UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Řešení základních geodetických výpočtů

Jan Skalický

Bakalářská práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Skalický**
Osobní číslo:	**I09256**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Řešení základních geodetických výpočtů**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce bude vytvoření programu umožňujícího řešení vybraných úloh z geodézie.
Při určování polohy nově zaměřovaného bodu se vychází z polohy stávajících bodů.
Tyto body jsou ale dány nepřesně a také navazující měření vzdáleností a úhlů je zatíženo chybou měření.
Pomocí metody nejmenších čtverců se hledá odhad souřadnic nového bodu.
Při řešení je třeba nejprve najít přibližné počáteční řešení. Další postup spočívá v linearizaci obecně nelineárních podmínek vzniklých ze schématu měření pomocí Taylorova rozvoje.
Při softwarovém zpracování je třeba použít symbolické výpočty resp. numerické metody.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

***Wimmer G.: Spracovanie a vyhodnocovanie meraní. VEDA. Bratislava, 2002. 183s.**
80-224-0734-8
***Kubáčková, L.: Metódy spracovania experimentálnych údajov. VEDA. Bratislava, 1979, 316 s**

Vedoucí bakalářské práce: **Mgr. Jaroslav Marek, Ph.D.**
Katedra matematiky a fyziky

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

Univerzita Pardubice
Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 10. května 2013 Jan Skalický

Poděkování

Chtěl bych poděkovat vedoucímu své práce, Mgr. Jaroslavu Markovi, Ph.D., za velmi cenné rady a skvělou spolupráci při zpracování této bakalářské práce.

Anotace

Tato bakalářská práce se zabývá řešením základních geodetických výpočtů. Jsou zde popsány historie geodézie, základní pojmy a matematický aparát vyrovnávací počet. Práce se dále zabývá řešením dvou typických úloh a tvorbou programu pro jejich řešení. Nechybí přiblížení vývojového prostředí NetBeans, programovacího jazyka Java a popis aplikace pro řešení základních geodetických výpočtů.

Klíčová slova

geodézie, vyrovnávací počet, Java

Title

Solving basic geodetic calculations.

Annotation

This thesis is about solving of basic geodetic calculations. There are describe geodetic history, basic terms and mathematical apparatus compensatory number. The work deal with the two typical tasks and program development for their solutions. There is description of development environment NetBeans, programming language Java and description of application for solving basic geodetic calculations.

Keywords

geodetic, compensatory number, Java

Obsah

Seznam zkratek

JTSK	Jednotný trigonometrický systém katastrální
grad	úhlová jednotka (1 grad = π/200 rad nebo 1 grad = 1/400 kruhu)
stadium	řecká délková míra (přibližně 600 stop), délka dráhy stadionu
JVM	Java Virtual Machine
SSZ	Souřadnicový Systém Zařízení
USS	Uživatelský Souřadnicový Systém
PNG	Portable Network Graphics
GUI	Graphical User Interface
IDE	Integrated Development Enviroment
GPS	Global Positioning System

Seznam obrázků

Úvod

Pod názvem „řešení geodetických výpočtů" si každý nepředstaví konkrétní typ úlohy, který by byl schopen vypracovat. Tato práce má čtenáři poukázat na fakt, co všechno se skrývá za výpočtem nové souřadnice, kterou se geodeti snaží vypočítat ze svých měření. Práce by měla působit jako pohled do odvětví geodézie, nastínit a popsat základní principy a na programu názorně předvést, jak celý výpočet funguje. U čtenáře se předpokládají základní znalosti ze statistiky a matematiky.

Cílem práce bude vytvoření funkční aplikace, která uživateli umožní zadat vstupní údaje ve formě souřadnic. Dále umožní vložit naměřené údaje a následně poskytne samotný náčrt řešené úlohy, pro názornější představu. Na základě vstupních dat zobrazí grafický výstup s vypočtenou (hledanou) souřadnicí.

Práce by měla poskytovat intuitivní ovládání a přehledný grafický výstup. Uživatel by neměl mít možnost jakkoliv poškodit svou činností běh celé aplikace. Samotná aplikace bude vytvořena ve vývojovém prostředí NetBeans 7.3 a napsána v programovacím jazyce Java.

V první části bude představena stručná historie geodézie a matematický aparát vyrovnávací počet, který se používá k výpočtu hledané souřadnice. V další části budou popsány dvě úlohy z oblasti geodézie. Jedna pro měření vzdáleností ze třech bodů pro určení hledaného bodu. Druhá pro měření ze dvou bodů a měření úhlu, který tento dva body svírají vzhledem k hledanému bodu. Nakonec bude představena samotná aplikace z hlediska programátorské a také uživatelské části.

1 Geodézie

1.1 Definice a základní pojmy

Pojem „*geodézie*" je řeckého původu a vznikl složením slov „*Země*" a „*dělit*". Česky lze geodézii označit jako zeměměřičství.

Zdroj: (3)

Učební texty pro Vysoké učení technické v Brně vysvětlují pojem „geodézie" následovně: „*Geodézie je široký vědní a technický obor zabývající se především zaměřováním bodů, objektů a terénů na povrchu Země i mimo něj, jejich zobrazováním do plánů a map a vytyčováním projektovaných staveb v terénu*."[1]

Zdroj: (2)

Ke zpracování výsledků geodetických měření se používá matematický aparát „*vyrovnávací počet*". Je to vlastně skupina metod, která se snaží o vyloučení hrubých chyb měření a zvýšení přesnosti cílového výsledku měření. Cílem je většinou určení souřadnic nového bodu ve zvolené soustavě souřadnic (např. JTSK). Tyto souřadnice jsou však určeny pomocí zprostředkujících měření vzdáleností a úhlů pomocí triangulace (podrobněji viz kapitola 1.2).

