VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MATEMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF MATHEMATICS

OPTIMÁLNÍ KOREKCE NEPŘESNÉ STŘELBY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN HORNÍČEK

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MATEMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF MATHEMATICS

OPTIMÁLNÍ KOREKCE NEPŘESNÉ STŘELBY

OPTIMALIZATION METHODS FOR COMPESATION OF SHOOTING ACCURACY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN HORNÍČEK
AUTHOR

VEDOUCÍ PRÁCE — RNDr. PAVEL POPELA, Ph.D.
SUPERVISOR

BRNO 2012

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav matematiky
Akademický rok: 2011/2012

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Jan Horníček

který/která studuje v **bakalářském studijním programu**

obor: **Matematické inženýrství (3901R021)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Optimální korekce nepřesné střelby

v anglickém jazyce:

Optimization Methods for Compensation of Shooting Inaccuracy

Stručná charakteristika problematiky úkolu:

Sestavení modelu a výpočetního algoritmu pro hledání optimálního cíle nepřesné střelby. Využití poznatků teorie pravděpodobnosti a matematického programování. Implementace, testování a vizualizace vypočtených dat.

Cíle bakalářské práce:

Rozvoj teoretických znalostí v následujících oblastech: základy balistiky, rozdělení pravděpodobnosti, teorie vícerozměrného Lebesgueova-Stieltjesova integrálu, simulace a optimalizace nepřesné střelby o daných parametrech s využitím metody Monte Carlo. Předpokládá se budoucí návaznost na obecné úlohy zásahu těles řešené M. Druckmullerem.

Seznam odborné literatury:

Kolmogorov, A. N. a Fomin, S. V. : Základy teorie funkcí a funkcionální analýzy, SNTL, Praha, 1975.
Zvára, K. a Štěpán. J.: Pravděpodobnost a matematická statistika. Matfyzpress, Praha 2006.
Rényi, A.: Teorie pravděpodobnosti. Academia, Praha 1972.

Vedoucí bakalářské práce: RNDr. Pavel Popela, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2011/2012.

V Brně, dne 29.10.2011

L.S.

prof. RNDr. Josef Šlapal, CSc.	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c.
Ředitel ústavu	Děkan fakulty

Abstrakt

V této práci je proveden rozbor nepřesné střelby a jejích optimálních korekcí na konkrétním příkladě házení šipek na terč. Nejprve je sestaven model popisující nepřesnou střelbu a na jeho základě jsou odvozeny předpoklady pro řešení úlohy hledání optimálních korekcí této střelby. Dále je sestrojen algoritmus numerického výpočtu, o kterém je dokázano, že jím lze úlohu řešit s libovolnou přesností. Tento algoritmus je implementován v prostředí MATLAB a následně modifikován poznatky z funkcionální analýzy tak, aby došlo k výrazné úspoře výpočetního času. Úloha je následně algoritmem vyřešena a výsledky jsou zpracovány formou jednoduché uživatelské aplikace.

Abstract

In this work there is performed analysis of inaccuracy shooting and its optimum corrections especially for playing darts. At first the model describing inaccurate shooting is made. Using this model we received prerequisites for numerical solving of our problem. Computing algorithm was made and than was proven that our problem can be solved with arbitrary precision by this algorithm. The algorithm was implemented in MATLAB and modified using functional Analysis to minimize computing time. Our problem was solved by this algorithm and in conclusion was made simple application for visualization of received data.

Klíčová slova

nepřesná střelba, házení šipek, metoda Monte Carlo, balistická křivky, úloha o posunutí funkce, integrální transformace, numerická integrace, algoritmus Pernstejn

Keywords

inaccuracy shooting, playing darts, ballistic curve, Monte Carlo method, function translation problem, integral transforming, Pernstejn algorithm

Citace

Jan Horníček: Optimální korekce nepřesné střelby, bakalářská práce, Brno, FSI VUT v Brně, 2012

Optimální korekce nepřesné střelby

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana RNDr. Pavla Popely Ph.D.

.....
Jan Horníček
24. května 2012

Poděkování

Chci poděkovat doktoru Pavlu Popelovi za jeho cenné připomínky v průběhu psaní celé práce, převážně pak v závěru za vyhledání řady prací na podobné téma, s jejichž výsledky porovnávám svá vypočtená data. Dále chci poděkovat doc. RNDr. Liboru Čermákovi CSc. za ochotnou pomoc při vyhledávání vhodné odborné literatury týkající se problematiky numerické integrace.

Obsah

Kapitola 1

Úvod

Loni na svatého Václava, tedy 28.9. jsme s několika kamarády podnikli výlet. Nebyl jsem toho dne ale příliš společensky použitelný. Mysl jsem měl příliš zaměstnanou úlohou, kterou jsem řešil už od předchozího večera.

Házím šipky na terč. Znám míru nepřesnosti svých hodů a ptám se, na které místo terče mířit, abych v průměru získal nejvyšší skóre. Hrál jsem si v hlavě s funkcemi, řezal je, posouval, dokud mi nenapadla transformace integrálů, která je jádrem algoritmu `Pernstejn` popsaného v páté kapitole. Tehdy už jsem neměl stání. Jen být u MATLABu!

Skutečně, když jsem pak v devět hodin večer přišel domů, okamžitě jsem zasedl k počítači a už v jednu ráno jsem měl první neumělé zprogramování úlohy a první výsledky.

Teď by se mohlo zdát, že jsem hlavní část této práce napsal za večer. Tak tomu samozřejmě není. Uvažuji házení šipek jako speciální případ nepřesné střelby a aby byly vypočtené výsledky použitelné, bylo nutné ukázat, že jsou splněna poměrně přísná kritéria kladená na vlastnosti této nepřesné střelby. K tomu bylo potřeba provést řadu počítačových simulací házení šipek na terč podél balistických křivek popsaných diferenciálnimi rovnicemi a testovat hypotézy o rozdělení náhodného vektoru souřadnic zasažených bodů terče.

Když jsem prokázal, že použití jistého specifického typu nepřesné střelby není statisticky významnou újmou na obecnosti úlohy, musel jsem ukázat, že integrální transformace, která je jádrem numerického řešení úlohy, je matematicky korektní. Dále bylo nutno dokázat, že řešení získané pomocí výpočetní techniky není zatížené numerickými chybami tak, že by bylo nepoužitelné. A v poslední řadě mě čekalo závěrečné zpracování velkého množství dat (221MB) a shrnutí formou uživatelsky jednoduché vizualizační aplikace.

Znamenalo to použít poznatky z širokého spektra matematických disciplín. Počínaje optimalizací, přes statistiku, pravděpodobnost, funkcionální analýzu až po numerické metody.

Na závěr tohoto poněkud nematematického povídání prozradím, proč jsem nosný výpočetní algoritmus celé práce pojmenoval `Pernstejn`. Ano, pozornější čtenář již jistě tuší; cílová destinace našeho svatováclavského výletu byl totiž právě hrad Pernštejn tyčící se nad řekou Svratkou.

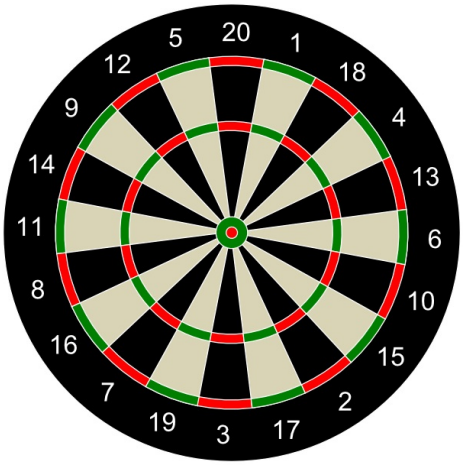
Abych už ale dále neodbíhal od tématu, doplním dvě poznámky k práci samotné. Ačkoli nese název Optimálni korekce nepřesné střelby, řešil jsem vlastně jen hledání optimálního místa terče pro různé míry nepřesnosti. Přesto věřím, že některé výsledky, případně algoritmy za jistého zobecnění budou použitelné i pro širší spektrum úloh týkajících se nepřesné střelby.

Žádný z důkazů, které jsem v práci sestrojil není myšlenkově složitý. Mnohdy, ale jeho zpracování bylo technicky poměrně komplikované a základní idea může být zamlžena. Proto nejsou důkazy rozhodně nutné pro pochopení celé práce a čtenář je může bez problémů přeskočit, aniž by ztratil myšlenkovou linii textu.

Kapitola 2

Sestavení modelu

Začněme matematickým popisem terče a pojmu nepřesné střelby. Tvar a rozměry terče jsou dány sportovními pravidly (viz přílohu Pravidla). Terč se tedy skládá z dvaaosmedáti polí s obecně různým bodovým ohodnocením. Pokud umístíme počátek soustavy souřadnic do středu terče, můžeme každému z polí terče přiřadit oblast $T_i \subset \mathbb{R}^2$ (Na pořadí očíslování oblastí samozřejmě nezáleží). Pokud každou z těchto oblastí budeme uvažovat jako otevřenou množinu, pak jsou všechny množiny T_i po dvou disjunktní.[1] K této množině oblastí připojme ještě oblast $T_H = \cup_{i=1}^{82} \partial T_i$, kde ∂T_i je hranice i-té množiny. Pak $T = (\cup_{i=1}^{82} T_i) \cup T_H$ je celá oblast terče.

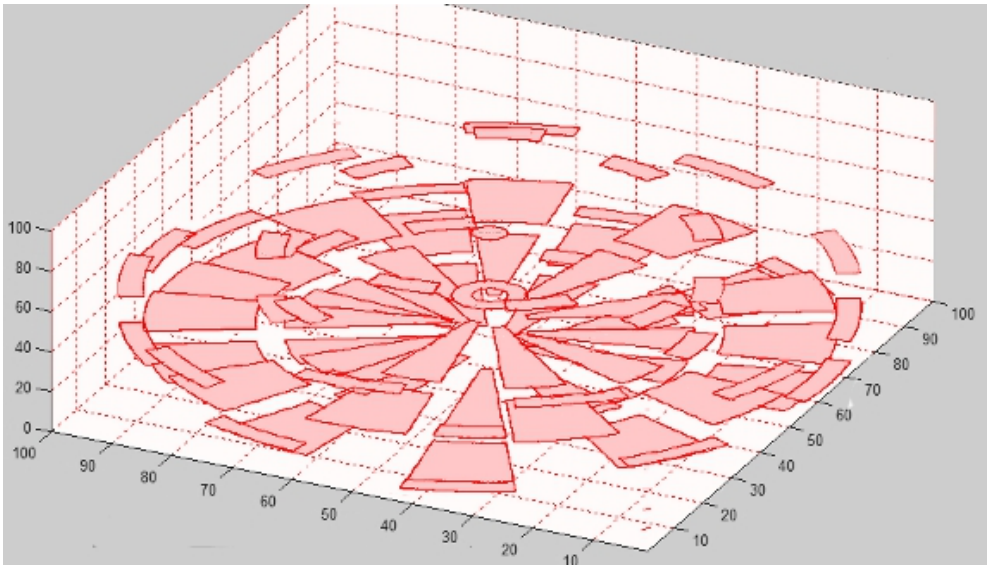


Obrázek 2.1: Terč

Nyní už můžeme definovat funkci $J : \mathbb{R}^2 \longrightarrow \mathbb{R}$, která je na každé oblasti T_i konstantní a nabývá zde hodnoty dané bodovým ohodnocením oblasti při jejím zasažením, na oblasti T_H je funkce J nulová a stejně tak mimo terč (mimo oblast T). Aby si čtenář mohl udělat lepší představu o tvaru funkce J, uvádíme její vykreslení na následující stránce (Obrázek 2.2).

Nepřesnou střelbu uvažujme jako náhodný vektor $\hat{\mathbb{X}}$ polohy zasaženého bodu terče $\mathbb{X} = (x, y)$. Střední hodnotou tohoto náhodného vektoru pak je bod $\mathbb{M} = (\mu_1, \mu_2) \in \mathbb{R}^2$, na který míříme. Rozdělení pravděpodobností náhodného vektoru $\hat{\mathbb{X}}$ může být prakticky libovolné. Pro náš konkrétní případ, tedy házení šipek na terč, ale budeme předpokládat, že tento vektor má dvourozměrné normální rozdělení, které je navíc rotačně symetrické. Respektive, že jeho odchylka od tohoto rozdělení není statisticky významná. Podrobněji jsou vlastnosti náhodného vektoru nepřesné střelby při házení šipek na terč studovány v následující kapitole. Za uvedených přepokladů, je hustota nepřesné střelby dána předpisem:

[1] Předpoklad otevřenosti množin nelze vynechat, protože pokud by obě sousední oblasti byly uzavřené, již by měly společnou hranici a nebyly by tedy disjuktní. Více o tzv. paradoxu matematického stříhání např. v [4].



Obrázek 2.2: Terčová funkce J

$$\Phi_{\sigma^2,(\mu_1,\mu_2)}(\mathbb{X}) = \Phi_{\sigma^2,(\mu_1,\mu_2)}(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu_1)^2+(y-\mu_2)^2}{2\sigma^2}},$$

kde σ^2 je rozptyl, tzn. míra nepřesnosti nepřesné střelby.

Připojme ještě poznámku, že se dopouštíme záměrně jisté odchylky od obvyklého značení. Hustota normální rozdělení se většinou značí φ, ale protože Φ vystupuje celou dobu s funkcí J, dovolíme si z důvodů estetických značit hustotu právě Φ.

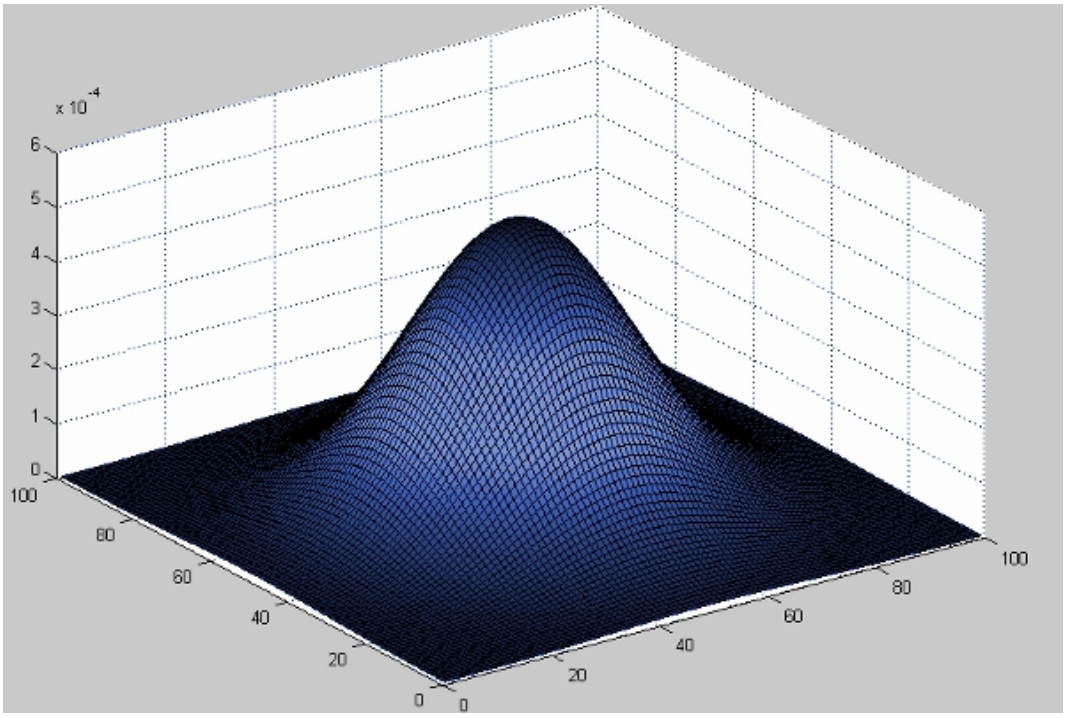
Pravděpodobnost zasažení libovolné oblasti T_i je dána integrálem:

$$\int_{T_i} \Phi_{\sigma^2,(\mu_1,\mu_2)}(\mathbb{X}) d\mathbb{X}.$$

Pokud každý z těchto integrálů vynásobíme hodnotou terčové funkce na dané množině a integrály nad všemi oblastmi sečteme, dostaneme vztah

$$\int_{\mathbb{R}^2} \Phi_{\sigma^2,(\mu_1,\mu_2)}(\mathbb{X}) \cdot J(\mathbb{X}) d\mathbb{X} = PS(\sigma^2,\mu_1,\mu_2), \tag{2.1}$$

kde $PS(\sigma^2,\mu_1,\mu_2)$ je označení pro průměrné získané skóre při dané střelbě, které obecně závisí na bodu (μ_1,μ_2), na nějž míříme a na míře nepřesnosti střelby σ^2. Naší úlohou je najít při každé míře nepřesnosti σ^2 takový bod $(\mu_1,\mu_2) \in \mathbb{R}^2$, aby $PS(\sigma^2,\mu_1,\mu_2) \geq PS(\sigma^2,x,y)$ pro $\forall (x,y) \in \mathbb{R}^2$. Tento bod nazveme **optimálním bodem střelby.**



Obrázek 2.3: Hustota náhodného vektoru nepřesné střelby $\Phi_{\sigma^2,(\mu_1,\mu_2)}$

Kapitola 3

Monte Carlo

Předpoklad, že hustota rozdělení pravděpodobností Φ náhodného vektoru $\hat{\mathbb{X}}$, jehož konkrétní realizací je zasažený bod $\mathbb{X}$, je rotačně symetrická kolem bodu $\mathbb{M} = (\mu_1, \mu_2)$, je velmi silný. Cílem této kapitoly proto bude ukázat, že ačkoli je silný, není tento předpoklad neúměrně omezující. Že i praktické realizace tohoto náhodného vektoru, tj. reálné házení šipek na terč, jsou tomuto rotačnímu rozdělení dostatečně blízké. Neboli, že odchylky od tohoto rozdělení nejsou statisticky významné.[1]

Základním aparátem této části bude metoda Monte Carlo. "*Metoda Monte Carlo je metodou stochastickou, což znamená, že hledaný výsledek je získáván na základě počtu pravděpodobnosti. Mezi tvůrce metody patří především J. von Neumann, E. Fermi a N. Metropolis. Její prudký rozvoj lze datovat obdobím konce druhé světové války, kdy jsou v oblasti atomového výzkumu využívány počítače. Statistické základy metody formuloval v této době J. von Neumann.*

Základní principy však byly vysloveny již v roce 1777, kdy Georges de Buffon formuloval svůj známý problém s jehlou (...). Na základě řešení této úlohy pak bylo možno určit s využitím náhodných jevů hodnotu čísla π. Nevýhodou však byla nutnost realizace velkého počtu náhodných jevů, což je problém, který je bez počítače těžko řešitelný.

Řešení problému metodou Monte Carlo (MMC) můžeme rozdělit do tří kroků.

1. *Rozbor problému a návrh modelu*

   z hlediska řešení problému se jedná o nejdůležitější krok. I když je MMC použitelná prakticky u všech problémů a její formulace není složitá, nalezení vhodného postupu může nezkušenému řešiteli činit nemalé potíže.

2. *Generování náhodných veličin, jejich transformace na veličiny s daným statistickým rozdělením.*

   Tento krok bývá zpravidla opakován v cyklu, dokud se hledaná hodnota příliš neliší od hodnoty dané výpočtem. Rychlost konvergence chyby výsledku k nulové hodnotě je u MMC přibližně rovna převrácené hodnotě odmocniny z počtu realizovaných pokusů N, z čehož plyne, že nepatří mezi metody nejefektivnější.

3. *Statistické zpracování výsledků.*" [10]

[1] Ještě uveďme jednu poznámku týkající se celé kapitoly. Probíraná látka je na pomezí mezi mechanikou pohybu těles a teorií pravděpodobnosti. Jistě by nebylo chybou veškeré použité symboly a značení vysvětlit a ujednotit. Ovšem s ohledem na to, že daná symbolika je použita právě jen v této kapitole a že použití symboliky intuitivně vychází z rozboru úlohy, dovolme si užít značení neúplně rigorózní.

3.1 Sestavení modelu

V první řadě tedy sestavme matematický model popisující pohyb šipky. Uvažujme tento pohyb jako pohyb hmotného bodu v gravitačním poli země při odporu vzduchu. Těla šipek se vyrábějí ze slitin s vysokou hustotou, jako jsou mosazi, či slitiny wolframu a niklu, převážná část hmotnosti je tak koncentrována do poměrně malého prostoru a zjednodušení formou hmotného bodu je přijatelné.

