Metody Monte Carlo – zápočtová práce

Šipky

aneb

Maximalizace očekávané hodnoty náhodné veličiny pomocí metod Monte Carlo

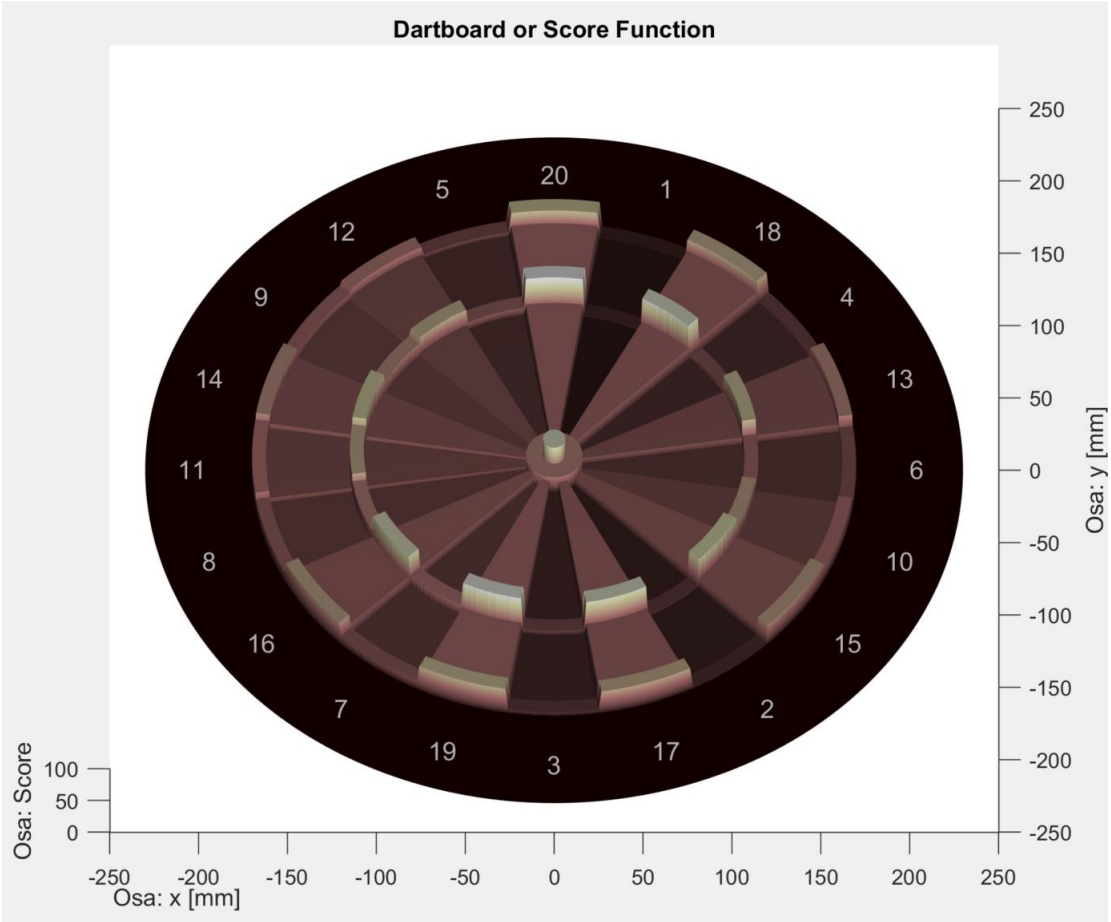
Cíl

Pomocí metod Monte Carlo určit optimální strategie hraní šipek hráčů různé úrovně v úvodní fázi hry.

Princip hry

Šipky proti sobě hrají zpravidla dva hráči, kteří střídavě hází trojici šipek na definovaný terč, přičemž za každé šipkou trefené políčko na terči, viz Obr 1, se hráči odečte určený počet bodů z jeho počátečního skóre, které je pro závodní hru zpravidla 501 bodů. Hru (leg) vyhrává hráč, který jako první vynuluje své počáteční skóre. V oficiálních turnajích platí navíc podmínka, že poslední hod musí být tzv. double (zavření doublem).

Z pravidel vyplívá, že v první fázi hry se hráči v každém kole snaží naházet co nejvíce bodů. Ke konci hry pak mění strategii, připravují si pozici pro možné zavření a následně se pokouší hru co nejrychleji ukončit hozeným doublem.



Obr 1: Obrázek standardního šipkařského terče, potažmo graf skórovací funkce použité při výpočtu.

Zavedení pojmů

- Každý hráčův hod šipkou na určité místo na terči představuje náhodný jev A.
- Hráč se při každém hodu dopouští určité nepřesnosti, proto výsledek jevu A, čili pozici (x, y) šipky v terči, modeluje dvojrozměrná náhodná veličina X.
- Střední hodnotu μ veličiny X považujeme za místo, kam hráč míří své hody. μ hráč dokáže během hry ovlivnit, tudíž μ představuje hráčovu strategii.
- Rozptyl σ^2 veličiny X pak uvádí hráčovu schopnost trefovat jím zvolený bod, čili popisuje kvalitu daného hráče.
- Terč modeluje výplatní funkce $s(x, y)$: $R^2 \rightarrow B \subset N_0$, přiřazující každé pozici v rovině terče získaný počet bodů, viz Obr 1.
- Počet bodů získaných jedním hodem popisuje náhodná veličina Y, která je transformací náhodné veličiny X skórovací funkcí s, čili $Y = s(X)$.

Předpoklady modelu

- Hráčovi hody jsou vzájemně nezávislé.
- Náhodná veličina $X \sim N^2(\mu, \sigma^2)$, kde $\mu = (\mu_x, \mu_y)$ je hráčova strategie a $\sigma^2 = (\sigma^2_x, \sigma_{xy}; \sigma_{yx}, \sigma^2_y)$ hráčova úroveň.
- Velikost chyby, které se hráč dopustí v horizontálním směru je nezávislá na odchylce od střední hodnoty ve směru vertikálním, $\sigma_{xy} = \sigma_{yx} = 0$.

Formulace problému

1) Určení očekávaného bodového zisku z hodu, při konkrétní strategii μ a hráčově úrovni σ^2
   Očekávaný bodový zisk odpovídá střední hodnotě veličiny Y, kterou spočítáme ze vztahu

$$EY_{\mu,\sigma^2} = \iint_{-\infty}^{+\infty} s(x,y)\phi_{\mu,\sigma^2}(x,y)dxdy \quad (1),$$

   kde $s(x,y)$ je výplatní funkce a $\phi_{\mu,\sigma^2}(x,y)$ součin dvou hustot normálního rozdělení o parametrech μ_x, σ^2_x resp. μ_y, σ^2_y. *(Pro jednoduchost.)*

2) Určení optimální strategie v první fázi hry pro hráče určité úrovně σ^2
   Nyní budeme chápat hodnoty EY jako funkci strategie μ,

$$EY_{\mu,\sigma^2} = EY_{\sigma^2}(\mu).$$

   Jelikož předpokládáme, že jsou jevy A nezávislé, platí

$$E(Y_1 + Y_2) = E(Y_1) + E(Y_2),$$

   tudíž optimální strategie μ* vyžaduje maximalizovat očekávaný bodový zisk každého hodu,

$$\mu_{\sigma^2}{}^* = argmax\, EY_{\sigma^2}(\mu).$$

3) Analýza výsledků. Průzkum závislost optimální strategie μ^* na schopnostech hráče σ^2
   Nyní budeme chápat optimální strategii μ* jako funkci úrovně hráče σ^2,

$$\mu_{\sigma^2}{}^* = \mu^*(\sigma^2),$$

   zjednodušíme na $\sigma^2_x = \sigma^2_y$ a zkoumáme, jak závisí optimální strategie a očekávané skóre na úrovni hráče.

Postup řešení

1) Generování náhodných čísel

Za tímto účelem byla vytvořena třída Generator, která poskytuje rozhraní pro použití v matlabu vestavěných RandomStreamů, včetně jejich nastavení a zároveň nabízí možnost dopsat do stejného rozhraní vlastní algoritmy.