Zdroj: (1)

Při určování vzdáleností se prakticky ve všech zemích používá „*metr*". Mezi výjimky patří USA, kde se používají yardy (1 yard = 0.9144 m). Metr byl původně definován jako $1/10\,000\,000$ délky zemského kvadrantu[2], jehož délka byla stanovena pomocí měření vzdálenosti mezi městy Dunkerque a Barcelóna, které se uskutečnilo v letech 1793-1799. Od roku 1983 je metr definován jako délka dráhy, kterou urazí světlo ve vakuu za $1/299\,792\,458$ sekundy.

Zdroj: (5), (6)

Pro měření úhlů se v geodézii používají „*grady*", když se jeden grad dále dělí na 100 gradových minut a 1 gradová minuta na 100 vteřin.

Zdroj: (7)

1.2 Vyrovnávací počet

Měření se stalo důležitou součástí většiny vědních disciplín. U geodézie je měření nenahraditelné. Je třeba si uvědomit, že měření je vždy zatížené chybou měření. Smysly a přístroje neposkytují stoprocentní záruku určení správné hodnoty a vnáší do měření

[1] Zdroj: (2), str. 1.
[2] Polovina délky poledníku.

tzv. „*nejistotu měření*". Na nejistotu měření vzdáleností a úhlů mají vliv např. otřesy způsobené větrem nebo teplotní faktory. Stav měřiče, přístroje a prostředí tvoří „*podmínky měření*". Všechny tyto podmínky se neustále mění a ovlivňují výsledky měření. Kvůli úkonům při měření a rušivým vlivům prostředí vznikají „*elementární chyby"*, které se algebraicky sčítají ve výslednou „*měřickou chybu*". Hodnota chyby je v čase proměnlivá a nelze ji přesně určit. Proto při měření stejné veličiny můžeme dostat rozdílné hodnoty nebo nám nemusejí vyjít hodnoty očekávané (např. při měření úhlů v rovinném trojúhelníku nezískáme výsledek 180°). Měřické chyby jsou tedy nevyhnutelné a náhodně se mění. To znamená, že o výsledcích nelze tvrdit, že jsou spolehlivé. Výsledky mohou být pouze spolehlivé v mezích chyb. Aby se předešlo hrubým chybám a zvýšila se přesnost výsledku měření, je třeba provést více měření dané veličiny anebo je třeba určit další veličiny, které mohou být s původní v nějakém vztahu (např. délky, úhly nebo určit pokaždé jiné funkční vztahy mezi více veličinami). Ke zpracování výsledků, které se určitým způsobem liší, se používá již zmíněný matematický aparát: „*vyrovnávací počet*".

Mezi úkoly vyrovnávacího počtu patří:

- vyrovnání přímých měření,
- vyrovnání zprostředkujících měření,
- vyrovnání podmínkových měření.

Zdroj: (1)

1.2.1 Vyrovnání přímých měření

Do této kategorie patří vyrovnání měření délky pásmem nebo měření úhlu teodolitem. Vlastně se snažíme o získání nejdůvěryhodnější hodnoty z výsledků měření téže veličiny. To znamená, že měření jediné neznámé veličiny bylo několikrát opakováno (replikace měření) za účelem vyrovnání této veličiny.

Zdroj: (1)

1.2.2 Vyrovnání zprostředkujících měření

Způsobu vyrovnání zprostředkujících měření používáme, pokud se hledané neznámé veličiny neměří přímo, ale určují se pomocí jiných měřených veličiny, které jsou s nimi v nějakém známém vztahu. Taková měření se nazývají nepřímá nebo zprostředkující. V tomto případě se vyrovnává více veličin. K vyrovnání použijeme tzv. „*nadbytečná měření*", ze kterých můžeme sestavit více rovnic, než je neznámých.

Zdroj: (1)

1.2.3 Vyrovnání podmínkových měření

Pokud jsou spolu měřené veličiny v určitém přesném matematickém vztahu (např. skutečné hodnoty v rovinném trojúhelníku splňují podmínku $\alpha + \beta + \gamma - 180° = 0$). Je

velmi pravděpodobné, že pokud budeme měřit tyto veličiny v nadbytečném počtu (máme změřeny dva úhly a měříme třetí), tak nedostaneme vlivem měřických chyb očekávané hodnoty a k tomu, abychom tyto nesouhlasy odstranili, provedeme jejich vyrovnání.

Zdroj: (1)

1.3 Historie

Historie geodézie sahají až do doby př. K., konkrétně do Starověku. Tvarem a velikostí Země se zabývalo mnoho učenců.

Zřejmě první představa o světě se nachází v Homérových zpěvech (asi 850 př. K.), kde je Země popsána jako kotouč obklopený oceánem. V 6. století př. K. Pythagoras zastával názor, že Země má tvar koule. Tuto hypotézu vyučoval i jeho žák Platón. Ale až Aristoteles (384 - 322 př. K.) podložil Pythagorův názor o kulatosti Země přesvědčivými důkazy. Argumentoval, že loď postupně mizí za obzorem při odplouvání nebo, že Země vrhá na Měsíc kruhový stín při zatmění. Aristoteles nám také zanechal první odhad obvodu Země. Tento odhad byl 400 000 stadií.

