(10^14,N);

r = 0;
root_N = sqrt(N);
a0 = floor(root_N);
R(1) = a0;
S(1) = N - a0^2;
A(1) = floor(nthConvergent(N,R(1),S(1)));
P(1) = a0*A(1)+1;
Q(1) = A(1);
T = zeros(16,2);
b = productMod(a0,a0,N,k);
if b>N/2%%%%%we find minimal positive b mod N.
pn_sqr = b-N;
else
pn_sqr = b;
end

[x,L] = B_Smooth(pn_sqr);
if x == 1
r = r + 1;
pn(r) = a0;
y_list(r) = pn_sqr;
T(:,r) = matrixMod2(B_Representation(L));
end


```

```
b = productMod(P(1),P(1),N,k);
if b>N/2%%%%%we find minimal positive b mod N.
pn_sqr = b-N;
else
pn_sqr = b;
end

[x,L] = B_Smooth(pn_sqr);
if x == 1
r = r + 1;
pn(r) = P(1);
y_list(r) = pn_sqr;
T(:,r) = matrixMod2(B_Representation(L));
end
T
if length(T(1,:)) > 1
v = F2LinearSolver(T);
if any(v == 0) == 1
v
end
end

i = 2;
while 1
R(i) = A(i-1)*S(i-1) - R(i-1);
S(i) = (N-r^2)/S(i-1);
x = nthConvergent(N,R(i),S(i));
A(i) = floor(x);
if i == 2
P(i) = A(i)*P(i-1)+a0;
Q(i) = A(i)*Q(i-1)+1;
else
P(i) = A(i)*P(i-1)+P(i-2);
Q(i) = A(i)*Q(i-1)+Q(i-2);
end

b = productMod(P(i),P(i),N,k);
if b>N/2%%%%%we find minimal positive b mod N.
pn_sqr = b-N;
else
pn_sqr = b;
end

[x,L] = B_Smooth(pn_sqr);
if x == 1
r = r + 1;
pn(r) = P(i);
y_list(r) = pn_sqr;
T(:,r) = matrixMod2(B_Representation(L));
end
if length(T(1,:)) > 1
v = F2LinearSolver(T);
if any(v == 0) == 1
v
```

```
break
end
end
if r == 17
break
end
i = i + 1;
end
T
pn
y_list



end
```

```

function [P,Q,R,S,A] = ContinuedFraction(N)
root_N = sqrt(N);
a0 = floor(root_N);
if (root_N - a0)== 0
P = a0;
Q = 1;
R = 0;
S = 1;
A = a0;
else
R(1) = a0;
S(1) = N - a0^2;
A(1) = floor(nthConvergent(N,R(1),S(1)));
P(1) = a0*A(1)+1;
Q(1) = A(1);
X = [a0];
i = 2;
while 1
r = A(i-1)*S(i-1) - R(i-1);
s = (N-r^2)/S(i-1);
x = nthConvergent(N,r,s);
if any(X==x)
P = P(1:length(P)-1);
Q = Q(1:length(Q)-1);
R = R(1:length(R)-1);
S = S(1:length(S)-1);
A = A(1:length(A)-1);
break
else
R(i) = r;
S(i) = s;
X = [X,x];
A(i) = floor(x);
if i == 2
P(i) = A(i)*P(i-1)+a0;
```

```
Q(i) = A(i)*Q(i-1)+1;
else
P(i) = A(i)*P(i-1)+P(i-2);
Q(i) = A(i)*Q(i-1)+Q(i-2);
end
end
i = i + 1;
end
A = [a0,A];
P = [a0,P];
Q = [1,Q];
R = [0,R];
S = [1,S];
end
end
```

```

function [x,L] = B_Smooth(n)
x = 1;%n is b smooth
L = [];
if n<0
n = -n;
L = [-1];
end
t = 2;
while t > 1
[a,b] = TrialDivFactorize(n);
if b > 49
x = 0;%x is not B-smooth.
t = 1;
else
L = [L,b];
if a == 1
t = 1;
else
n = n/b;

end
end
end
end
```

```

function T = B_Representation(L)
T = zeros(16,1);
T(1) = multiplicity(-1,L);
T(2) = multiplicity(2,L);
T(3) = multiplicity(3,L);
T(4) = multiplicity(5,L);
T(5) = multiplicity(7,L);
```

```
T(6) = multiplicity(11,L);
T(7) = multiplicity(13,L);
T(8) = multiplicity(17,L);
T(9) = multiplicity(19,L);
T(10) = multiplicity(23,L);
T(11) = multiplicity(29,L);
T(12) = multiplicity(31,L);
T(13) = multiplicity(37,L);
T(14) = multiplicity(41,L);
T(15) = multiplicity(43,L);
T(16) = multiplicity(47,L);
end
```

```
L = zeros(1,6);
for i = 0:5
counter = 0;
for j = 10^i:10^(i+1)-1
if B_Smooth(j) == 1
counter = counter + 1;
end
end
L(i+1)=counter/(10^(i+1)-10^(i));
end
```

```

for i = 1:50
[P,Q,R,S,A] = ContinuedFraction(i);
filename = 'PartialQuotients.xlsx';
filename2 = 'R.xlsx';
filename3 = 'S.xlsx';
sheet = 1;
xlRange = strcat('A',num2str(i));
xlswrite(filename,A,sheet,xlRange)
xlswrite(filename2,R,sheet,xlRange);
xlswrite(filename3,S,sheet,xlRange);
end
```

```
filename = 'PellTabulate.xlsx';
sheet = 1;
xlRange = strcat('A',num2str(1));
xlswrite(filename,PellTabulate(5),sheet,xlRange);
xlRange = strcat('A',num2str(2));
xlswrite(filename,PellTabulate(13),sheet,xlRange);
xlRange = strcat('A',num2str(3));
xlswrite(filename,PellTabulate(17),sheet,xlRange);

%%%%%%%%%%%%
filename = 'PellSolution.xlsx';
sheet = 1;
```

```
for i = 1:100
xlRange = strcat('A',num2str(i));
xlswrite(filename,PellEqnSolver(i),sheet,xlRange);
end

for i = 0:50
xlRange = strcat('A',num2str(500+i));
xlswrite(filename,PellEqnSolver(500+i),sheet,xlRange);
end
```

```

filename = 'PnModN.xlsx';
sheet = 1;
filename2 = 'PnSqrModN.xlsx';

N = 1449774329;
xlRange = strcat('A',num2str(1));
[pn,pn_sqr] = pnModN(N);
xlswrite(filename,[N,pn],sheet,xlRange);
xlswrite(filename2,[N,pn_sqr],sheet,xlRange);

N = 3333999913;
xlRange = strcat('A',num2str(2));
[pn,pn_sqr] = pnModN(N);
xlswrite(filename,[N,pn],sheet,xlRange);
xlswrite(filename2,[N,pn_sqr],sheet,xlRange);

N = 7686335197;
xlRange = strcat('A',num2str(3));
[pn,pn_sqr] = pnModN(N);
xlswrite(filename,[N,pn],sheet,xlRange);
xlswrite(filename2,[N,pn_sqr],sheet,xlRange);
```