V rámci práce byl implementován lineární kongruenční generátor rovnoměrně rozdělených náhodných veličin. Dále algoritmy pro transformaci uniformě rozdělených bodů na normálně rozdělené a sice sčítací, Box-Mullerův[3] a Marsagliův[4].

Následně byly tyto generátory porovnány s matlabovskými ve smyslu rychlosti, přesnosti požadovaných parametrů a průchodů vestavěných testů normality. Výsledky jsou v tabulce na Obr 10.

Z testu vychází ručně vyrobené generátory podstatně hůře než implicitní, což je velké štěstí pro tak nákladný software jako je matlab, viz Obr 10. Výsledky testů více méně odpovídají předpokladům o složitostech algoritmů, i když zde hraje výraznou roli i samotná implementace a práce s pamětí.

Nejlépe v testu dopadl uniformní generátor Shift Register[1] s transformačním algoritmem Ziggurat[5] bez plné přesnosti. Toto nastavení je dále používáno při dalších výpočtech. Za zmínku stojí, že zřejmě nejde o implicitní variantu.

Pozn. Zajímavé může být, že od zhruba 50 000 generovaných bodů, žádná z realizací neprošla žádným testem normality viz. Obr 11, chybu generování dokládají i histogramy na Obr 12.

2) Výpočet EY dle (1)

Funkce přes kterou integrujeme nemusí být spojitá, nicméně z podstaty víme, že je omezená, nezáporná a může být nenulová pouze v kruhu se středem v počátku a poloměrem stejným, jako má terč, viz Obr 3. Integrál tudíž konverguje a my můžeme použít metodu Monte Carlo Vzorkování funkce, která integrál odhaduje pro konkrétní parametry μ, σ^2 jako

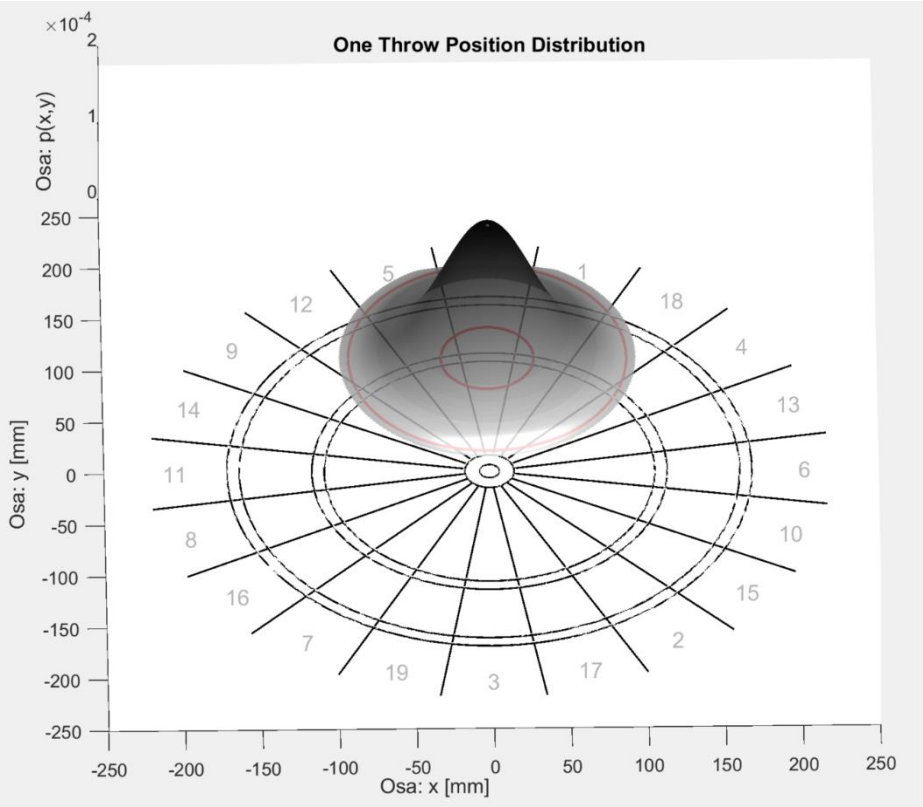
$$EY = \frac{1}{N}\sum_{i=1}^{N} s(x_i, y_i),\ x_i, y_i \sim N(\mu, \sigma^2)[1],$$

kde x_i, y_i byly generovány připraveným generátorem. Některé takto spočtené hodnoty jsou uvedeny v tabulce na Obr 4. Tento postup je v konečném důsledku nejintuitivnější možný. Simulujeme hody a počítáme průměr. Chyba takového odhadu lze poté odhadnout směrodatnou odchylkou průměru, tedy odmocninou z jeho rozptylu,

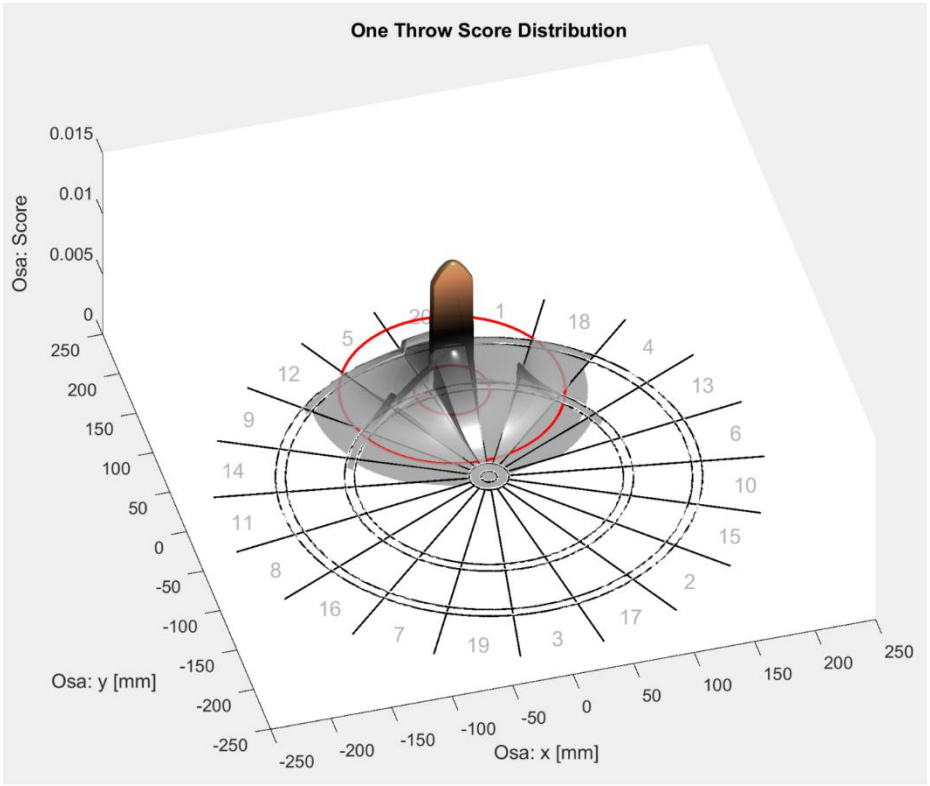
$$varEY = \frac{1}{N(N-1)}\sum_{i=1}^{N}(s(x_i, y_i) - EY)^2[1],$$

případně násobenou koeficientem mezi 2-3, záleží, jaký intervalový odhad chceme.

Funkce, viz Obr 3, není pro tuto metodu ideální, nabývá relativně úzkých maxim, proto je nutné použít pro přesný výpočet poměrně velké množství bodů. Z analýzy v tabulce na Obr 5 směrodatné odchylky EY v závislosti na parametrech a N vyplívá, že nepřesnost skutečně klesá s odmocninou z N a že pokud chceme mít jisté alespoň jedno desetinné místo pro všechny kombinace parametrů, musíme generovat zhruba N = 400 000 bodů pro jediný výpočet.



Obr 2: Graf normálního rozdělení použitého pro výpočet očekávané střední hodnoty skóre s parametry $\sigma_x = \sigma_y = 30$ a $\mu = (0, 110)$.