Zdroj: (4), (9)

O určení velikosti Země, na základě výpočtu, se pokusil Eratosthenés roku 250 př. K.. Eratosthenes předpokládal, že země je ideálně kulaté těleso. Pro výpočet použil vzdálenost mezi Alexandrií a Asuánem, která měla činit 5000 stádií. Vzdálenost údajně nezměřil Eratosthenes. Jednou z možností jak údaj o vzdálenosti získal, je od poslů ptolemaiovské říše, kteří museli zaznamenávat vzdálenosti, které urazili na svých cestách. Dále předpokládal, že Alexandrie i Asuán leží na stejném poledníku, a že sluneční paprsky v době letního slunovratu (21. - 22. června) dopadají v Asuánu kolmo na zemský povrch. O kolmosti dopadu slunečních paprsků rozhodl na základě faktu, že sluneční paprsky, dopadající na dno studní a tělesa, jako jsou obelisky, nevrhají žádný stín. Ve stejnou dobu, jako dopadaly v Asuánu paprsky kolmo na zemský povrch, musel provést měření tzv. zenitové vzdálenosti (úhel mezi svislicí a směrem ke Slunci) v Alexandrii. Použil přístroj zvaný skafé, což jsou vlastně sluneční hodiny. Pomocí skafé určil zenitovou vzdálenost 7,2°. K výpočtu obvodu Země použil jednoduchou úvahu. Poměr mezi obvodem Země ve stupních (360°) a zenitovou vzdáleností v Alexandrii (7,2°) je stejný jako poměr obvodu Země ve stadiích a vzdálenosti mezi Alexandrií a Asuánem. Tudíž je výsledný obvod 250 000 stadií (360 / 7,2 * 5000).

Zdroj: (4), (6), (8), (9), (14)

Podobným způsobem jako Eratosthenes určil poloměr Země Poseidónios. Poseidónios použil pro délku oblouku spojnice Alexandrie a Rhodu (také o délce přibližně 5000 stadií) a místo Slunce použil hvězdu Canopus. Poloměr mu tehdy vyšel 240 000 stadií. Z tohoto údaje vycházel například Klaudios Ptolemaios (90 - 160 n. l.) při tvorbě své první mapy světa a Kryštof Kolumbus při odhadu doby plavby z Evropy do Indie (1492). Chyba

Ptolemaiovy mapy byla odhalena až později, když se zjistilo, že Kolumbus doplul do Ameriky, místo do Indie.

Zdroj: (4), (9)

V Asii přibližně roku 1000 provedl Ahmad al Bírúní měření zemského poloměru. K měření použil vrch v Indii, který se zvedal nad okolní rovinu. Nejprve zjistil výšku vrchu a z tohoto údaje určil poloměr a obvod Země. Celé měření uskutečnil pomocí dvou provazců a hodnotu poloměru Země stanovil na 12 951 369 loktů. Jedná se o tzv. černý loket, jehož délka je stanovena na 49.29 - 50 cm. Když dosadíme tuto veličinu do Bírúního výpočtu, vyjde velmi přesná hodnota poloměru Země, 6383-6475 km.

Zdroj: (4), (6)

V Evropě byly myšlenky řeckých učenců z počátku 1. tisíciletí zapomenuty. Nositelem vzdělanosti byla církev, která zastávala názor, že Země je plochá. Po zámořských objevech je církev nucena své názory revidovat. Církvi oponovali také Mikuláš Koperník, Galileo Galilei a Johannes Kepler, kteří přišli s názorem označovaným jako heliocentrismus[3]. V 18. století se objevili první názory, že Země není dokonalá koule, ale rotační elipsoid (koule zploštělá na pólech), což bylo i potvrzeno na základě přesnějších stupňových měření[4] v Peru a Laponsku(1735-1741). Dnes se uvádí, že Země má tvar geoidu.

Zdroj: (4)

V dnešní době se k určení polohy hojně využívá GPS, který provozuje ministerstvo obrany Spojených států amerických. Pomocí GPS lze určit polohu kdekoliv na Zemi nebo nad Zemí s přesností do deseti metrů. Přesnost lze zvýšit pomocí speciálních metod až na jednotky centimetrů. Výpočet polohy je prováděn na základě 24 družic, které po kruhových drahách obíhají Zemi.

Zdroj: (16)

[3] Slunce je středem vesmíru a sluneční soustavy.
[4] Určí se délka jednoho stupně (ve směru poledníku) a z něj se určí délka poloměru Země

2 Řešení typických úloh

Nyní si detailněji probereme dvě typické úlohy. V první budeme znát tři body a k nim tři naměřené hodnoty (3 vzdálenosti). U druhé úlohy budeme mít dva body a úhel. Naměřené hodnoty tedy budou dvě vzdálenosti a jeden úhel.