Nebudeme dále uvažovat efekt plachtění, případně jiné efekty, které ale způsobují jen mírnou odchylku reality od našeho modelu. Nabízí se ještě možnost popisu zanedbávajícího odpor vzduchu. Tento přístup by neznamenal přílišnou odchylku v našem případě, kdy náhodnou střelbou je házení šipek na terč, pro jiné případy nepřesné střelby už by však rozdíl obou přístupů mohl být značný. Z tohoto důvodu budeme uvažovat nepřesnou střelbu obecněji a odpor vzduchu nezanedbáme.

Ještě než se pustíme do samotného odvozování, je nutné uvést poznámku k použitým souřadným systémům. V průběhu této kapitoly budou vystupovat různé souřadné systémy. Zaveďme si nyní první tři. Čtvrtý, pomocný, dodefinujeme později.

Realizace $\mathbb{X}$ náhodného vektoru $\hat{\mathbb{X}}$ má význam polohy v rovině terče.[2] Jeho složky v této kapitole tedy značme T_x, T_y, abychom zdůraznili, že jde o souřadnice v systému, který odpovídá rovině terče a jehož počátek leží ve středu terče.

Druhým vystupujícím systémem bude systém hlavní, ve kterém probíhá pohyb šipky. Souřadnice značme x, y, z. Rovina určená osami y, z je rovnoběžná s rovinou terče a osa x je na ni kolmá. Počátek je umístěn do bodu startu šipky.

Poslední souřadný systém má společný počátek a osu z s hlavním souřadným systém, je ale skloněn o úhel α, který svírá vektor počáteční rychlosti šipky s osou x. Přes úhel α také můžeme provést rozklad počáteční rychlosti v_0 do složek v_{x0} a v_{y0} takto: $v_{x0} = v_0 \cos\alpha, v_{y0} = v_0 \sin\alpha$. Souřadnice tohoto systému značme $\hat{x}, \hat{y}, \hat{z} (\hat{z} \equiv z)$. Viz obrázek 3.1.

Rovnic pohybu šipek odvozeny podle [12]. Vyjdeme z druhého Newtonova zákona a dostaneme dvě rovnice:

$$\Sigma F_x = -C_d v_x = m a_x,$$

kde ΣF_x je suma vnějších sil působících na těleso ve směru osy x, C_d je součinitel odporu vzduchu, v_x je složka rychlosti ve směru osy x, m je hmostnost a a_x je změna rychlosti (zrychlení) ve směru osy x, a

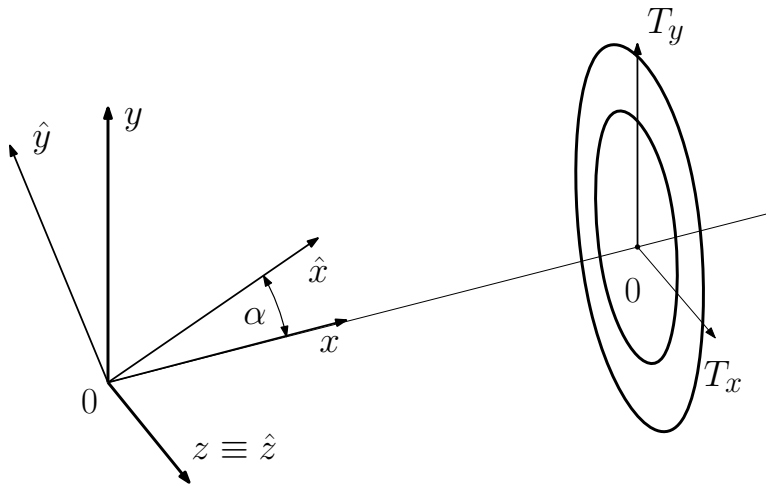
$$\Sigma F_y = -C_d v_y - mg = m a_y,$$

kde ΣF_y je suma vnějších sil působících na těleso ve směru osy y, v_y je složka rychlosti ve směru osy y, g je tíhové zrychlení a a_y je změna rychlosti (zrychlení) ve směru osy y. Protože $a_x = \frac{dv_x}{dt}$ a $a_y = \frac{dv_y}{dt}$ dostáváme dvě obyčejné diferenciální rovnice prvního řádu:

$$\frac{-C_d v_x}{m} = \frac{dv_x}{dt}, \qquad \frac{-C_d v_y}{m} - g = \frac{dv_y}{dt}.$$

Řešíme je pro počáteční podmínky: $v_x = v_{x0}$ a $v_y = v_{y0}$v čase $t = 0$, pak

[2] Dopouštíme se samozřejmě odchylky od značení obvyklého u statistických textů, kde se $\hat{\mathbb{X}}$ používá pro značení odhadu. Zde užíváme tohoto značení pro náhodný vektor namísto obvyklého $\mathbb{X}$, protože $\mathbb{X}$ má již význam realizace náhodného vektoru $\hat{\mathbb{X}}$.



Obrázek 3.1: Souřadné systémy

$$v_x = v_{x0} e^{-\frac{C_d}{m} t} \tag{3.1}$$

a integrací (protože $v_x = \frac{ds_x}{dt}$ s počátečními podmínkami $s_x = 0$ pro $t = 0$)

$$s_x = \frac{m}{C_d} v_{x0} (1 - e^{-\frac{C_d}{m} t}) \tag{3.2}$$

Druhá rovnice už je nehomogenní, řešení uvedené v [12] využívá teorii integračního faktoru (viz třeba: [13]). Mějme integrační faktor $M(t) = e^{\int \frac{C_d}{m} dt}$, z $\frac{-C_d v_y}{m} - g = \frac{dv_y}{dt}$ potom dostaneme

$$e^{\frac{C_d}{m} t} (\frac{dv_y}{dt} + \frac{C_d v_y}{m}) = e^{\frac{C_d}{m} t} (-g)$$

$$(e^{\frac{C_d}{m} t} v_y)' = e^{\frac{C_d}{m} t} (-g)$$

$$\int (e^{\frac{C_d}{m} t} v_y)' dt = e^{\frac{C_d}{m} t} v_y = \int e^{\frac{C_d}{m} t} (-g) dt$$

$$e^{\frac{C_d}{m} t} v_y = \frac{m}{C_d} e^{\frac{C_d}{m} t} (-g) + K$$

$$v_y = -\frac{mg}{C_d} + K_1 e^{\frac{-C_d}{m} t}$$

z čehož integrací

$$s_y = -\frac{mg}{C_d} t - \frac{m}{C_d} K_1 e^{-\frac{C_d}{m} t} + K_2.$$

Po dosazení počátečních podmínek $v_y = v_{y0}$, $s_y = 0$ pro $t = 0$ pak už dostaneme výsledné řešení ve tvaru

$$v_y = -\frac{mg}{C_d} + (v_{y0} + \frac{mg}{C_d})e^{\frac{-C_d}{m}t} \tag{3.3}$$

$$s_y = -\frac{mg}{C_d}t - \frac{m}{C_d}(v_{y0} + \frac{mg}{C_d})e^{-\frac{m}{C_d}t} + \frac{C_d}{m}(v_{y0} + \frac{mg}{C_d})$$

a po úpravě

$$s_y = -\frac{mg}{C_d}t + \frac{m}{C_d}(v_{y0} + \frac{mg}{C_d})(1 - e^{-\frac{C_d}{m}t}) \tag{3.4}$$

Pro tyto sestavené rovnice ((3.1), (3.2), (3.3), (3.4))potřebujeme ještě znát velikosti vystupujících veličin. Potřebné rozměry byly určeny podle sportovních pravidel pro házení šipek, viz přílohu Pravidla.

Nyní se přesuneme na půdu nepříliš exaktní. Budeme určovat parametry letu šipky. Provádět to experimentálně s vysokou přesností by ale nedávalo valný smysl. Jde nám jen o hodnoty přibližné. Při simulaci budeme navíc každou hodnotu volit v jistém spektru tak, abychom výsledky nedostali jen pro nepřesnou střelbu s jedním konkrétním specifickým počátečním nastavením.

O hledaných parametrech se můžeme dočíst toto: "Průměrná rychlost šipky při nárazu do terče je $64km/h$ $(40mil/h)$."[3] Rychlost šipky při opouštění ruky střelce je pak jen o málo větší. Součinitel odporu vzduchu šipek se pohybuje kolem $0,2$.[4] "Lehčí jsou šipky softové, které mají max povolenou hmotnost $20g$. Nejpoužívanější gramáže jsou v rozpětí $14 - 18g$. Steelové šipky jsou vyráběny s gramáží až přes $30g$. Přesto je nejběžnější gramáž $21 - 30g$."[5] Tíhové zrychlení budeme brát $g = 9,81ms^{-1}$.

Simulaci samotnou budeme provádět pro tři různá nastavení každé z veličin, u kterých to má smysl, a jejich kombinace. Viz následující tabulku.

hmotnost (m) [kg]	rychlost (v_0) [ms^{-1}]	C_d	S_y [m]
0,020	20	0,15	-0,2
0,025	25	0,2	0
0,030	30	0,25	0,2

Hodnoty neproměnných veličin pak jsou: poloha středu terče od místa výstřelu (x-ová složka) $Sx = 2,37m$ a tíhové zrychlení $g = 9,81ms^{-1}$.

Zbývá ještě určit hodnotu úhlu α a tedy i konkrétní rozklad rychlosti v_0 do složek v_{x0}, v_{y0}. Hodnoty (v radiánech) úhlu výstřelu α pro jednotlivá nastavení jsou uvedeny v nasledující tabulce.

[3] **http://www.sipky.cz/method/method.php**
[4] **http://en.allexperts.com/q/Aeronautical-Engineering-1809/Dart-Drag-Coefficient.htm**
[5] **http://www.sipky.org/cs/clanky/detail/?section_ id=2& article_ id=19**

	C_d	0,15	0,15	0,15	0,20	0,20	0,20	0,25	0,25	0,2500
v	$m \backslash S_y$	0	0,20	-0,2000	0	0,20	-0,2000	0	0,2000	-0,2000
30	0,030	0,0178	0,1021	-0,0664	0,0206	0,1049	-0,0636	0,0248	0,1092	-0,0594
30	0,025	0,0194	0,1037	-0,0648	0,0238	0,1082	-0,0604	0,0320	0,1166	-0,0523
30	0,020	0,0225	0,1068	-0,0617	0,0320	0,1166	-0,0523	0,1044	NaN	NaN
25	0,030	0,0279	0,1123	-0,0563	0,0343	0,1188	-0,0500	0,0461	0,1311	-0,0383
25	0,025	0,0314	0,1158	-0,0529	0,0429	0,1278	-0,0414	0,0857	0,1796	-0,0013
25	0,020	0,0391	0,1238	-0,0452	0,0857	0,1796	-0,0013	NaN	NaN	NaN
20	0,030	0,0507	0,1354	-0,0338	0,0724	0,1584	-0,0126	NaN	NaN	NaN
20	0,0250	0,0613	0,1465	-0,0234	0,1415	NaN	0,0467	NaN	NaN	NaN
20	0,0200	0,0985	0,1886	0,0118	NaN	NaN	NaN	NaN	NaN	NaN

Hodnoty úhlu α byly vypočteny MATLABem numerickým řešením rovnice, která je odvozena ze vztahů (3.2) a (3.4).

$$\frac{m^2 g}{C_d^2} \ln(1 - \frac{S_x C_d}{m v_0 \cos\alpha}) + S_x \tan\alpha + \frac{S_x g m}{C_d v_0 \cos\alpha} - S_y = 0 \tag{3.5}$$

Symbol `NaN` je MATLABovský výpis proměnné, která není číslem (Not a Number), je v těch pozicích tabulky, kde pro zvolené nastavení neexistuje žádný reálný úhel α, který by byl kořenem této rovnice. V praxi to znamená, že při daných parametrech nejsme schopni najít úhel, pod kterým bychom terč zasáhli. To je logické, neboť například pro malou rychlost střelby a velký odpor vzduchu šipka k terči prostě nedoletí.

Protože rovnici (3.5) budeme v dalším textu ještě hojně využívat, uveďme si její odvození. Z (3.2) můžeme vyjádřit:

$$(1 - e^{\frac{-C_d}{m}t}) = \frac{S_x C_d}{m v_{x0}} \tag{3.6}$$

a dále

$$1 - \frac{S_x C_d}{m v_{x0}} = e^{\frac{-C_d}{m}t}, \quad \ln(1 - \frac{S_x C_d}{m v_{x0}}) = -\frac{C_d}{m}t$$

$$t = \frac{-m}{C_d} \ln(1 - \frac{S_x C_d}{m v_{x0}}) \tag{3.7}$$

Pokud vztahy (3.6), (3.7) dosadíme do rovnice (3.4), dostaneme již výsledný vztah (3.5).

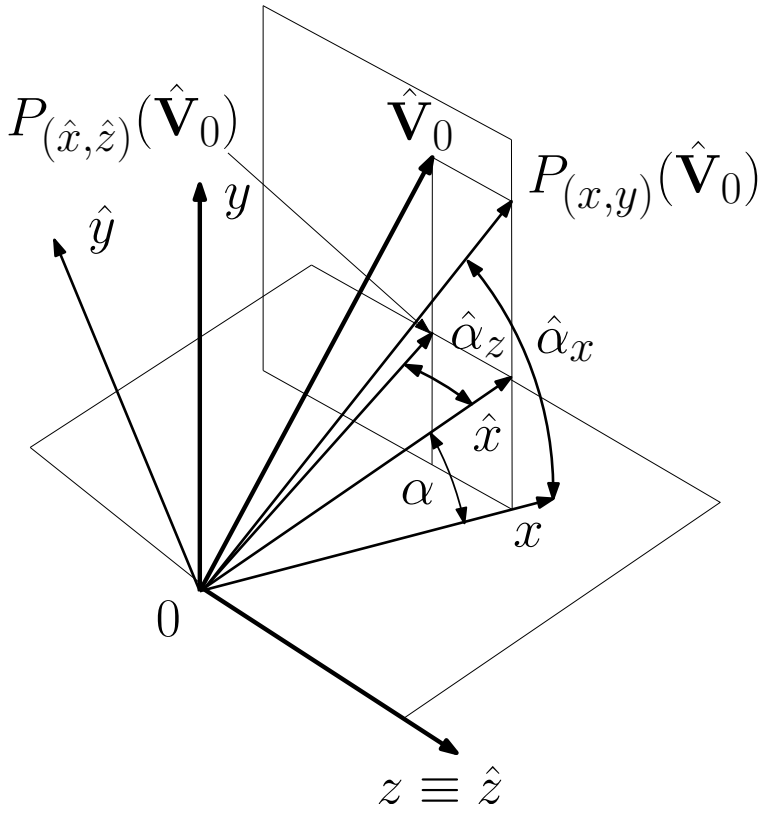
Místo vektoru $\mathbb{V}_0$ počáteční rychlosti přesné střelby, kterým při daném souboru vstupních parametrů (C_d, m, S_y) zasáhneme přesně požadovaný bod terče, uvažujme nyní vektor $\hat{\mathbb{V}}_0$ počáteční rychlosti nepřesné střelby, který je zatížen nějakými náhodnými chybami. Zatímco složka v_{z0} vektoru $\mathbb{V}_0 = (v_{x0}, v_{y0}, v_{z0})$ byla nulová, u vektoru $\hat{\mathbb{V}}_0 = (\hat{v}_{x0}, \hat{v}_{y0}, \hat{v}_{z0})$ tomu tak již obecně není.

Chybovou složku vektoru $\hat{\mathbb{V}}_0$ je nyní nutné nějak účelně zvolit, aby co nejpřesněji popisovala reálnou nepřesnost nepřesné střelby (v našem případě házení šipek), ale také, aby pravděpodobnostní model byl pokud možno co nejjednodušší.

Uvažujme, že vlivem lidského faktoru bude docházet k náhodnému kolísání velikosti v_0 počáteční rychlosti $\hat{\mathbb{V}}_0$ a dále k nezávislému kolísání prostorového úhlu počáteční rychlosti ve dvou na sebe kolmých rovinách. Parametry $\hat{v}_0, \hat{\alpha}_x, \hat{\alpha}_z$, které jednoznačně určují $\hat{\mathbb{V}}_0$, jsou tedy konkrétními realizacemi náhodných veličin $V_0, \Lambda_x, \Lambda_z$ a rozdělení těchto veličin zvolme takto: $V_0 \sim N(v_0, \kappa^2 \cdot \nu^2), \Lambda_x \sim N(\alpha, \nu^2), \Lambda_z \sim N(0, \nu^2)$. Rozptyly jednotlivých rozdělení jsou tedy u úhlových složek voleny stejné, nedávalo by dobrý smysl, aby nepřesnost v některé

ose míření byla větší než v jiné. Rozptyl rozdělení rychlostí jsme pak pro jednoduchost položili jako násobek rozptylu úhlových složek. Při simulaci opět volíme κ ve třech různých variantách, konkrétně 0,2; 1; 5.

Konkrétní geometrický význam parametrů $\hat{v}_0, \hat{\alpha}_x, \hat{\alpha}_z$ je následující: $\hat{v}_0$ určuje velikost vektoru počáteční rychlosti $\hat{\mathbb{V}}_0$, její střední hodnota je v_0; $\hat{\alpha}_x$ se střední hodnotou α je skutečný úhel mezi osou x a projekcí vektoru počáteční rychlosti $\hat{\mathbb{V}}_0$ do roviny (x, y) označme ji $P_{(x,y)}(\hat{\mathbb{V}}_0)$; $\hat{\alpha}_z$ se střední hodnotou 0 je skutečný úhel mezi osou $\hat{x}$ a projekcí náhodného vektoru $\hat{\mathbb{V}}_0$ do roviny $(\hat{x}, \hat{z})$ označme ji $P_{(\hat{x},\hat{z})}(\hat{\mathbb{V}}_0)$. Viz obrázek 3.2, jen upozorněme, že v obrázku je místo znaku $\hat{\mathbb{V}}_0$ použit znak $\hat{\mathbf{V}}_0$.

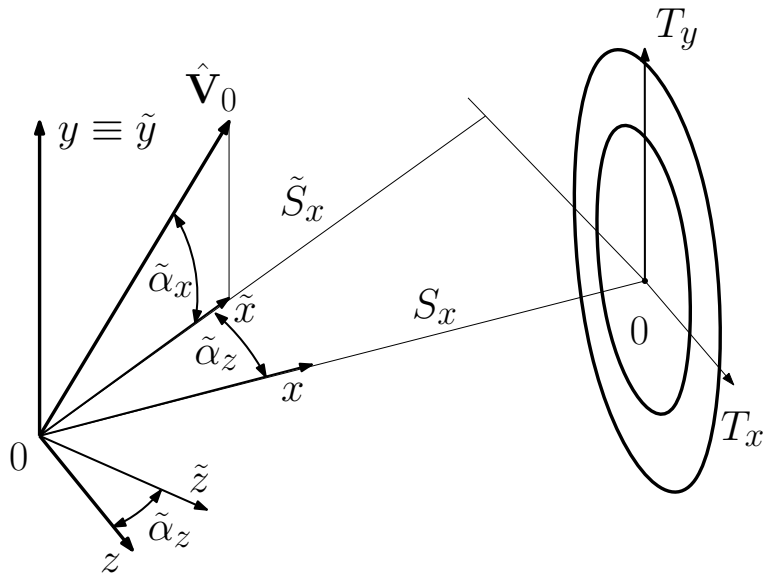


Obrázek 3.2: Geometrický význam vstupních nepřesností střelby

Pomocí $\hat{v}_0, \hat{\alpha}_x, \hat{\alpha}_z$ můžeme vyjádřit jednotlivé složky $\hat{v}_{x0}, \hat{v}_{y0}, \hat{v}_{z0}$ vektoru $\hat{\mathbb{V}}_0 = (\hat{v}_{x0}, \hat{v}_{y0}, \hat{v}_{z0})$. Rovnice pro pohyb hmotného bodu v gravitačním poli jsme si ale odvodili pouze ve dvou dimenzích, pokud chceme, aby pro nás byly použitelné, musíme najít pomocný souřadný systém, ve kterém bude z-tová složka počáteční rychlosti nulová. To však není nijak komplikované; souřadným systém (x, y, z) pouze otočíme kolem osy y o úhel $\tilde{\alpha}_z$, viz obrázek 3.3.

