Doktori (PhD) értekezés tézisei

Álvéletlenszám-generátorok konstrukciója

Padányi Viktória

Témavezető: Dr. Herendi Tamás

Debreceni Egyetem
Informatikai Tudományok Doktori Iskola

Debrecen, 2024.

Tartalomjegyzék / Table of contents

Bevezetés

Gyakran használnak álvéletlenszám-generátorokat különböző elméleti és gyakorlati problémák megoldására. Ezeket a generátorokat alkalmazzák minden olyan területen, ahol szükség van véletlenszerű tesztadatokra vagy mintákra, például mesterséges intelligencia alkalmazásokban, szimulációkban, véletlenszerű algoritmusokban vagy pedig kriptográfiában.

Két módszer létezik véletlenszámok sorozatának generálására: Az előállítás történhet álvéletlenszám-generátorokkal (PRNG) és valódi véletlenszám-generátorokkal (TRNG). Az álvéletlenszám-generátorok esetében általános megközelítés, hogy a sorozat elemeit rekurzív módon számítjuk ki az előző elemekből. A rekurzió a kezdeti mag (seed) felhasználásával indítható, amelyet iteratív módon kiszámítunk, majd a véletlenszerű értékeket ennek alapján származtatjuk.

Az álvéletlenszám-generálásra léteznek elméleti és gyakorlati algoritmusok. Az elméleti megközelítés esetében a műveletek tetszőleges pontosságú számokon történnek (vagy végtelen halmazon), míg a gyakorlati módszer esetén fix pontosságú számokat használunk (általában véges halmazon, gyakran algebrai struktúrával). Mindkét esetben azonban elvárás a kapott sorozat egyenletes eloszlása, amit csak véges esetre definiáltunk. Azonban a sorozatok teljesen kiszámíthatóak és rendkívül hatékonyan tömöríthetőek, így nem képesek valódi, információelméletileg bizonyítható véletlen számok előállítására.

Használat szempontjából érdekes kérdés az álvéletlenszám-generátorok tulajdonsága. Az eltérő alkalmazások más-más tulajdonságokat várnak el a generátoroktól. A felhasználás során fontos szempont lehet a generátor sebessége, valamint az erőforrásigénye, illetve az adott generátorok minősége, melyek statisztikai tesztekkel mérhetők. A statisztikai tesztek kulcsfontosságúak az álvéletlenszám-generátorok elemzésében.

Kutatómunkám első lépéseként összegyűjtöttem néhány használatban lévő, egyenletes eloszlású elméleti és gyakorlati álvéletlenszám-generátort. A generátorokat tulajdonságuk és a használatban lévő próbák és vizsgálatok alapján rendeztem. A NIST teszt csomaggal vizsgáltam mélyebben az felsorolt generátorokat. Az összefüggő eredményeket [4]-ben publikáltam, míg ezeknek az eredményeknek a javítása és az ehhez kapcsolódó kísérleti eredmények a [6]-ben jelentek meg.

A kutatásom következő periódusában részletes elemzést végeztem a négyzetközép módszerek csoportjába tartozó generátorokról, tanulmányoztam, hogy milyen fontos tulajdonságokkal rendelkeznek, illetve melyek az előnyös tulajdonságaik.

A Neumann-féle négyzetközép módszerre fókuszáltam, amely során a számsorozatokat úgy generáljuk, hogy a kezdeti magértéket négyzetre emeljük, majd a kapott szám középső jegyei alkotják a következő számot. A periódus hossza a kezdeti értéktől függ, és rövid ciklusúnak bizonyult, azaz rövid számsorozat után ismétli önmagát.

Ezt követően az általánosított számrendszerekkel folytattam a kutatást, Neumann János négyzetközép módszerét általánosítottam kanonikus számrendszerekre. A rendszerben két számjegy van (0,1) a sorozat pedig hasonló módon van definiálva: egy megfelelően megválasztott m-bites kezdőérték a kiindulás. Négyzetre kell emelni a magot, ki kell vágni a középső m-bitet, és ez a generátor következő magja. Ezekben a számrendszerekben a műveletvégzés hasonló a racionális egészekhez, viszont az átvitel számítsa kicsit komplikáltabb. Tehát, némileg bonyolultabb az átvitel reprezentációja. Dolgozatomban egy példát használok ennek a számításnak a levezetéséhez.

A disszertációmban megtalálható a négyzetközép-módszer általánosítása, illetve ezen generátorok elemzése bináris kanonikus számrendszerekben. Az eredményeket a [5]-ben publikáltam.

Kutatásom másik témája az automatákkal kapcsolatos. Definiáltam egy $A \in \mathbb{Z}[x]$ polinom információmennyiségét a CNS polinomhoz viszonyítva, és megmutattam, hogy szoros kapcsolatban áll a számrendszerbeli reprezentáció hosszával. Eredményeim alapján sikerült igazolnom, hogy minden kanonikus számrendszer polinomhoz létezik egy véges automata, amely képes a P alapú kanonikus reprezentációban a polinomok összeadásának végrehajtására. Végül elemeztem ezeknek az automatáknak a méretét, azaz az állapotok számát. Az eredményem közlésre benyújtott állapotban van. [2]

Finomítottam Kovács-Pethő számolását, amivel a számok kanonikus számrendszerekben való felírásának hossza megbecsülhető. Az elért becslések segítségével korlátokat számolhatunk a fix hosszúságú négyzetek hosszára. A továbbiakban, a műveletvégzés volt fontos. Ennek célja elsősorban az volt, hogy megvizsgáljam hogyan alakul a négyzetszámok hossza számított korlátokhoz képest. A vizsgálatok elsődlegesen a négyzetközép-módszer miatt voltak érdekesek, ezért a négyzetre emelés tulajdonságait elemeztem mind gyakorlati mind elméleti szempontból.

Többek között bizonyítottam, hogy tetszőleges kanonikus számrendszerhez létezik egy átalakító automata, amely az adott számrendszerben összeadás műveletét hajtja végre. Ehhez készítettem egy algoritmust, ami meghatározza a minimális összeadó automatát, ha a számrendszer egy bináris kanonikus számrendszer. Az összeadás tulajdonságai alapján elkészítettem egy algoritmust a Gray-kód felhasználásával, amely $O(n \cdot 2^n)$ időben felsorolja az algebrai egész számok négyzetét. A konstrukció segítségével elemeztem a négyzetre emelés néhány tulajdonságát bináris kanonikus számrendszerekben.

Kutatásom ezen részének eredménye, hogy fel tudom sorolni az összes n hosszúságú négyzetszámokat $O(n \cdot 2^n)$ időben. Az értekezés utolsó fejezetében bemutatom az algoritmust és azok megfigyeléseit.

1. Álvéletlenszám-generátorok vizsgálata

A kutatásom során összegyűjtöttem néhány olyan álvéletlenszám-generátort, amelyek elméleti és gyakorlati szempontból egyaránt egyenletes eloszlást biztosítanak. Az értekezésemben leginkább a gyakorlati álvéletlenszám-generátorokkal foglalkoztam. Az elméleti álvéletlenszám-generátorok gyakorlati hasznossága korlátozott, ezért nem is végeztem részletes elemzést, mivel azok erősen reprezentáció függőek. A gyakorlati álvéletlenszám-generátorokat több kategóriába sorolhatjuk, amelyek olyan rekurzív relációkat használnak, amelyeknek a mélysége korlátozott. Emiatt ezek a sorozatok periodikusak. Többségük moduláris aritmetikát használ alapműveletként a sorozatok elemeinek kiszámításához.

Gyakorlati álvéletlenszám-generátorok osztályozása:

1. Maradékosztályok felett definiált rekurziókon alapuló generátorok
   a) Homogén lineáris módszerek
   b) Nem homogén lineáris módszerek
   c) Hibrid módszerek
   d) Nem lineáris rekurzión alapuló generátorok
2. Egyéb rekurzión alapuló generátorok
   a) Négyzetközép-módszeren alapuló generátorok
   b) Keveréses módszerek
3. Vegyes módszerek

A továbbiakban a generátorok tulajdonságait vizsgáltam meg, mint például a periódus hossza, a számítási bonyolultság és a memóriahasználat. Az álvéletlenszám-generátorok tulajdonságait az **1**. táblázat mutatja be.

Mivel a generátorokat széles körben alkalmazzák különböző környezetekben, nehéz összehasonlítani a sebességüket. Néhányuk jobban teljesít korlátozott eszközökön, míg más generátorok hatékonyabbak CISC processzorokon. Ezen kettősség alapján, az algoritmusok sebessége helyett a komplex (lassú) és egyszerű (gyors) műveletek számát emeltem ki. A legtöbb generátorhoz elérhető implementáció is található. [6] Amelyekhez nem áll rendelkezésre implementáció, azok egyszerű, gyengén teljesítő generátorok.

Generátorok	Periódus-hossz[1]	Műveletek[2]		Memória (bit)
		Lassú	Gyors	
Lehmer's MLCG	2^s	1 mul, 1 mod		$2s$
GFSR	2^k		s-bit xor	$k \cdot s$
TGFSR	2^{ks}	$s \times s$-bit m.mul	s-bit xor	$k \cdot s + s^2$
LCG	2^s	1 mul, 1 mod	1 sum	$4s$
LRS[3]	2^{k+s}		k sum	$k \cdot s$
MT[4]	2^{ks}	$s \times s$-bit m.mul	s-bit xor	$k \cdot s + s^2$
WELL[5]	2^{ks}	8 $s \times s$-bit m.mul	9 s-bit op	
W-H algorithm	2^{3s}	3 LCG, 3 div	2 sum	$12s$
ACORN	2^{ks}	k mod	k sum	$k \cdot s$
PCG	2^{128}	mul	s-bit perm,xor	$4s$
QCG	s^s	4 mul, 1 mod	2 sum	$4s$
BBS	2^s	1 mul, 1 mod		$2s$
Multipl. Fibonacci	2^{2s}	1 mul, 1 mod		$3s$
Power Congr. Gen.	2^s	s mul, 1 mod		$4s$
ICG	2^s	1 mul, 1 inv, 1 mod	1 sum	$4s$
Neumann's MSM[6]	8^d	1 mul	1 xor, 1 shr	$4d$
Metropolis MSM[7]	2^s	1 mul	1 xor, 1 shr	s
Coveyou's SMSM	2^s	1 mul		s
GMSM	2^s	1 mul	1 xor, 1 shr	s
MS Weyl	2^{2s}	1 LCG, 1 mul	1 sum	$5s$
Algorithm M	2^{2s}	2 gen, 1 mul, 1 div		$2 \cdot * + k \cdot s$
Algorithm B	2^s	1 gen, 1 mul, 1 div		$* + k \cdot s$
RC4	2^{1024}		7 byte op	256 bytes
Xorshift	2^s		1 xor, 1 shr	s
Legendre symbol	2^s	s mod		s
Enc.-based methods[8]	2^s	*	*	$2s$

1. táblázat. A generátorok általános tulajdonságai

A táblázatban kapott eredmények a generátorok paramétereivel vannak kifejezve, hogy világosabb képet kapjunk a különböző módszerek jellemzőiről:

- s: Az adat (mag) hossza bitben kifejezve.
- d: Az adat decimális hossza.
- k: A következő mag meghatározásához használt legtávolabbi mag távolsága vagy a magvektor dimenziója.