2.1 Úloha č.1

2.1.1 Zadání a rozbor

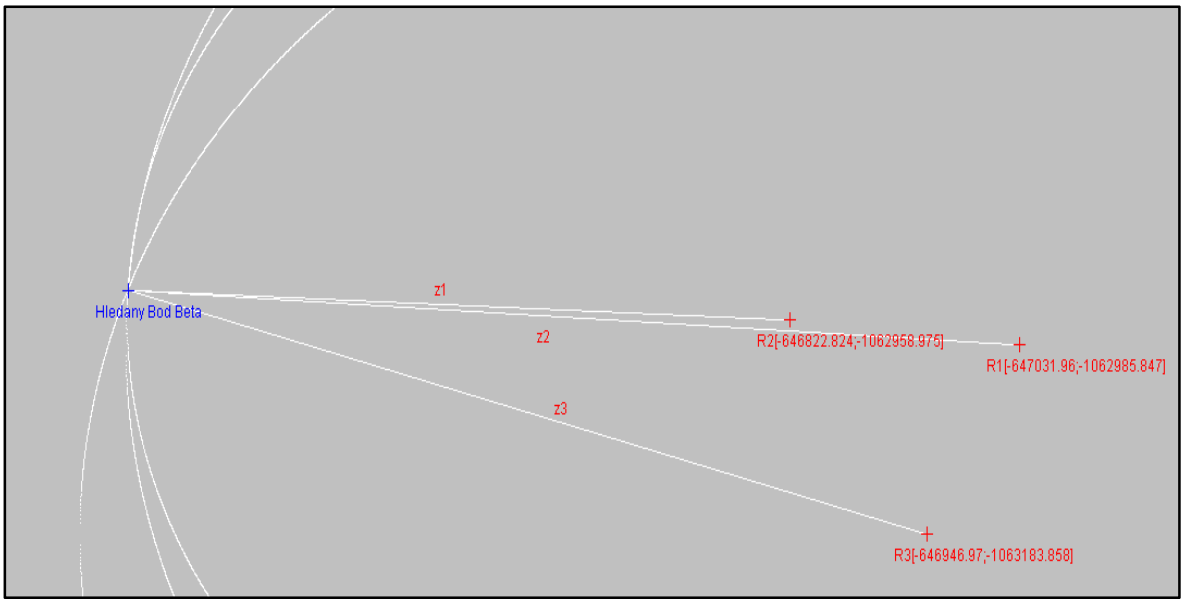
Označme bodem $\beta=(\beta_1;\beta_2)$ polohu neznámého bodu. Polohu bodu β určíme nepřímo pomocí měření vzdáleností. Nechť body $R_1=[X_1;Y_1]$, $R_2=[X_2;Y_2]$, $R_3=[X_3;Y_3]$ jsou dány deterministicky. Mějme měření vzdáleností z_1, z_2, z_3 se směrodatnou odchylkou σ^2. Označme z_1 hodnotu měření vzdálenosti mezi R_1 a β, z_2 hodnotu měření vzdálenosti mezi R_2 a β a z_3 hodnotu měření vzdálenosti mezi R_3 a β. Bod β se nachází na průsečíku kružnic (kružnice o poloměru z_1 se středem v bodě R_1, kružnice o poloměru z_2 se středem v bodě R_2 a kružnice o poloměru z_3 se středem v bodě R_3).

Naměřené hodnoty jsou:

$X_1 = -647031{,}960$ m, $Y_1 = -1062985{,}847$ m, $z_1 = 603{,}849$ m,

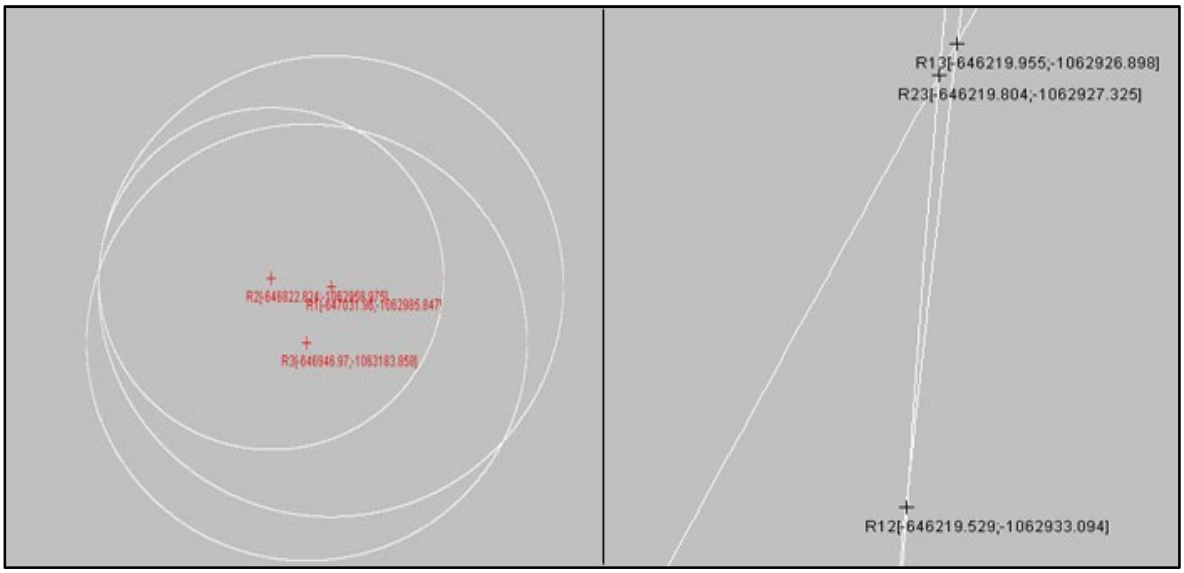
$X_2 = -646822{,}824$ m, $Y_2 = -1062958{,}975$ m, $z_2 = 814{,}141$ m,

$X_3 = -646946{,}970$ m, $Y_3 = -1063183{,}859$ m, $z_3 = 771{,}089$ m.