Využijeme-li připravenou rovnici (3.5), můžeme již spočítat souřadnice T_x, T_y zasaženého bodu $\mathbb{X}$ nepřesné střelby $\hat{\mathbb{X}}$ takto:

$$T_y = \frac{m^2 g}{C_d^2} \ln(1 - \frac{\tilde{S}_x C_d}{m \tilde{v}_0 \cos \tilde{\alpha}_x}) + \tilde{S}_x \tan \tilde{\alpha}_x + \frac{\tilde{S}_x g m}{C_d \tilde{v}_0 \cos \tilde{\alpha}_x} - S_y \tag{3.8}$$



Obrázek 3.3: Výsledný souřadný systém nepřesné střelby

$$T_x = S_x \tan(\tilde{\alpha}_z), \tag{3.9}$$

přičemž hodnoty parametrů m, g, C_d, S_y, α a $S_x = 2,37m$ známe. Pak již jsme schopni zbylé parametry $\tilde{S}_x, \tilde{\alpha}_x, \tilde{\alpha}_z, \tilde{v}_0$ vyjádřit jako funkce realizací $\hat{v}_0, \hat{\alpha}_x, \hat{\alpha}_z$ náhodných veličin $V_0 \sim N(v_0, \kappa^2 \cdot \nu^2), \Lambda_x \sim N(\alpha, \nu^2), \Lambda_z \sim N(0, \nu^2)$.

$$\tilde{v}_0 = \hat{v}_0 \tag{3.10}$$

$$\tilde{\alpha}_z = \arctan(\tan \hat{\alpha}_z \cos \alpha) \tag{3.11}$$

$$\tilde{S}_x = \frac{S_x}{\cos \tilde{\alpha}_z} = \frac{S_x}{\cos[\arctan(\tan \hat{\alpha}_z \cos \alpha)]} \tag{3.12}$$

$$\tilde{\alpha}_x = \arctan(\tan \hat{\alpha}_x \cos \tilde{\alpha}_z) = \arctan\{\tan \hat{\alpha}_x \cos[\arctan(\tan \hat{\alpha}_z \cos \alpha)]\} \tag{3.13}$$

3.2 Samotná simulace

V druhém kroku MMC budeme generovat trojice náhodných veličin z rozdělení

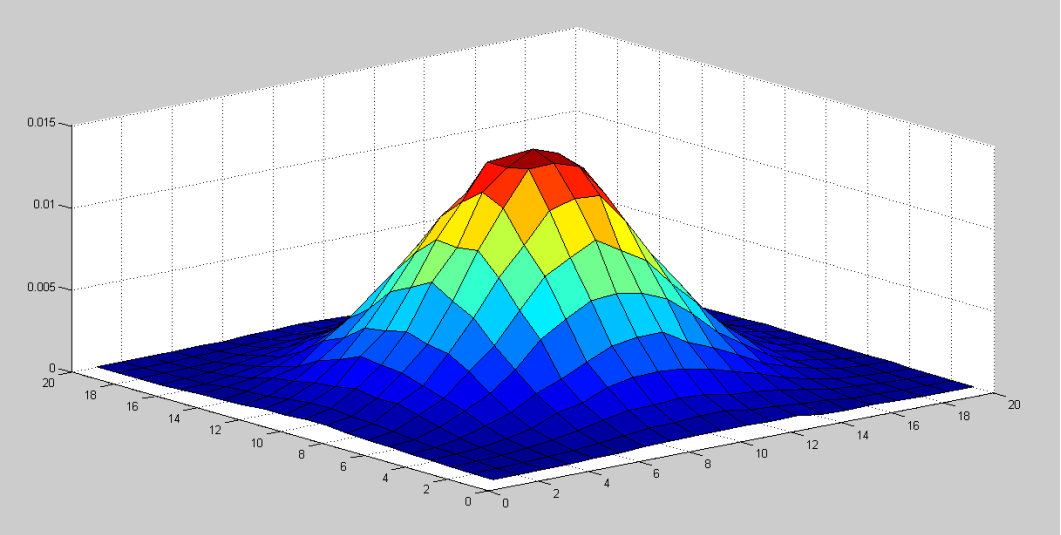
$$V_0 \sim N(v_0, \kappa^2 \cdot \nu^2), \Lambda_x \sim N(\alpha, \nu^2), \Lambda_z \sim N(0, \nu^2).$$

Model sestavený z rovnic (3.8) až (3.13) pak každé trojici $\hat{v}_0, \hat{\alpha}_x, \hat{\alpha}_z$ realizací těchto náhodných veličin přiřadí jednoznačně bod $\mathbb{X} = (T_x, T_y)$ jako realizaci náhodného vektoru nepřesné střelby $\hat{\mathbb{X}}$, který je transformací náhodných veličin $V_0, \Lambda_x, \Lambda_z$ podél balistické křivky o daných parametrech.

Tato část je prováděna za pomoci počítače. Pro každý soubor vstupních parametrů m, g, C_d, S_y, α a κ je provedena simulace 200 000 nepřesných střeleb. Dostaneme tak náhodný výběr z $\hat{\mathbb{X}}$ náhodného vektoru nepřesné střelby o rozsahu 200 000. Výběr z $\hat{\mathbb{X}}$ je průběžně tříděn ukládán do matice `SberMat`, jejíž vlastnosti jsou ve třetím kroku MMC testovány.

Pro generování pseudonáhodných čísel s normálním rozdělením používáme MATLABovskou funkci `randn`.

Uveďme ještě pro názornost vykreslení utříděného výběru z $\hat{\mathbb{X}}$, o kterém tvrdíme, že jeho odlišnost od náhodného vektoru majícího rotačně symetrické normální rozdělení není statisticky významná, viz obrázek 3.4.



Obrázek 3.4: Vykreslení sběrné matice

3.3 Statistické zpracování výsledků

Výběr z rozdělení náhodné vektoru $\hat{\mathbb{X}}$ byl roztříděn a jednotlivé realizace uloženy do sběrné matice `SberMat`.

O náhodném vektoru $\hat{\mathbb{X}}$ chceme tvrdit, že $\hat{\mathbb{X}} \sim N_2(0, 0, \sigma^2, \sigma^2, 0)$, tedy že $\hat{\mathbb{X}}$ má dvourozměrné rotačně symetrické normální rozdělení. To mimo jiné znamená, že hustoty marginálních rozdělení

$$\Phi_1(T_x) = \int_\Omega \Phi(T_x, T_y) dT_y,$$

$$\Phi_2(T_y) = \int_\Omega \Phi(T_x, T_y) dT_x$$

jsou hustoty jednorozměrného normálního rozdělení o stejných parametrech. Dále pak také, že i hustoty

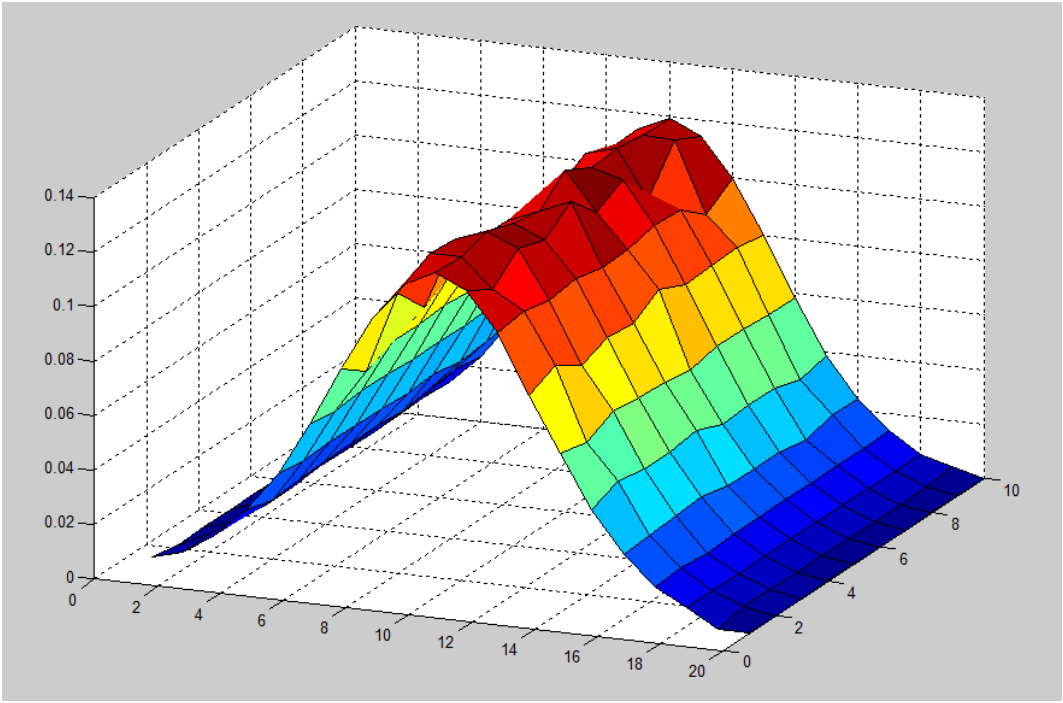
$$\Phi_\gamma = \int_\Omega \Phi(T_x, T_y) d\lambda,$$

kde $\lambda : y = \gamma T_x$, jsou také hustoty normálního rozdělení o stejných parametrech jako $\Phi_1(T_x), \Phi_2(T_y)$ pro všechny $\gamma \in (-\frac{\pi}{2}, \frac{\pi}{2})$.

Při vyšetřování vlastností $\hat{\mathbb{X}}$ budeme tedy postupovat takto: Z matice `SberMat` nejprve pro různé hodnoty $\gamma \in (-\frac{\pi}{2}, \frac{\pi}{2})$ spočteme Φ_γ jako jednorozměrná rozdělení. Pro každé zvlášť budeme testovat normalitu a poté shodnost rozptylů jednotlivých rozdělení.

Na obrázku (3.5) jsou pro porovnání uvedeny hustoty jednotlivých funkcí Φ_γ řazené za sebou do jednoho grafu.

Pro testování hypotézy o normalitě použijeme Pearsonův χ - kvadrát test.



Obrázek 3.5: Porovnání hustot funkcí Φ_γ

"*Testujeme hypotézu H, že pozorovaná náhodná veličina X má distribuční funkci $F(X)$, proti alternativní hypotéze $\bar{H}$, že X nemá distribuční funkci $F(X)$. Roztřídíme získaný statistický soubor $(x_1, ..., x_n)$ do m tříd s četností f_i a vypočteme teoretické absolutní četnosti $\tilde{f} = n(F(x_j^+) - F(x_{j-1}^+))$ pro $j = 1, ..., m$, kde x_j^+ značí pravý koncový bod j-té třídy, přičemž klademe $x_0^+ = -\infty$ a $x_m^+ = +\infty$. Statistický soubor roztřídíme tak, aby ve všech třídách byly dostatečně velké teoretické absolutní četnosti - obvykle požadujeme, aby $\tilde{f}_j > 5$. Toho lze při dostatečně velkém rozsahu n dosáhnout vhodnou volbou tříd nebo sloučením již získaných sousedních tříd. Pozorovaná hodnota testového kritéria je*

$$t = \sum_{j=1}^{m} \frac{(f_j - \tilde{f}_j)^2}{\tilde{f}_j} = (\sum_{j=1}^{m} \frac{f_j^2}{\tilde{f}_j}) - n$$

a $\bar{W}_\alpha = \langle 0; \chi^2_{1-\alpha} \rangle$, kde $\chi^2_{1-\alpha}$ je $(1-\alpha)$ - kvantil Pearsonova rozdělení $\chi^2(k)$ s $k = m - g - 1$ stupni volnosti. (...) Číslo q je počet parametrů, hypotetického rozdělení náhodné veličiny X, které jsme nuceni odhadnout z roztříděného statistického souboru pro určení hodnot distribuční funkce $F(x)$. Uvedený test je zjednodušenou, ale obvykle používanou variantou přesného testu chí-kvadrát. " [9] str. 135,136.

V MATLABu (viz přílohu Nezmíněné skripty) byl tento test proveden pro všechny hustoty Φ_γ. Na hladině významnosti $\alpha = 0,05$ nebyla ani pro jednu hustotu Φ_γ zamítnuta hypotéza, že jde o hustotu normálního rozdělení. Budeme tedy předpokládat, že se opravdu jedná o hustoty normálních rozdělení a v dále testovat shodnost roztylů.

Test o hodnotách rozptylů pro jednotlivé $\Phi\gamma$ uděláme opět podle [9].

" **Test hypotézy** $H : \sigma^2 = \sigma_0^2$. *Pozorovaná hodnota testového kritéria je*

$$t = \frac{ns^2}{\sigma_0^2}$$

a $\bar{W}_\alpha = \langle \chi^2_{\alpha/2}; \chi^2_{1-\alpha/2} \rangle$, *kde* χ^2_P *je P-kvantil Pearsonova rozdělení* $\chi^2(k)$ *s* $k = n - 1$ *stupni volnosti.*", strana 128.

Doplňme, že $s^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$ (kde $\bar{X}(= \frac{1}{n}\sum_{i=1}^{n} X_i)$ je průměrná hodnota statistického souboru) je výběrový rozptyl, který ale není nestranný. Někteří autoři proto zavádí výběrový rozptyl $\hat{s}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$, který už je nestranný. V našem případě, kdy rozsah výběru je 200 000, je ale rozdíl obou přístupů zanedbatelný, zůstaňme tedy u značení podle [9].

Našim cílem je testovat hypotézu, že rozptyly všech rozdělení s hustotami Φ_γ jsou shodné. Provedeme to tak, že spočteme s_0^2 průměrnou hodnotu výběrových rozptylů $\hat{s}^2_\gamma = \frac{1}{n}\sum_{i=1}^{n}(X_i^\gamma - \bar{X}^\gamma)^2$ a položíme $\sigma^2 = \sigma_0^2 = s_0^2$. Pro jednotlivé náhodné veličiny odpovídající hustotám Φ_γ jsme testovali hypotézu $H : \sigma^2 = s_0^2$ opět na hladině významnosti $\alpha = 0,05$. Tento test jsem opět provedli pomocí MATLABu. Ani v tomto případě nebyla hypotéza H zamítnuta pro žádnou náhodnou veličinu danou hustotou Φ_γ.

Nyní již můžeme uvést následující tvrzení:

Pro házení šipek na terč nemá náhodný vektor zasažených bodů terče statisticky významně odlišné rozdělení od rozdělení normálního rotačně symetrického.[6]

[6] Způsob testování hypotézy o shodnosti rozptylů není standartní. Běžně se pro tento případ používá Bartlettův test viz třeba [11].

Kapitola 4

Příprava numerického řešení

Vraťme se k druhé kapitole a předpisu 2.1

$$\int_{\mathbb{R}^2} \Phi_{\sigma^2,(\mu_1,\mu_2)}(\mathbb{X}) \cdot J(\mathbb{X}) d\mathbb{X} = PS(\sigma^2, \mu_1, \mu_2).$$

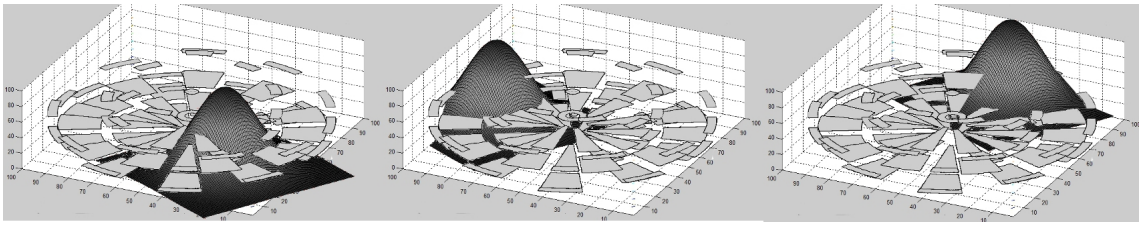
Úlohu najít maximum funkce $PS(\sigma^2, \mu_1, \mu_2)$ můžeme nazvat úlohou o posunutí funkce, kterou můžeme obecně formulovat takto:

Obecná úloha o posunutí funkce: Mějme funkce $J : \mathbb{R}^N \to \mathbb{R}$ a $\Phi : \mathbb{R}^N \to \mathbb{R}$ definované a integrovatelné s kvadrátem na $\mathbb{R}^N$ ($\int_{\mathbb{R}^N} \Phi^2 d\mathbb{X} < \infty$, respektive $\int_{\mathbb{R}^N} J^2 d\mathbb{X} < \infty$). Najděme parametr $\mathbb{M}_{Opt} \in \mathbb{R}^N$ tak, aby integrál

$$\int_{\mathbb{R}^N} \Phi(\mathbb{X} - \mathbb{M}_{Opt}) \cdot J(\mathbb{X}) d\mathbb{X}$$

byl maximální, tj. aby

$$\int_{\mathbb{R}^N} \Phi(\mathbb{X} - \mathbb{M}_{Opt}) \cdot J(\mathbb{X}) d\mathbb{X} \geq \int_{\mathbb{R}^N} \Phi(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X}) d\mathbb{X}, \forall \mathbb{M} \in \mathbb{R}^N.$$



Obrázek 4.1: Úloha o posunutí funkce - pro představu

Obecné řešení takto formulovaná úlohy jistě není cílem této práce. Uveďme si jen, jak by bylo možno postupovat, pokud by obě funkce J, Φ byly "rozumné" a k funkci $\Phi(\mathbb{X}-\mathbb{M})\cdot J(\mathbb{X})$ by existovala funkce primitivní. Pak bychom po provedení integrace obdrželi výraz závislý pouze na $\mathbb{M}$ a hledali bychom jeho maximum. Toto řešení je však použitelné opravdu jen pro úzkou skupinu funkcí.

Ani pro naši konkrétní úlohu není bohužel vhodný. Místo přesného analytického řešení se musíme spokojit s řešením přibližným numerickým. Idea je jednoduchá. Sestrojíme nějakou konečnou δ- síť M_δ[1] množiny T (pro tuto chvíli budeme předpokládat, že $\mathbb{M} = (\mu_1, \mu_2) \in T$, více o tomto viz kapitolu 5) a $\forall\, \mathbb{M} \in M_\delta$ spočteme

$$PS_{\sigma^2}(\mathbb{M}) = \int_{\mathbb{R}^2} \Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X}) d\mathbb{X}$$

Ze všech bodů $\mathbb{M} \in M_\delta$ pak vybereme ten, ve kterém je hodnota integrálu nejvyšší. Takto jsme našli přibližný optimální bod střelby při dané míře nepřesnosti σ^2. Označme ho $\hat{\mathbb{M}}_{Opt}$.

Lemma: Parametr δ jsme schopni volit tak, aby pro libovolné $\varepsilon > 0$ platilo, že

$$PS_{\sigma^2}(\mathbb{M}_{Opt}) - PS_{\sigma^2}(\hat{\mathbb{M}}_{Opt}) < \varepsilon.$$

Důkaz: Nechť $\mathbb{M}_{Opt}$ je optimální bod střelby a $\mathbb{N} \in M_\delta$ je bod ze všech bodů sítě M_δ bodu $\mathbb{M}_{Opt}$ nejbližší v euklidovské metrice. Pokud je těchto bodů více, vezmeme libovolný z nich. Dále nechť $\vec{v} = \mathbb{M}_{Opt} - \mathbb{N}$ je směrový vektor a J_i je hodnota funkce $J(\mathbb{X})$ nad oblastí T_i.

$$PS_{\sigma^2}(\mathbb{M}_{Opt}) - PS_{\sigma^2}(\mathbb{N}) = \sum_{i=1}^{82} \int_{T_i} \Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}_{Opt}) J_i d\mathbb{X} - \sum_{i=1}^{82} \int_{T_i} \Phi_{\sigma^2}(\mathbb{X} - \mathbb{N}) J_i d\mathbb{X} =$$

$$= \sum_{i=1}^{82} J_i \int_{T_i} [\Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}_{Opt}) - \Phi_{\sigma^2}(\mathbb{X} - \mathbb{N})] d\mathbb{X} \leq \sum_{i=1}^{82} J_i \int_{T_i} [\delta \max_{\mathbb{X} \in \mathbb{R}^2} \frac{\partial \Phi_{\sigma^2}(\mathbb{X})}{\partial \vec{v}}] d\mathbb{X} =$$

$$= \sum_{i=1}^{82} J_i \delta \text{meas} T_i \max_{\mathbb{X} \in \mathbb{R}^2} \frac{\partial \Phi_{\sigma^2}(\mathbb{X})}{\partial \vec{v}} \leq \delta \max_{i=1,\ldots,82} J_i \text{meas} T \max_{\mathbb{X} \in \mathbb{R}^2} \frac{\partial \Phi_{\sigma^2}(\mathbb{X})}{\partial \vec{v}}$$

(meas$T$ znamená Lebesqueovu míru množiny T)

Pokud volíme $\delta < \frac{\varepsilon}{\max_{i=1,\ldots,82} J_i \text{meas} T \max_{\mathbb{X} \in \mathbb{R}^2} \frac{\partial \Phi_{\sigma^2}(\mathbb{X})}{\partial \vec{v}}}$ pak

$$PS_{\sigma^2}(\mathbb{M}_{Opt}) - PS_{\sigma^2}(\mathbb{N}) \leq \delta \max_{i=1,\ldots,82} J_i \text{meas} T \max_{\mathbb{X} \in \mathbb{R}^2} \frac{\partial \Phi_{\sigma^2}(\mathbb{X})}{\partial \vec{v}} < \varepsilon$$

Protože v bodě $\hat{\mathbb{M}}_{Opt}$ nabývá fce PS_{σ^2} na celé množině M svého maxima, zřejmě platí $PS_{\sigma^2}(\hat{\mathbb{M}}_{Opt}) \geq PS_{\sigma^2}(\mathbb{N})$, tedy i

$$PS_{\sigma^2}(\mathbb{M}_{Opt}) - PS_{\sigma^2}(\hat{\mathbb{M}}_{Opt}) < \varepsilon.$$

Což jsme chtěli dokázat.