Az **1.** táblázatban a következő megjegyzések érvényesek:

1. A *Periódushossz* oszlopban lehető legjobb értékek vannak feltüntetve.

2. A *Műveletek* oszlop mutatja az algoritmusok bonyolultságát, azaz a szükséges műveletek számát egyetlen véletlen minta kiszámításához. A műveletek a megadott adatméretet használják (s bit, vagy d decimális számjegy).

3. Az LRS generátor bonyolultsága k összeadásra csökkenthető a megfelelő paraméterek kiválasztásával.

4. A széles körben használt paraméterek mellett az MT periódus hossza $2^{19937} - 1$.

5. A széles körben használt paraméterek mellett a WELL periódus hossza 2^{44497}.

6. Neumann MSM módszere eredetileg decimális számokra van definiálva.

7. Metropolis eredeti konstrukciója 20-bites számokkal maximális periódushossza 142.

8. Az időbonyolultság az alapul szolgáló kriptográfiai algoritmusoktól függ.

A **2.** táblázat magyarázatot ad az **1.** táblázatban hivatkozott műveletekre.

Lassú műveletek		Gyors műveletek	
mul	szorzás	sum	összeadás
mod	maradékos osztás	xor	bitenkénti vagy (xor)
div	osztás	shl, shr	bitenkénti eltolás
m.mul	vektor szorzása mátrixszal	op	memória műveletek
inv	inverz	perm	bitenkénti permut.
gen	egymásba ágyazott gen.-okból szárm. minta		
LCG	minta egy adott LCG generátorból		

2. táblázat. Műveletek

Mielőtt elkezdünk álvéletlenszám-generátort használni, tudnunk kell, hogy alkalmas-e az adott célra. Ezért fontos általános tulajdonságokat vizsgálni, amelyet standard statisztikai tesztekkel lehet elvégezni. A különböző tesztek a tulajdonságokat különböző szempontokból fejezik ki. Azonban még mindig kérdés, hogy milyen tulajdonságokra van szükség az adott alkalmazás szempontjából. Az álvéletlenszám-generátorok statisztikai tesztjei általában olyan általános statisztikai teszteknek az alkalmazásai, amelyek egy képzeletbeli tökéletesen egyenletes eloszlású valódi véletlen sorozat elvárt tulajdonságaira vonatkoznak.

Részletesen bemutattam egy statisztikai tesztcsomagot, melyet a NIST (Nemzeti Szabványügyi és Technológiai Intézet) tervezett és ajánl véletlen- és álvéletlen-számgenerátorok tesztelésére. A választásom azért esett erre a teszt csomagra, mert ez a standard. A NIST tesztcsomag 15 tesztből áll, és célja, hogy tesztelje a hardveres vagy szoftveres generátorok által előállított tetszőlegesen hosszú bináris sorozatok véletlenszerűségét. A tesztek a különböző típusú nem véletlen jelenségekre fókuszálnak, amelyek előfordulhatnak egy sorozatban.

1.1. Kísérleti eredmények

A disszertációban bemutatottam a vizsgálataimat és teszteredményeimet az általam leírt generátorok által generált sorozatokkal kapcsolatban. Az álvéletlenszám-generátorokat C++ nyelven implementáltam, és egy Intel(R) Core(TM) i9-9900K CPU-n (3,60GHz), 64GB RAM-mal futtattam. A kapott sorozatokat ugyanebben a környezetben teszteltem a NIST tesztcsomaggal. Az implementációk és a részletes eredmények elérhetőek a http://www.prng.hu weboldalon.

A tesztekhez 10^9 hosszú 0,1-bites sorozatokat használtam minden generátor esetén. Minden sorozatot 1000 azonos hosszúságú részsorozatra osztottam fel. Mind a 15 tesztet alkalmaztam az összes generátorra.

A legfontosabb eredményeket a dolgozatom NIST eredmények táblázataiban gyűjtöttem össze. A szignifikancia szintet 0,01-re állítottam be. Az egyes statisztikai tesztek minimális teljesítési aránya (átment az adott teszten) – a Módosított véletlen bolyongás teszt kivételével – körülbelül 0,981819, egy 1000 minta méretű bináris sorozathoz. A Módosított véletlen bolyongás teszt esetében körülbelül 0,979517 egy 609 minta méretű bináris sorozathoz.

Azok a generátorok, amelyek nem teljesítették az adott tesztet, csillaggal (*) jelölve vannak a táblázatokban. Azok a tesztek, amelyek több altesztől függenek különböző paraméterekkel – Kumulatív összeg tesztek, Nem átfedő sablonillesztési teszt, Véletlen bolyongás, Módosított véletlen bolyongás teszt és Sorozat teszt – a táblázatokban az eredmények átlaga került be.

A tárgyalt álvéletlenszám-generátorok többsége átment a tesztcsomag összes tesztjén. Ilyen összeállításban nem találtam hasonló elemzéseket, amelyeket NIST teszttel végeztek volna, ugyanolyan feltételek és paraméterek mellett ezekre a generátorokra. A generátorok ismertek; igyekeztem áttekinteni a klasszikus (korai) generátoroktól kezdve a modern speciális alkalmazással rendelkező generátorokig többféle sorozatot is. Célom volt egy átfogó elemzés elvégzése (beleértve a generátorok műveleteinek számának kiemelését), hogy a definíció szerint működő generátorok hogyan viselkednek NIST teszttel vizsgálva őket.

Néhány esetben érdekes viselkedést tapasztaltam. A legmeglepőbb az, hogy az LCG és a Lehmer-féle MLCG generátorok jó eredményeket mutattak, annak ellenére, hogy mindkettő bizonyítottan kriptográfiailag nem biztonságos. Ellentétes viselkedést találtam a BBS-nél, amely gyengén teljesített a teszteken, viszont - kriptográfiai szempontból is - biztonságos generátornak mondható. A Neumann János-féle Négyzetközépmódszerre épülő generátoroknál jelentősen növelnem kellett a magjaik számjegyeit, hogy a teszteken átmenő szekvenciákat elérjem. (Például az eredeti NMSM-generátor magját 72-jegyűre kellett bővítenem.) Ehhez képest a kanonikus számrendszerekre általánosított Neumann-féle négyzetközép generátor csak egy 92 bites, hasonló tulajdonságokkal rendelkező magot igényelt, még egy viszonylag egyszerű algebrai kiterjesztés esetén is.

2. Általánosított Neumann-féle négyzetközép módszeren alapuló álvéletlenszámgenerátor

Neumann János Négyzetközép módszert általánosítottam kanonikus számrendszerekre. A disszertációban bemutattam a generált sorozatok definícióit, vizsgálatait és statisztikai tesztjeit, amelyek fontosak a generátorok minőségének értékeléséhez.

Létezik általánosabb definíciója (rácsok segítségével) is a számrendszereknek, de az általam elért eredményekhez teljesen megfelelő a klasszikus definíció is.

2.1. Definíció. *Legyen R egy integritási tartomány, $\alpha \in R$ és $\mathcal{N} = \{n_1, \ldots, n_m\} \subseteq \mathbb{Z}$. Az $(\alpha, \mathcal{N})$ párt számrendszernek nevezzük, ha bármely $\gamma \in R$ egyértelműen felírható $\gamma = \sum_{i=0}^{h} c_i \alpha^i$ alakban, ahol $c_i \in \mathcal{N}$ $0 \leq i \leq h$ és $c_h \neq 0$, ha $h \neq 0$. A számrendszert kanonikusnak nevezzük, ha $\mathcal{N} = \{0,1, \ldots, m-1\}$.*

A továbbiakban a γ hosszára $(\alpha, \mathcal{N})$ számrendszerben az $L(\gamma, \alpha, \mathcal{N}) = h + 1$ jelölést használom.

2.2. Tétel. *Legyen $p \in \mathbb{Z}[x]$ egy irreducibilis polinom, ahol $\deg(p) = d$, és $p(x) = a_d x^d + \cdots + a_0$, amelyekre $1 = a_d \leq \leq a_{d-1} \leq \ldots \leq a_0$ és $2 \leq a_0$. Továbbá, legyen α a gyöke p-nek és $\mathcal{N} = \{0,1, \ldots, a_0 - 1\}$. Ekkor $(\alpha, \mathcal{N})$ kanonikus számrendszer $\mathbb{Z}[\alpha]$-ban.*

2.3. Tétel. *Legyen β, $n \geqq 1$ fokú algebrai egész, és legyen $(\alpha, \mathcal{N})$ egy kanonikus számrendszer $\mathbb{Z}[\beta]$-ban. Ekkor léteznek a következő algoritmikusan kiszámítható konstansok:*

$C_1 = C_1(\alpha, \mathcal{N})$ és $C_2 = C_2(\alpha, \mathcal{N})$, melyek csak az α-tól és $\mathcal{N}$-től függenek, amelyekre

$$|\log|_\alpha \gamma + C_1 \leqq L(\gamma, \alpha, \mathcal{N}) \leqq |\log|_\alpha \gamma + C_2 \tag{2.1}$$

minden $0 \neq \gamma \in \mathbb{Z}[\beta]$-ra igaz.

2.4. Megjegyzés. *$C_3 = C_1 - 2C_2$ és $C_4 = C_2 - 2C_1$ ismeretében az alábbi állítást tudom felírni:*

$$2L(\gamma) + C_3 \leq L(\gamma^2) \leq 2L(\gamma) + C_4 \ . \tag{2.2}$$

A **3**. táblázatban összefoglaltam, hogy hogyan alakul rögzített hosszúságú számok négyzeteinek hossza. A négy kanonikus számrendszer halmazát két racionális bináris számrendszerrel bővítettem, amelyek alapja 2 és -2.

Az összes 20 és 30 számjegyű számot figyelembe vettem. A táblázat tartalmazza a h hosszúságú számok négyzetei hosszának minimális és maximális eltérését az elvárt $2h$-tól. Ezen felül az átlagos négyzethosszúságokat is bemutattam a táblázatban. Az utolsó oszlopban a megfelelő távolságértékek elméleti alsó és felső korlátjai láthatók. A cél többek között az volt, hogy mennyire közelíti meg a gyakorlatban az elméleti számítást. Továbbá, vizsgáltuk azt is, hogy ezek a korlátok milyen hosszúságú számok esetén érik el az adott értéket.

Bár az elméleti értékek eltérnek a gyakorlati mérésektől, még akkor is, ha finomítottuk őket. Például 30 hosszúságú számok esetén az elméleti értékekhez közeli eredményeket kaptunk, de még mindig gyakorlati szempontból releváns. A további cél, hogy a gyakorlati mérések alapján bizonyítsuk, hogy ezek az értékek véglegesek. Az elméleti korlátok további finomítása még lehetséges, legalábbis a gyakorlati mérések alapján. A gyakorlati eredmények véglegesítése további kutatási cél.