Obrázek 1 - Schéma zadání úlohy č.1. Zdroj: vlastní.

Měřené hodnoty (z_1,z_2,z_3) určí popsané kružnice (viz Obrázek 1). Na obrázku se zdá, že všechny kružnice mají hledaný průsečík v tomtéž bodě. Při detailnějším pohledu ale zjistíme, že tomu tak není (viz Obrázek 2). Důvodem jsou nejistoty měření vzdáleností.



Obrázek 2 - Schéma zadání úlohy č.1 a)oddálené, b)přiblížené. Zdroj: vlastní.

Na obrázku 2 je jasně vidět, že kružnice nemá pouze jeden průsečík. Jednak musíme brát v potaz dvojici průsečíku každých dvou kružnic, což je patrné z přiblíženého obrázku, kde je první průsečík a z oddáleného obrázku, kde je vidět průsečík druhý. Takže musíme určit, který ze dvou nalezených průsečíku je ten správný a poté z výsledných třech průsečíků dopočítat počáteční řešení, které použijeme v dalších výpočtech.

2.1.2 Praktické řešení

Při dalších výpočtech bude potřebná znalost přibližné polohy bodu β. Celé řešení tedy nejprve zobecníme. Mějme tři kružnice (k_1, k_2, k_3). Pro kružnice k_1 a k_2 chceme spočítat jejich průsečík pomocí obecných rovnic těchto kružnic:

$$(x - m_1)^2 + (y - n_1)^2 = r_1^2, \tag{2.1}$$

$$(x - m_2)^2 + (y - n_2)^2 = r_2^2. \tag{2.2}$$

Z těchto rovnic se pokusíme obecně získat jejich průsečíky, které si označíme jako P_1 a P_2. Rovnice roznásobíme a odečteme od sebe. Výsledek bude vypadat takto:

$$-2xm_1 + 2xm_2 + m_1^2 - m_2^2 - 2yn_1 + 2yn_2 + n_1^2 - n_2^2 = r_1^2 - r_2^2.$$

(2.3)

Po úpravě získáme:

$$x = \frac{r_1^2 - r_2^2 - m_1^2 + m_2^2 + 2yn_1 - 2yn_2 - n_1^2 + n_2^2}{(2m_2 - 2m_1)}.$$

(2.4)

Zlomek si rozdělíme na dvě části (pro pozdější přehlednější vyjádření):

$$x = \frac{r_1^2 - r_2^2 - m_1^2 + m_2^2 - n_1^2 + n_2^2}{(2m_2 - 2m_1)} + \frac{y.(2n_1 - 2n_2)}{(2m_2 - 2m_1)}.$$

(2.5)

Použijeme substituci:

$$x = P + y.Z,$$

(2.6)

kde $P = \frac{r_1^2 - r_2^2 - m_1^2 + m_2^2 - n_1^2 + n_2^2}{(2m_2 - 2m_1)}$,

$Z = \frac{(2n_1 - 2n_2)}{(2m_2 - 2m_1)}$.

Poté dosadíme rovnici 2.6 zpět do původní rovnice 2.1 a získáme kvadratickou rovnici ve tvaru:

$$Ay^2 + By + C = 0,$$

(2.7)

kde $A = Z^2 + 1$,

$B = 2PZ - 2m_1Z - 2n_1$,

$C = P^2 - 2m_1P + m_1^2 + n_1^2 - r_1^2$.

Koeficienty P a Z lze spočítat, protože jsou známé všechny hodnoty k jejich výpočtu. Vyřešíme kvadratickou rovnici, ze které získáme dvě hodnoty, což jsou souřadnice P_{1y} a P_{2y}. Tyto dvě hodnoty dosadíme do rovnice 2.6 a získáme P_{1x} a P_{2x}. Nyní je třeba určit, který bod je ten správný. K tomu využijeme fakt, že správný bod bude ten, jehož hodnota bude bližší nule po dosazení do rovnice kružnice k3:

$$\sqrt{(R_x - P_x)^2 + (R_y - P_y)^2} - d = 0,$$

(2.8)

kde R_x - souřadnice x středu kružnice k_3,

R_y - souřadnice y středu kružnice k_3,

Px - souřadnice x daného průsečíku,

P_y - souřadnice y daného průsečíku,

d - poloměr kružnice k_3.

Teď již máme hledaný průsečík kružnic k_1 a k_2. Stejným způsobem, jako jsme spočítali průsečík kružnic k_1 a k_2, za pomoci jejich obecných rovnic a dosazením do rovnice kružnice k_3, můžeme dopočítat i průsečíky k_1 a k_3 a také k_2 a k_3. Nyní máme všechny potřebné průsečíky (P_1, P_2, P_3) , takže můžeme určit střed trojúhelníka a tím získáme počáteční řešení $\beta_0=(\beta_1;\beta_2)$. Toto řešení je přibližné, protože na něho nejsou uplatněny žádné chyby měření. Je tedy třeba určit přesnější hodnotu hledaného bodu. Nejprve je třeba určit teoretický model měření (podmínky, na měřené a odhadované parametry, které musejí platit):

$$f_1(\beta_0) = \sqrt{(X_1 - \beta_1)^2 + (Y_1 - \beta_2)^2} - z_1 = 0,$$

$$f_2(\beta_0) = \sqrt{(X_2 - \beta_1)^2 + (Y_2 - \beta_2)^2} - z_2 = 0,$$

$$f_3(\beta_0) = \sqrt{(X_3 - \beta_1)^2 + (Y_3 - \beta_2)^2} - z_3 = 0,$$

(2.9)

kde $(X_n; Y_n)$ - souřadnice splňující rovnici n-té kružnice,

$(\beta_1; \beta_2)$ - souřadnice počátečního řešení,

z_n - poloměr n-té kružnice (naměřená hodnota).