Nyní je na řadě numerický výpočet integrálu, který budeme v dalším nazývat **pilotním integrálem**.

$$PS_{\sigma^2}(\mathbb{M}) = \int_{\mathbb{R}^2} \Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X}) d\mathbb{X} \tag{4.1}$$

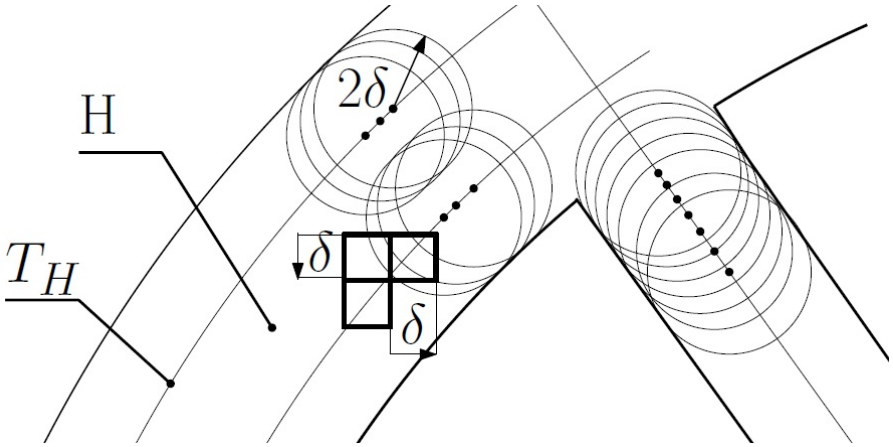
[1] Množinu M_δ nazveme δ- sítí množiny T, jestliže pro každé $\mathbb{X} \in T$ existuje $\mathbb{M} \in M_\delta$ takové, že $\rho(\mathbb{X}, \mathbb{M}) < \delta$. (Definice je udělána podle [2] strana 22 a $\rho(\mathbb{X}, \mathbb{M})$ značí euklidovskou metriku.)

S ohledem na fakt, že terčová funkce $J(\mathbb{X})$ je mimo oblast $O = \langle -R, R\rangle \times \langle -R, R\rangle \subset \mathbb{R}^2$, kde R je poloměr oblasti terče T, nulová, stačí provádět výpočet integrálu (4.1) pouze na oblasti O. Proveďme ekvidistatní dělení oblasti O do sítě čtverců o hraně $\delta = \frac{2R}{n}$, zřejmě pak n^2 je celkový počet čtverců daného dělení. Formuli pro numerickou integraci dostaneme nahrazením funkce $J \cdot \Phi_{\sigma^2}$ po částech interpolačním polynomem stupně nula, tedy po částech konstantní funkcí F. Více o této volbě v následující kapitole.

Vzhledem k nespojitosti funkce $J \cdot \Phi_{\sigma^2}$ na hranicích oblastí T_i musíme brát celkovou chybu numerické integrace jako $\triangle\epsilon = \sum_{i=1}^{82} \epsilon_{NI}^i + \epsilon_{NC}$, kde $\sum_{i=1}^{82} \epsilon_{NI}^i$ je součet chyb numerické integrace na jednotlivých oblastech T_i a ϵ_{NC} je chyba, které se dopustíme při numerickém integrování na všech elementárních čtvercích, které obsahují nějaký bod nespojitosti funkce J (bod množiny T_H). Sjednocení těchto čtverců označme Q.

Lemma: Pro libovolné $\varepsilon > 0$ jsme schopni volit δ tak, aby $\triangle\epsilon < \varepsilon$.

Důkaz: Zkonstruujme množinu $H = \{\mathbb{X} \in \mathbb{R}^2 | \parallel \mathbb{X} - \mathbb{Y} \parallel \geq 2\delta, \mathbb{Y} \in T_H\}$, viz obrázek (4.2).



Obrázek 4.2: Množina H

Je zřejmé, že $H \supset Q$, kde Q je sjednocení čtverců dělení, které obsahují nějaký bod nespojitosti. Chybu ϵ_{NC}, pak můžeme omezit takto:

$$\epsilon_{NC} \leq \int_Q \Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X}) d\mathbb{X} \leq \int_H \Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X}) d\mathbb{X} \leq$$

$$\leq \text{meas}H \cdot [min_{\mathbb{X}\in\mathbb{R}^2}\{\Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X})\} - max_{\mathbb{X}\in\mathbb{R}^2}\{\Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X})\}]$$

Výpočet samozřejmě probíhá pro bod $\mathbb{M}$ daný pevně. Oblast T_H je složena z dvaceti úseček a šesti kružnic. Dále nechť je Σ_{T_H} je součet délek všech těchto křivek, pak zřejmě $4\delta \cdot \text{meas}\sum_{T_H} \geq \text{meas}H$. Tedy

$$\epsilon_{NC} \leq \text{meas}H \cdot [\min_{\mathbb{X}\in\mathbb{R}^2}\{\Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X})\} - \max_{\mathbb{X}\in\mathbb{R}^2}\{\Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X})\}] \leq$$
$$\leq 4\delta \cdot \text{meas}\Sigma_{T_H} \cdot [\min_{\mathbb{X}\in\mathbb{R}^2}\{\Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X})\} - \max_{\mathbb{X}\in\mathbb{R}^2}\{\Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X})\}].$$

Pokud položíme
$\delta < \frac{\varepsilon}{8\text{meas}\sum_{T_H}\cdot[min_{\mathbb{X}\in\mathbb{R}^2}\{\Phi_{\sigma^2}(\mathbb{X}-\mathbb{M})\cdot J(\mathbb{X})\}-max_{\mathbb{X}\in\mathbb{R}^2}\{\Phi_{\sigma^2}(\mathbb{X}-\mathbb{M})\cdot J(\mathbb{X})\}]} = \delta_1$, zřejmě už $\epsilon_{NC} < \frac{\varepsilon}{2}$.

Dále najděme δ_2 tak, aby chyba $\sum_{i=1}^{82}\epsilon_{NI}^i$ byla menší než $\frac{\varepsilon}{2}$. Chybu ϵ_{NI}^i můžeme podle [7] strana 44 vyjádřit takto:

$$\epsilon_{NI}^i = \int_{T_i}(J\cdot\Phi_{\sigma^2} - F)d\mathbb{X}$$

Jen připomeňme, že F je nahrazení $J\cdot\Phi_{\sigma^2}$ po částech interpolačním polynomem stupně nula. Na každém z elementárních čtverců c_{ij} dělení oblasti O, kde $c_{ij}\subset T_i$ můžeme chybu integrace vyjádřit takto: $\epsilon_{NI}^{c_{ij}} = \int_{C_x^{ij}-\frac{\delta}{2}}^{C_x^{ij}+\frac{\delta}{2}}\int_{C_y^{ij}-\frac{\delta}{2}}^{C_y^{ij}+\frac{\delta}{2}}(J\cdot\Phi_{\sigma^2}-F)dydx$, kde $(C_x^{ij}, C_y^{ij}) = \mathbb{C}_{ij}$ je střed čtverce c_{ij}. Potom

$$\epsilon_{NI}^i = \int_{T_i}(J\cdot\Phi_{\sigma^2}-F)d\mathbb{X} = \sum_{\forall\, j}\int_{C_x^{ij}-\frac{\delta}{2}}^{C_x^{ij}+\frac{\delta}{2}}\int_{C_y^{ij}-\frac{\delta}{2}}^{C_y^{ij}+\frac{\delta}{2}}(J\cdot\Phi_{\sigma^2}-F)dydx$$

a

$$\sum_{i=1}^{82}\epsilon_{NI}^i = \sum_{i=1}^{82}\sum_{\forall\, j}\int_{C_x^{ij}-\frac{\delta}{2}}^{C_x^{ij}+\frac{\delta}{2}}\int_{C_y^{ij}-\frac{\delta}{2}}^{C_y^{ij}+\frac{\delta}{2}}(J\cdot\Phi_{\sigma^2}-F)dydx$$

Funkce $J\cdot\Phi_{\sigma^2}$ a F nabývají v bodě $\mathbb{C}_{ij}$ společné hodnoty. F je na elementárním čtverci c_{ij} konstantní a růst $J\cdot\Phi_{\sigma^2}$ je omezen hodnotou maximální parciální derivace funkce Φ_{σ^2}. Proto můžeme každou elementární chybu $\epsilon_{NI}^{c_{ij}}$ omezit

$$\epsilon_{NI}^{c_{ij}} = \int_{C_x^{ij}-\frac{\delta}{2}}^{C_x^{ij}+\frac{\delta}{2}}\int_{C_y^{ij}-\frac{\delta}{2}}^{C_y^{ij}+\frac{\delta}{2}}(J\cdot\Phi_{\sigma^2}-F)dydx \leq$$

$$\leq J_i\int_{C_x^{ij}-\frac{\delta}{2}}^{C_x^{ij}+\frac{\delta}{2}}\int_{C_y^{ij}-\frac{\delta}{2}}^{C_y^{ij}+\frac{\delta}{2}}\frac{\sqrt{2}}{2}\delta\max_{\mathbb{X}\in\mathbb{R}^2}\frac{\partial\Phi_{\sigma^2}(\mathbb{X})}{\partial\vec{v}}dxdy = J_i\frac{\sqrt{2}}{2}\delta^3\max_{\mathbb{X}\in\mathbb{R}^2}\frac{\partial\Phi_{\sigma^2}(\mathbb{X})}{\partial\vec{v}} \leq$$

$$\leq \max_{i=1,\ldots,82}J_i\frac{\sqrt{2}}{2}\delta^3\max_{\mathbb{X}\in\mathbb{R}^2}\frac{\partial\Phi_{\sigma^2}(\mathbb{X})}{\partial\vec{v}},$$

pak

$$\sum_{i=1}^{82}\epsilon_{NI}^i \leq \sum_{\forall c_{ij}}\max_{i=1,\ldots,82}J_i\frac{\sqrt{2}}{2}\delta^3\max_{\mathbb{X}\in\mathbb{R}^2}\frac{\partial\Phi_{\sigma^2}(\mathbb{X})}{\partial\vec{v}} \leq$$

$$\leq n^2\delta^2\max_{i=1,\ldots,82}J_i\frac{\sqrt{2}}{2}\delta\max_{\mathbb{X}\in\mathbb{R}^2}\frac{\partial\Phi_{\sigma^2}(\mathbb{X})}{\partial\vec{v}} \leq 4R^2\max_{i=1,\ldots,82}J_i\frac{\sqrt{2}}{2}\delta\max_{\mathbb{X}\in\mathbb{R}^2}\frac{\partial\Phi_{\sigma^2}(\mathbb{X})}{\partial\vec{v}}$$

(připomeňme, že R je poloměr oblasti O a n je počet dílků dělení).

Při volbě $\delta < \frac{\varepsilon}{8R^2\max_{i=1,\ldots,82}J_i\frac{\sqrt{2}}{2}\delta\max_{\mathbb{X}\in\mathbb{R}^2}\frac{\partial\Phi_{\sigma^2}(\mathbb{X})}{\partial\vec{v}}} = \delta_2$ je už $\sum_{i=1}^{82}\epsilon_{NI}^i < \frac{\varepsilon}{2}, \quad \forall\varepsilon > 0.$

Připojme tři poznámky. Princip omezení rozdílu funkčních hodnot $(J\cdot\Phi_{\sigma^2}-F)$ pomocí maximální hodnoty parciálních derivací je obdobný jako u důkazu předchozí věty. Využíváme spojitosti funkce $(J\cdot\Phi_{\sigma^2}-F)$ společně se všemi jejími derivacemi na jednotlivých čtverečcích c_{ij}.

Odhad chyby, který se při tomto způsobu numerické integrace většinou používá je uveden například v [5] na straně 351. Pro naše použití v této části důkazu je ale nepříliš vhodný, neboť by znamenal zavést řadu nových pojmů, zejména některé z variačního počtu.

Námi uvedený odhad je velmi pesimistický. Skutečná chyba bude výrazně menší. Ovšem vzhledem k tomu, že nám nejde o přesnou hodnotu δ_2, pouze tvrdíme, že takové δ_2 existuje, se spokojíme s odhadem uvedeným.

Dále už zřejmě pro volbu $\delta < min(\delta_1, \delta_2)$ dostaneme $\triangle\epsilon = \sum_{i=1}^{82} \epsilon_{NI}^{i} + \epsilon_{NC} < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$. Což jsme chtěli dokázat.

Nyní můžeme uvést algoritmus výpočtu průměrného skóre, který nazveme `PSnaive`.

```
function PSnaive = PSnaive( TERC )
%naivni algoritmus vypoctu prumerneho skore PS

n = size(TERC);
n = n(1);

%pro vsechny body terce
for i = 1:n
    for j = 1:n
        %pro vsechny miry nepresnosti
        for k = 1:n %k = sigma^2
            %sestrojeni odpovidajici fce s 2D normal rozdelenim
            %ktera je ovsem rotacni
            fun=@(x,y)(1/(2*pi*k))*exp(-0.5*(((x-i)^2+(y-j)^2)/k));

            %vypocet soucinu funkci nad jednotlivymi body
            for l = 1:n
                for m = 1:n
                    %vypocet
                    SUM(l,m) = TERC(l,m)*fun(l,m);
                end
            end

            PSnaive(i,j,k) = sum(sum(SUM));
        end
    end
end
```

Výpočetní náročnost tohoto naivního algoritmu je však neúměrně vysoká, jak ukazuje následující tabulka, kde n je počet dílků dělení a připomeňme že $\delta = \frac{2R}{n}$ tedy, $n = \frac{2R}{\delta}$.

PSnaive									
n	3	5	7	9	11	13	15	17	19
t [s]	0,005	0,035	0,161	0,479	1,219	2,715	5,432	9,988	17,281

Pro vyšší n už je tento algoritmus nepoužitelný. Připojme ještě v poznámce, že ani míru nepřesnosti σ^2 nemůžeme měnit v našem výpočtu spojitě. Pro jednoduchost jsme ji volili $\{\delta, 2\delta, 3\delta, ...R\}$, kde R značí poloměr množiny T.

Kapitola 5

Algoritmus Pernstejn

Aby byly výsledky algoritmu `PSnaive` použitelné, museli bychom volit n řádově desítky, lépe však stovky. Což je ale v přímém rozporu s délkou výpočetní doby pro takováto n. K problému hledání optimálního bodu terče je tedy nutno přistoupit jiným způsobem.

Nabízí se jistě možnost pilotní integrál (4.1) počítat po částech nad jednotlivými oblastmi, kde je funkce $\Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X})$ spojitá. Použít nějakou integrační formuli vysokého řádu a přesnosti a jednotlivé integrály pak prostě sečíst. Potíž nastane opět v tom, že funkce J a tedy samozřejmě ani oblasti T_i nejsou v kartézských souřadnicích explicitně zadatelné. Museli bychom přejít do souřadnic polárních. Tam by se sice všechny oblasti T_i transformovaly na obdélníky, na kterých už je numerická integrace některou z formulí elementární problém, potíž by však nastala s funkcí Φ_{σ^2}. Tu bychom samozřejmě museli také transformovat do polárních souřadnic a to obecně pro všechny vektory středních hodnot $\mathbb{M} \in M_\delta$. Tedy pro každý z těchto bodů bychom nejen počítali 82 různých dvourozměrných integrálů, ale v první řadě ještě náročnou transformaci funkce Φ_{σ^2} Ani tato cesta tedy zřejmě nevede k cíli.

Bylo tedy nutno vymyslet nový a efektivní přístup k danému problému. Postup, který byl nakonec zvolen, využívá některých příjemných vlastností Φ_{σ^2}. A to především rotační symetrie.[1]

Ukážeme si, že pilotní integrál přes dvourozměrnou oblast se dá transformovat na integrál jednoduchý. Podle Fubiniovy věty převedeme (4.1) na integrál dvojnásobný, provedeme posunutí a transformaci do polárních souřadnic.

$$PS_{\sigma^2}(\mathbb{M}) = \int_{\mathbb{R}^2} \Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) \cdot J(\mathbb{X}) d\mathbb{X} =$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Phi_{\sigma^2}(x - \mu_1, y - \mu_2) \cdot J(x, y) dx dy =$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Phi_{\sigma^2}(x, y) \cdot J(x + \mu_1, y + \mu_2) dx dy = \left| \begin{array}{ll} x = \rho\cos\varphi & J_c = \rho \\ y = \rho\sin\varphi & \end{array} \right| =$$

$$= \int_0^{\infty}\int_0^{2\pi} \rho\Phi_{\sigma^2}(\rho, \varphi) \cdot J(\rho\cos\varphi + \mu_1, \rho\sin\varphi + \mu_2) d\varphi d\rho$$

Protože Φ_{σ^2} je podle předpokladu rotačně symetrická, $\Phi_{\sigma^2}(\rho, \varphi) = \Phi_{\sigma^2}(\rho)$ nezávisí na φ. Můžeme tedy psát:

[1] Algoritmus PernstejnMOD, který je uveden dále se pak dá zobecnit i pro případy, že Φ_{σ^2} má jiný speciální tvar.

$$PS_{\sigma^2}(\mathbb{M}) = \int_0^\infty \rho\Phi_{\sigma^2}(\rho) \int_0^{2\pi} J(\rho\cos\varphi + \mu_1, \rho\sin\varphi + \mu_2)d\varphi d\rho =$$

$$= \int_0^\infty \rho\Phi_{\sigma^2}(\rho) \cdot I(\rho, \mu_1, \mu_2)d\rho,$$

$$I(\rho, \mu_1, \mu_1) = \int_0^{2\pi} J(\rho\cos\varphi + \mu_1, \rho\sin\varphi + \mu_2)d\varphi. \tag{5.1}$$

Funkce I nikterak nezávisí na tvaru Φ_{σ^2} tedy na míře nepřesnosti, její výpočet je tedy nutno provést jen jednou a pro všechny míru nepřesnosti pak už počítat pouze jednoduchý integrál

$$\int_0^\infty \rho\Phi_{\sigma^2}(\rho) \cdot I(\rho, \mu_1, \mu_2)d\rho. \tag{5.2}$$

.