Length of base numbers

Digits	20	21	22	23	24	25	26	27	28	29	30	T
Defining polynomial: $x-2$												
Decrease	1	1	1	1	1	1	1	1	1	1	1	1
Increase	0	0	0	0	0	0	0	0	0	0	0	0
Average	39.6	41.6	43.6	45.6	47.6	49.6	51.6	53.6	55.6	57.6	59.6	
Defining polynomial: $x+2$												
Decrease	3	3	3	3	3	3	3	3	3	3	3	4
Increase	1	1	1	1	1	1	1	1	1	1	1	2
Average	38.9	40.9	42.9	44.9	46.9	48.9	50.9	52.9	54.9	56.9	58.9	
Defining polynomial: x^2+x+2												
Decrease	8	8	8	8	8	8	8	8	8	8	8	12
Increase	5	5	5	5	5	5	5	5	5	5	5	10
Average	39.6	41.6	43.6	45.6	47.6	49.6	51.6	53.6	55.6	57.6	59.6	
Defining polynomial: x^2+2x+2												
Decrease	12	12	12	12	12	12	12	12	12	12	12	16
Increase	9	9	9	9	9	9	9	9	9	9	9	12
Average	40.6	42.6	44.6	46.6	48.6	50.6	52.6	54.6	56.6	58.6	60.6	
Defining polynomial: x^3+x^2+x+2												
Decrease	11	14	14	14	14	14	14	14	14	14	14	31
Increase	12	12	12	12	12	12	12	12	12	12	12	27
Average	41.6	43.5	45.5	47.5	49.5	51.5	53.5	55.5	57.5	59.5	61.5	
Defining polynomial: $x^4+x^3+x^2+x+2$												
Decrease	17	18	18	18	18	18	20	20	20	20	20	46
Increase	21	21	21	21	21	21	21	21	21	21	21	32
Average	43.8	45.6	47.6	49.7	51.9	53.9	55.7	57.7	59.7	61.8	63.8	

3. táblázat. A négyzetek hossza

A Neumann-féle négyzetközép módszer általánosítása kanonikus számrendszerekre a következő:

Legyen $p(x) \in \mathbb{Z}[x]$ egy irreducibilis d-fokú polinom, melynek együtthatói $1 = a_d \leq a_{d-1} \leq \ldots \leq a_0 = 2$. A megfelelő kanonikus számrendszerben 2 számjegy van, 0 és 1. Az egyszerűség kedvéért, számjegyeket biteknek és a $\mathbb{Z}[\alpha]$-beli algebrai egész számok számjegyes ábrázolását bináris ábrázolásnak fogom nevezni.

A generátor meghatározásánál egy $m \in \mathbb{N}$ bites magot használok. A véletlenszám sorozatot az eredeti definícióhoz hasonló módon határozom meg:

$u_0 \in \mathbb{Z}[\alpha]$ véletlen m-bites szám;

ha $n > 0$,

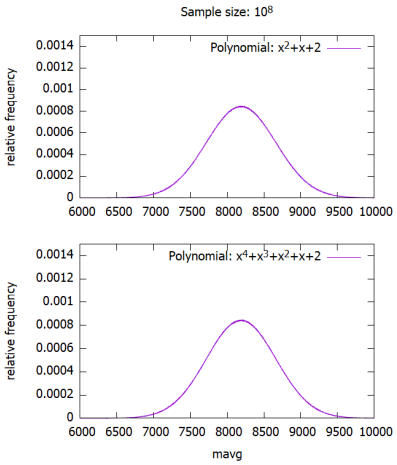
$$u_{n-1}^2 = \sum_{i=0}^{h} b_i \alpha^i \ , b_h \neq 0 \ , \ t = \left\lfloor \frac{h-m}{2} \right\rfloor \text{ és}$$
$$u_k = \sum_{i=0}^{m-1} b_{i+t+1} \alpha^i \ .$$

Az m értékét úgy kell megválasztani, hogy elég nagy legyen, egész pontosan úgy, hogy $2m + C_3 > m$, azaz $m > -C_3$, ahol C_3 a fent meghatározott.

Egy másik megközelítés, ha $t = \lfloor \frac{m}{2} \rfloor$, de akkor teljesülnie kell annak, hogy $\frac{m}{2} > -C_3$.

2.1. Kísérleti eredmények

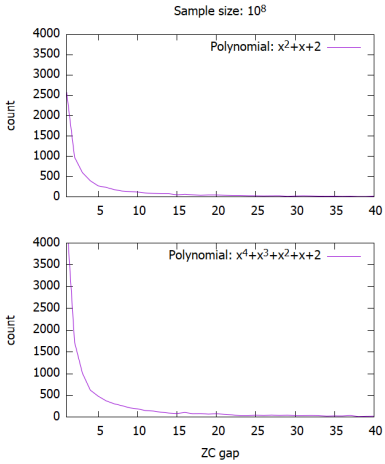
Az **1**. ábra a sorozatok mozgóátlagának eloszlását mutatja.



1. ábra. Mozgó átlag

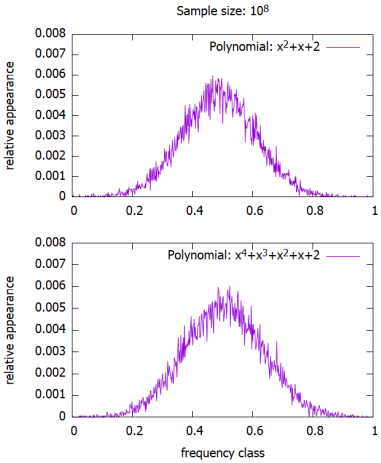
A sorozatokat véletlenszerűen választott egészekkel inicializáltam. A vizsgálatok során a mintavételnek a mérete 10^8 volt. A mag mérete 63-bites, és a generált sorozatból kinyert véletlenszámokat a mag 14-bites kezdőszeletére való redukcióval állítottam elő (a legalacsonyabb 49 bitet elhagytam). Az összegzési ablak mérete 100, ami azt jelenti, hogy 100 egymás utáni számot adtam össze.

Ezt követően a generátorok viselkedését vizsgáltam meg a véletlen bolyongás teszt során.



2. ábra. Véletlen bolyongás vizsgálat

A kapott sorozatokat kiegyensúlyoztam a 0 körül, az átlaggal való eltolás által. Az így kapott kiegyensúlyozott mintákból, kiszámoltam a sorozat kezdőszeletének teljes összegét. Azt vizsgáltam, hogy az így kapott teljes összeg 0 átmenetei közötti különböző távolságoknak mekkora a gyakorisága. Ennek eredménye a **2**. ábrán látható.
Végül, mozgó átlaghoz hasonlóan, a generált véletlen számokból képzett redukált - 14 bites - minták eloszlását vizsgáltam. A **3**. ábrákon azt láthatjuk, hogy mennyi a relatív gyakorisága annak az eseménynek, hogy egy adott redukált érték relatív gyakorisága pontosan t, azaz, hogy egy gyakorisági osztálynak mennyi a valószínűsége.



3. ábra. Relatív gyakoriság

A NIST teszt csomaggal (ld. [3]), két generátort vizsgáltam. Ezek az általánosított Neumann-féle négyzetközép módszerén alapulnak, ahol a számrendszer alapjául szolgáló polinomok x^2+x+2 és az $x^4+x^3+x^2+x+2$. Mindkét sorozatban 63 bites magot használtam.

Összehasonlítottam az eredményeimet két NIST beépített generátorával, az LCG és az SHA1 generátorokkal. Az összehasonlítás azt mutatja, hogy a GMSM sorozatok tulajdonságai a két beépített generátor között helyezkednek el. A NIST ajánlására hivatkozva elmondható, hogy mindkét generátor átment az összes teszten.

A **4**. táblázatban elemeztem azoknak a GMSM sorozatoknak a periodicitási tulajdonságát, amelyek az **3**. táblázatban szerepelnek. A táblázatban 6 generátor látható, az első blokkban az eredeti sorozat van. A blokkok elemei: definiáló polinom, a ciklusok száma, a leghosszabb periódus és az egyhosszú ciklusok száma. Az oszlopokban a sorozat elemek méretét adtam meg.

Fontos megjegyezni, hogy még akkor is használhatók a sorozatok a generátorokhoz, ha rövid periódushosszuk van, a hosszú előperiódusuk miatt. A magok méretének növelése növeli a periódushosszt és a leghosszabb periódus hosszát, de nem monoton módon. Azt jelenti, hogy attól, hogy növeljük a mag méretét, még nem biztos, hogy minden esetben nőni fog a periódushossz. Előfordulhat, hogy nagyobb magméret esetén mégiscsak kisebb a periódushossz, de általánosan elmondható, hogy a növekvő magmérettel nő a periódushossz is.

Nontrivial cycles

Digits (seed)	10	11	12	13	14	15	16	17	18	19	20

Defining polynomial: $x - 2$

Cycles	4	4	6	4	9	12	12	10	11	6	12
Max period length	5	5	10	2	56	70	111	203	197	2	142
Stability points	2	3	3	3	3	3	4	4	6	5	6

Defining polynomial: $x + 2$

Cycles	2	6	7	7	11	12	16	11	13	18	18
Max period length	3	3	2	34	10	27	51	30	2	39	4
Stability points	1	3	4	3	5	4	6	5	8	9	8

Defining polynomial: $x^2 + x + 2$

Cycles	3	4	4	2	4	6	3	3	4	9	7
Max period length	2	2	10	19	10	13	34	21	13	256	476
Stability points	2	3	1	1	1	1	1	1	2	2	2

Defining polynomial: $x^2 + 2x + 2$

Cycles	2	4	6	5	5	7	5	4	7	12	13
Max period length	1	2	2	5	5	11	20	2	7	24	117
Stability points	2	3	4	2	2	2	2	3	5	9	8

Defining polynomial: $x^3 + x^2 + x + 2$

Cycles	10	13	6	6	3	1	5	6	7	11	5
Max period length	5	5	9	5	1	1	7	67	20	165	57
Stability points	8	10	4	5	3	1	3	3	3	3	1

Defining polynomial: $x^4 + x^3 + x^2 + x + 2$

Cycles	5	8	6	5	5	7	10	11	6	6	8
Max period length	13	19	4	12	83	22	57	54	270	125	258
Stability points	2	1	3	3	3	3	6	7	2	2	3

4. táblázat. Periodicitás

3. Műveletvégzés kanonikus számrendszerekben

A kanonikus számrendszerek műveletének néhány tulajdonságának a vizsgálatával foglalkoztam algebrai egész számok felett. Többek között bebizonyítottam egy véges automatának a létezését, amely elvégzi az összeadási műveletet egy adott kanonikus számrendszerben, és algoritmust írtam a minimális összeadó automatának meghatározására, ha a számrendszer egy bináris kanonikus számendszer. Az összeadás tulajdonságai alapján egy olyan algoritmust konstruáltam, amely $O(n \cdot 2^n)$ időben felsorolja az összes n hosszúságú algebrai egész számok négyzetét. Végezetül ismertetem ezeknek az automatáknak a méretét, azaz az állapotok számát, melyeket vizsgáltam.