Obr 3: Graf integrované funkce pro parametry $\sigma_x = \sigma_y = 30$ a $\mu = (0, 110)$.

EX_is_0_and_EY_is_110	N_100	N_1000	N_10000	N_100000	N_1000000
'SigmaXY = 0'	60	60	60	60	60
'SigmaXY = 5'	47.2	42.36	42.475	42.812	42.736
'SigmaXY = 10'	29.24	30.012	30.044	30.173	30.104
'SigmaXY = 15'	19.6	22.38	22.411	22.398	22.415
'SigmaXY = 20'	20.76	17.172	17.899	17.88	17.87
'SigmaXY = 25'	14	15.06	15.372	15.37	15.312
'SigmaXY = 30'	13.08	13.816	14.103	13.827	13.806
'SigmaXY = 35'	11.92	12.959	12.735	12.84	12.827
'SigmaXY = 40'	12.64	11.961	12.057	12.031	12.048
'SigmaXY = 45'	11.08	10.898	11.483	11.402	11.421
'SigmaXY = 50'	8.72	11.32	10.959	10.865	10.884
'SigmaXY = 55'	10.08	10.317	10.296	10.392	10.389
'SigmaXY = 60'	9.61	10.149	9.8684	9.8767	9.9481
'SigmaXY = 65'	10.02	9.68	9.6054	9.5265	9.5405
'SigmaXY = 70'	11.27	8.775	9.1448	9.1632	9.1618

Obr 4: Tabulka spočtených odhadů střední hodnoty počtu bodů při strategii $\mu = (0, 110)$ a různých úrovních hráče, pro kterého ovšem $\sigma_x = \sigma_y$ po použití různých hodnot N.

Parameters_values	MEAN_is_mean_of_score	N_100	N_1000	N_10000	N_100000	N_1000000
'EX = 0 EY = 0'	'mean(s(MEAN))'	0.993	0.28816	0.092163	0.029163	0.0092076
'for SigmaXY = 0 : 5 : 70'	's(s(MEAN))'	0.34595	0.09706	0.030518	0.0097269	0.0030707
''	'max(s(MEAN))'	1.429	0.43438	0.13479	0.042883	0.013508
'EX = 0 EY = 110'	'mean(s(MEAN))'	1.4066	0.41406	0.13065	0.041286	0.013037
'for SigmaXY = 0 : 5 : 70'	's(s(MEAN))'	0.48395	0.14631	0.046913	0.014801	0.004694
''	'max(s(MEAN))'	1.9842	0.62522	0.19801	0.062688	0.019821
'EX = -40 EY = -100'	'mean(s(MEAN))'	1.2187	0.38037	0.12039	0.03804	0.012034
'for SigmaXY = 0 : 5 : 70'	's(s(MEAN))'	0.46648	0.13495	0.042692	0.01347	0.0042622
''	'max(s(MEAN))'	1.9014	0.60519	0.19143	0.060615	0.019168
'EX = -100 EY = 0'	'mean(s(MEAN))'	0.77664	0.24138	0.077646	0.024396	0.0077067
'for SigmaXY = 0 : 5 : 70'	's(s(MEAN))'	0.23965	0.071864	0.022775	0.007136	0.002254
''	'max(s(MEAN))'	1.0724	0.29017	0.093409	0.028967	0.0091596
'EX = 0 EY = -50'	'mean(s(MEAN))'	0.83395	0.25397	0.080256	0.025294	0.0080133
'for SigmaXY = 0 : 5 : 70'	's(s(MEAN))'	0.2997	0.082167	0.025806	0.0081263	0.0025812
''	'max(s(MEAN))'	1.2498	0.31455	0.097348	0.03083	0.0097809
'EX = 70 EY = 70'	'mean(s(MEAN))'	1.0815	0.33833	0.10539	0.033358	0.010551
'for SigmaXY = 0 : 5 : 70'	's(s(MEAN))'	0.36583	0.099315	0.031272	0.009856	0.0031188
''	'max(s(MEAN))'	1.5861	0.43962	0.13522	0.042661	0.013466
'EX = 120 EY = -100'	'mean(s(MEAN))'	0.87554	0.27421	0.085522	0.027168	0.0085874
'for SigmaXY = 0 : 5 : 70'	's(s(MEAN))'	0.27647	0.084935	0.026494	0.008437	0.002666
''	'max(s(MEAN))'	1.1628	0.31801	0.099415	0.031527	0.0099945

Obr 5: Tabulka analyzující odhady směrodatných odchylek střední hodnoty počtu bodů při různých strategiích a úrovních hráče. Uvedena je zde hodnota průměrné směrodatné odchylky pro konkrétní strategii počítanou ze směrodatných odchylek výpočtu pro různou úroveň hráče. Dále směrodatná odchylka těchto souborů směrodatných odchylek a odchylka maximální. Vše je uvedeno pro intuitivně dobré strategie a různá N. Tabulka naznačuje, že chyba klesá s odmocninou N, chyba odhadu chyby s N a že pro jisté jedno desetinné místo potřebujeme generovat zhruba 400 000 bodů.

3) Určení optimální strategie

Hledáme extrém funkce závisející na parametru σ. Pro jeho různé hodnoty může vypadat různě, viz Obr 6 a 7, nicméně pro všechny bude z kontextu omezená, spojitá a bude nabývat globální maximum na kruhu představující terč, nicméně zde může také mít desítky lokálních extrémů.

K nalezení optimální strategie byla použita optimalizace hejnem částic, konkrétně algoritmus SPSO_2006, implementovaný v rámci bakalářské práce spolužáka Ondry Pánka. Jde o heuristickou metodu hledání minima funkce na omezeném intervalu. Je příbuzná k Metodám Monte Carlo, jelikož je princip také v generování náhodných bodů, pouze chytřeji. S metodou se autor pouze chtěl seznámit, další informace o ní naleznete např. ve zmíněné bakalářské práci.[2]

Díky vlastnostem zkoumané funkce by jediným možným problémem pro částice mohl být případ, kdy nabývají dvě lokální maxima podobné hodnoty, která aspiruje na globální maximum, což pro určité σ jistě nastává. Nicméně zde největší problém znamená přesnost našeho výpočtu EY a i v případě chyby částic je nalezená strategie stále velmi dobrá.

Vzhledem k testům a výsledkům další části se zdá, že algoritmus svou roli zvládá. Nicméně určení jeho chyby je problematické, ale bude zřejmě představovat menší problém než chyba výpočtu funkční hodnoty.

4) Analýza optimálních strategií

V poslední části je, za zjednodušujícího předpokladu $\sigma_x = \sigma_y$, zkoumán, jak se mění optimální strategie a očekávané skóre v závislosti na úrovni hráče σ, a to od dokonalého stroje až po člověka, který je rád, když trefí terč.