Nyní se ještě zmiňme o tom, jak lze efektivně provést výpočet funkce $I(\rho, \mu_1, \mu_2)$. Počítat transformace funkce J do polárních souřadnic v každém bodě $\mathbb{M} = (\mu_1, \mu_2)$ a následně integrovat, není zrovna efektivní. Platí totiž následující tvrzení:

Lemma: Pro libovolnou kružnici Γ se středem $\mathbb{M} = (\mu_1, \mu_2)$ a poloměrem ρ v $\mathbb{R}^2$ platí:

$$\rho \int_0^{2\pi} J(\rho\cos\varphi + \mu_1, \rho\sin\varphi + \mu_2)d\varphi =$$

$$= \int_\Gamma J(x + \mu_1, y + \mu_2)d\gamma. \tag{5.3}$$

Důkaz: Mějme libovolnou kružnici Γ se středem $\mathbb{M} = (\mu_1, \mu_2)$ a poloměrem ρ. Pak můžeme vyjádřit integrální průměr funkce J nad touto kružnicí pomocí rovnice (5.1) takto:

$$IP_{pol} = \frac{1}{2\pi} \int_0^{2\pi} J(\rho\cos\varphi + \mu_1, \rho\sin\varphi + \mu_2)d\varphi.$$

Stejně můžeme vyjádřit integrální průměr funkce J nad kružnicí Γ v kartézských souřadnicích pomocí rovnice (5.3),

$$IP_{kart} = \frac{1}{2\pi\rho} \int_\Gamma J(x + \mu_1, y + \mu_2)d\gamma.$$

Je zřejmě, že integrální průměr funkce, nezáleží na souřadné soustavě, ve které je počítán. Tedy, že $IP_{pol} = IP_{kart}$ pro libovolnou kružnici Γ. Pak

$$\frac{1}{2\pi} \int_0^{2\pi} J(\rho\cos\varphi + \mu_1, \rho\sin\varphi + \mu_2)d\varphi = \frac{1}{2\pi\rho} \int_\Gamma J(x + \mu_1, y + \mu_2)d\gamma,$$

odkud už tvrzení okamžitě plyne.

Uveďme si, k čemu je toto tvrzení dobré. Výpočet integrálů (5.1) budeme provádět numericky. Protože funkci J lze velmi snadno vyjádřit ve velkém počtu uzlových bodů, zvolme pro integraci složenou obdélníkovou formuli. To pak znamená pro získání funkce I pouze sčítat hodnoty funkce J nad danou kružnicí. Ovšem pozor, protože funkci J máme

kartézské soustavě souřadnic, dostaneme sečtením všech uzlových nad kružnicí Γ hodnotu $\rho\int_0^{2\pi} J(\rho\cos\varphi+\mu_1,\rho sin\varphi+\mu_2)d\varphi$ nikoli pouze $\int_0^{2\pi} J(\rho\cos\varphi+\mu_1,\rho sin\varphi+\mu_2)d\varphi$, jak by se mohlo zdát logické. Tato úvaha plynoucí z předchozího tvrzení je tedy pro implementaci algoritmu `Pernstejn` velmi důležitá.

Pro potřeby výpočtu funkce I ještě uveďme toto tvrzení:

Lemma: Pro libovolnou kružnici Γ se středem $\mathbb{M}=(\mu_1,\mu_2)$ a poloměrem ρ v $\mathbb{R}^2$ platí

$$\frac{1}{4\pi\rho\Delta}\lim_{\Delta\to 0}\int_A Jd\mathbb{S}=\frac{1}{2\pi r}\int_\Gamma Jd\gamma,$$

kde $A=B(\mathbb{M},\rho+\Delta)-B(\mathbb{M},\rho-\Delta)$, přičemž $B(\mathbb{M},\rho)$ značí kouli v $\mathbb{R}^2$ tj. kruh se středem $\mathbb{M}=(\mu_1,\mu_2)$ a poloměrem ρ. Tedy křivkový integrál nad kružnicí Γ jsme schopni libovolně přesně nahradit integrálem plošným nad mezikružím $A=B(\mathbb{M},\rho+\Delta)-B(\mathbb{M},\rho-\Delta)$.

Důkaz: První integrál rozepišme podle Fubiniovy věty a použijeme integrální větu o střední hodnotě[2].

$$\frac{1}{2\pi\rho}\lim_{\Delta\to 0}\frac{1}{2\Delta}\int_A Jd\mathbb{X}=\frac{1}{2\pi\rho}\lim_{\Delta\to 0}\frac{1}{2\Delta}\int_{\rho-\Delta}^{\rho+\Delta}\int_\Gamma Jd\gamma d\psi=\frac{1}{2\pi\rho}\int_\Gamma Jd\gamma \quad \square$$

Všech poznatků uvedených v této kapitole využívá nový upravený algoritmus výpočtu optimálního bodu terče pojmenovaný `Pernstejn`[3]. Rozeberme si nyní ještě základní princip jeho fungování.

Funkce J vstupuje do algoritmu v maticovém tvaru, přičemž velikost matice odpovídá jemnosti dělení oblasti O z předchozí kapitoly. Matice obsahující terčovou funkci je v algoritmu označena jako `TERC`. Jak bylo předesláno, pro výpočet budeme používat složenou obdélníkovou formuli, přičemž nebudeme sčítat pouze uzlové body ležící na kružnici, ale podle předchozího tvrzení sečteme uzlové body nějakého úzkého mezikruhového pásu kolem této kružnice. Konečným počtem takovýchto pásů pokryjeme celou oblast O. Součet mezikruhového pásu se středem v bodě $\mathbb{M}=(\mu_1,\mu_2)$ a středním poloměr ρ označme $\hat{I}(\rho,\mu_1,\mu_2)$. Pak podle předchozích dvou tvrzení

$$\hat{I}(\rho,\mu_1,\mu_2)\doteq\rho I(\rho,\mu_1,\mu_1)=\rho\int_0^{2\pi} J(\rho\cos\varphi+\mu_1,\rho sin\varphi+\mu_2)d\varphi$$

a

$$PS_{\sigma^2}(\mathbb{M})=\int_0^\infty \rho\Phi_{\sigma^2}(\rho)\cdot I(\rho,\mu_1,\mu_2)d\rho\doteq\int_0^\infty \Phi_{\sigma^2}(\rho)\cdot\hat{I}(\rho,\mu_1,\mu_2)d\rho. \tag{5.4}$$

Funkce $\hat{I}=\hat{I}(\rho,\mu_1,\mu_2)$ je funkce tří proměnných, je tedy reprezentována třídimenzionální maticí[4].

Dalo by se to představit i tak, že jsme na M_δ množině uzlových bodů zkonstruovali topologii, tedy že jsme pro každý bod $\mathbb{M}$ definovali všechna jeho okolí. Pokud pak sečteme funkční hodnoty funkce J ve všech bodech jednotlivých okolí, dostaneme opět funkci $\hat{I}=\hat{I}(\rho,\mu_1,\mu_2)$ v diskrétním (maticovém) tvaru.

[2] Viz např. [3] strana 46.

[3] Jeho konkrétní implementaci lze nalézt v příloze Nezmíněné skripty.

[4] Aby nedošlo k nedorozumění, pojmem třídimenzionální matice není myšlena matice, jejíž vektory generují třídimenzionální prostor, ale matice $\hat{I}=\{\hat{i}_{\rho,\mu_1,\mu_2}\}$.

Máme již hodnoty funkce $\hat{I}$ a zbývá tedy dořešit úlohu:

$$PS_{\sigma^2}(\mathbb{M}) = \int_0^\infty \Phi_{\sigma^2}(\rho) \cdot \hat{I}(\rho, \mu_1, \mu_2) d\rho, \tag{5.5}$$

Opět se nabízí výpočet provést některou z integračních formulí vysokého řádu a přesnosti. Opět ale zvolíme jednoduchou složenou obdélníkovou formuli. Vysvětlení je nasnadě. Chování funkce $\hat{I} = \hat{I}(\rho, \mu_1, \mu_2)$ v proměnné ρ může být poměrně divoké. Skutečně, pokud zvolíme bod $\mathbb{M}$ ve středu terče, pak pro ρ menší než průměr středové oblasti terče je hodnota integrálních průměrů zřejmě 50, pak se skokově mění na 25, následně pak opět skokově nabývá hodnot $10{,}5; 31{,}5; 10{,}5; 21; 0$. To je zřejmé ze tvaru terče a jednotlivých jeho oblastí (viz přílohu Pravidla). Funkce $\hat{I}$ je tedy v proměnné ρ dokonce nespojitá, proto by bylo použití integračních formulí vysokého řádu značně riskantní. Dále pokud budeme mít hodnoty funkce Φ_{σ^2} zadány ve stejných uzlových bodech jako funkci $\hat{I}$, pak řešit integrál (5.5) znamená vlastně jen počítat skalární součin vektorů funkčních hodnot funkcí Φ_{σ^2} a $\hat{I}$.

Protože množina M_δ obsahuje n^2 bodů a počet měr nepřesnosti jsme volili také n, znamenala původní úloha hledání optimálního bodu střelby výpočet n^3 dvourozměrných pilotních integrálů (4.1). Provedenými úpravami jsme tuto úlohu transformovali na výpočet funkce $\hat{I}$ sčítáním prvků matice `TERC` odpovídající terčové funkci J a na výpočet n^3 skalárních součinů. Pro vysoká n, která ale ostatně ze samotného principu úlohy potřebujeme, jsou pak výsledky dané oběma principy téměř identické.

Doplňme ještě, jak vypadá transformace funkce Φ_{σ^2} do polárních souřadnic. (V předchozím výpočtu jsme samozřejmě brali funkční hodnoty této transformace.)

$$\Phi_{\sigma^2}(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \left| \begin{array}{ll} x = \rho\cos\varphi & J_c = \rho \\ y = \rho sin\varphi & \end{array} \right|$$

$$\Phi_{\sigma^2} = \frac{1}{2\pi\sigma^2} e^{-\frac{(\rho\cos\varphi)^2+(\rho sin\varphi)^2}{2\sigma^2}} = \frac{1}{2\pi\sigma^2} e^{-\frac{\rho^2}{2\sigma^2}} \tag{5.6}$$

Už nejde o normálního rozdělení, neboť předpis (5.6) vůbec nedává rozdělení pravděpodobnosti

$$\int_{-\infty}^{\infty} \frac{1}{2\pi\sigma^2} e^{-\frac{\rho^2}{2\sigma^2}} \neq 1,$$

pro zjednodušení si ale dovolme označení Φ_{σ^2} zachovat, abychom zdůraznili, že jde o transformaci dvourozměrného rotačně symetrického normálního rozdělení.

Následující tabulka ukazuje porovnání rychlostí výpočtů podle algoritmů `Pernstejn` a `PSnaive`.

Porovnání rychlostí algoritmů PSnaive a Pernstejn										
	n	3	5	7	9	11	13	15	17	19
PSnaive	t [s]	0,005	0,035	0,161	0,479	1,219	2,715	5,432	9,988	17,281
Pernstejn	t [s]	0,001	0,003	0,009	0,024	0,054	0,112	0,212	0,378	0,632

Algoritmus `Pernstejn` však ještě není nijak rychlostně optimalizovaný. Vhodným přeskládáním jednotlivých kusů kódu a využitím některých dalších poznatků byl sestrojen algoritmus `PernstejnMOD`, který je ještě několikanásobně rychlejší.

Nyní tedy zmíníme ještě dvě základní modifikace, jimiž jsme z algoritmu `Pernstejn` přešli na algoritmus `PernstejnMOD` liší, kterým jsme nakonec úlohu hledání optimálního bodu terče řešili.

Algoritmus `Pernstejn` počítá hodnotu pilotního integrálu ve všech bodech $\mathbb{M} \in M_\delta$. Platí ale následující tvrzení:

Lemma: Optimální bod střelby $\mathbb{M}_{Opt}$ leží v množině $T \cap M_\delta$. Tedy neleží mimo oblast terče.

Důkaz: Předpokládejme, že $\mathbb{M} \in \overline{T}$ ($\overline{T}$ značí doplněk množiny T) je optimální bod, tedy, že $\mathbb{M} \equiv \mathbb{M}_{Opt}$. Je ovšem zřejmé, že celá oblast T je konvexní množina v $\mathbb{R}^2$. Tedy, pro bod $\mathbb{M} \in \overline{T}$ existuje bod $\tilde{\mathbb{M}} \in T$ takový, že jeho vzdálenost od všech bodů množiny T je menší, než vzdálenost bodu $\mathbb{M}$. Pak $\forall\, \mathbb{X} \in T, \quad \Phi_{\sigma^2}(\mathbb{X} - \mathbb{M}) < \Phi_{\sigma^2}(\mathbb{X} - \tilde{\mathbb{M}})$ a tedy i $PS_{\sigma^2}(\mathbb{M}) < PS_{\sigma^2}(\tilde{\mathbb{M}})$. Docházíme tak ke sporu s předpokladem, že $\mathbb{M} \equiv \mathbb{M}_{Opt}$, tedy optimální bod opravdu nemůže ležet mimo T. Což jsme chtěli dokázat.

Konvexnost množiny je nutná podmínka. Uveďme si protipříklad. Mějme množinu $\hat{T} = T - \mathbb{M}_{Opt}$, ta zřejmě není konvexní a bod $\mathbb{M}_{Opt} \in \overline{\hat{T}}$.

V algoritmu `PernstejnMOD` jsme tedy počítali hodnoty pilotního integrálu jen v bodech $\mathbb{M} \in T \cap M_\delta$, čímž jsem uspořili přibližně $\frac{4-\pi}{4} n^3$ operací.

Podstatou druhé modifikace bylo zefektivnění vyhledávání prvků matice `TERC` na požadovaných mezikružích. Oproti algoritmu `Pernstejn`, kde jsme při hledání bodů daného mezikruží procházeli celou matici `TERC`, si v `PernstejnMOD` spočítáme tvary jednotlivých mezikruží předem a požadované mezikruží z matice `TERC` pouze vyřízneme a sečteme. Zmiňme aspoň tuto část zdrojového kódu.

```
%vyrizni mezikruzi
meziA = Ar.*TERC;

%suma
I(r+1) = sum(sum(meziA));
```

V předchozí kapitole je uvedeno, že výpočet pomocí algoritmu `PernstejnMOD` je možno zobecnit nejen pro rotačně symetrické normální rozdělení. Vskutku, pokud bychom proceduru `mezikruziA`, která algoritmu `PernstejnMOD` předpočítává jednotlivá mezikruží, nahradili procedurou generující například elipsy o zadaných parametrech, bude algoritmus `PernstejnMOD` stejně dobře použitelný i pro vektor náhodné střelby, jehož hustota je konstatní na daných elipsách. Tímto způsobem je možno zobecnit algoritmus `PernstejnMOD` pro velmi širokou třídu vektorů náhodné střelby. Odvození korektnosti výpočtu by se provedlo stejně jako pro rotačně symetrický náhodný vektor. Jen by bylo technicky výrazně komplikovanější.

Zbytek změn mezi algoritmy `Pernstejn` a `PernstejnMOD` spočívá v efektivnějším poskládání kusů kódu a lepším nakládání s proměnnými. Závěrem ještě uveďme srovnání rychlostí algoritmů `Pernstejn` a `PernstejnMOD`.

Porovnání rychlostí algoritmů Pernstejn a PernstejnMOD										
	n	45	47	49	51	53	55	57	59	61
Pern.	t [s]	38,97	48,38	59,41	71,53	86,68	104,50	124,77	148,23	173,30
PMOD	t [s]	10,44	13,01	15,44	18,32	21,48	25,34	30,62	35,44	41,37

Algoritmus `PernstejnMOD` je opět možno nalézt v příloze Nezmíněné skripty.

Kapitola 6

Zpracování výsledků

Pomocí algoritmu `PernstejnMOD` jsme již mohli provést výpočet s dostatečnou přesností v uspokojivém čase. Funkci J na oblasti $O = \langle -R, R\rangle \times \langle -R, R\rangle$ jsme diskretizovali pomocí matice 341×341, to je možno představit si tak, že jsme terč nadělili na čtverečky menší než jeden milimetr. Budeme tuto diskretizaci považovat za vhodný kompromis mezi přesností výsledků a výpočetní dobou. Jen ještě dodejme, že celý výpočet optimálních korekcí nepřesné střelby s touto jemností dělení trval na osobním počítači Samsung 305U padesát tři hodiny.

V této kapitole popíšeme, jak bylo postupováno, aby bylo možné získaná data interpretovat a jednoduchou vizuální formou přiblížit uživateli. Algoritmus `PernstejnMOD` (stejně jako `PSnaive` a `Pernstejn`) vrací výsledky ve formě třídimenzionální matice $\mathbb{A}$ (v m-skriptu byla označena jako `vysledky`)[1]. Pro lepší pochopení uvádíme matici $\mathbb{A}$ pro diskretizaci terče pomocí matice 5×5.

```
A(:,:,1) =

         0         0   63.6620         0         0
         0   19.0986   31.8310    6.3662         0
   35.0141   17.5070   79.5775    9.5493   19.0986
         0   25.4648    4.7746    3.1831         0
         0         0    9.5493         0         0

A(:,:,2) =

         0         0    3.1446         0         0
         0    3.3805    3.5152    2.6143         0
    2.1009    2.9658    4.4624    2.0844    0.9920
         0    2.6924    1.8932    1.5649         0
         0         0    0.7873         0         0

A(:,:,3) =

         0         0    2.0442         0         0
         0    2.8731    2.8528    2.3701         0
    1.5497    2.5807    3.0269    1.9575    0.7845
         0    2.1697    1.8258    1.5110         0
         0         0    0.7315         0         0

A(:,:,4) =

         0         0    1.6369         0         0
         0    2.3639    2.3517    1.9965         0
    1.3276    2.1785    2.3977    1.7439    0.7576
         0    1.8028    1.6297    1.3552         0
         0         0    0.7254         0         0

A(:,:,5) =

         0         0    1.4071         0         0
         0    1.9948    1.9948    1.7141         0
    1.1891    1.8726    2.0093    1.5569    0.7467
         0    1.5534    1.4585    1.2271         0
         0         0    0.7162         0         0
```

Obrázek 6.1: $\mathbb{A}$ pro matici terče 5×5

[1] Opět jen doplňme, že pojmem třídimenzionální matice není myšlena matice, jejíž vektory generují třídimenzionální prostor, ale matici $\mathbb{A} = \{a_{ijk}\}$.

Vrstvy matice $\mathbb{A}$ odpovídají různým mírám nepřesnosti střelby. V každé vrstvě pak byly v uzlových bodech dělení vypočteny hodnoty pilotního integrálu (4.1). Poznamenejme, že tyto hodnoty neodpovídají skutečnému průměrnému skóre získanému při míření na daný uzlový bod. Není to jenom v důsledku chyb výpočtu, ale především ve faktu, že jsme při výpočtu pilotních integrálů (4.1) neuvažovali reálnou velikost plochy terče. Toto zjednodušení budeme ale považovat za přijatelné s ohledem na to, že nás v první řadě zajímá pouze poloha optimálního bodu nikoli přesná hodnota pilotního integrálu v tomto bodě. Navíc, jak ukážeme dále, tuto hodnotu přece jen získáme a to poměrně elegatně a s malým úsilím.

V první části zpracování dat procházíme matici $\mathbb{A}$ a v každé vrstvě hledám optimální bod[2]. Hodnoty souřadnic optimálního bodu i, j uložíme do dvousloupcové matice `MatViz`, jejíž řádky odpovídají jednotlivým vrstvám třídimenzionální matice $\mathbb{A}$.

V dalším budeme interpretovat míru nepřesnosti σ^2. Jde samozřejmě o rozptyl jednotlivých souřadnic náhodného vektoru polohy zasažených bodů nepřesné střelby. Konkrétní představa je ovšem již poněkud složitější. Co například prakticky znamená $\sigma^2 = 34mm$? Velice názornou představu o míře nepřesnosti však dává poloměr kružnice, do níž padne 95% střel dané nepřesné střelby. Například vím-li, že poloměr, do nějž padne 95% střel, je stejný jako poloměr středové oblasti terče, a mířím-li na střed terče, pak středovou oblast terče zasáhnu s 95%ní pravděpodobností. Míru nepřenosti danou řádkem matice `MatViz` respektive vrstvou matice A tedy budeme interpretovat právě pomocí R poloměru kružnice, do níž padne 95% střel dané nepřesné střelby.

Ze sr pořadového čísla řádku matice `MatViz` je možno vyjádřit hodnotu σ^2. Odhad R lze pak získat pomocí Čebyševovy nerovnosti (viz například [8] strana 315). Takto bychom ale obdrželi pouze odhad R a navíc je tento postup poněkud komplikovaný, protože náhodný vektor polohy zasaženého místa terče je dvourozměrný.