Abychom získali $\delta\widehat{\boldsymbol{\beta}}$, což bude vlastně vektor, který bude obsahovat hledané hodnoty pro určení přesnějšího výsledku, aplikujeme metodu nejmenších čtverců[5]:

$$\delta\widehat{\boldsymbol{\beta}} = (F'.F)^{-1}.F'.\delta\mathbf{Y},$$

(2.10)

[5] viz zdroj (10), str. 81

kde $F = \begin{pmatrix} \frac{\partial f_1(\beta_0)}{\partial \beta_1} & \frac{\partial f_1(\beta_0)}{\partial \beta_2} \\ \frac{\partial f_2(\beta_0)}{\partial \beta_1} & \frac{\partial f_2(\beta_0)}{\partial \beta_2} \\ \frac{\partial f_3(\beta_0)}{\partial \beta_1} & \frac{\partial f_3(\beta_0)}{\partial \beta_2} \end{pmatrix}$,

F' - transponovaná matice F.

Takže matice F (označována jako matice plánu)[6] je vlastně maticí derivací jednotlivých funkcí, podle β_1 a β_2. Matici F jsme získali linearizací teoretického modelu měření[7] pomocí Taylorova rozvoje, u kterého bereme v potaz pouze první dva členy, ostatní zanedbáváme. V předchozím vzorci se nám ještě objevuje vektor $\delta\mathbf{Y}$, což je:

$$\delta \boldsymbol{Y} = \boldsymbol{Y}_0 - \boldsymbol{Y},$$

(2.11)

kde $\boldsymbol{Y_0}$ - vektor hodnot po dosazení do příslušné podmínky, čili $f_n(X_n, Y_n, \beta_1, \beta_2)$,

$\boldsymbol{Y}$ - vektor naměřených hodnot, čili hodnoty (z_1, z_2, z_3).

Nyní již máme vše potřebné k určení vektoru $\delta\widehat{\boldsymbol{\beta}}$ ze vzorce (2.10). Vypočítáme si hodnotu matice F, tuto matici transponujeme, poté převedeme na inverzní, vynásobíme s transponovanou maticí F a jako poslední vynásobíme s vektorem $\delta\mathbf{Y}$, který lze spočítat (z rovnice 2.11). Jako poslední krok je třeba určit přesnější výsledek. K tomu je zapotřebí dosadit do vzorce:

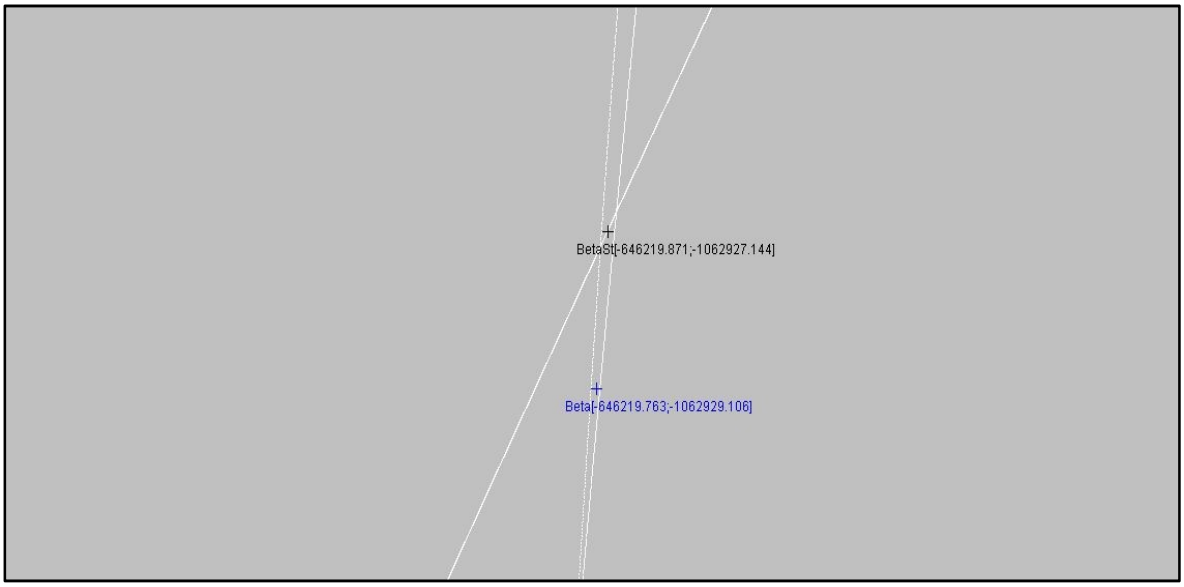
$$\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta_0} + \delta\widehat{\boldsymbol{\beta}},$$

(2.12)

kde $\boldsymbol{\beta_0}$ - vektor počátečního řešení,

$\delta\widehat{\boldsymbol{\beta}}$ - korekce počátečního řešení (vektor spočítaný ze vzorce 2.10).

[6] viz zdroj (10), str. 81
[7] viz zdroj (10), str. 21



Obrázek 3 - Výsledky úlohy č.1. Zdroj: vlastní.