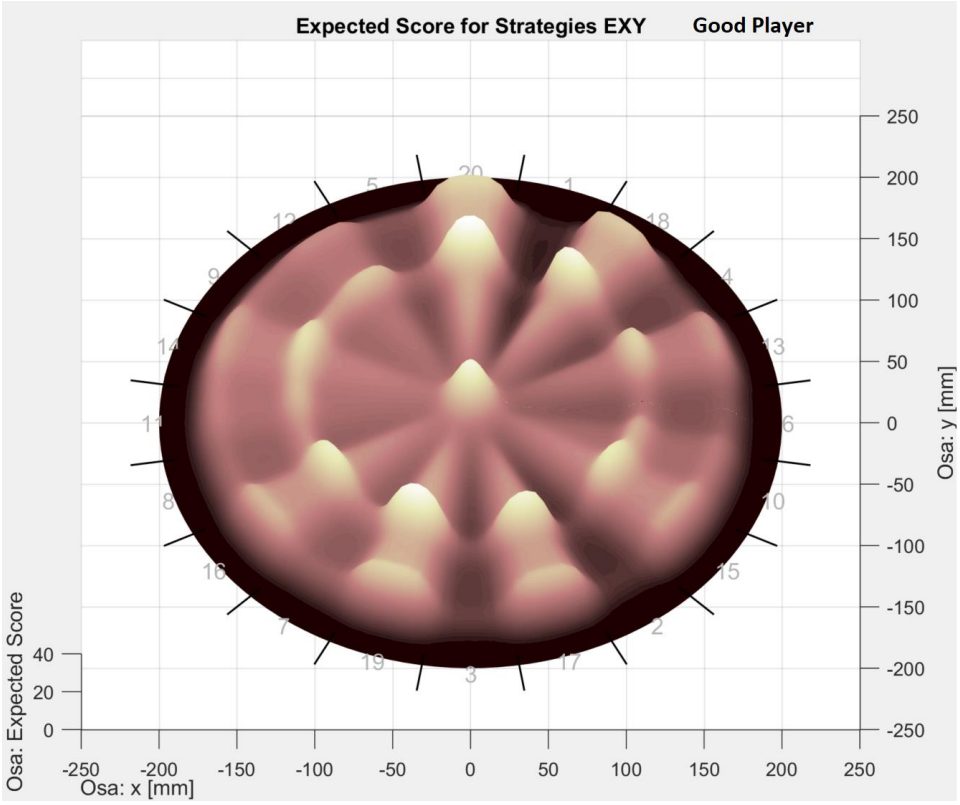
To, že profesionál hází na nejvíce bodované pole triple 20 a naprostý začátečník by měl mířit na střed, je intuitivní. Nicméně vývoj optimální strategie mezi těmito extrémy je relativně zajímavý, viz Obr 9. K získání takto plynulého grafu byla použitá určitá korekce. Strategie se se zvýšením σ změní pouze tehdy, pokud je nová hodnota prokazatelně vyšší než hodnota v bodu původní strategie, čili alespoň o chyby obou výpočtů.

Nakonec není bez zajímavosti také porovnání různých strategií z Obr 10, které ukazuje, že volbou chybné, a přece intuitivní strategie, se hráč může připravit v průměru i o několik bodů z každé odhozené šipky.

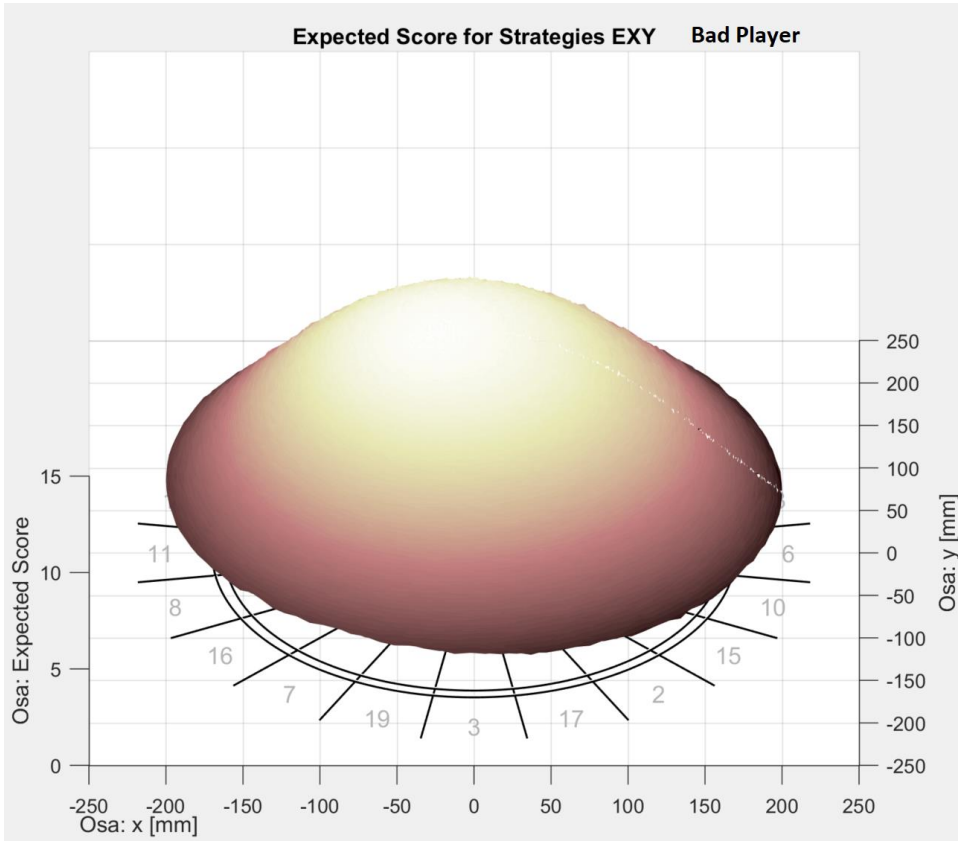
Závěr

V rámci této práce bylo implementováno několik algoritmů pro transformaci bodů z uniformního rozdělení na normální s danými parametry a jeden lineárně kongruenční generátor uniformě rozdělených bodů. Dále byly porovnány jejich vlastnosti s vlastnostmi generátorů vestavěných v matlabu a vybrán ten nejvhodnější.

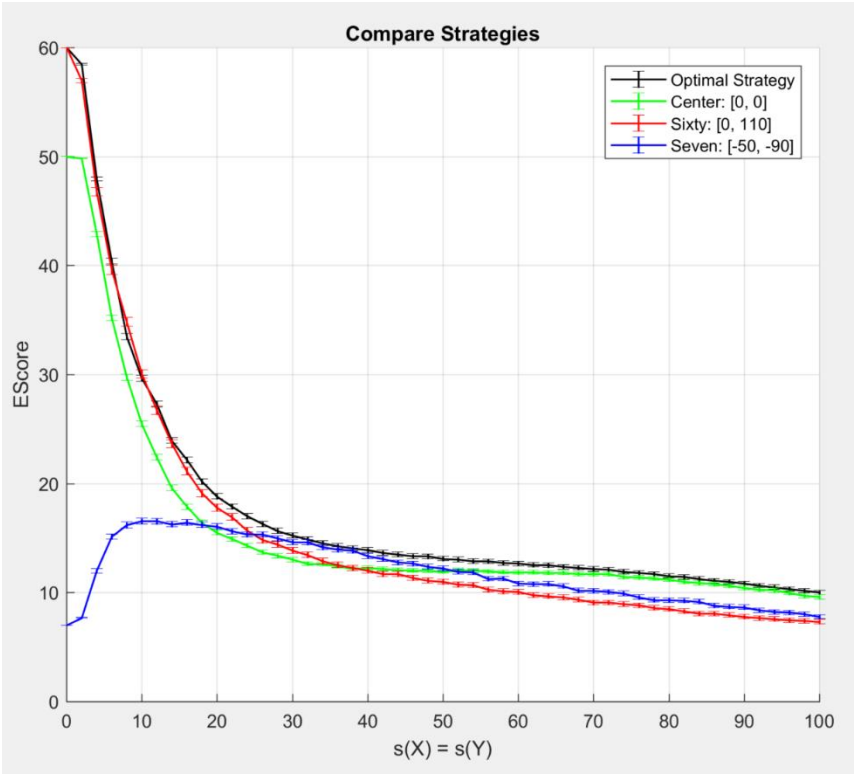
Pomocí metody Vzorkování funkce byla dále vytvořena funkce pro určení střední hodnoty skóre při konkrétní strategii a úrovni hráče. Tato funkce byla následně použita pro určení optimální strategie daného hráče pomocí hledání extrému hejnem částic. Poslední část práce vedla k popisu vývoje optimální strategie v závislosti na schopnostech hráče a ukázání významu zohlednění svých schopností při volbě strategie porovnáním výsledků optimální strategie s několika dalšími intuitivně dobrými strategiemi.