3.1. A kanonikus számrendszerrel kapcsolatos definíciók és eredmények

A bizonyítások megtalálhatók a disszertációban.

3.1. Definíció. *Legyen $P \in \mathbb{Z}[x]$ főegyüttható polinom, $R = \mathbb{Z}[x]/P\mathbb{Z}[x]$ a megfelelő faktorgyűrű, $\varphi : \mathbb{Z}[x] \to R$ az a homomorfizmus, amelyikre $\varphi(A) = A \mod P$. Továbbá legyen $m > 1$, $\mathcal{N} = n_1, \ldots, n_m \subseteq \mathbb{Z}$, és $\mathcal{N}[x] \subseteq \mathbb{Z}[x]$ azon polinomok halmaza, melyeknek minden együtthatója $\mathcal{N}$-ből származik. A $(P, \mathcal{N})$ párt számrendszernek nevezzük, ha bármely $A \in \mathbb{Z}[x]$ esetén létezik $B \in \mathcal{N}[x]$, amelyikre $\varphi(A) = \varphi(B)$.*

Ha $\mathcal{N} = 0,1,\ldots,m-1$, akkor a számrendszer kanonikusnak, és $P(x)$-et CNS polinomnak nevezzük.

Az A polinom P általi reprezentációjának hossza

$$L(A) = L(A, P) = \deg(B) + 1.$$

3.2. Tétel. *Legyen $P \in \mathbb{Z}[x]$ egy d fokú egy fő együtthatós polinom, és $P(x) = p_d x^d + \cdots + p_0$, ahol $1 = p_d \leq p_{d-1} \leq \leq \ldots \leq p_0$, valamint $2 \leq p_0$. Ha P nem osztható körosztási polinommal, akkor P egy CNS polinom.*

3.3. Tétel. *Ha $P(x)$ egy CNS polinom, akkor $P(x)$ összes gyöke a zárt egységkörön kívül helyezkedik el, és $P(x)$ összes valós gyöke -1-nél kisebb.*

3.4. Definíció. *Legyen $P(x)$ egy d fokú CNS polinom, és $\alpha_1, \ldots, \alpha_d$ a P gyökei (melyek nem feltétlenül különbözőek), valamint $A(x) \in \mathbb{Z}[x]$. A relatív információ mennyiségét A-nak a következőképpen definiáljuk:*

$$\mathbb{I}(A) = \mathbb{I}_P(A) = \max_{\substack{1 \leq i \leq d \\ A(\alpha_i) \neq 0}} \frac{\log |A(\alpha_i)|}{\log |\alpha_i|} \ .$$

Speciális esetben, ha P osztója A-nak, akkor legyen $\mathbb{I}(A) = 0$.

3.5. Lemma. *Legyen $P(x)$ egy CNS polinom és $A(x), B(x) \in \in \mathbb{Z}[x]$. Ekkor*

i) $0 \leq \mathbb{I}(A)$

ii) Ha $\varphi(A) = \varphi(B)$, akkor $\mathbb{I}(A) = \mathbb{I}(B)$.

iii) $\mathbb{I}(A + B) \leq \max \mathbb{I}(A), \mathbb{I}(B) + \mathbb{I}(2)$*;*

iv) $\mathbb{I}(A \cdot B) \leq \mathbb{I}(A) + \mathbb{I}(B)$.

v) $\mathbb{I}(x \cdot A) = \mathbb{I}(A) + 1$.

vi) $\mathbb{I}(A^n) = n\mathbb{I}(A)$.

3.6. Tétel. *Ha α egy olyan komplex szám, amelyiknek abszolút értéke 1-nél nagyobb és nem pozitív valós, akkor létezik $c < 1$ konstans, ami csak és kizárólag α-tól függ, amire igaz, hogy bármely egész N és $a_0, a_1, \ldots$ sorozat esetén, ahol $0 \leq a_i \leq N$ minden $i = 0,1,\ldots$-re, teljesül az alábbi egyenlőtlenség:*

$$\left| \sum_{i=0}^{k-1} a_i \alpha^i \right| \leq cN \frac{|\alpha|^k}{|\alpha| - 1} \ ,$$

minden $1 < k \in \mathbb{Z}$ esetén.

3.7. Következmény. *Legyen $\alpha \in \mathbb{C}$, ahol $1 < |\alpha|$, és tegyük fel, hogy α nem pozitív valós szám. Legyen $0 < N \in \mathbb{Z}$, és*

$$M_n = \max \left\{ \left| \sum_{i=0}^{n-1} b_i \cdot \alpha^i \right| 0 \leq b_0, b_1, \ldots, b_{n-1} \leq N, \text{egészek} \right\} \ ,$$

minden $0 < n \in \mathbb{Z}$-re. Ekkor létezik $c < 1$, ami csak α-tól függ, úgy hogy

$$\lim_{n \to \infty} \frac{M_n}{|\alpha|^n} = \frac{cN}{|\alpha| - 1} \ .$$

Az előző eredmények alapján egy egzakt algoritmust valósíthatunk meg a c érték meghatározására a **3.6** tételben.

3.8. Tétel. *Legyen $P(x) = x^d + p_{d-1}x^{d-1} + \cdots + p_0$ egy CNS polinom. Ekkor léteznek hatékonyan kiszámítható c_1 és c_2 konstansok úgy, hogy az alábbi egyenlőtlenség teljesül minden $A(x) \in \mathbb{Z}[x]$ esetén:*

$$\mathbb{I}(A) - c_1 \leq L(A) \leq \mathbb{I}(A) + c_2 \tag{3.1}$$

3.2. Összeadó automata kanonikus számrendszerekben

Konstanssal való összeadás és szorzás megvalósítható automata segítségével kanonikus számrendszerekben. Az ilyen automata létezése nem nyilvánvaló, mivel két algebrai egész összegének hossza a CNS-ben rövidebb lehet, mint az összeadandók hossza. A disszertációban bizonyításra került, hogy létezik a számjegyenkénti összeadást végrehajtó véges automata.

3.9. Definíció. *Az* $\mathcal{A} = (k, Q, X, Y, s, \delta)$ *hatost véges (jelátalakító) automatának nevezzük,* k *bemenő szalaggal, ha*

$k \in \mathbb{Z}^+$,

Q *véges, nem üres halmaz; állapotok halmaza,*

X *véges, legalább* 2 *elemű halmaz; bemenő ábécé,*

Y *véges, legalább* 2 *elemű halmaz; kimenő ábécé,*

$s \in Q$ *kezdő állapot,*

$\delta : Q \times X^k \to Q \times Y$ *; átmenetfüggvény.*

3.10. Tétel. *Legyen* $(P, \mathcal{N})$ *egy kanonikus számrendszer. Ekkor létezik egy* $\mathcal{A}$ *véges átalakító automata két bemeneti szalaggal, amelyre* $\bar{c} = \mathcal{A}(\bar{a}, \bar{b})$ *minden* $\bar{a}, \bar{b}, \bar{c} \in \mathcal{N}^*$ *esetén, ha* $\varphi(Q_c) = \varphi(Q_a + Q_b)$.

3.11. Tétel. *A* ***3.10.*** *tételben meghatározott automata minimális.*

3.12. Tétel. *Legyen* $(P, \mathcal{N})$ *egy kanonikus számrendszer. Ekkor nem létezik olyan* S *kétszalagos véges átalakító automata, amelyre* $\bar{c} = S(\bar{a}, \bar{b})$ *minden* $\bar{a}, \bar{b}, \bar{c} \in \mathcal{N}^*$ *esetén, feltéve, hogy* $\varphi(Q_c) = \varphi(Q_a \cdot Q_b)$.

3.3. Az összeadó automata állapotainak becslése

Ahogy azt korábban már megvizsgáltam, az összeadó automata állapotai a lehetséges átvitelek. Ezeket az állapotokat polinomok reprezentálják $\mathcal{N}[x]$-ben. Mivel az állapotpolinomok fokszámai korlátozottak a **3.8.** tételben szereplő $c = \mathbb{I}(2) + c_2$ konstans által, így az alábbi egyszerű eredményt kaphatjuk.

3.13. Tétel. *Legyen $(P, \mathcal{N})$ egy kanonikus számrendszer, és legyen $\mathcal{A}$ az összeadást végző minimális automata. Ekkor $\mathcal{A}$ állapotainak száma $|\mathcal{N}|^c$-nel korlátozott.*

Különböző kanonikus számrendszerbeli polinomok esetén végzett kísérletem során az összeadó automaták konstrukcióját vizsgáltam. A polinomok Kovács-típusúak voltak és különböző fokúak, ahol az együtthatók a következő tulajdonsággal rendelkeztek: $1 = p_d \leq p_{d-1} \leq \ldots \leq p_0 = 2$. Egyes esetekben meghatároztam az automaták állapotainak számát.

A **5.** táblázatban az adott fokszámú kanonikus polinomokhoz tartozó összeadó automaták állapotainak lehetséges számának maximumát és minumumát láthatjuk.

deg	Min. NoS	Max. NoS
3	62	125
4	239	1195
5	1185	13037
6	6248	155765
7	33387	2041141
8	185695	27270180
9	1074944	370005950
10	6419041	$\geq$ 380278058

5. táblázat. Állapotok száma

Eltérően a megszokottól, a további számításokat nem az alkalmazott algoritmus idő-, hanem tárbonyolultsága korlátozza.

4. Négyzetek hosszának elemzése

Vizsgáltam a kanonikus számrendszerek műveletének néhány tulajdonságát algebrai egész számok felett. Az összeadás tulajdonságai alapján egy olyan algoritmust konstruáltam, amely $O(n \cdot 2^n)$ időben felsorolja az összes n hosszúságú algebrai egész számok négyzetét. A konstrukció felhasználásával elemeztem néhány négyzetre emelési tulajdonságot bináris kanonikus számrendszerben.

A GMSM tulajdonságainak elemzéséhez szükség van arra, hogy megvizsgáljuk a négyzetszámok általános tulajdonságait az öszszes négyzetszámra vonatkoztatva. Ehhez elő kell állítani egy adott mérettartományon belül az összes olyan számot, amely egy algebrai egésznek a négyzete. A racionális egészek körében használható az a formula, amely szerint
$(n+1)^2 = n^2 + (2n+1)$.