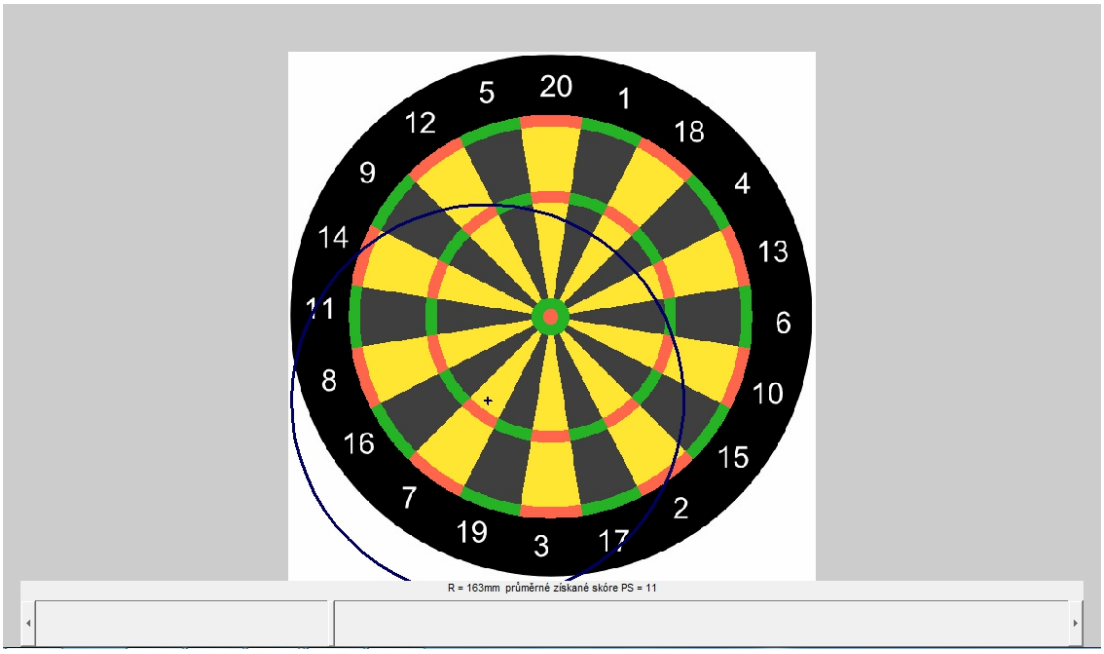
Zvolili jsme proto postup jiný. Jde vlastně opět o metodu Monte Carlo. Pro všechna sr jsme opakovali následující postup. Simulovali jsme 1 000 000 nepřesných střel s mírou nepřesnosti danou σ^2, kterou jsme určili z sr, přičemž jsme mířili na bod určený souřadnicemi i, j na sr-tém řádku v matici `MatViz`. Závislost R na σ^2 je lineární, bylo nutno jen empiricky najít takovou konstantu λ, aby $R = \lambda \cdot \sigma^2$. Tu jsme našli po několikátém opakování simulace bez jakýchkoli problémů.

Tento postup měl i další výhodu. Pouhým zprůměrováním zasažených hodnot při simulacích jsme obdrželi odhad průměrného skóre PS pro danou nepřesnost střelby, který je ale vzhledem k vysokému počtu střel (1 000 000) velmi dobrý.

Matici `MatViz` jsme dále rozšířili na čtyřsloupcovou, přičemž třetí sloupec obsahuje odhady PS a čtvrtý sloupec velikosti poloměrů R. Posledním krokem bylo z dat uložených v matici `MatViz` sestrojit vizualizační aplikaci, která je současně hlavním výstupem celé práce. Uživatel v této aplikaci navolí míru nepřesnosti, odezvou mu je zobrazení optimálního bodu střelby společně s kružnicí, do níž padne 95% střel a vypsání hodnoty poloměru R a průměrného skóre PS získaného při dané střelbě. Viz obrázek 6.2.

Ještě se krátce zmiňme o tom, jak může hráč jednoduše zjistit míru nepřesnosti své střelby. Stačí, aby hráč zamířil na střed a hodil na terč sto šipek. Pokud pak změří vzdálenost šesté nejvzdálenější šipky od středu terče, bude již tato hodnota poměrně dobrým odhadem poloměru R kružnice, do níž padne 95% střel. Tuto hodnotu pak hráč použije jako vstup do vizualizační aplikace.

[2]Mluvíme o optimálním bodu, což není přesné, neboť jde jen o optimální bod mezi body sítě, na níž provádíme výpočty (viz kapitolu Příprava numerického řešení), ale pro jednoduchost u této terminologie zůstaňme.



Obrázek 6.2: Výsledná uživatelská aplikace

Závěrem spíše pro zajímavost ještě uveďme jeden z velkých problémů vizualizace, který bylo nutno řešit. Šlo o to najít vhodné spektrum měr nepřesnosti σ^2 tak, aby byly pokryty všechny smysluplné míry nepřesnosti, na druhou stranu, abychom drahocenný výpočetní čas nemrhali na mírách nepřesnosti, které již nepřinesou žádnou novou skutečnost.

Toto spektrum bylo potřeba zkonstruovat adaptivní pro různé přesnosti výpočtu (velikosti diskretizační matice), aby jej bylo možné odladit na malé jemnosti dělení oblasti O a pak pouze aplikovat na jemnosti dělení velké. Provádět třiapadesátihodinový výpočet, abychom na jeho konci zjistili, že maximální míra nepřesnosti, kterou jsme ještě do výpočtu zahrnuli, je tak malá, že ji žádný reálný hráč není schopen dosáhnout, by bylo nemilé.

Ačkoli šlo o problém týkající se pouze závěrečné vizualizace, bylo nutné zasáhnout přímo do jádra algoritmu `PernstejnMOD` konkrétně do procedury `GAUSS2` a naškálovat zde některé z vystupujících proměnných. Tato poznámka je zde uvedena proto, aby bylo zřejmé, že jsme řešení hledali nejen z pohledu jeho matematické korektnosti, ale i jako dobře a rychle numericky spočitatelné a dále jako počítané ve vhodných bodech s ohledem na uživatelskou použitelnost.

Kapitola 7

Závěr

Přes všechny diskutované těžkosti, se mi nakonec podařilo požadovanou úlohu vyřešit s velmi příjemnou přesností. Podívejme se ještě na porovnání s konkurencí. Samozřejmě nejsem první, kdo danou úlohu hledání optimálního bodu terče řešil. Z přechozích prací o tomto tématu jsem ale do své práce nic nečerpal. Dokonce jsem ani podobné práce nehledal. Chtěl jsem úlohu řešit po svém s využitím algoritmu `Pernstejn`. Teprve po dokončení základní části práce vyhledal můj vedoucí RNDr. Pavel Popela Ph. D. některé práce, které se tématikou optimalizace házení šipek zaobíraly přede mnou. Zmiňme aspoň jednu z nich a srovnejme ji s prací, kterou jsem vytvořil já.

Pro porováníní jsem vybral, předpokládám, zatím nejnovější práci na toto téma vytvořenou týmem statistiků ze Standfortské univerzity, která byla publikována v Journal of the Royal Statistical Society [14]. V této práci je také kriticky zmíněno několik dalších autorů, kteří naši úlohu řešili.

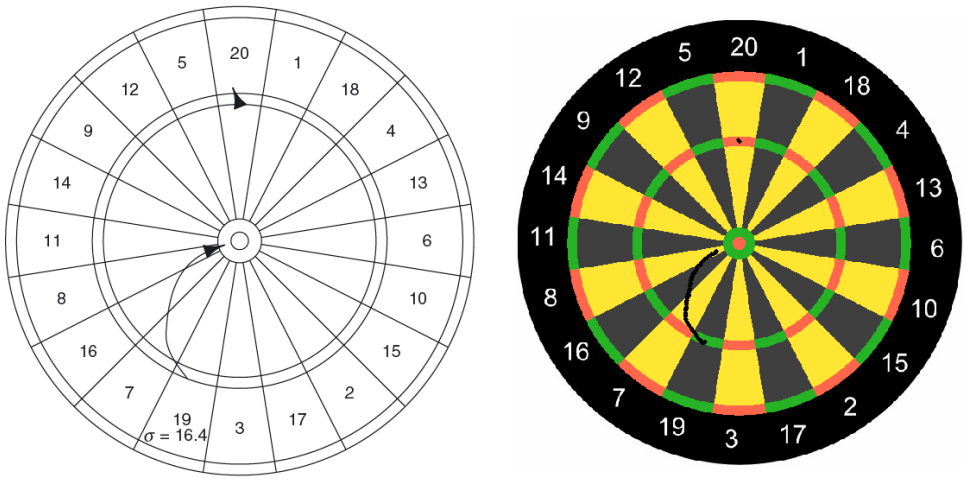
Podívejme se, ale na srovnání mé práce s [14]. Mé řešení využívá poznatků funkcionální analýzy a je speciálním případem úlohy o posunutí funkce (viz Kapitolu 4), dále je řešena otázka šíření vstupní nepřesnosti pomocí MMC a fyzikálního modelu. Rovněž jsou ověřeny předpoklady symetrické normality. V [14] je publikový postup řešení zaměren pouze na výstupní neurčitosti, k výpočtu ohodnocení bodů terče je použita Fast Fourier Transform, dále se autoři zaměřují na odhad parametrů výstupních rozdělení hráčů pomocí EM algoritmu a navazují zobecněním původního symetrického gaussovského modelu.

Rozdílnou bych pak viděl i cílovou skupinu, na kterou jsou výsledky práce orientovány. Zatímco v případě [14] je určena spíše pro odbornou veřejnost, výsledná aplikace mé práce je určena uživatelům bez jakéhokoli matematického vzdělání.

Ještě zbývá porovnání výsledků obou prací, to je ukázáno na obrázku 7.1. Levý obrázek je převzat z [14], pravý pak je zobrazení mých dat ve stejném grafickém formátu.

Uvedu ještě, proč je mezi citacemi uvedena kniha Základy teorie funkcí a funkcionální analýzy autorů Kolmogorova a Fomina [1], z které v práci vůbec necituji. Je to tak proto, že právě jen díky jejímu studiu jsem dokázal nalézt transformaci integrálu (4.1) na integrál (5.2). A dále bych rozhodně nebyl schopen sestavit uvedené důkazy, nevypracoval-li bych si důkazovou techniku právě na důkazech z této knihy.

A dovolil bych si zakončit citátem svého oblíbeného autora Friedricha Nietzscheho, který myslím vystihuje rozdíl mezi původní naivní představou o jednoduchosti úlohy a výsledným řešením se všemi složitostmi a obtížemi: “Když si člověk postaví dům, vždycky zpozoruje, že se přitom přiučil něčemu, co měl rozhodně vědět, než začal stavět.”



Obrázek 7.1: Porovnání výsledků prací

Literatura

[1] A. N. Kolmogorov, S. V. Fomin: *Základy teorie funkcí a funkcionální analýzy (překlad z ruštiny: Nauka, Moskva, 1972)*, SNTL, Praha 1975.

[2] Jan Franců: *Funkcionální analýza I*, CERM, učební text FSI VUT Brno, Brno 2009.

[3] Jan Franců: *Parciální diferenciální rovnice*, CERM, učební text FSI VUT Brno, Brno 2003.

[4] Alexander Ženíšek: *Samozřejmosti nebo paradoxy*, VUTIUM, Brno 2000.

[5] Philip J. Davis and Philip Rabinowitz: *Methods of Numerical Integration*, Academic Press, Inc., San Diego 1984.

[6] Libor Čermák, Rudolf Hlavička: *Numerické metody*, CERM, učební text FSI VUT Brno, 2008.

[7] Libor Čermák: *Vybrané statě z numerických metod*, [online], dostupné z `http://mathonline.fme.vutbr.cz/download.aspx?id_file=1035`. učební text.

[8] Alfréd Rényi: *Teorie pravděpodobnosti*, Academia, Praha 1972

[9] Zdeněk Karpíšek: *Matematika IV, Statistika a pravděpodobnost*, CERM, učební text FSI VUT Brno, 2007

[10] Jiří Tesař, Petr Bartoš: *Metoda Monte Carlo a programovací jazyk MATLAB při přípravě učitelů na pedagogických fakultách*, Konference Technical Computing Prague 2006 [online], [cit. 5. 5. 2012]. Dostupné na: `http://dsp.vscht.cz/konference_matlab/MATLAB06/prispevky/ tesar_bartos/tesar_bartos.pdf`.

[11] *Bartlett's test*, Wikipedia [online], [cit. 5. 5. 2012]. Dostupné na: `http://en.wikipedia.org/wiki/Bartlett%27s_test`

[12] *Trajectory of a projectile*, Wikipedia [online], [cit. 5. 5. 2012]. Dostupné na: `http://en.wikipedia.org/wiki/Trajectory_of_a_projectile`

[13] Tomáš Bárta: *Úlohy na integrační faktor*, učební text [online], [cit. 5. 5. 2012]. Dostupné na: `http://www.karlin.mff.cuni.cz/~barta/pcODR/Kapitola-IntegracniFaktor/ IntegracniFaktor-1250.pdf`

[14] Ryan J. Tibshirani, Andrew Price and Jonathan Taylor: *A statistician plays darts*, J. R. Statist. Soc. A (2011) 174, Part 1, pp. 213-226

Příloha A

Pravidla

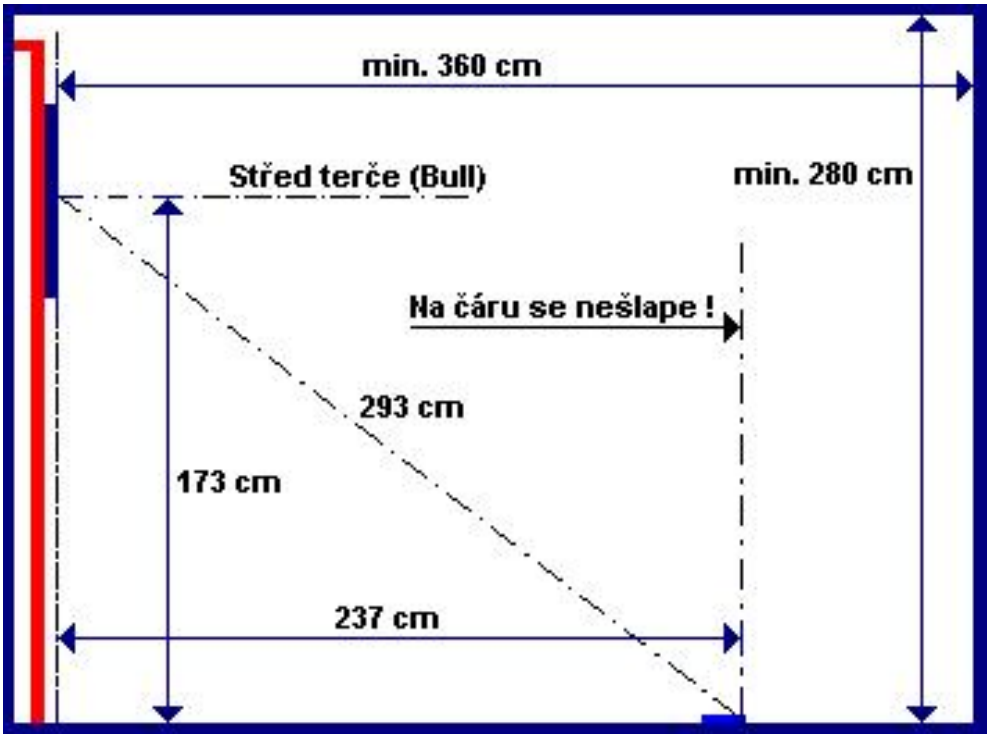
Pro seznámení se základními pravidly šipkové hry, použijeme výňatek ze všeobecných pravidel Unie šipkových organizací, viz: `www.sipky.org/download.php?file_id=8546`.

B.2.1. šipka se skládá z vlastního těla šipky, které je z jedné strany opatřeno umělohmotným hrotem a z druhé strany násadkou, která může být složena z více částí - zadní závaží, násadka, upevnění letky, letka, koncovka apod.
B.2.2. Předepsané míry a váhy: minimální délka = $60mm$, maximální hmotnost = $20g$. Další míry, materiál ani tvar nejsou limitovány.[1]
B.4.1. Čára hodu je vodorovná přímka vedená po podlaze hracího prostoru ve vzdálenosti $2,37m$ od rysky umístěné na boku přístroje, což je kolmice spuštěná od přední hrany segmentů terče. Střed terče (Bull) je tak zároveň ve výšce $173cm$ po kolmici od podkladu, na kterém stojí šipkový přístroj.
B.4.2. Odhodová čára musí být pevně vyznačena ve vzdálenosti $2,37m$ a nejméně v šíři 25 cm od osy, která je vyznačena středem dolní hrany šipkového přístroje. Odhodová čára je vyznačena vždy tak, že se na této nestojí.
B.5.3. Při hodu musí hráč stát v daném hracím prostoru a nesmí překročit linii čáry hodu. Pomyslné linie vyznačující hrací prostor nesmí přesahovat žádná část těla hráče.

Pro lepší přehled uveďme rozměry terče a hracího prostoru podle:
`http://www.sipky.cz/method/method_size.html`

- Terč musí mít dvacet rozměrově shodných výsečí po čtyřech segmentech a dvojitý střed
- Bodové hodnoty výsečí musí být od 1 do 20 bodů, vnějšího segmentu středu 25 bodů a vnitřního segmentu středu 50 bodů
- Pořadí bodových hodnot výsečí musí být následující: výseč mající hodnotu 20 bodů se nachází v horní polovině terče a je kolmá k vodorovné ose terče, dále musí po směru hodinových ručiček následovat tyto hodnoty výsečí:
- $20, 1, 18, 4, 13, 6, 10, 15, 2, 17, 3, 19, 7, 16, 8, 11, 14, 9, 12, 5$
- První a třetí segment výseče, počítáno od středu, musí mít bodovou hodnotu jednonásobku bodové hodnoty výseče. Druhý segment výseče, počítáno od středu, musí

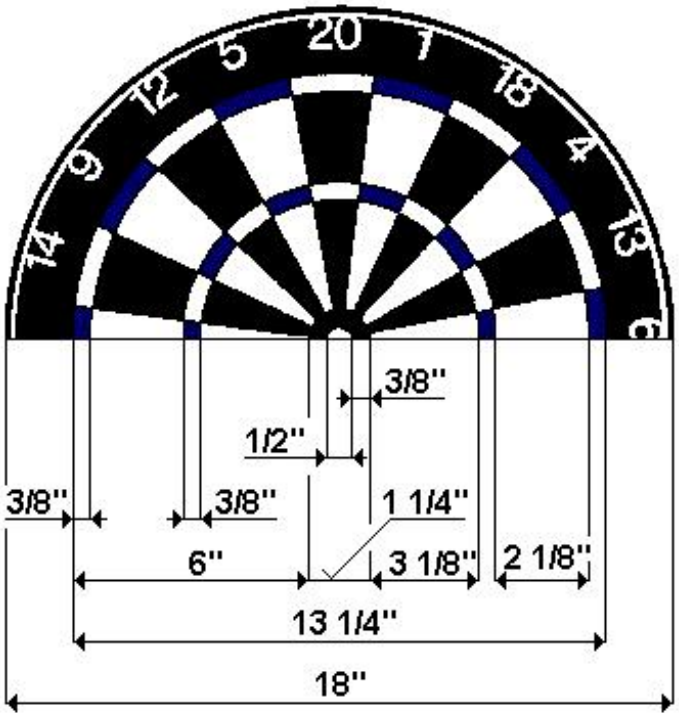
[1] Předpis pro maximální hmotnost = $20g$ neplatí pro stealové šipky, pozn. citujícího.



Obrázek A.1: Hrací prostor

mít bodovou hodnotu trojnásobku bodové hodnoty výseče. Čtvrtý segment výseče, počítáno od středu, musí mít bodovou hodnotu dvojnásobku bodové hodnoty výseče

- Veškeré rozměry jednotlivých výsečí musí být shodné (...)
- Průměr terče včetně mezikruží musí být $340mm \pm 2mm$
- Vnější rozměr středu musí být $34mm$. Vnitřní segment středu musí mít po obvodu 8 otvorů a jeden ve středu, celkem tedy 9 otvorů a jeho průměr musí být $12mm$
- Segmenty trojnásobků bodových hodnot musí mít dvě řady po 10i otvorech
- Segmenty dvojnásobků bodových hodnot musí mít dvě řady po 16i otvorech
- Vnitřní šířka mezikruží, v němž jsou uloženy segmenty dvoj a trojnásobků musí být maximálně $11mm$



Obrázek A.2: Plocha terče

Příloha B

Nezmíněné skripty

Aby byl v m-skriptech zmíněných v této příloze nějaký pořádek, uvedeme si hned zpočátku jejich strukturu a okomentujeme je. V první řadě jsou skripty rozděleny na dvě základní části. Skripty týkající se metody Monte Carlo a druhé kapitoly a skripty použité přímo k výpočtu optimálních korekcí nepřesné střelby. V těchto dvou sekcích je pak vždy uvedena nejprve struktura uspořádání jednotlivých kódů s krátkým komentářem. Zdrojové kódy, které fungují jen jako součást jiného zdrojového kódu, jsou uvedeny v podsekci daného nadřazeného kódu. Po tomto počátečním rozčlenění následuje odpovídající seřazený výčet zdrojových kódů.