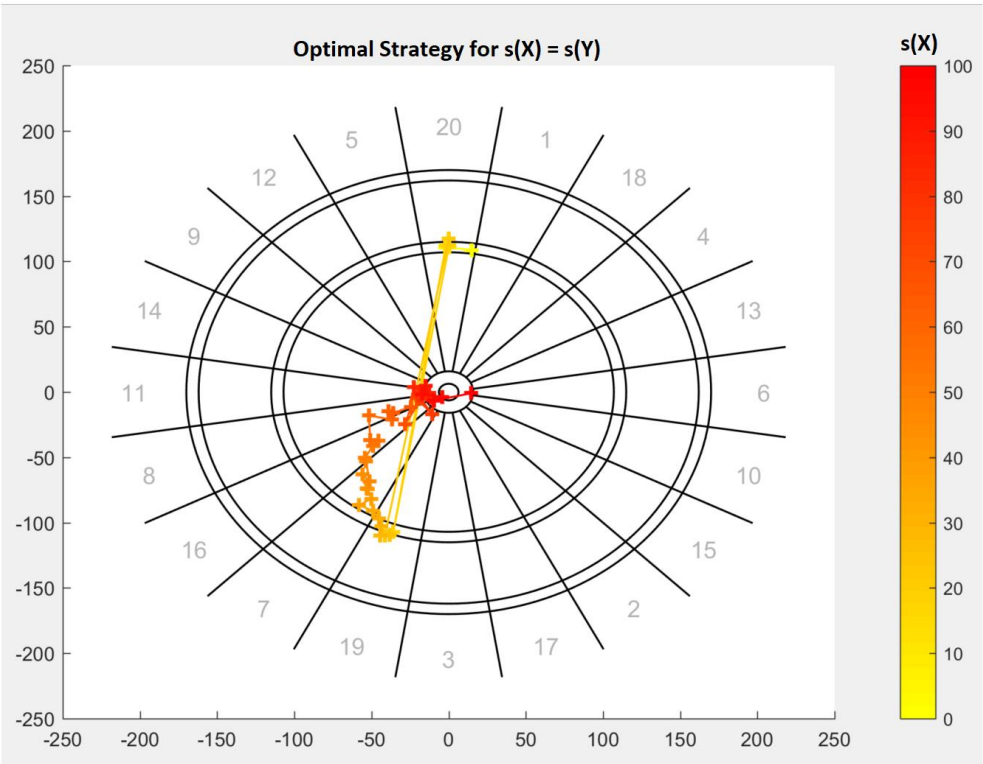
Obr 6: Graf středních hodnot očekávaného skóre v závislosti na volbě strategie (x, y) pro velmi dobrého hráče $\sigma_x = \sigma_y = 7$.



Obr 7: Graf středních hodnot očekávaného skóre v závislosti na volbě strategie (x, y) pro začínajícího hráče $\sigma_x = \sigma_y = 70$.



Obr 8: Graf porovnání hodnot výsledných očekávaných skóre pro 3 různé konstantní strategie s výsledkem strategie optimální pro danou úroveň hráče.



Obr 9: Graf vývoje pozice nalezené optimální strategie (argmax) v závislosti na úrovni hráče.

Zdroje

[1] *M. Virius*, METODA MONTE CARLO, skriptum, Česká technika - nakladatelství ČVUT, Praha 2010.

[2] *O. Pánek*, Algoritmus optimalizace hejnem částic: vývoj a jeho aplikace, bakalářská práce, Praha 2018.

[3] *Wikipedie*, Box-Muller transform, ONLINE, 13. 12. 2018. Dostupné z: https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform

[4] *Wikipedie*, Marsaglia polar method, ONLINE, 13. 12. 2018. Dostupné z: https://en.wikipedia.org/wiki/Marsaglia_polar_method

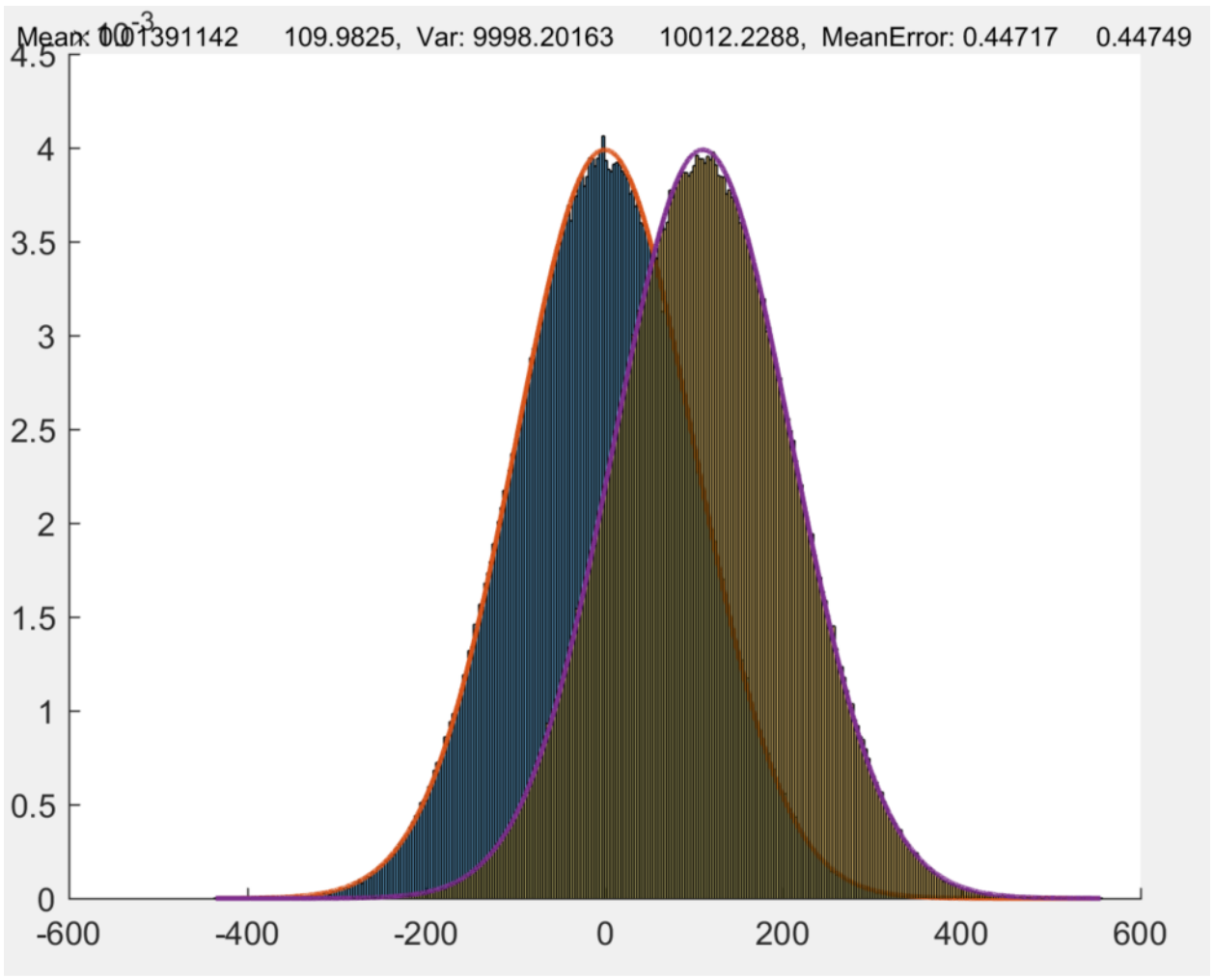
[5] *Wikipedie*, Ziggurat algorithm, ONLINE, 13. 12. 2018. Dostupné z: https://en.wikipedia.org/wiki/Ziggurat_algorithm