Takže nyní máme vektor $\widehat{\boldsymbol{\beta}}$, která obsahuje odhad souřadnice hledaného bodu. Tato souřadnice označuje místo, kde by měl přibližně ležet hledaný bod. Na obrázku 3 je názorně předvedeno, o kolik se liší počáteční řešení a řešení pomocí metody nejmenších čtverců. Počáteční řešení je zde označeno Beta(modrá barva) a řešení s uplatněním metody nejmenších čtverců je označeno BetaSt(černá barva). Jako poslední si ukážeme konstrukci konfidenční elipsy[8]. Tato elipsa pokrývá (znázorňuje) s pravděpodobností (1-α) skutečný bod β. Lze tedy říci, že s danou pravděpodobností leží hledaný bod uvnitř této elipsy. K získání elipsy potřebujeme kovarianční matici, kterou označíme $U_{\widehat{\beta}}$. Nejprve je třeba určit počáteční matici chyb:

$$U_w = \begin{pmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{pmatrix},$$

(2.13)

kde σ^2 - směrodatná odchylka.

Poté dosadíme do vzorce[9]:

$$U_{\widehat{\beta}} = (F'.{U_w}^{-1}.F)^{-1},$$

(2.14)

kde F - matice plánu,

U_w - matice chyb ze vzorce (2.13).

[8] viz zdroj (10), str. 81, 82
[9] viz zdroj (10), str. 81

V tomto případě bude $U_{\hat{\beta}}$ maticí o rozměrech 2x2. Spočítáme vlastní čísla a vlastní vektory matice $U_{\hat{\beta}}$. Vlastní čísla nám popíší rozměr elipsy a vlastní vektory směr, kterým se bude elipsa ubírat. Ještě je třeba zmínit, že vlastní vektory jsou navzájem kolmé. Jednotlivé prvky elipsy (v nerotovaném tvaru) lze vyjádřit parametrickými rovnicemi:

$$x = a_1.\sin(t),$$

$$y = a_2.\cos(t) \qquad t\,\epsilon\langle 0|2\pi\rangle, \tag{2.15}$$

kde $a_1 = \sqrt{\frac{\chi_2^2}{\lambda_1}}$,

$a_2 = \sqrt{\frac{\chi_2^2}{\lambda_2}}$.

Pro vysvětlení je λ_n označením pro n-té vlastní číslo a pod χ_2 rozuměj kritickou hodnotu chí-kvadrátu o dvou stupních volnosti (při hladině významnosti 0,05 je kritická hodnota 5,59). Hodnoty a_1 a a_2 nám vlastně určují délky os hledané konfidenční elipsy. Nyní je zapotřebí posledního kroku, kterým je rotace spočítané elipsy pomocí vlastních vektorů. Rotace se provede podle následující rovnice:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = (\mathbf{V}_1, \mathbf{V}_2).\begin{pmatrix} x \\ y \end{pmatrix}, \tag{2.16}$$

kde x', y' - souřadnice po uplatnění rotace,

$\mathbf{V}_1$, $\mathbf{V}_2$ - vlastní vektory,

x, y - původní (nerotovaná) souřadnice.

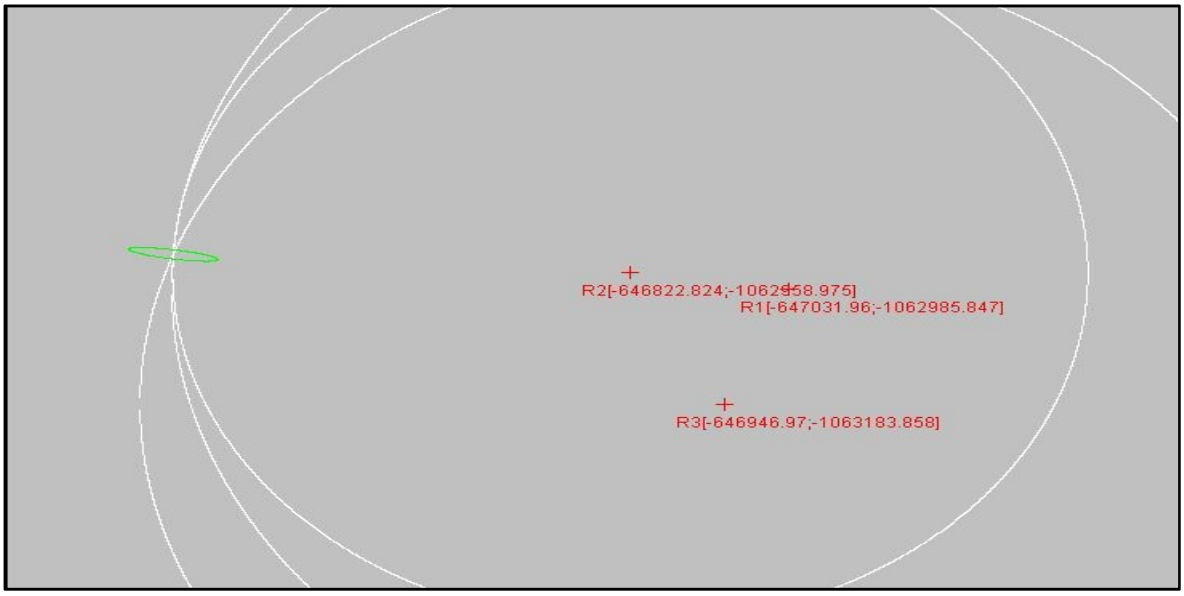
Pro názornou ukázku jsme ze zadaných hodnot úlohy č.1 získali následující dělky os konfidenční elipsy[10] (hladina významnosti byla zvolena 0,05):

$a_1 = 57{,}513$ m,

$a_2 = 7{,}554$ m.