Monte Carlo

1. `MMCRunner` - řídí celou simulaci
   (a) `vypocetuhlu` - provádí výpočet úhlu α podle rovnice (3.5)
   (b) `vykresleni` - provede vykreslení všech v úvahu připadajících balistik
   (c) `MatAlfa` - sestaví z jednotlivých nastavení přehlednou tabulku použitou v textu
      i. `MakeDec` - pomocná procedura `MatAlfa`, převadí číslo zadané v libolné soustavě do soustavy desítkové
2. `MonteCarlo` - samotná simulace
3. `TESTHyp` - testuje hypotezu o dvourozměrném rotačně symetrickém normálním rozdělení
   (a) `ChiKvadrat` - testuje hypotézu o normalitě statistického souboru

MMCRunner

```
%kmen simulace
%ridi cely proces
%rozdeluje ulohy
clear; clc;

%pocet nastaveni
sets = 3;

%Konstantni parametry
Sx = 2.37;  %vzdalenost terce
g = 9.81;  %gravitacni zrychleni

%promenne parametry

    %rychlosti
    v0 = [30,25,20];

    %vahy sipek
    m = [0.030 0.025 0.020];

    %drag of dart (soucinitel odporu vzduchu)
    Cd = [0.15 0.20 0.25];

    %vyska
    Sy = [0 0.2 -0.2];

    %vypocet uhlu alfa pro kazde nastaveni
    alfa = zeros(1, sets*sets);

%permutace ruznych nastaveni
nastaveni = zeros(sets^4,4);

j = 0;
%pres vsechna nastaveni
for i1 = 1:sets
    for i2 = 1:sets
        for i3 = 1:sets
            for i4 = 1:sets
                j = j + 1;
                alfa(j) = vypocetuhlu(Sx, g, v0(i1), m(i2), Cd(i3), Sy(i4));
                %ulozeni nastaveni
                nastaveni(j,1) = v0(i1);
                nastaveni(j,2) = m(i2);
                nastaveni(j,3) = Cd(i3);
                nastaveni(j,4) = Sy(i4);
                %vykresleni jednotlivych balistik
```

```
                vykresleni(alfa(j),Sx, g, v0(i1), m(i2), Cd(i3), Sy(i4));
            end
        end
    end
end
%doplnim nastaveni o vypoctena alfa
nastaveni = [nastaveni transpose(alfa)];

%prehledne usporadane do tabulky pro pisemnou cast prace
MatA = MatAlfa(nastaveni);

%odstranim ta nastaveni ktera netrefuji terc
    %nuluju pozice techto nastaveni
    for i = 1:sets^4
        nastaveni(i,1) = isfinite(alfa(i)) * nastaveni(i,1);
    end

    %pocet nenulovych nastaveni
    nenul = sum(isfinite(alfa));

    %smazu znulovana nastaveni
    nastaveni = sortrows(nastaveni);
    nastaveni = nastaveni((sets^4-nenul + 1):end,:);

--------------------------------------------------------------------------
    %% samotna simulace Monte Carlo

%parametry simulace
numberOF = 200000; %pocet hodu
nu = 0.01; %mira nepresnosti
SizeMat = 10; %velikost sberne matice SberMat

%predefinovani potrebnych promennych
hypoteza = zeros(1,nenul*3);
SberMat = zeros(2*SizeMat,2*SizeMat,nenul*3);

%poradove cislo nastaveni
j = 0;

%pro vsechny konstanty pomeru rozptylu uhlu a rychlosti
for kappa = [0.2 1 5]
    for i = 1:nenul %pres vsechna pripustna nastaveni
        j = j + 1;

        %Samotna simulace
        %vraci uz rovnou roztridena data
        SberMat(:,:,j) = MonteCarlo(Sx, kappa, g,...
            numberOF,nu, SizeMat, nastaveni(i,:));
```

```
        %uvazuji dale jen sipky ktere zasahly SberMat
        numberOF = sum(sum(SberMat(:,:,j)))*numberOF;

        %test hypotezy 0 - zamitneme / 1 - nezamitneme
        hypoteza(j) = TESTHyp(SberMat(:,:,j),numberOF);

    end
end
```

vypocetuhlu

```
function alfa = vypocetuhlu(Sx, g, v0, m, Cd, Sy)
%VYPOCETUHLU pro dane strelecke parametry spocita
%uhel hodu ktery se bude vyuzivat k dalsim vypoctum

%predpocitane konstanty
K1 = ((m^2)*g)/(Cd^2);
K2 = (Cd*Sx)/(m*v0);
K3 = (m*g*Sx)/(Cd*v0);

%funkce fun = 0
fun = @(alfa)K1*log(1-K2/cos(alfa)) + Sx*tan(alfa)...
    + K3/cos(alfa) - Sy;

%Hledani korenu nelinearni rovnice
    %startovaci uhel
    alfa1 = 0.1;

    %zajistim realnost a konecnost fce ve start bode
    while isreal(fun(alfa1)) == 0 ||...
            isfinite(fun(alfa1)) == 0
        alfa1 = alfa1 - 0.001;
    end

    %vypocet uhlu
    alfa = fzero(fun,alfa1);

end
```

vykresleni

```
function vykresleni(alfa,Sx, g, v, m, Cd, Sy)
%vykresli balistickou krivku o danych parametrech

%cas
t = 0:0.01:0.5;

%pohyb v ose x
X = ((cos(alfa)*m*v)/Cd)*(1-exp((-Cd*t)/m));

%pohyb v ose y
Y = (-m/Cd)*(v*sin(alfa)+ (m*g)/Cd)*...
    (exp((-Cd*t)/m)-1) - t*((m*g)/Cd);

%trajektorie
plot(X,Y); hold on;

%vykresli stred terce
%zavisi na velikosti Sy
plot(Sx,Sy,'--rs');

end
```

MatAlfa

```
function MatA = MatAlfa( nastaveni )
%MATALFA vytvori tabulku uhlu alfa pro ruzne nastaveni

%parametry
velikost = size(nastaveni);

%tabulka
sets = sqrt(sqrt(velikost(1)));
MatA = zeros(sets*sets,sets*sets);
j = 1;
for i1 = 1:sets
    for i2 = 1:sets
        for i3 = 1:sets
            for i4 = 1:sets
                %prepocitani souradnic fiktivnich i1,...,i4
                %na realne souradnice pozic v tabulce
                radek = MakeDec([i1-1,i2-1],sets) + 1;
                sloupec = MakeDec([i3-1,i4-1],sets) + 1;

                %zapis uhlu do prislusne pozice
                MatA(radek, sloupec) = nastaveni(j,5);
                j = j + 1;
            end
        end
    end
end

%hlavicka
%radky
radky = zeros(sets^2,2);
j = 1;
for i = 1:sets^2
    radky(i,:) = nastaveni(1+(i-1)*(sets^2),1:2);
end

%sloupce
sloupce = transpose(nastaveni(1:(sets^2),3:4));

%slozeni vysledne tabulky
MatA = [zeros(2,2) sloupce;radky MatA];

end
```

MakeDec

```
function cislo = MakeDec( A, sets )
%MakeDec prevadi libovolne cislo
%v libovolnem cifernem zapisu na zapis dekadicky
%cifry se zadavaji zvlast jako vektor
%druhy vstupni parametr je "desitka" prevadene soustavy

%delka prevadeneho cisla
cifer = length(A);


%predpokladam spravne zadani uzivatele
null = 1;

%pres cele prevadene cislo
for i = 1:cifer
    %zadana cifra neni v zadane soustave
    %je prilis velke
    if A(i) >= sets
        %pripravim vypis cislo = Inf
        null = 0;

    %cifra lze prevest
    else
        A(i) = A(i)*(sets^(cifer - i));
    end
end

%pokud jdou prevest vsechny cifry/jinak vraci Inf
cislo = sum(A)/null;

end
```

MonteCarlo

```
function SberMat = MonteCarlo(Sx, kappa, g, numberOF, nu, SizeMat, nastaveni)
%pro zadane veskere parametry strelby vcetne poctu strel
%a konstanty nepresnoti vytvori sbernou matici podle pozadovane presnosti
SberMat = zeros(2*SizeMat);

%rozdeleni pozice
v0 = nastaveni(1);
m = nastaveni(2);
Cd = nastaveni(3);
Sy = nastaveni(4);
alfa = nastaveni(5);

%vzdalenost na pixel sberne matice
%pricemz orezeme jednotliva rozdeleni podle 3 sigma
    %nastaveni pri nemz dojde k nejvetsi odchylce
    %od zamysleneho bodu pri dodrzeni 3 sigma
    MaxAlfaX = alfa - 3*nu;
    MaxAlfaZ = 3*nu;
    MaxV0 = v0-3*kappa*nu;

    %prepocet do pomocneho sour systemu
    MaxtildealfaZ = atan(tan(MaxAlfaZ)*cos(alfa));
    MaxtildeSx = Sx/cos(MaxAlfaZ);
    MaxtildealfaX = atan(tan(MaxAlfaX)*cos(MaxtildealfaZ));

    %souradnice v rovine terce
    MaxTx = Sx*tan(MaxtildealfaZ);

        %predpocitane konstanty
        K1 = ((m^2)*g)/(Cd^2);
        K2 = (Cd*MaxtildeSx)/(m*MaxV0);
        K3 = (m*g*MaxtildeSx)/(Cd*MaxV0);

    MaxTy = K1*log(1-K2/cos(MaxtildealfaX))+MaxtildeSx...
        *tan(MaxtildealfaX)+K3/cos(MaxtildealfaX)-Sy;

    %maximalni vzdalenost
    MaxVzd = max(MaxTx,MaxTy);

    %vysledna vzdalenost na pixel
    VzdPix = MaxVzd/SizeMat;

%nahodne variujeme tri parametry a opakujeme numberOFkrat
for i = 1:numberOF
    %vypocet nahodnych cisel
    hatalfaX = alfa+nu*randn;
```

```
    hatalfaZ = nu*randn;
    hatv0 = v0+kappa*nu*randn;

    %v pomocnem souradnem systemu
    tildealfaZ = atan(tan(hatalfaZ)*cos(alfa));
    tildeSx = Sx/cos(tildealfaZ);
    tildealfaX = atan(tan(hatalfaX)*cos(tildealfaZ));

    %vypocet souradnic v rovine terce
    Tx = Sx*tan(tildealfaZ);

        %predpocitane konstanty
        K1 = ((m^2)*g)/(Cd^2);
        K2 = (Cd*tildeSx)/(m*hatv0);
        K3 = (m*g*tildeSx)/(Cd*hatv0);

    Ty = K1*log(1-K2/cos(tildealfaX))+ tildeSx*...
        tan(tildealfaX)+K3/cos(tildealfaX)-Sy;

    %zapsani bodu do sberne matice SberMat
    Tx = Tx/VzdPix;
    Tx = round(Tx+SizeMat);
    Ty = Ty/VzdPix;
    Ty = round(Ty+SizeMat);

    %aspon jedna souradnice je komplexni
    if isreal(Tx) + isreal(Ty) < 2
        %hod vynecham
        Tx = 0;
        Ty = 0;
    end

    %pojistka pred hody ktere nepadnou do SberMat
    if Tx <= 2*SizeMat && Tx >= 1 &&...
            Ty <= 2*SizeMat && Ty >= 1
        %pri zasahu prictu sipku ovsem jako zlomek 1/numberOF
        %abych dostal hustotu pravdepodobnosti
        SberMat(Tx,Ty) = SberMat(Tx,Ty) + 1/numberOF;
    end
end

end
```

TESTHyp

```
function hypoteza = TESTHyp( SberMat, numberOF )
%TESTHYP testuje hypotezu zda zaslany vyber je
%z normalniho rotacniho rozdeleni

%pozadovana velikost
n = size(SberMat);
n1 = n(1);
PocRez = round(n1/2);

%rez(Crezu,s)
rez = zeros(PocRez,n1);

%kostrukce vsech uhlu rezu
ALFA = linspace(0,pi,PocRez);

%Najit stred rotace rezu - vyberovy prumer
%oznacme ho E
    E = zeros(1,2);

    %marginalni fce p1(x)
    p(1,:) = sum(SberMat');

    %marginalni fce p2(x)
    p(2,:) = sum(SberMat);

    %stredni hodnota E(X)
    for J = 1:2 %pro obe souradnice
        E(J) = 0;
        for i = 1:n(J)
            E(J) = E(J) + i*p(J,i);
        end
    end

%pres vsechny rezy
Crezu = 0;
for alfa = ALFA
    Crezu = Crezu +1;

    %Podle Radonovy transformace
    for i = 1:n1
        for j = 1:n(2)
            %poloha \s v primce rezu
            x = i-E(1);
            y = j-E(2);
            s = y*cos(alfa)+x*sin(alfa);
```

```
            %orezu \s na delku n1/2
            if abs(s)+1 >= n1/2
                %nedelam nic
            else
                %scitam hodnoty rezu
                s = round(s+n1/2);
                rez(Crezu,s) = rez(Crezu,s)+SberMat(i,j);
            end
        end
    end

end

%test normality
    hypoteza1 = zeros(1,PocRez);

    %vyberovy rozptyl rezu
    Drez = zeros(1,PocRez);
    for i = 1:PocRez
        [hypoteza1(i),Drez(i)] = Chikvadrat(rez(i,:));
    end

%test o shodnosti rozptylu
    hypoteza2 = zeros(1,PocRez);

    %prumerny vyberovy rozptyl S20
    S20 = sum(Drez)/PocRez;

    %pocet stupnu volnosti
    k = numberOF - 1;

    %pres vsechny rezy
    for i = 1:PocRez
        %testove kriterium
        t = (numberOF*Drez(i))/S20;

        %interval spolehlivosti
        Wa = chi2inv(0.025,k);
        Wb = chi2inv(0.975,k);

        %test
        if t > Wa && t < Wb
            hypoteza2(i) = 1;
        else
            hypoteza2(i) = 0;
        end
    end
```

```
%vysledna hypoteza
hypoteza = [hypoteza1; hypoteza2];
[S1 S2] = size(hypoteza);

%hypoteza o rotacnim rozdeleni byla prokazana
%tedy v kazdem rezu byla prokazana normalita
%a dale shodnost rozptylu
if sum(sum(hypoteza)) == S1*S2
    hypoteza = 1;
else
    hypoteza = 0;
end

end
```

ChiKvadrat

```
function [hypoteza, k, D] = ChiKvadrat( StSoubor )
%CHIKVADRAT testuje statisticky soubor
%zda se jedna o dvourozmerne nahodne rozdeleni

n = length(StSoubor);
fsvlnkou = zeros(1,n);

%vyberovy prumer znacme E
E = 0;
for i = 1:n
    E = E + i*StSoubor(i);
end

%vyberovy rozptyl znacme D
Ex2 = 0;
for i = 1:n
    Ex2 = Ex2 + i*i*StSoubor(i);
end
D = (Ex2 - E^2); %vychyleny, ale jen velmi malo
%200000/199999 = 1.0000;

%sestrojeni odpovidajici fce s normalnim rozdelenim
    %odhadnute parametry
    sigma = sqrt(D);
    mu = E;

%funkce
fun = @(x) 1/(sigma*sqrt(2*pi))*exp(-(x-mu)^2/(2*D));

%hodnoty normalniho rozdeleni s odhadnutymi parametry
for x = 1:n
    fsvlnkou(x) = fun(x);
end

%Chi-kvadrat test
q = 2; %odhadnute parametry (E,D)
k = n - q - 1; %pocet stupnu volnosti

%interval spolehlivosti
Wb = chi2inv(0.95,k);
Wa = 0;

%testovaci kriterium
DiferMat = ((StSoubor - fsvlnkou).^2)./fsvlnkou;
t = sum(sum(DiferMat));
```

```
%vyvracim, nebo nevyvracim hypotezu
if t < Wb && t > Wa
    hypoteza = 1; %true
else
    hypoteza = 0; %false
end

end
```

Optimální korekce nepřesné střelby

1. `PSnaive` - zmíněn v textu
   (a) `tercovnice` - vytvoří matice `TERC`, která je diskretizací funkce J, o požadované velikosti, její implementaci byl jistě bylo možné provést rafinovaněji, použitý způsob, je však jednoduchý a bezproblémově fungující, proto ponecháváme tuto proceduru v původním tvaru
2. `Pernstejn`
3. `PernstejnMOD`
   (a) `QAUSS2` - procedura použitá v `Pernstejn` a `PernstejnMOD` pro generování hodnot funkce dané předpisem (5.6)
   (b) `mezikruziA` - předpočítá jednotlivá mezikruží na nichž poté `PernstejnMOD` počítá jednotlivé integrální průměry funkce J
   (c) `orezA` - doplňuje fungování `mezikruziA`, mezikruží ořeže podle matice `TERC`
4. `TimeMeasure` - porovnává rychlosti algoritmů `PSnaive`, `Pernstejn` a `PernstejnMOD`
5. `CompLeader` - řídí hlavní výpočet optimální korekcí nepřesné střelby, vrací nezpracovaná data
   (a) `Obrazek` - vytvoří podkladový obrázek pro finální simulaci
   (b) `tercovnice2` - úplně stejná procedura jako `tercovnice` jen hodnoty návratové matice neurčují bodové ohodnocení dané oblasti, ale její barvu, z důvodu zbytečnosti ji tedy neuvádíme
6. `mezivys` - provede první krok zpracování vypočtených dat, vrací dvousloupcovou matici `MatViz`
7. `TESTmezivys` - MMC testuje správnost vypočtených výsledků, doplňuje matici `MatViz` a sloupec obsahující průměrné skóre PS a o sloupec poloměrů R
8. `vizualizaceFIN` - provede závěrečnou vizualizaci, v závěru je z ní vytvořen soubor `.exe`