Ilyen módon a következő négyzetszámot szorzás helyett egy egyszerű összeadással tudjuk előállítani az előzőből. Így az egymás utáni előállítás sokkal kevesebb műveletet igényel. Ezt a módszert közvetlenül nem lehet alkalmazni az általánosított számrendszerekben. Helyette az eggyel való növelés műveletét egy másik egyszerűen reprezentálható műveletre cseréljük, amellyel a számokat úgy tudjuk felsorolni, hogy minden lépésben csak egy számjegyet változtatunk meg. Ez a módszer a bináris számok esetén a közismert Gray-kód. [1] Ilyen módon az n+a alakú szám négyzetét tudjuk könnyen meghatározni az n négyzetéből. A formula ennek megfelelően $(n+a)^2 = = n^2 + (2an + a^2)$. Bináris rendszerek esetén a $2an$ kiszámolása lényegében egy kettővel való szorzás és egy a-nak megfelelő eltolással számolható.

Olyan eljárás programozásával foglalkoztam, amely egy adott általánosított számrendszerben felsorolja az összes rögzített n-től rövidebb szám négyzetét.

Sikerült felsorolnom a négyzetszámokat szorzás nélkül, kizárólag az összeadás műveletére támaszkodva. A négyzetszámok felsorolása gyorsabb, mint az egyes számok négyzetének kiszámítása.

Továbbá, ellenőriztem, hogy a négyzetek hossza mennyiben tér el az elvárt $2n$-től. Az elemzés a disszertációban található. Az n a négyzetre emelt szám hossza. Igazából ez egy általános elemzés volt, négyzetek hosszának az elemzése konkrét polinomok esetén (gyakorlati közelítő eredményeket sorolja fel).

5. Hitelesített Publikációs Lista

DEBRECENI EGYETEM
EGYETEMI ÉS NEMZETI KÖNYVTÁR
H-4002 Debrecen, Egyetem tér 1, Pf.: 400
Tel.: 52/410-443, e-mail: publikaciok@lib.unideb.hu

Nyilvántartási szám: DEENK/69/2025.PL
Tárgy: PhD Publikációs Lista

Jelölt: Padányi Viktória
Doktori Iskola: Informatikai Tudományok Doktori Iskola
MTMT azonosító: 10072238

A PhD értekezés alapjául szolgáló közlemények

Idegen nyelvű tudományos közlemények hazai folyóiratban (1)

1. **Padányi, V.**, Herendi, T.: Generalized Middle-Square Method.
   Ann. Math. Inform. 56, 95-108, 2022. ISSN: 1787-5021.
   DOI: http://dx.doi.org/10.33039/ami.2022.12.003
   IF: 0.3

Idegen nyelvű tudományos közlemények külföldi folyóiratban (2)

2. Herendi, T.*, **Padányi, V.***: Automata and Arithmetics in Canonical Number Systems.
   Algorithms. 18 (3), 1-22, 2025. EISSN: 1999-4893.
   DOI: http://dx.doi.org/10.3390/a18030122
   * Megosztott első szerzős közlemény.
   IF: 1.8 (2023)

3. **Padányi, V.**, Herendi, T.: A study on comparison of pseudorandom number generator.
   Int. J. Math. Comp. Eng. 1 (1), 25-44, 2023. EISSN: 2956-7068.
   DOI: http://dx.doi.org/10.2478/ijmce-2023-0003

DEBRECENI EGYETEM * Egyetemi és Nemzeti Könyvtár *

DEBRECENI EGYETEM
EGYETEMI ÉS NEMZETI KÖNYVTÁR
H-4002 Debrecen, Egyetem tér 1, Pf.: 400
Tel.: 52/410-443, e-mail: publikaciok@lib.unideb.hu

Idegen nyelvű konferencia közlemények (1)

4. **Padányi, V.**, Herendi, T.: Metaanalysis of pseudorandom number generators. In: Tavaszi Szél 2020 Konferencia = Spring Wind 2020 : Tanulmánykötet. Szerk.: Szabó Csaba, Doktoranduszok Országos Szövetsége, Budapest, 474-486, 2020. ISBN: 9786155586729

A közlő folyóiratok összesített impakt faktora: 2,1
A közlő folyóiratok összesített impakt faktora (az értekezés alapjául szolgáló közleményekre): 2,1

A DEENK a Jelölt által a Tudóstérbe feltöltött adatok bibliográfiai és tudománymetriai ellenőrzését a tudományos adatbázisok és a Journal Citation Reports Impact Factor lista alapján elvégezte.

Debrecen, 2025.02.24.

DEBRECENI EGYETEM * Egyetemi és Nemzeti Könyvtár *

Short thesis for the degree of doctor of philosophy (PhD)

Construction of Pseudorandom Number Generators

by Viktória Padányi

Supervisor: Dr. Herendi Tamás

University of Debrecen
Doctoral School of Informatics

Debrecen, 2024

Introduction

Random numbers are used in various fields of science. Their usage is a must wherever random test data or random samples are required for instance artificial intelligence applications, scientific simulations, randomized algorithm, or cryptography. Two methods are possible to generate a sequence of random numbers: generating with Pseudorandom Number Generators (PRNG) and generating with True Random Number Generators (TRNG). PRNGs are using some recursive algorithm to compute the elements of the sequence from the previous values. There are theoretical and practical recurrences: in the theoretical case the algorithms use operations on arbitrary precision numbers (or on an infinite set), while in the practical case we use operations on fixed precision numbers (i.e., the base structure is a finite set, usually with some algebraic structure).

In both cases natural requirement is the uniform distribution, what we defined only for the finite case. However, the sequences are completely predictable and highly losslessly compressible by definition. It is not possible for these schemes to generate TRNs with information-theoretically provable randomness.

As the first step of my research, I gathered several widely-used theoretical and practical PRNGs with uniform distributions. I categorized the generators based on their properties and the tests and examinations commonly used. I delved deeper into the listed generators using the NIST test suite. The relevant results were published in [4], and the improvement of these results with the experimental achievements were published in [6].

I continued my research work by analyzing PRNGs based on the Middle Square Method. I performed a detailed analysis of the generators belonging to the group of MSMs, and I studied what important properties they have and what their beneficial properties are. I focused on John von Neumann's MSMs. This method is now primarily of historical significance, as it is no longer in use due to the limitations of the sequences it generates. These sequences have extremely short period lengths, which means that after a certain number of iterations, the method will produce either the same number repeatedly or cycle back to a previously generated number, ultimately leading to a repeating pattern or converging to zero.

One can start with an n-digit number (seed), which becomes the first element of the sequence. The generator seed is squared, and then the least and most significant digits are removed, leaving only the middle part of the digits of the squared number. This middle part is then used as the subsequent seed for the next iteration of the method.

After that, I continued my research with generalized number systems. I generalized John von Neumann's MSM to canonical number systems (CNSs). The number system has two digits (0,1), and the sequence is defined in a similar way: the starting point is a properly chosen m-bit initial value. Square the seed, cut out the middle m-bit, and this is the next seed of the generator. The arithmetics in these number systems are similar to the arithmetics in rational integers with the classical digit representation. However, the calculation of the next digit requires a more difficult reduction operation. I used an example to derive this calculation.

The exact definition of the generalized version of MSM binary CNS and the analysis properties of these generators are described in the dissertation. I tested the generators with the NIST statistical test suite. The collected most essential results will be presented. The results have been published in [5].

Another topic of my research was related to automata, which is summarized in the dissertation. I discussed some properties of arithmetic in canonical number systems (CNSs) over algebraic integers. I defined the information quantity of a polynomial $A \in \mathbb{Z}[x]$ relative to the base of the CNS and proved that it has a strong relation with the length of the representation in the number system. Based on this result, I showed that for every CNS polynomial P, there exists a finite transducer automaton executing the addition operation of polynomials in canonical representation of base P. Finally, I observed the size – i.e., the number of states – of such automata. The results have been under publishing in [2].

I have proven, among other things, the existence of a transducer automaton that performs the addition operation in a given canonical number system, and I have provided an algorithm to determine the minimal addition automaton if the number system is binary CNS. Based on the properties of addition, I have developed an algorithm using Gray code to count the all square numbers of length n of algebraic integers in $O(n \cdot 2^n)$ time. Using the construction, I analyzed some properties of squaring in binary CNS. I focused on the squaring, because it was one of the most important operation. Therefore, I applied these estimates to the square of the numbers and determined in some cases analytical bounds. Henceforth, I studied with the primary aim of being able to enumerate the square numbers and investigate how their lengths actually behave, that is, how their lengths are related to the constraints. In the last Chapter of the dissertation, I am going to present the algorithm and their observations.

1 Pseudorandom Number Generators

I collected some theoretical and practical PRNGs with uniform distribution. In my dissertation, I focused mainly on practical PRNGs. The practical effectiveness of theoretical pseudorandom number generators is limited, so I did not perform a detailed analysis, as they are highly representation-dependent. Practical PRNGs can be classified into several categories, which use recursive relations with limited depth. As a result, these sequences are periodic. Most of them use modular arithmetic as a basic operation to calculate the elements of the sequences.

1. PRNGs based on recursions with modular arithmetic
   a) Homogeneous linear methods
   b) Non-homogeneous linear methods
   c) Hybrid methods
   d) PRNGs based on non-linear recursion
2. PRNGs based on other recursion methods
   a) MSM-based generators
   b) Shuffling methods
3. Miscellaneous methods

I considered further analytical attributes of the generators, such as period length, computational complexity, and memory usage. The properties of the analyzed PRNGs are presented in Table **6**.

Since there is a wide range of platforms they are implemented on, it is hard to compare the speed of the generators. Some perform better on constrained devices, while others are more efficient on CISC processors. Owing to this duality, instead of the speed of the algorithms, I have highlighted the number of complex (slow) and simple (fast) operations. The available implementations for most of the generators can be found in [6]. Those without released implementations are either weak or very simple.

Generators	Period length[1]	Operations[2]		Memory (bits)
		Slow	Fast	
Lehmer's MLCG	2^s	1 mul, 1 mod		$2s$
GFSR	2^k		s-bit xor	$k \cdot s$
TGFSR	2^{ks}	$s \times s$-bit m.mul	s-bit xor	$k \cdot s + s^2$
LCG	2^s	1 mul, 1 mod	1 sum	$4s$
LRS[3]	2^{k+s}		k sum	$k \cdot s$
MT[4]	2^{ks}	$s \times s$-bit m.mul	s-bit xor	$k \cdot s + s^2$
WELL[5]	2^{ks}	8 $s \times s$-bit m.mul	9 s-bit op	
W-H algorithm	2^{3s}	3 LCG, 3 div	2 sum	$12s$
ACORN	2^{ks}	k mod	k sum	$k \cdot s$
PCG	2^{128}	mul	s-bit perm,xor	$4s$
QCG	s^s	4 mul, 1 mod	2 sum	$4s$
BBS	2^s	1 mul, 1 mod		$2s$
Multipl. Fibonacci	2^{2s}	1 mul, 1 mod		$3s$
Power Congr. Gen.	2^s	s mul, 1 mod		$4s$
ICG	2^s	1 mul, 1 inv, 1 mod	1 sum	$4s$
Neumann's MSM[6]	8^d	1 mul	1 xor, 1 shr	$4d$
Metropolis MSM[7]	2^s	1 mul	1 xor, 1 shr	s
Coveyou's SMSM	2^s	1 mul		s
GMSM	2^s	1 mul	1 xor, 1 shr	s
MS Weyl	2^{2s}	1 LCG, 1 mul	1 sum	$5s$
Algorithm M	2^{2s}	2 gen, 1 mul, 1 div		$2 \cdot * + k \cdot s$
Algorithm B	2^s	1 gen, 1 mul, 1 div		$* + k \cdot s$
RC4	2^{1024}		7 byte op	256 bytes
Xorshift	2^s		1 xor, 1 shr	s
Legendre symbol	2^s	s mod		s
Enc.-based methods[8]	2^s	*	*	$2s$

Table 6: General properties of the generators

The entries of the table are expressed in terms of the parameters of the generators:

- s: the length (bit size) of the data (seed). If the generator under consideration is based on modular arithmetics, then for the used modulus $m \approx 2^s$.
- d: the decimal length of the data. Some of the early generators are defined in decimal representation.
- k: the farthermost previous seed value used to determine the next one or the dimension of the seed vector.