Distribution	UniformGenerator	NormalTransformator	FullPrecision	TimeNeeded	FailedTests	RealMean	RealsqrVar	RealSize
'Uniform'	'MultiFibonacci'	'---'	true	0.20333	NaN	'50.23: :49.77'	'29.15: :28.74'	'10000.00: :2.00'
'Uniform'	'ShiftRegister'	'---'	false	0.20417	NaN	'49.91: :49.53'	'28.73: :28.93'	'10000.00: :2.00'
'Uniform'	'ShiftRegister'	'---'	true	0.2049	NaN	'49.91: :49.53'	'28.73: :28.93'	'10000.00: :2.00'
'Uniform'	'MultiFibonacci'	'---'	false	0.21061	NaN	'50.23: :49.77'	'29.15: :28.74'	'10000.00: :2.00'
'Uniform'	'FastMersenneTwister'	'---'	false	0.23064	NaN	'49.88: :49.75'	'28.95: :29.01'	'10000.00: :2.00'
'Uniform'	'MultiCongruential'	'---'	true	0.24292	NaN	'50.27: :49.93'	'28.93: :29.09'	'10000.00: :2.00'
'Uniform'	'FastMersenneTwister'	'---'	true	0.25054	NaN	'49.88: :49.75'	'28.95: :29.01'	'10000.00: :2.00'
'Uniform'	'MultiCongruential'	'---'	false	0.26083	NaN	'50.27: :49.93'	'28.93: :29.09'	'10000.00: :2.00'
'Uniform'	'CombiMultiRecursive'	'---'	false	0.29687	NaN	'50.00: :50.06'	'28.96: :28.91'	'10000.00: :2.00'
'Uniform'	'SubtractBorrow'	'---'	false	0.36844	NaN	'49.82: :50.00'	'28.77: :29.12'	'10000.00: :2.00'
'Uniform'	'SubtractBorrow'	'---'	true	0.37249	NaN	'49.82: :50.00'	'28.77: :29.12'	'10000.00: :2.00'
'Uniform'	'MersenneTwister'	'---'	false	0.38506	NaN	'49.60: :50.03'	'28.78: :28.81'	'10000.00: :2.00'
'Uniform'	'CombiMultiRecursive'	'---'	true	0.42577	NaN	'49.97: :50.49'	'28.90: :28.66'	'10000.00: :2.00'
'Uniform'	'MersenneTwister'	'---'	true	0.46247	NaN	'49.43: :50.14'	'29.05: :28.68'	'10000.00: :2.00'
'Normal'	'ShiftRegister'	'Ziggurat'	false	0.55716	1957	'0.27: :110.26'	'101.13: :100.05'	'10000.00: :2.00'
'Normal'	'ShiftRegister'	'Ziggurat'	true	0.59892	1957	'0.27: :110.26'	'101.13: :100.05'	'10000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Ziggurat'	false	0.63313	1921	'0.01: :109.25'	'100.54: :100.93'	'10000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Ziggurat'	true	0.63806	1921	'0.01: :109.25'	'100.54: :100.93'	'10000.00: :2.00'
'Normal'	'MultiFibonacci'	'Ziggurat'	false	0.67317	1932	'0.47: :109.41'	'99.28: :99.33'	'10000.00: :2.00'
'Normal'	'MultiFibonacci'	'Ziggurat'	true	0.68173	1932	'0.47: :109.41'	'99.28: :99.33'	'10000.00: :2.00'
'Normal'	'MersenneTwister'	'Ziggurat'	false	0.71394	1946	'-1.45: :110.93'	'100.15: :99.75'	'10000.00: :2.00'
'Normal'	'MersenneTwister'	'Ziggurat'	true	0.76068	1956	'-1.67: :110.47'	'100.33: :99.41'	'10000.00: :2.00'
'Normal'	'MultiCongruential'	'Ziggurat'	false	0.77886	1960	'0.28: :109.60'	'100.80: :99.06'	'10000.00: :2.00'
'Normal'	'MultiCongruential'	'Ziggurat'	true	0.79147	1960	'0.28: :109.60'	'100.80: :99.06'	'10000.00: :2.00'
'Normal'	'MultiFibonacci'	'Polar'	false	0.83681	1863	'0.94: :110.02'	'99.00: :100.38'	'10000.00: :2.00'
'Normal'	'ShiftRegister'	'Polar'	true	0.83911	1883	'0.93: :111.40'	'101.39: :99.91'	'10000.00: :2.00'
'Normal'	'MultiFibonacci'	'Polar'	true	0.85566	1863	'0.94: :110.02'	'99.00: :100.38'	'10000.00: :2.00'
'Normal'	'ShiftRegister'	'Inversion'	true	0.86287	1909	'-0.40: :108.48'	'99.54: :100.48'	'10000.00: :2.00'
'Normal'	'MultiFibonacci'	'Inversion'	false	0.86414	1946	'1.19: :109.39'	'101.81: :99.20'	'10000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Polar'	false	0.86639	1953	'-1.51: :109.68'	'99.67: :99.92'	'10000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Polar'	true	0.86983	1953	'-1.51: :109.68'	'99.67: :99.92'	'10000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Ziggurat'	false	0.87664	1922	'-0.83: :110.46'	'100.21: :100.60'	'10000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Inversion'	true	0.8877	1919	'0.19: :109.00'	'100.14: :100.65'	'10000.00: :2.00'
'Normal'	'ShiftRegister'	'Inversion'	false	0.89103	1909	'-0.40: :108.48'	'99.54: :100.48'	'10000.00: :2.00'
'Normal'	'ShiftRegister'	'Polar'	false	0.89491	1883	'0.93: :111.40'	'101.39: :99.91'	'10000.00: :2.00'
'Normal'	'MersenneTwister'	'Polar'	false	0.89586	1924	'-0.29: :109.83'	'100.42: :99.58'	'10000.00: :2.00'
'Normal'	'MultiCongruential'	'Inversion'	false	0.91123	1957	'1.45: :109.51'	'100.28: :101.65'	'10000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Inversion'	false	0.91652	1919	'0.19: :109.00'	'100.14: :100.65'	'10000.00: :2.00'
'Normal'	'MultiFibonacci'	'Inversion'	true	0.91883	1946	'1.19: :109.39'	'101.81: :99.20'	'10000.00: :2.00'
'Normal'	'MultiCongruential'	'Inversion'	true	0.92282	1957	'1.45: :109.51'	'100.28: :101.65'	'10000.00: :2.00'
'Normal'	'MersenneTwister'	'Inversion'	false	0.94092	1874	'-1.43: :110.46'	'99.64: :99.41'	'10000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Polar'	false	0.94144	1973	'1.58: :109.59'	'99.45: :99.41'	'10000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Inversion'	false	0.98496	1884	'-0.23: :110.14'	'100.94: :100.30'	'10000.00: :2.00'
'Normal'	'MersenneTwister'	'Polar'	true	0.98703	1879	'-3.19: :110.38'	'99.73: :99.28'	'10000.00: :2.00'
'Normal'	'MersenneTwister'	'Inversion'	true	1.0126	1934	'-1.62: :110.78'	'100.66: :98.97'	'10000.00: :2.00'
'Normal'	'SubtractBorrow'	'Ziggurat'	true	1.0179	1901	'-1.70: :110.68'	'100.81: :99.41'	'10000.00: :2.00'
'Normal'	'SubtractBorrow'	'Ziggurat'	false	1.0436	1901	'-1.70: :110.68'	'100.81: :99.41'	'10000.00: :2.00'
'Normal'	'SubtractBorrow'	'Inversion'	true	1.0614	1950	'-0.64: :109.99'	'99.73: :101.65'	'10000.00: :2.00'
'Normal'	'ShiftRegister'	'my BoxMuller'	false	1.0643	1911	'-0.26: :110.39'	'100.39: :99.69'	'10000.00: :2.00'
'Normal'	'MultiCongruential'	'my_BoxMuller'	true	1.0887	1838	'1.11: :109.79'	'99.79: :98.87'	'10000.00: :2.00'
'Normal'	'SubtractBorrow'	'Polar'	true	1.0892	1942	'0.09: :108.82'	'100.03: :99.29'	'10000.00: :2.00'