tercovnice

```
function Tercovnice = tercovnice( n )
%TERCOVNICE vytvori tercovnici pozadovanych rozmeru
%samotny terc
Tercovnice = zeros(2*n+1);

%rozdeleni do trojuhelnicku
%probiram matici postupne po jednotlivych bodech.
%spocitam na jake tangente vzhledem ke stredu lezi
%a podle toho jim priradim cislo v terci
for i = n : -1 : -n
   for j = -n : n
        if j ~= 0
            %spocitam tangenc bodu vzhledem ke stredu terce
            tangenc = i/j;
            if tangenc < tan(pi/20) &&...
               tangenc >= -tan(pi/20) && j > 0
                Tercovnice(i+n+1,j+n+1) = 6; %sestka
            elseif tangenc < tan(pi/10+pi/20) &&...
                    tangenc >= tan(pi/20) && j > 0
                Tercovnice(i+n+1,j+n+1) = 10; %desitka
            elseif tangenc < tan(2*pi/10+pi/20) &&...
                    tangenc >= tan(pi/10+pi/20) && j > 0
                Tercovnice(i+n+1,j+n+1) = 15; %patnactka
            elseif tangenc < tan(3*pi/10+pi/20) &&...
                    tangenc >= tan(2*pi/1+pi/20) && j > 0
                Tercovnice(i+n+1,j+n+1) = 2; %dvojka
            elseif tangenc < tan(4*pi/10+pi/20) &&...
                    tangenc >= tan(3*pi/10+pi/20) && j > 0
                Tercovnice(i+n+1,j+n+1) = 17; %sedmnactka
            elseif tangenc < tan(6*pi/10+pi/20) &&...
                    tangenc >= tan(5*pi/10+pi/20) && j < 0
                Tercovnice(i+n+1,j+n+1) = 19; %devatenactka
            elseif tangenc < tan(7*pi/10+pi/20) &&...
                    tangenc >= tan(6*pi/10+pi/20) && j < 0
                Tercovnice(i+n+1,j+n+1) = 7; %sedmicka
            elseif tangenc < tan(8*pi/10+pi/20) &&...
                    tangenc >= tan(7*pi/10+pi/20) && j < 0
                Tercovnice(i+n+1,j+n+1) = 16; %sestnactka
            elseif tangenc < tan(9*pi/10+pi/20) &&...
                    tangenc >= tan(8*pi/10+pi/20) && j < 0
                Tercovnice(i+n+1,j+n+1) = 8; %osmicka
            elseif tangenc < tan(10*pi/10+pi/20) &&...
                    tangenc >= tan(9*pi/10+pi/20) && j < 0
                Tercovnice(i+n+1,j+n+1) = 11; %jedenactka
            elseif tangenc < tan(11*pi/10+pi/20) &&...
                    tangenc >= tan(10*pi/10+pi/20) && j < 0
```

```
                Tercovnice(i+n+1,j+n+1) = 14; %ctrnactka
            elseif tangenc < tan(12*pi/10+pi/20) &&...
                    tangenc >= tan(11*pi/10+pi/20) && j < 0
                Tercovnice(i+n+1,j+n+1) = 9; %devitka
            elseif tangenc < tan(13*pi/10+pi/20) &&...
                    tangenc >= tan(12*pi/10+pi/20) && j < 0
                Tercovnice(i+n+1,j+n+1) = 12; %dvanactka
            elseif tangenc < tan(14*pi/10+pi/20) &&...
                    tangenc >= tan(13*pi/10+pi/20) && j < 0
                Tercovnice(i+n+1,j+n+1) = 5; %petka
            elseif tangenc < tan(16*pi/10+pi/20) &&...
                    tangenc >= tan(15*pi/10+pi/20) && j > 0
                Tercovnice(i+n+1,j+n+1) = 1; %jednicka
            elseif tangenc < tan(17*pi/10+pi/20) &&...
                    tangenc >= tan(16*pi/10+pi/20) && j > 0
                Tercovnice(i+n+1,j+n+1) = 18; %osmnactka
            elseif tangenc < tan(18*pi/10+pi/20) &&...
                    tangenc >= tan(17*pi/10+pi/20) && j > 0
                Tercovnice(i+n+1,j+n+1) = 4; %ctyrka
            elseif tangenc < tan(19*pi/10+pi/20) &&...
                    tangenc >= tan(18*pi/10+pi/20) && j > 0
                Tercovnice(i+n+1,j+n+1) = 13; %trinactka
            elseif i < 0
                Tercovnice(i+n+1,j+n+1) = 20; %dvacitka
            else
                Tercovnice(i+n+1,j+n+1) = 3; %trojka
            end
        end
    end
end

%doplneni nuloveho sloupce, kde tangenc neni definovan

%dvacitka
for i = 1 : n
    Tercovnice(i,n+1) = 20;
end

%trojka
for i = 1 : n
    Tercovnice(i+n+1,n+1) = 3;
end

%postupne vykrajim a dodam kruhy terce
%po jednotlivych pixelech
%jednotlive polomery
      r1 = (1-0.375/6.625)*n; %polomer doublu
      r21 = (1-2.5/6.625)*n; %horni polomer triplu
```

```
      r22 = (1-2.875/6.625)*n; %dolni polomer triplu
      r3 = (0.625/6.625)*n; %horni polomer stredu
      r4 = (0.25/6.625)*n; %dolni polomer stredu

%cyklus
for i = n : -1 : -n
   for j = -n : n
      %spocitam polomer
      R = sqrt(j*j+i*i);

      %pro jednotlive polomery...
            if R > n+0.1 %orez kraju
                Tercovnice(i+n+1,j+n+1) = 0;

            elseif R >= r1 %double
                %zdvojnasobim
                Tercovnice(i+n+1,j+n+1) =...
                    Tercovnice(i+n+1,j+n+1)*2;

            elseif R <= r21 && R > r22 %triple
                %ztrojnasobim
                Tercovnice(i+n+1,j+n+1) =...
                    Tercovnice(i+n+1,j+n+1)*3;

            elseif R <= r4 %uplny stred
                %padesatka
                Tercovnice(i+n+1,j+n+1) = 50;

            elseif R <= r3 %neuplny stred
                %petadvacitka
                Tercovnice(i+n+1,j+n+1) = 25;
            end

   end

end

end
```

Pernstejn

```
function PS = Pernstejn( TERC )
%PERNSTEJN algoritmus vypoctu jiz vyrazne sofistikovanejsi
%nez puvodni PSnaive vyuziva vysledku funcionalni analyzy
%ale jeste bez numerickych modifikaci

%vytvorim prazdnou matici potrebne velikosti
n = size(TERC); n = n(1); I = zeros(n,n,n); %trojDIM matice

%prochazim matici I bod po bodu a kazdemu bodu
%priradim soucet bodu z mezikruzi
for r = 0 : round((n*1.4)) %pres vsechny polomery
    for x = 1 : n
        for y = 1 : n
            %hledam body, ktere patri do mezikruzicka
            %o danych parametru a scitam
            Mezikruzi = 0;
            for i = 1 : n
                for j = 1 : n
                    %polomer od momentalniho stred (x,y)
                    R = round(sqrt((x-i)^2+(y-j)^2));
                    if R == r %jsme na uvazovanem mezikruzicku
                        %pricitame hodnotu dle tercovnice
                        Mezikruzi = Mezikruzi + TERC(i,j);
                    end
                end
            end
            I(r+1,x,y) = Mezikruzi;
        end
    end
end

%vypocet GM
delka = size(I);
delka = delka(1);
GM = GAUSS2(n,delka);

%pres vsechny body
for i = 1:n
    for j = 1:n
        %pro vsechny presnosti
        for k = 1:n
            PS(i,j,k) = GM(k,:)*I(:,i,j);
        end
    end
end
end
```

PernstejnMOD

```
function PS = PernstejnMOD( TERC )
%PERNSTEJN modifikovany algoritmus Pernstejn ktery
%krom jiného vyuziva predpocitani mezikruzicek

%vytvorim prazdnou matici potrebne velikosti
n = size(TERC);
n = n(1);
PS = zeros(n,n,n); %trojDIM matice vysledku

%vypocty co si mohu provest napred

    %vypocet GM
    GM = GAUSS2(n,n+1);

    %vytvor matici mezikruzi
    A = mezikruziA(n);
    C = ones(n); %pomocna matice

%prochazim predpokladanou matici PS bod po bodu
%pro jednotlive body pocitam hodnoty funkce I
%a primo je skalarne nasobim s vektory funkcnich hodnot norm rozdeleni
for x = 1 : n
    for y = 1 : n
        if TERC(x,y) ~= 0
            for r = 0 : n %pres vsechny polomery

                %sectu cele mezikruzi!

                    %orez matici mezikruzi
                    AOREZ = orezA(A,n,2*n,x,y);

                    %vyber mezikruzi
                    Ar = AOREZ == r*C;

                    %vyrizni mezikruzi
                    meziA = Ar.*TERC;

                    %suma mezikruzi
                    I(r+1) = sum(sum(meziA));

               %prechod na dalsi polomer
            end



            %pro vsechny nepresnosti
```

```
            for k = 1:n
                PS(x,y,k) = GM(k,:)*I(:);
            end
        end
    end
end

end
```

GAUSS2

```
function GM = GAUSS2( n, delka )
%GAUSS2 dava gaussovu matici GM kde v kazdem radku
%je vektor funkcnich hodnot hustoty
%dvourozmerneho normalniho rozdeleni se stredni
%hodnotou 0 a ruznymi rozptyly pri jedne souradnici nulove

GM = zeros(n,delka);

%vyplnuji matici
for i = 0:(n-1)
    %sigma^2
    %naskalovana tak aby vypoctene vysledky
    %byly vhodne pro zaverecnou vizualizace
    sigma2 = 0.1 + i*(10/(n-1));
    for j = 0:delka-1
        %x je skalovano tak aby odpovidalo realne velikosti terce
        %ne jemnosti diskretizace pri numerickem reseni
        x = (15*j)/delka;
        %vzorec pro normalni rozdeleni
        GM(i+1,j+1) = (1/(sigma2*2*pi))*exp(-(x^2)/(2*sigma2));
    end

end

end
```

mezikruziA

```
function A = mezikruziA( n )
%MEZIKRUZIA vytvori matici mezikruzi A
%ta bude mi velikost 4n+1, aby byla
%dvojnasob vetsi nez TERC a mohla byt
%na potrebny rozmer orezana

N = 4*n + 1;
A = zeros(N);

%pro kazdy bod matice
for i = 1:N
    for j = 1:N
        %spocitam vzdalenost od stredu [2n+1,2n+1]
        %zaokrouhlim a ulozim do pozice
        A(i,j) = round(sqrt((i-(2*n+1))^2+ (j-(2*n+1))^2));
    end
end

end
```

orezA

```
function AOREZ = orezA( A,n,m,i,j)
%OREZA oreze matici A podle TERCE

%TERC bude nekde presahovat matici
%A tedy musim poskladat
%obecne z deviti submatic

    %zjistuji pro kazdou stranu zda matice A presahuje
    %a urcuji hodnoty indexu presahu
    if i-(m+1) > 0
        indexI1 = i-(m+1);
    else
        indexI1 = 0;
    end
    if n-(m+1) > 0
        indexI2 = n-(m+i);
    else
        indexI2 = 0;
    end
    if j-(m+1) > 0
        indexJ1 = j-(m+1);
    else
        indexJ1 = 0;
    end
    if n-(m+j) > 0
        indexJ2 = n-(m+j);
    else
        indexJ2 = 0;
    end

    %Sestavim matici AOREZ po castech
    %zakladni nenulova matice A0
    A0i = n-(indexI1+indexI2);
    A0j = n-(indexJ1+indexJ2);
    A0 = A(((m+2)-(i-indexI1)):((m+n+1)-(i+indexI2)),...
        ((m+2)-(j-indexJ1)):((m+n+1)-(j+indexJ2)));

    %nulove doplnkove matice
    A1 = zeros(indexI1,indexJ1);
    A2 = zeros(indexI1,A0j);
    A3 = zeros(indexI1,indexJ2);
    A4 = zeros(A0i,indexJ1);
    A5 = zeros(A0i,indexJ2);
    A6 = zeros(indexI2,indexJ1);
    A7 = zeros(indexI2,A0j);
    A8 = zeros(indexI2,indexJ2);
```

```
    %celkova matice
    AOREZ = [A1 A2 A3;A4 A0 A5;A6 A7 A8];

end
```

TimeMeasure

```
%Zmerim rychlost vypoctu optimalniho bodu
%pro algoritmy PSnaive, Pernstejn a PernstejnMOD

%maximalni velikost
N = 30;

%matice casu:
%velikost & PSnaive & Pernstejn & PernstejnMOD
MT = zeros(N,4);

%pres vsechny pokusy
for n = 1:N
    %vytvoreni terce
    TERC = tercovnice(n);

    %prvni sloupec MT jemnost deleni n
    MT(n,1) = 2*n+1;

    %PSnaive
    %jen do desitky, pak uz je to zbytecne
    if n < 10
        startTime = tic;%spustim cas
        PSnaive(TERC);
        MT(n,2) = toc(startTime); %zastavim cas
    end

    %Pernstejn
    startTime = tic;%spustim cas
    Pernstejn(TERC);
    MT(n,3) = toc(startTime); %zastavim cas

    %PernstejnMOD
    startTime = tic;%spustim cas
    PernstejnMOD(TERC);
    MT(n,4) = toc(startTime); %zastavim cas
end
```

CompLeader

```
%Procedura ridici hlavni cast vypoctu
clear; clc;

n = 170; %po milimetru

%sestrojim terc pozadovanych rozmeru
terc = tercovnice(n);

%provedu vypocet
vysledky = PernstejnMOD(terc);

%pripravim si obrazek terce
%pro pozdejsi vizualizaci
terc = Obrazek;

%ulozim vysledky i obrazek terce
save('vysledky','vysledky','terc');
```

Obrazek

```
function terc = Obrazek
%Procedura slouzici jen pro zkvalitneni vysledneho grafickeho vystupu
%z matice TERC vytvori obrazek a doplni ji rameckem s ciselnymi hodnotami

%sestrojim si terc
n = 231;
N = 2*n+1;
%matice rozvrzeni barev
terc = tercovnice2(n);

%prazdna matice obrazku
%vrstvy: R & G & B
A = zeros(N,N,3);

%prebarvim potrebne pixely
for i = 1:N
    for j = 1:N
        if terc(i,j) == 0 %cerna
        elseif terc(i,j) == 1 %zluta
            A(i,j,:) = [1 0.9 0.2];
        elseif terc(i,j) < 4 && terc(i,j) > 1  %cervena
            A(i,j,:) = [1 0.4 0.3];
        elseif terc(i,j) == 4 %cerna
            A(i,j,:) = [0.25 0.25 0.25];
        else  %zelena
            A(i,j,:) = [0.15 0.7 0.15];
        end
    end
end

%doplnim terc rameckem

%nahraju obrazek terce
ramecek = imread('Terc1.jpg');
rozmer = size(ramecek);

%smazu stred
for i = 1:rozmer(1)
    for j = 1:rozmer(2)
        %velikost polomeru na nemz jsem
        R = (i-303)^2 + (j-302)^2;
        if p < 235^2 %jsem ve stredu
            %mazu stred
            ramecek(i,j,:) = [0 0 0];
        end
    end
```

```
end

%sectu terc a ramecek
ramecek = (1/256)*double(ramecek);
ramecek(73:535,70:532,:) = ramecek(73:535,70:532,:)+A;

%vratim vysledny upraveny obrazek
terc = ramecek;

end
```

mezivys

```
%Prvni krok zpracovani velkeho mnozstvi dat
%obdrzeneho z procedury CompLeader
clc; clear;

%nahraju vypoctene vysledky
load('vysledky');
vysledky;
terc;

velikost = size(vysledky);
velikost = velikost(3);

%dvousloupcova matice vizualizace
MatViz = zeros(velikost,2);

%hledam optimalni body strelby
%tj. maximalni prvky jednotlivych vrstev matice vysledky
for k = 1:velikost
    [t i] = max(vysledky(:,:,k));
    [t j] = max(t);

    %prepocet na interval (0,1)
    %aby nezaviselo na jemnosti diskretizace
    %vztahuju k velikosti TERCe
    MatViz(k,1) = i(j)/velikost;
    MatViz(k,2) = j/velikost;

end

%ulozim mezivysledky
save('mezivysledky','MatViz','terc');
```

TESTmezivys

```
%%testovani vysledku metodou Monte Carlo
%k empirickemu urceni konstanty kappa
%plus doplneni sloupce R a PS do matice MatViz
clc; clear;

%nahraju pripravene mezivysledky
load('mezivysledky');
[delka a] = size(MatViz);
terc2 = terc;

%sestrojim opet terc vyplneny bodovymi hodnotami
%na ktery budu hazet
terc = tercovnice((delka-1)/2);

%rozsah simulace
n = 1000000;

%pro vsechny radky matice MatViz
%tedy pro kazdou miru nepresnosti
for sr = 1:delka
    %z poradoveho cisla radku vyjadrim sigma^2
    sigma2 = (0.1 + sr*(10/(delka)))/15;

    %konstanta kappa odvozena empiricky
    kappa = 2.499;
    %polomer r \in (0,1) pro vysledne zpracovani
    r(sr) = 2.449*sigma2;
    %polomer R95% v pixelech pro test spravne velikosti
    R = r(sr)*delka;
    %prepocet sigma^2 na pixely
    sigma2 = sigma2*delka;

    %stredni hodnoty
    EX = MatViz(sr,2)*delka;
    EY = MatViz(sr,1)*delka;

    %MMC
    %zasazenych bodu
    k = 0;

    %prumerne nahazene skore
    ps = 0;

    %simulace ve zvolenem rozsahu
    for j = 1:n
        %generuji nahodnou strelu
```

```
        X = randn*sigma2;
        Y = randn*sigma2;

        %test ze trefim
        if X < R && X > -R %spravne souradnice X
            if Y < sqrt(R^2-X^2) && Y > -sqrt(R^2-X^2)
                %padne do kruhu
                k = k + 1; %pridam zasazeny bod
            end
        end

        %pricitam prumerneho skore
        Y = round(Y + EY);
        X = round(X + EX);
        if Y > 0 && Y < delka+1 && X > 0 && X < delka+1
            %nejsem mimo tercovou matici
            %prictu hodnotu zasazeneho bodu terce
            ps = ps + terc(Y,X);
        end

    end

    %prumerne skore
    PS(sr) = ps/n;

    %procento zasazenych bodu
    K(s) = k/n;
end

%procenta zasazenych bodu
%K, %K(i) pozaduji rovno 0.95 pro vsechna i
%odsud empiricky urcena konstanta kappa

%Rozsireni matice MatViz
MatViz = [MatViz PS' r'];

%ulozeni doplnenych vysledku
save('mezivysledkyMOD','MatViz','terc2');
```

vizualizaceFIN

```
%%Vizualizuje pripravena data podle MatViz
clc; clear;

%pripravne vypocty
%nahraji mezivysledkyMOD s rozsirenou MatViz
%a obrazkem terce pro podklad vizualizace
load('mezivysledkyMOD');
%MatViz;
%terc2;

%velikost vysledku
[velikost a] = size(MatViz);

%pripravim si graficke prostredi
OKNO = figure;

%podkladovy obrazek
img = image(terc2);

%naformatuji graficke prostredi
prostor = get(img,'parent');
set(prostor,'Position',[0.10 0.10 0.8 0.8])
set(prostor,'NextPlot','add')
axis off
axis equal

%velikost terce v podkladovem obrazku
%propacitavam pres ni souradnice z int (0,1)
velikost2 = 463;

%Priprava vykresleni kruhu R95% forall sigma^2
for k = 1:velikost
    %prepocet R na pixely podkladoveho obrazku
    R = velikost2*MatViz(k,4);

    %vykresleni kruznice
    t = -pi:0.1:pi+0.1;
    T(k,1) = plot(R*cos(t)+70*ones(1,length(t))+...
        velikost2*MatViz(k,2),R*sin(t)+...
        72*ones(1,length(velikost))+velikost2*MatViz(k,1),...
        'Color',[0 0 0.4],'LineWidth',2.5,'Visible','off');
    %zneviditelnim

    %vykresleni stredu
    T(k,2) = plot(70*ones(1,length(velikost))+...
        velikost2*MatViz(k,2),72*ones(1,length(velikost))...
        +velikost2*MatViz(k,1),'+',...
```

```
        'Color',[0 0 0.4],'LineWidth',2,'Visible','off');
    %zneviditelnim

    %prepocet R na milimetry *2R(=6.625")*25.4(mm/")
    T(k,4) = round(MatViz(k,4)*2*6.625*25.4);

    %%PS
    T(k,3) = round(MatViz(k,3));
end


%text pro startovni polohu posuvniku (tj. pri min mire nepresnosti)
poloha = 1;
textik = ['R = ',num2str(round(T(poloha,4))),'mm',...
    '  průměrné získané skóre PS = ',num2str(T(poloha,3))];
text = uicontrol('Unit','Normalized','Position',[0.1 0 0.8 0.1],...
    'Style','Text','String',textik);

%zviditelni prvni miru nepresnosti
set(T(1,1),'Visible','on');
set(T(1,2),'Visible','on');

%cast kodu ktera pro kazde posunuti posuvniku
%zobrazi R95% vybrane miry nepresnosti

    %prikaz pro uvozovky v textovem retezci
    c = char(39);

    %nahradu retezce Visible
    a = 'Visible';

    %kostrukce posuvniku
    %ve funkci Callback je napsan kod ktery vyhodnoti pozici posuvniku
    %zneviditelni puvodni R95% a zobrazi zamysleny polomer
    posuvnik = uicontrol('Unit','Normalized','Position',...
        [0.1 0 0.8 0.07],'Style','Slider','Min',1,'Max',...
        velikost,'Value',1,'SliderStep',...
        [1/velikost 2/velikost],'Callback',...
        ['set(T(poloha,1),' c a c ',' c 'off' c...
        '); set(T(poloha,2),' c a c,',' c 'off' c...
        ');poloha = round(get(posuvnik,',c 'Value' c...
        ')); set(T(poloha,1),' c a c ',' c 'on' c,...
        '); set(T(poloha,2),' c a c ',' c 'on' c...
        '); textik = [' c 'R = ' c,',num2str(T(poloha,4)),'...
        c 'mm' c ',' c '  průměrné získané skóre PS = ',...
        c ',num2str(T(poloha,3))]; set(text,' c 'String'...
        c ',textik), ']);
```