In Table **6**, I assumed the following:

1. in the column *Period length*, the best (largest) possible values are shown;

2. the column *Operations* displays the complexity of the algorithms, i.e., the number of necessary operations to calculate a single random sample. Operations use the given size of data (s bit, or d decimal digit).

3. The complexity of the LRS generator can be reduced to k additions by choosing the proper parameters.

4. With the widely used parameters, the period length of the MT is $2^{19937} - 1$.

5. With the widely used parameters, the period length of the WELL is 2^{44497}.

6. Neumann's MSM is originally defined for decimal numbers.

7. The original construction of Metropolis with 20-bit numbers has maximal period length of 142.

8. The time complexity depends on the underlying cryptographic algorithms.

Table **7** explains the operations referred in Table **6**.

Slow operations		Fast operations	
mul	multiplication	sum	addition
mod	modular reduction	xor	bitwise xor
div	division	shl, shr	bitwise left and right shift
m.mul	multiplication of a vector by a matrix	op	memory operations
inv	modular inversion	perm	bitwise permutation
gen	sample from an embedded generator		
LCG	sample from an LCG generator		

Table 7: Operations

Before using a PRNG, we need to know whether it is suitable for the given purpose. Knowing the underlying algorithm is usually not informative enough. The users often do not even care about the technical details. This is why it is important to have general performance observations, which can be realized by standard statistical tests. The different tests express the properties from different points of view. But it is still a question of which properties are expected for the application concerned. Statistical tests of PRNGs are usually applications of the general statistical tests for some expected properties of an imaginary perfect uniformly distributed true random sequence.

I provided a detailed overview of a statistical test suite designed and recommended by the NIST (National Institute of Standards and Technology) for testing random and pseudorandom number generators. My choice fell on this test suite because it is the standard. The NIST Test Suite consisting of 15 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software-based pseudorandom number generators. These tests focus on a variety of different types of non-randomness that could exist in a sequence.

1.1 Experimental results

I implemented the PRNGs discussed in the dissertation in C++ and ran on an Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz with 64GB RAM. I tested the obtained sequences with the NIST test suite in the same environment. The implementations and the detailed results are available at http://www.prng.hu .

In the tests, 10^9-long 0,1-bit sequences were used for each generator. Each sequence was split into 1000 equal-length subsequences to have reasonably large independent test cases. All fifteen tests were applied for every sequence.

The most essential results are collected in the NIST result tables in the dissertation. The significance level is set to 0.01.

The minimum pass rate for each statistical test – with the exception of the random excursion (variant) test – is approximately 0.981819 for a sample size of 1000 binary sequences.

The minimum pass rate for the random excursion (variant) test is approximately 0.979517 for a sample size of 609 binary sequences.

The generators which did not pass a particular test are marked by an * in the tables. For those tests which stand off several subtests with different parameters – cumulative sums, non-overlapping template, random excursions, random excursions variant, serial – I displayed the average of the results instead.

As a result of the observations, most of the discussed PRNGs have successfully passed all tests within the NIST test battery. However, in particular cases, I encountered some interesting behavior.

In this compilation, I did not find similar analyses performed with the NIST Test Suite under the same conditions and parameters for the generators mentioned in this Capter. The generators are well-known; I aimed to review various sequences from classic (early) generators to modern generators with specific applications. My goal was to conduct a comprehensive analysis (including highlighting the number of operations of the generators) to see how these generators, which operate according to their definitions, perform when tested with the NIST Test Suite.

The most surprising finding is that the LCG and Lehmer's MLCG showed good scores in the tests, while both of them are proven to be cryptographically insecure. One can find an opposite behavior for the BBS, which performed weak on the tests, but it has proven to be a secure one.

Moreover, for the generators based on von Neumann's MSM, I had to increase the number of digits of their seeds considerably to achieve sequences passing the tests. For instance, the seed of the original Neumann's MSM was extended to 72-digit. In contrast, the improved GMSM generator, which is defined over algebraic number fields, showed good scores with significantly shorter seed length. For instance, the GMSM required only a 92-bit seed with similar properties, even for a relatively simple algebraic extension. The dissertation presented this generator in more detail.

Overall, this part of my research, emphasized the importance of the empirical testing and the potential discrepancies between theoretical guarantees and practical performance in the evaluation of PRNGs, which is important from the point of view of usage.

2 Generalized Middle-Square Method

I generalized John von Neumann's Middle-Square Method (MSM) for canonical number systems (CNSs). The dissertation also includes definitions, observations, and statistical tests for the generated sequences.

There exists a more general definition of number systems using lattices, but for the results I have achieved, this classical definition is entirely sufficient.

Definition 2.1. *Let R be an integral domain, $\alpha \in R$ and $\mathcal{N} = \{n_1, \ldots, n_m\} \subseteq \mathbb{Z}$. The pair $(\alpha, \mathcal{N})$ is called a number system in R, if any $\gamma \in R$ has a unique representation in the form $\gamma = \sum_{i=0}^{h} c_i \alpha^i$, where $c_i \in \mathcal{N}$ for all $0 \leq i \leq h$ and $c_h \neq 0$, if $h \neq 0$. The number system is called canonical, if $\mathcal{N} = \{0,1,\ldots,m-1\}$.*

I will use the notation $L(\gamma, \alpha, \mathcal{N}) = h + 1$, i.e. the length of the representation of γ in the number system $(\alpha, \mathcal{N})$.

Theorem 2.2. *Let $p \in \mathbb{Z}[x]$ be an irreducible polynomial with $\deg(p) = d$, and $p(x) = a_d x^n + \cdots + a_0$ such that $1 = a_d \leq a_{d-1} \leq \ldots \leq a_0$ and $2 \leq a_0$. Furthermore, let α be a root of p and $\mathcal{N} = \{0,1,\ldots,a_0-1\}$. Then $(\alpha, \mathcal{N})$ is a canonical number system in $\mathbb{Z}[\alpha]$.*

Theorem 2.3. *Let β be an algebraic integer of degree $d \geqq 1$, and let $(\alpha, \mathcal{N})$ be a number system in $\mathbb{Z}[\beta]$. Then there exist computable constants $C_1 = C_1(\alpha, \mathcal{N})$ and $C_2 = C_2(\alpha, \mathcal{N})$ depending only on α and $\mathcal{N}$, such that*

$$|\log|_\alpha \gamma + C_1 \leqq L(\gamma, \alpha, \mathcal{N}) \leqq |\log|_\alpha \gamma + C_2 \tag{2.1}$$

holds for every $0 \neq \gamma \in \mathbb{Z}[\beta]$.

Remark 2.4. *With the notations* $C_3 = C_1 - 2C_2$ *and* $C_4 = C_2 - 2C_1$*, I have*

$$2L(\gamma) + C_3 \leq L(\gamma^2) \leq 2L(\gamma) + C_4 \ . \tag{2.2}$$

In Table **8**, I have collected the results of how the sizes of the squares changed after squaring in the considered canonical number systems. The set of four CNSs is extended by the two rational binary number systems with bases 2 and -2.

I consider all the numbers of 20 to 30 digits. For a given length h, the table contains the distances of the minimal and maximal lengths of the squares from the expected $2h$. Additionally, the average lengths of the squares are presented. The last column displays the theoretical bounds for the corresponding values of distances. An important objective was to evaluate how closely the practical results align with the theoretical calculations. Additionally, I have looked at the specific lengths at which these bounds are reached. The findings indicate that even after refinement, the theoretical values differ significantly from the practical results. While the outcomes seem to approach a constant value. The aim was to assess, for instance, whether numbers of length 30, which are still practically relevant, come close to the theoretical bounds. A potential future goal would be to demonstrate, based on practical values, that these results are final. Further refinement of the theoretical bounds remains possible, particularly in light of practical measurements, which would also allow for the finalization of the practical results.

Length of base numbers

Digits	20	21	22	23	24	25	26	27	28	29	30	T
Defining polynomial: $x - 2$												
Decrease	1	1	1	1	1	1	1	1	1	1	1	1
Increase	0	0	0	0	0	0	0	0	0	0	0	0
Average	39.6	41.6	43.6	45.6	47.6	49.6	51.6	53.6	55.6	57.6	59.6	
Defining polynomial: $x + 2$												
Decrease	3	3	3	3	3	3	3	3	3	3	3	4
Increase	1	1	1	1	1	1	1	1	1	1	1	2
Average	38.9	40.9	42.9	44.9	46.9	48.9	50.9	52.9	54.9	56.9	58.9	
Defining polynomial: $x^2 + x + 2$												
Decrease	8	8	8	8	8	8	8	8	8	8	8	12
Increase	5	5	5	5	5	5	5	5	5	5	5	10
Average	39.6	41.6	43.6	45.6	47.6	49.6	51.6	53.6	55.6	57.6	59.6	
Defining polynomial: $x^2 + 2x + 2$												
Decrease	12	12	12	12	12	12	12	12	12	12	12	16
Increase	9	9	9	9	9	9	9	9	9	9	9	12
Average	40.6	42.6	44.6	46.6	48.6	50.6	52.6	54.6	56.6	58.6	60.6	
Defining polynomial: $x^3 + x^2 + x + 2$												
Decrease	11	14	14	14	14	14	14	14	14	14	14	31
Increase	12	12	12	12	12	12	12	12	12	12	12	27
Average	41.6	43.5	45.5	47.5	49.5	51.5	53.5	55.5	57.5	59.5	61.5	
Defining polynomial: $x^4 + x^3 + x^2 + x + 2$												
Decrease	17	18	18	18	18	18	20	20	20	20	20	46
Increase	21	21	21	21	21	21	21	21	21	21	21	32
Average	43.8	45.6	47.6	49.7	51.9	53.9	55.7	57.7	59.7	61.8	63.8	

Table 8: Lengths of squares

The generalization of the John von Neumann's Middle-Square Method to canonical number systems is the following:

Let $p(x) \in \mathbb{Z}[x]$ be an irreducible polynomial of degree d, and with coefficients $1 = a_d \leq a_{d-1} \leq \ldots \leq a_0 = 2$. The corresponding CNS has only 2 digits: 0 and 1. For the sake of simplicity, I will call the digits bits and the digit representation of algebraic integers in $\mathbb{Z}[\alpha]$ as a binary representation.