'Normal'	'MultiFibonacci'	'my_BoxMuller'	false	1.0925	1848	'-1.37: :111.51'	'99.80: :100.03'	'10000.00: :2.00'
'Normal'	'FastMersenneTwister'	'my_BoxMuller'	true	1.1049	1909	'1.02: :110.62'	'100.17: :99.54'	'10000.00: :2.00'
'Normal'	'MultiCongruential'	'my_BoxMuller'	false	1.105	1838	'1.11: :109.79'	'99.79: :98.87'	'10000.00: :2.00'
'Normal'	'ShiftRegister'	'my_BoxMuller'	true	1.1094	1911	'-0.26: :110.39'	'100.39: :99.69'	'10000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'my_BoxMuller'	false	1.1388	1927	'1.11: :109.06'	'100.29: :100.51'	'10000.00: :2.00'
'Normal'	'FastMersenneTwister'	'my_BoxMuller'	false	1.1563	1909	'1.02: :110.62'	'100.17: :99.54'	'10000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Inversion'	true	1.1573	1909	'-0.25: :111.67'	'100.28: :98.52'	'10000.00: :2.00'
'Normal'	'MersenneTwister'	'my_BoxMuller'	false	1.206	1922	'-0.10: :110.43'	'101.02: :100.05'	'10000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Ziggurat'	true	1.2242	1962	'1.79: :109.84'	'100.12: :100.92'	'10000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Polar'	true	1.2314	1907	'1.19: :111.62'	'100.66: :100.01'	'10000.00: :2.00'
'Normal'	'MultiCongruential'	'Polar'	true	1.2361	1905	'0.07: :111.10'	'100.30: :99.58'	'10000.00: :2.00'
'Normal'	'MultiCongruential'	'Polar'	false	1.2688	1905	'0.07: :111.10'	'100.30: :99.58'	'10000.00: :2.00'
'Normal'	'SubtractBorrow'	'my_BoxMuller'	false	1.2781	1899	'2.05: :110.21'	'100.51: :99.88'	'10000.00: :2.00'
'Normal'	'SubtractBorrow'	'my_BoxMuller'	true	1.3211	1899	'2.05: :110.21'	'100.51: :99.88'	'10000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'my_BoxMuller'	true	1.3338	1890	'-0.75: :109.02'	'100.28: :100.10'	'10000.00: :2.00'
'Normal'	'MersenneTwister'	'my_BoxMuller'	true	1.3438	1847	'-1.09: :110.96'	'100.37: :101.38'	'10000.00: :2.00'
'Normal'	'MultiCongruential'	'my_Marsaglia'	true	1.369	1853	'-0.94: :111.34'	'100.07: :99.94'	'9923.00: :2.00'
'Normal'	'MersenneTwister'	'my_Marsaglia'	false	1.3805	1920	'0.81: :109.21'	'100.42: :99.66'	'10079.00: :2.00'
'Normal'	'ShiftRegister'	'my_Marsaglia'	true	1.3867	1902	'-0.47: :109.24'	'99.74: :100.73'	'9959.00: :2.00'
'Normal'	'MultiFibonacci'	'my_Marsaglia'	false	1.4092	1890	'0.01: :110.04'	'100.36: :99.03'	'9958.00: :2.00'
'Normal'	'MultiCongruential'	'my_Marsaglia'	false	1.4129	1853	'-0.94: :111.34'	'100.07: :99.94'	'9923.00: :2.00'
'Normal'	'ShiftRegister'	'my_Marsaglia'	false	1.4339	1902	'-0.47: :109.24'	'99.74: :100.73'	'9959.00: :2.00'
'Normal'	'MultiFibonacci'	'my_Marsaglia'	true	1.4517	1890	'0.01: :110.04'	'100.36: :99.03'	'9958.00: :2.00'
'Normal'	'FastMersenneTwister'	'my_Marsaglia'	true	1.5243	1899	'-0.44: :110.46'	'99.60: :100.41'	'9934.00: :2.00'
'Normal'	'CombiMultiRecursive'	'my_Marsaglia'	false	1.5358	1915	'0.93: :109.94'	'100.14: :100.86'	'9914.00: :2.00'
'Normal'	'FastMersenneTwister'	'my_Marsaglia'	false	1.5475	1899	'-0.44: :110.46'	'99.60: :100.41'	'9934.00: :2.00'
'Normal'	'MersenneTwister'	'my_Marsaglia'	true	1.5522	1924	'-2.40: :110.52'	'99.47: :100.43'	'9951.00: :2.00'
'Normal'	'SubtractBorrow'	'my_Marsaglia'	false	1.5794	1904	'1.00: :110.61'	'98.34: :100.21'	'9935.00: :2.00'
'Normal'	'SubtractBorrow'	'my_Marsaglia'	true	1.6189	1904	'1.00: :110.61'	'98.34: :100.21'	'9935.00: :2.00'
'Normal'	'CombiMultiRecursive'	'my_Marsaglia'	true	1.6697	1963	'2.34: :110.17'	'100.07: :99.14'	'10031.00: :2.00'
'Normal'	'ShiftRegister'	'my_Sum'	true	3.38	2643	'-0.03: :109.88'	'99.11: :100.39'	'10000.00: :2.00'
'Normal'	'MultiFibonacci'	'my_Sum'	false	3.5842	2655	'-1.09: :108.71'	'99.72: :98.79'	'10000.00: :2.00'
'Normal'	'ShiftRegister'	'my_Sum'	false	3.6235	2643	'-0.03: :109.88'	'99.11: :100.39'	'10000.00: :2.00'
'Normal'	'MultiFibonacci'	'my_Sum'	true	3.6329	2655	'-1.09: :108.71'	'99.72: :98.79'	'10000.00: :2.00'
'Normal'	'FastMersenneTwister'	'my_Sum'	true	3.6778	2664	'1.84: :110.93'	'101.68: :100.58'	'10000.00: :2.00'
'Normal'	'FastMersenneTwister'	'my_Sum'	false	3.8445	2664	'1.84: :110.93'	'101.68: :100.58'	'10000.00: :2.00'
'Normal'	'MersenneTwister'	'my_Sum'	false	4.185	2694	'1.08: :109.28'	'100.37: :98.24'	'10000.00: :2.00'
'Normal'	'MultiCongruential'	'my_Sum'	false	4.2842	2649	'1.79: :107.57'	'100.50: :100.61'	'10000.00: :2.00'
'Uniform'	'my_Linear'	'---'	true	4.3028	NaN	'49.86: :50.18'	'28.77: :28.92'	'10000.00: :2.00'
'Uniform'	'my_Linear'	'---'	false	4.3435	NaN	'49.86: :50.18'	'28.77: :28.92'	'10000.00: :2.00'
'Normal'	'MultiCongruential'	'my_Sum'	true	4.3852	2649	'1.79: :107.57'	'100.50: :100.61'	'10000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'my_Sum'	false	4.879	2655	'-2.60: :111.28'	'100.01: :99.87'	'10000.00: :2.00'
'Normal'	'my_Linear'	'my_BoxMuller'	true	4.9826	1895	'1.12: :110.01'	'99.66: :100.24'	'10000.00: :2.00'
'Normal'	'my_Linear'	'my_BoxMuller'	false	5.0922	1895	'1.12: :110.01'	'99.66: :100.24'	'10000.00: :2.00'
'Normal'	'MersenneTwister'	'my_Sum'	true	5.4336	2656	'-1.07: :109.42'	'98.93: :100.33'	'10000.00: :2.00'
'Normal'	'SubtractBorrow'	'my_Sum'	true	5.7025	2656	'0.67: :108.21'	'99.74: :99.90'	'10000.00: :2.00'
'Normal'	'SubtractBorrow'	'my_Sum'	false	5.7108	2656	'0.67: :108.21'	'99.74: :99.90'	'10000.00: :2.00'
'Normal'	'my_Linear'	'my_Marsaglia'	false	6.7224	1937	'-0.53: :108.82'	'99.70: :100.09'	'10032.00: :2.00'
'Normal'	'my_Linear'	'my_Marsaglia'	true	6.778	1937	'-0.53: :108.82'	'99.70: :100.09'	'10032.00: :2.00'
'Normal'	'CombiMultiRecursive'	'my_Sum'	true	6.889	2636	'-0.07: :109.78'	'99.41: :99.43'	'10000.00: :2.00'
'Normal'	'my_Linear'	'my_Sum'	false	33.277	2693	'0.21: :110.78'	'100.34: :100.41'	'10000.00: :2.00'
'Normal'	'my_Linear'	'my_Sum'	true	33.554	2693	'0.21: :110.78'	'100.34: :100.41'	'10000.00: :2.00'