In the design of the generator, I use a seed of $m \in \mathbb{N}$ bits. Similarly, as it is done in the original construction, let u be a sequence over $\mathbb{Z}[\alpha]$ defined by the following:

$u_0 \in \mathbb{Z}[\alpha]$ is a random m-bit number;

if $n > 0$, let

$$u_{n-1}^2 = \sum_{i=0}^{h} b_i \alpha^i \ , \text{ with } b_h \neq 0 \ , \ t = \left\lfloor \frac{h-m}{2} \right\rfloor \text{ and}$$
$$u_k = \sum_{i=0}^{m-1} b_{i+t+1} \alpha^i \ .$$

The value of m should be chosen to be large enough, in particular such that $2m + C_3 > m$, i.e. $m > -C_3$, where C_3 is as defined above.

Another approach is if $t = \lfloor \frac{m}{2} \rfloor$, but then $\frac{m}{2} > -C_3$ should hold.

2.1 Experimental results

Figure **4** displays the distributions of the moving average of the sequences.

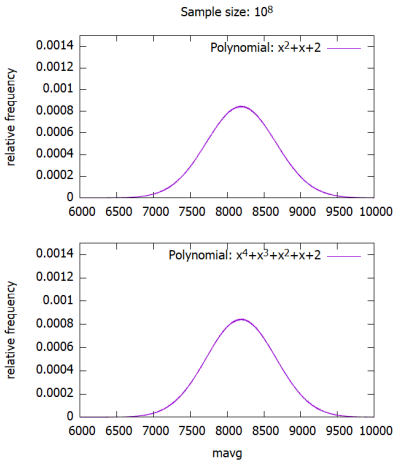


Figure 4: Moving average distribution

I have initialized the sequences with randomly chosen integers. The sizes of the samples are 10^8. The seeds are 63-bit words, and the pseudorandom values are obtained by a reduction to the 14-bit prefixes (the least significant 49 bits are eliminated). The length of the window for the summation is 100.

Next, I observed the generators' behavior under the random walk test.

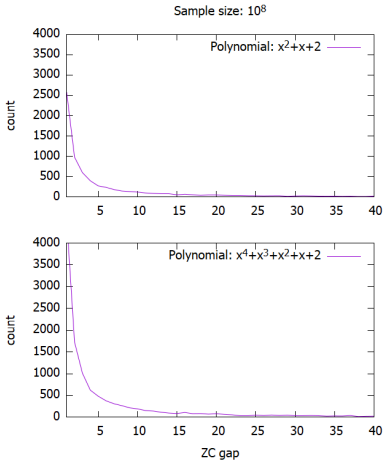


Figure 5: Random walk

The generated sequences are balanced around 0 by a shift with the mean value. The test calculates the frequency of the lengths of the gaps between consecutive zero crossings of f. The results are presented in Figure **5**.

Finally, I have investigated the distribution of the frequency classes. The values of the sequences are arranged into 2^{14} intervals of equal lengths. My objective is to describe the probability of the event when the same (reduced) random value appears exactly t times for a given t.

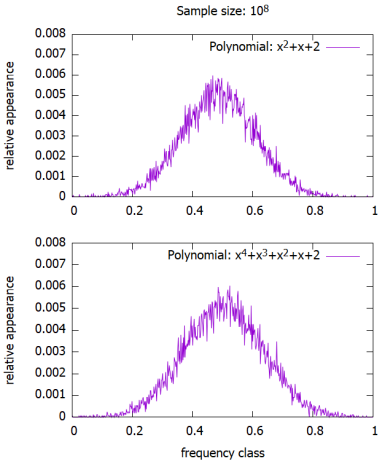


Figure 6: Frequency distribution

Figure **6** displays the distributions of the relative frequencies of the cardinalities of U_k.

I have tested two of my generators with the NIST Statistical Test Suite (c.f. [3]). The results are summarized in the dissertation. These two are the MSMs corresponding to the polynomials x^2+x+2 and $x^4+x^3+x^2+x+2$. I denote them by GMSM1 and GMSM2 in the tables, respectively. In both sequences, I have used a 63-bit seed.

I have compared the results with two of the NIST's built-in generators, the LCG and SHA1. The comparison shows that the properties of GMSM sequences are between the two built-in ones. A NIST recommendation indicates that both generators passed all the tests.

Last but not least, in Table **9**, I have collected the periodicity properties of the same GMSM sequences as in Table **8**.

Again, one block corresponds to the CNS given by the defining polynomial of its base.

The entries are:

– the number of disjoint cycles;
– the maximal length of the cycles;
– the number of the length-1 cycles.

The first block contains test results in the CNS with base 2, i.e., the simple binary representation of non-negative rational integers. In the second block, the number system is the extension of the previous one to the whole set of integers with base -2.

It must be added that even if they have small period lengths, the sequences can be used for pseudorandom number generators because of the long preperiod. Increasing the size of the seed typically increases both the period length and the length of the longest period, but not in a monotonous way. This means that with a larger seed size, the period length might sometimes be shorter; however, in general, as the seed size increases, the period length also tends to increase, although there are cases where it may decrease.

Nontrivial cycles

Digits (seed)	10	11	12	13	14	15	16	17	18	19	20

Defining polynomial: $x - 2$

Cycles	4	4	6	4	9	12	12	10	11	6	12
Max period length	5	5	10	2	56	70	111	203	197	2	142
Stability points	2	3	3	3	3	3	4	4	6	5	6

Defining polynomial: $x + 2$

Cycles	2	6	7	7	11	12	16	11	13	18	18
Max period length	3	3	2	34	10	27	51	30	2	39	4
Stability points	1	3	4	3	5	4	6	5	8	9	8

Defining polynomial: $x^2 + x + 2$

Cycles	3	4	4	2	4	6	3	3	4	9	7
Max period length	2	2	10	19	10	13	34	21	13	256	476
Stability points	2	3	1	1	1	1	1	1	2	2	2

Defining polynomial: $x^2 + 2x + 2$

Cycles	2	4	6	5	5	7	5	4	7	12	13
Max period length	1	2	2	5	5	11	20	2	7	24	117
Stability points	2	3	4	2	2	2	2	3	5	9	8

Defining polynomial: $x^3 + x^2 + x + 2$

Cycles	10	13	6	6	3	1	5	6	7	11	5
Max period length	5	5	9	5	1	1	7	67	20	165	57
Stability points	8	10	4	5	3	1	3	3	3	3	1

Defining polynomial: $x^4 + x^3 + x^2 + x + 2$

Cycles	5	8	6	5	5	7	10	11	6	6	8
Max period length	13	19	4	12	83	22	57	54	270	125	258
Stability points	2	1	3	3	3	3	6	7	2	2	3

Table 9: All cycles in GMSM sequences

3 Arithmetics in CNSs

In this Chapter, I am going to discuss some properties of arithmetics in CNSs over algebraic integers. I defined the information quantity of a polynomial $A \in \mathbb{Z}[x]$ relative to the base of the CNS and proved that it has a strong relation with the length of the representation in the number system. Based on this result, I am going to show that for every CNS polynomial P, there exists a finite transducer automaton executing the addition operation of polynomials in canonical representation of base P. Finally, I will present the observed size – i.e., the number of states – of such automata.

3.1 Definitions and results related to CNSs

The proofs can be found in the dissertation.

Definition 3.1. *Let $P \in \mathbb{Z}[x]$ be a monic polynomial, $R = \mathbb{Z}[x]/P\mathbb{Z}[x]$ be the corresponding quotient ring, and let φ: $\mathbb{Z}[x] \to R$ be the homomorphism $\varphi(A) = A \mod P$. Furthermore, let $m > 1$, $\mathcal{N} = \{n_1, \ldots, n_m\} \subseteq \mathbb{Z}$, and let $\mathcal{N}[x] \subseteq \mathbb{Z}[x]$ be the set of polynomials with all coefficients from $\mathcal{N}$. The pair $(P, \mathcal{N})$ is called a number system, if for any $A \in \mathbb{Z}[x]$ there exists a unique $B \in \mathcal{N}[x]$, such that $\varphi(A) = \varphi(B)$.*

The number system is called canonical, and $P(x)$ is called a CNS polynomial if $\mathcal{N} = \{0,1,\ldots,m-1\}$.

The length of the representation of A by P is $L(A) = L(A,P) = \deg(B) + 1$.

Theorem 3.2. *Let $P \in \mathbb{Z}[x]$ be a monic polynomial with $\deg(P) = d$, and $P(x) = p_d x^d + \cdots + p_0$ be such that $1 = p_d \leq p_{d-1} \leq \ldots \leq p_0$ with $2 \leq p_0$. If P is not divisible by a cyclotomic polynomial, then P is a CNS polynomial.*

Theorem 3.3. *If $P(x)$ is a CNS polynomial, then all roots of $P(x)$ lie outside of the closed unit circle, and all real roots of $P(x)$ are less than -1.*

Definition 3.4. *Let $P(x)$ be a CNS polynomial of degree d, $\alpha_1, \ldots, \alpha_d$ be the (not necessarily different) roots of P, and let $A(x) \in \mathbb{Z}[x]$. We define the relative information quantity of A by*

$$\mathbb{I}(A) = \mathbb{I}_P(A) = \max_{\substack{1 \leq i \leq d \\ A(\alpha_i) \neq 0}} \frac{\log |A(\alpha_i)|}{\log |\alpha_i|} .$$

In the special case $P|A$ we set $\mathbb{I}(A) = 0$.

Lemma 3.5. *Let $P(x)$ be a CNS polynomial and $A(x), B(x) \in \mathbb{Z}[x]$. Then*

i) $0 \leq \mathbb{I}(A)$

ii) *If $\varphi(A) = \varphi(B)$, then $\mathbb{I}(A) = \mathbb{I}(B)$.*

iii) $\mathbb{I}(A + B) \leq \max\{\mathbb{I}(A), \mathbb{I}(B)\} + \mathbb{I}(2)$;

iv) $\mathbb{I}(A \cdot B) \leq \mathbb{I}(A) + \mathbb{I}(B)$.

v) $\mathbb{I}(x \cdot A) = \mathbb{I}(A) + 1$.

vi) $\mathbb{I}(A^n) = n\mathbb{I}(A)$.

Theorem 3.6. *Let $\alpha \in \mathbb{C}$ with $1 < |\alpha|$, and assume that α is not a positive real number. Then there exists a constant $c < 1$ depending only on α such that for every $0 < N \in \mathbb{Z}$, and for any sequence $a_0, a_1, \ldots$ of integers satisfying $0 \leq a_i \leq N$ for all $i = 0,1,\ldots,$ we have*

$$\left| \sum_{i=0}^{k-1} a_i \alpha^i \right| \leq cN \frac{|\alpha|^k}{|\alpha| - 1} \ ,$$

for any $1 < k \in \mathbb{Z}$.