Obr 10: Tabulka s výsledky testu porovnávajícího vlastnosti různých generátorů pseudonáhodných čísel. Každý generátor tisíckrát generoval 20 000 náhodných čísel, přičemž v každé iteraci prošla jeho realizace náhodné veličiny 3 testy normality (celkem 3000 testů).

Distribution	UniformGenerator	NormalTransformator	FullPrecision	TimeNeeded	FailedTests	RealMean	RealsqrVar	RealSize
'Normal'	'ShiftRegister'	'Ziggurat'	false	0.40592	30	'0.16: :109.87'	'100.09: :100.03'	'1000000.00: :2.00'
'Normal'	'ShiftRegister'	'Ziggurat'	true	0.41664	30	'0.16: :109.87'	'100.09: :100.03'	'1000000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Ziggurat'	true	0.46331	30	'-0.13: :109.91'	'99.97: :100.04'	'1000000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Ziggurat'	false	0.47334	30	'-0.13: :109.91'	'99.97: :100.04'	'1000000.00: :2.00'
'Normal'	'MultiFibonacci'	'Ziggurat'	true	0.48247	30	'0.10: :109.88'	'99.85: :100.07'	'1000000.00: :2.00'
'Normal'	'MersenneTwister'	'Ziggurat'	false	0.50358	30	'-0.01: :110.14'	'99.95: :99.99'	'1000000.00: :2.00'
'Normal'	'MultiFibonacci'	'Ziggurat'	false	0.50416	30	'0.10: :109.88'	'99.85: :100.07'	'1000000.00: :2.00'
'Normal'	'MultiCongruential'	'Ziggurat'	false	0.54259	30	'-0.02: :109.93'	'99.92: :100.00'	'1000000.00: :2.00'
'Normal'	'MultiCongruential'	'Ziggurat'	true	0.54537	30	'-0.02: :109.93'	'99.92: :100.00'	'1000000.00: :2.00'
'Normal'	'MersenneTwister'	'Ziggurat'	true	0.5503	30	'0.08: :110.11'	'99.96: :100.02'	'1000000.00: :2.00'
'Normal'	'MultiFibonacci'	'Polar'	false	0.6014	30	'-0.10: :110.02'	'100.02: :100.09'	'1000000.00: :2.00'
'Normal'	'ShiftRegister'	'Polar'	true	0.60888	30	'-0.01: :110.13'	'100.16: :100.04'	'1000000.00: :2.00'
'Normal'	'ShiftRegister'	'Polar'	false	0.62098	30	'-0.01: :110.13'	'100.16: :100.04'	'1000000.00: :2.00'
'Normal'	'MultiFibonacci'	'Polar'	true	0.62442	30	'-0.10: :110.02'	'100.02: :100.09'	'1000000.00: :2.00'
'Normal'	'ShiftRegister'	'Inversion'	false	0.6294	30	'0.04: :109.95'	'99.91: :100.03'	'1000000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Inversion'	false	0.64341	30	'-0.06: :109.80'	'99.97: :99.85'	'1000000.00: :2.00'
'Normal'	'MersenneTwister'	'Polar'	false	0.64989	30	'-0.07: :110.07'	'100.09: :99.95'	'1000000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Polar'	false	0.65201	30	'-0.09: :109.74'	'99.97: :100.01'	'1000000.00: :2.00'
'Normal'	'MultiFibonacci'	'Inversion'	false	0.65239	30	'-0.02: :109.92'	'100.10: :99.86'	'1000000.00: :2.00'
'Normal'	'MultiFibonacci'	'Inversion'	true	0.65374	30	'-0.02: :109.92'	'100.10: :99.86'	'1000000.00: :2.00'
'Normal'	'MultiCongruential'	'Inversion'	false	0.66981	30	'-0.22: :109.95'	'99.88: :100.06'	'1000000.00: :2.00'
'Normal'	'MultiCongruential'	'Inversion'	true	0.67318	30	'-0.22: :109.95'	'99.88: :100.06'	'1000000.00: :2.00'
'Normal'	'ShiftRegister'	'Inversion'	true	0.67336	30	'0.04: :109.95'	'99.91: :100.03'	'1000000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Polar'	true	0.67431	30	'-0.09: :109.74'	'99.97: :100.01'	'1000000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Ziggurat'	false	0.67742	30	'-0.15: :109.99'	'100.01: :100.08'	'1000000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Polar'	false	0.71476	30	'-0.06: :110.08'	'100.07: :100.02'	'1000000.00: :2.00'
'Normal'	'MersenneTwister'	'Inversion'	false	0.71684	30	'-0.14: :110.01'	'100.00: :100.00'	'1000000.00: :2.00'
'Normal'	'SubtractBorrow'	'Ziggurat'	false	0.75879	30	'-0.00: :110.03'	'100.00: :99.98'	'1000000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Inversion'	false	0.76343	30	'0.08: :109.96'	'99.82: :100.11'	'1000000.00: :2.00'
'Normal'	'SubtractBorrow'	'Ziggurat'	true	0.78717	30	'-0.00: :110.03'	'100.00: :99.98'	'1000000.00: :2.00'
'Normal'	'MersenneTwister'	'Polar'	true	0.78977	30	'-0.00: :109.97'	'100.10: :100.08'	'1000000.00: :2.00'
'Normal'	'SubtractBorrow'	'Polar'	false	0.80204	30	'0.05: :110.00'	'100.01: :99.95'	'1000000.00: :2.00'
'Normal'	'SubtractBorrow'	'Polar'	true	0.80645	30	'0.05: :110.00'	'100.01: :99.95'	'1000000.00: :2.00'
'Normal'	'SubtractBorrow'	'Inversion'	false	0.80937	30	'0.02: :109.92'	'99.97: :100.16'	'1000000.00: :2.00'
'Normal'	'FastMersenneTwister'	'Inversion'	true	0.81108	30	'-0.06: :109.80'	'99.97: :99.85'	'1000000.00: :2.00'
'Normal'	'MersenneTwister'	'Inversion'	true	0.81593	30	'0.03: :110.05'	'100.01: :99.98'	'1000000.00: :2.00'
'Normal'	'SubtractBorrow'	'Inversion'	true	0.83238	30	'0.02: :109.92'	'99.97: :100.16'	'1000000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Polar'	true	0.91296	30	'-0.20: :110.13'	'99.94: :99.94'	'1000000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Inversion'	true	0.91876	30	'-0.10: :109.97'	'99.94: :100.04'	'1000000.00: :2.00'
'Normal'	'MultiCongruential'	'Polar'	false	0.97076	30	'-0.22: :110.10'	'100.02: :99.93'	'1000000.00: :2.00'
'Normal'	'MultiCongruential'	'Polar'	true	0.97538	30	'-0.22: :110.10'	'100.02: :99.93'	'1000000.00: :2.00'
'Normal'	'ShiftRegister'	'my_BoxMuller'	false	0.99327	30	'-0.08: :109.96'	'99.96: :99.93'	'1000000.00: :2.00'
'Normal'	'FastMersenneTwister'	'my_BoxMuller'	false	1.0053	30	'-0.18: :110.13'	'100.02: :100.00'	'1000000.00: :2.00'
'Normal'	'CombiMultiRecursive'	'Ziggurat'	true	1.0059	30	'0.00: :109.99'	'100.00: :100.11'	'1000000.00: :2.00'
'Normal'	'MultiFibonacci'	'my_BoxMuller'	true	1.0269	30	'-0.26: :110.08'	'99.99: :100.10'	'1000000.00: :2.00'
'Normal'	'MultiCongruential'	'my_BoxMuller'	false	1.0541	30	'0.04: :110.05'	'100.05: :100.07'	'1000000.00: :2.00'

Obr 11: První polovina tabulky s výsledky testu porovnávajícího vlastnosti různých generátorů pseudonáhodných čísel. Každý generátor desetkrát generoval 20 000 000 náhodných čísel, přičemž v každé iteraci prošla jeho realizace náhodné veličiny 3 testy normality (celkem 30 testů). Žádný test nikdy neprošel.



Obr 12: Histogram jedné z realizací vygenerované normální veličiny jedním z matlabovských generátorů o rozsahu 20 000 000 bodů, který ukazuje na nedokonalost tohoto generátoru.