Corollary 3.7. *Let $\alpha \in \mathbb{C}$ with $1 < |\alpha|$, and assume that α is not a positive real number. Let $0 < N \in \mathbb{Z}$ and*

$$M_n = \max \left\{ \left| \sum_{i=0}^{n-1} b_i \cdot \alpha^i \right| 0 \leq b_0, b_1, \ldots, b_{n-1} \leq N, \textit{integers} \right\} \ ,$$

for all $0 < n \in \mathbb{Z}$. Then there exists $c < 1$ depending only on α such that

$$\lim_{n \to \infty} \frac{M_n}{|\alpha|^n} = \frac{cN}{|\alpha| - 1} \ .$$

Based on the previous results, a fast algorithm to find the value of c in Theorem **3.6** can be constructed.

Theorem 3.8. *Let $P(x) = x^d + p_{d-1}x^{d-1} + \cdots + p_0$ be a CNS polynomial. Then, there exist computable constants c_1, c_2 such that*

$$\mathbb{I}(A) - c_1 \leq L(A) \leq \mathbb{I}(A) + c_2 \tag{3.1}$$

holds for every $A(x) \in \mathbb{Z}[x]$.

3.2 Automata and canonical number systems

Adding and multiplying by a constant are implementable with automata in CNSs. The existence of such an automaton is not obvious since the length of the sum of two algebraic integers represented in the CNS may have a shorter length than the summands'. The dissertation contains the proof that there is an automaton that executes the digit-wise addition.

Definition 3.9. *The* 6*-tuple* $\mathcal{A} = (k, S, X, Y, s, \delta)$ *is called a finite transducer automaton with* k *input tape if*

$k \in \mathbb{Z}^+$;

S *is a finite, non-empty set of states;*

X *is a finite set of at least* 2 *elements (the input alphabet);*

Y *is a finite set of at least* 2 *elements (the output alphabet);*

$s \in S$ *is the initial state;*

δ*:* $S \times X^k \longrightarrow S \times Y$ *is a unique mapping.*

Theorem 3.10. *Let* $(P, \mathcal{N})$ *be a CNS. Then there exists a finite transducer automaton* $\mathcal{A}$ *with two input tapes, such that* $\bar{c} = \mathcal{A}(\bar{a}, \bar{b})$ *for all* $\bar{a}, \bar{b}, \bar{c} \in \mathcal{N}^*$ *if* $\varphi(Q_c) = \varphi(Q_a + Q_b)$.

Theorem 3.11. *The addition automaton, constructed in the Theorem* ***3.10*** *is minimal.*

Theorem 3.12. *Let* $(P, \mathcal{N})$ *be a CNS. Then there are no finite transducer automaton* S *with two input tapes, such that* $\bar{c} = S(\bar{a}, \bar{b})$ *for all* $\bar{a}, \bar{b}, \bar{c} \in \mathcal{N}^*$ *if* $\varphi(Q_c) = \varphi(Q_a \cdot Q_b)$.

3.3 Estimation of the number of states of an addition automaton

As I observed before, the states of the addition automaton are the possible carries. According to the construction, these are polynomials in $\mathcal{N}[x]$. Since the degrees of the state polynomials are bounded by the constant $c = \mathbb{I}(2) + c_2$ of Theorem **3.8**, the following simple result can be established.

Theorem 3.13. *Let $(P, \mathcal{N})$ be a CNS, and $\mathcal{A}$ be the minimal automaton implementing the addition. Then the number of states of $\mathcal{A}$ is bounded by $|\mathcal{N}|^c$.*

I made an experiment on the construction of addition automata for different CNS polynomials. I determined and counted the number of states for different degree binary Kovács-type CNS polynomials. (i.e., the coefficients of P have the property $1 = p_d \leq p_{d-1} \leq \ldots \leq p_0 = 2$.) The results are collected in the following table.

deg	Min. NoS	Max. NoS
3	62	125
4	239	1195
5	1185	13037
6	6248	155765
7	33387	2041141
8	185695	27270180
9	1074944	370005950
10	6419041	$\geq$ 380278058

Table 10: Number of states

The further computations are limited unusually not by the time but by the space complexity of the applied algorithm and data representation.

4 Observation of the length of the squares

The existence of a transducer automaton that executes the addition operation in a given CNSs has been proven. Furthermore, an algorithm has been provided for determining the minimal addition automaton if the number system is binary CNS. I constructed an algorithm for enumerating all square numbers of length n of algebraic integers in $O(n \cdot 2^n)$ time. By employing this method, I have analyzed various properties of squaring within binary CNSs.

For understanding the properties of sequences generated by GMSM, it is essential to comprehend the behavior of squaring operations within CNSs. Consequently, I will present a method for listing and analyzing all squares of fixed-length numbers within the given binary CNSs. To analyze the properties of generators, it was imperative to investigate the general properties of square numbers applied to all algebraic integers. This was achieved by generating all numbers within a given size range that are the squares of algebraic integers. While the formula $(n+1)^2 = n^2 + (2n+1)$ is applicable among rational integers, it cannot be directly utilized in generalized number systems. Instead, I have replaced the operation of incrementing by 1 with another simpler operation, enabling the enumeration of numbers by changing only one digit at each step. In the case of binary systems, this operation is represented by the well-known Gray code. [1] Thus, I could efficiently determine the square of a number of the form $n+a$ from the square of n. Accordingly, the formula becomes $(n+a)^2 = n^2 + (2an + a^2)$.

Additionally, I have proven the existence of a transducer automaton that performs the addition operation within a given canonical CNS, along with providing an algorithm to determine the minimal addition automaton in the case of binary CNS.

Using the properties of addition, I have developed an algorithm utilizing the Gray code to enumerate all square numbers of length n of algebraic integers in $O(n \cdot 2^n)$ time. This construction facilitated the analysis of various properties associated with squaring within binary CNSs.

As a result of this research, I have successfully enumerated all squares of length n in $O(n \cdot 2^n)$ time.

I managed to list the number of squares without multiplication, relying solely on addition. I have verified how the length of squares differs from the expected $2n$, where n is the length of the base number. The analysis, which is detailed in the dissertation, involves an analytical calculation of the maximum possible change in the length of the square numbers, and it exactly calculates this for a few polynomials with limited size. Therefore, the analysis of the lengths of squares for specific polynomials was a general examination that includes the listing of practical approximations and results.

5 Certified Publication List

UNIVERSITY of DEBRECEN

UNIVERSITY AND NATIONAL LIBRARY
UNIVERSITY OF DEBRECEN
H-4002 Egyetem tér 1, Debrecen
Phone: +3652/410-443, email: publikaciok@lib.unideb.hu

Registry number: DEENK/69/2025.PL
Subject: PhD Publication List

Candidate: Viktória Padányi
Doctoral School: Doctoral School of Informatics
MTMT ID: 10072238

List of publications related to the dissertation

Foreign language scientific articles in Hungarian journals (1)

1. **Padányi, V.**, Herendi, T.: Generalized Middle-Square Method.
   Ann. Math. Inform. 56, 95-108, 2022. ISSN: 1787-5021.
   DOI: http://dx.doi.org/10.33039/ami.2022.12.003
   IF: 0.3

Foreign language scientific articles in international journals (2)

2. Herendi, T.*, **Padányi, V.***: Automata and Arithmetics in Canonical Number Systems.
   Algorithms. 18 (3), 1-22, 2025. EISSN: 1999-4893.
   DOI: http://dx.doi.org/10.3390/a18030122
   * These authors contributed equally to this work.
   IF: 1.8 (2023)
3. **Padányi, V.**, Herendi, T.: A study on comparison of pseudorandom number generator.
   Int. J. Math. Comp. Eng. 1 (1), 25-44, 2023. EISSN: 2956-7068.
   DOI: http://dx.doi.org/10.2478/ijmce-2023-0003

Foreign language conference proceedings (1)

4. **Padányi, V.**, Herendi, T.: Metaanalysis of pseudorandom number generators.
   In: Tavaszi Szél 2020 Konferencia = Spring Wind 2020 : Tanulmánykötet. Szerk.: Szabó Csaba, Doktoranduszok Országos Szövetsége, Budapest, 474-486, 2020. ISBN: 9786155586729

Total IF of journals (all publications): 2,1
Total IF of journals (publications related to the dissertation): 2,1

DEBRECENI EGYETEM Egyetemi és Nemzeti Könyvtár

The Candidate's publication data submitted to the Tudóstér have been validated by DEENK on the basis of the Journal Citation Report (Impact Factor) database.

24 February, 2025

Tézisfüzet irodalomjegyzéke / References

[1] F. Gray. *Pulse Code Communication.* US patent #2,632,058, 1953.

[2] T. Herendi and V. Padányi. Automata and arithmetic in canonical number systems. *in publishing*, 2024.

[3] National Institute of Standards and Technology. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications: NIST SP 800-22, 2012. URL https://www.nist.gov/publications/statistical-test-suite-random-and-pseudorandom-number-generators-cryptographic.

[4] V. Padányi and T. Herendi. Metaanalysis of pseudorandom number generators. *23rd annual Spring Wind conference*, 23:474–486, 2020.

[5] V. Padányi and T. Herendi. Generalized Middle-Square Method. *Annales Mathematicae et Informaticae*, 56:95–108, 2022.

[6] V. Padányi and T. Herendi. A study on comparison of pseudorandom number generators. *International Journal of Mathematics and Computer in Engineering*, 1:1–20, 2023.

Publikációs lista / List of papers of the author

1. V. Padányi and T. Herendi (2020) *Metaanalysis of Pseudorandom Number Generators*, 23rd annual Spring Wind conference, vol. 23, pp. 474–486.

2. V. Padányi and T. Herendi (2022) *Generalized Middle-Square Method*, Annales Mathematicae et Informaticae, vol. 56, pp. 95-108.

3. V. Padányi and T. Herendi (2023) *A Study on Comparison of Pseudorandom Number Generators*, International Journal of Mathematics and Computer in Engineering, vol. 1, pp. 1-20.

4. T. Herendi and V. Padányi (2025) *Automata and Arithmetic in Canonical Number Systems*, Algorithms, vol. 18(3), pp. 1-22. EISSN: 1999-4893. https://doi.org/10.3390/a18030122

Konferencia előadások / List of talks of the author

1. The 11^{th} International Conference on Applied Informatics. Eger, Hungary, January 29–31, 2020.

2. The 1^{st} Conference on Information Technology and Data Science. Debrecen, Hungary, November 6–8, 2020.

3. The 23^{rd} Spring Wind Conference. Budapest, Hungary, October 16, 2020.

4. The 2^{nd} Conference on Information Technology and Data Science. Debrecen, Hungary, May 16–18, 2022.