Tyto hodnoty jsou velmi vysoké z důvodu velkých měřických chyb. V praxi by hodnoty museli být rapidně menší. Příklad byl konstruován jako názorná ukázka, není proto žádný problém s rozsáhlostí elipsy. Důležité je, že bod leží s 95% pravděpodobností v této elipse.

[10] viz zdroj (10), str. 82



Obrázek 4 - Konfidenční elipsa pro úlohu č.1. Zdroj: vlastní.

Konfidenční elipsa je na obrázku vykreslena zelenou barvou. Právě její velikost je zde názorně vidět. V praxi by byla elipsa o mnoho menší. Ke zmenšení by bylo zapotřebí získat měřené údaje přesněji nebo použít dalšího měření jiného bodu a vzdálenosti.

2.2 Úloha č.2

2.2.1 Zadání a rozbor

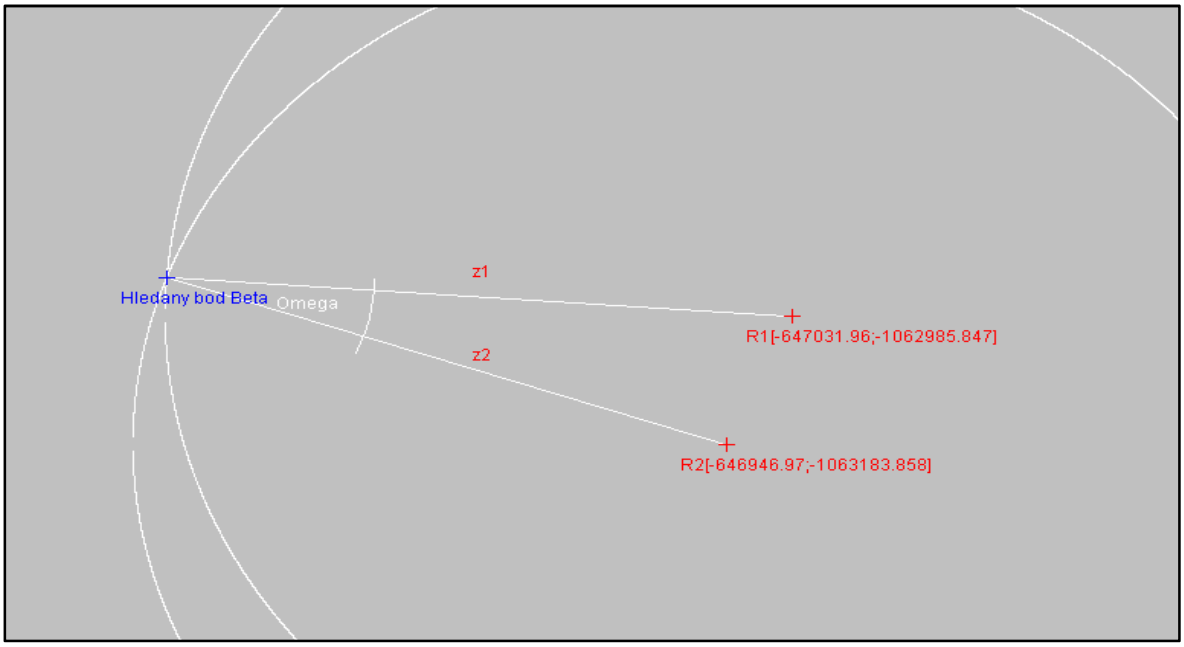
Označme bodem $\beta=(\beta_1;\beta_2)$ polohu neznámého bodu. Polohu bodu β určíme nepřímo pomocí měření vzdáleností. Nechť $R_1=[X_1; Y_1]$, $R_2=[X_2; Y_2]$ jsou dány deterministicky. Mějme měření vzdáleností z_1 a z_2 se směrodatnou odchylkou σ_d^2 a měření úhlu se směrodatnou odchylkou σ_u^2. Označme z_1 vzdálenost mezi R_1 a β, z_2 vzdálenost mezi R_2 a ω jako úhel $R_1\beta R_2$. Hledaný bod β je vlastně vrchol trojúhelníka o stranách z_1 a z_2 a vrcholovém úhlu ω.

Naměřené hodnoty jsou:

X_1 = -647031,96 m, Y_1 = -1062985,847 m, z_1 = 603,849 m,

X_2 = -646946,97 m, Y_2 = -1063183,858 m, z_2 = 771,089 m,

ω = 0.28859 rad (≈16° 54').



Obrázek 5 - Schéma zadání úlohy č.2. Zdroj: vlastní.

Měřené hodnoty (z_1 a z_2) určí popsané kružnice (viz Obrázek 4). Na obrázku je zobrazen pouze jeden průsečík těchto kružnic. Měřič totiž určí, ze kterého místa se uskutečnilo měření úhlu ω (přibližné místo). Není tedy třeba rozhodovat mezi dvěma průsečíky, ale pouze vybrat ten správný. Počáteční řešení tedy získáme jako průsečík kružnic, z bodů R_1 a R_2 o poloměrech z_1 a z_2,. Kružnice označíme k_1 a k_2.

2.2.2 Praktické řešení

Stejně jako v případě určení průsečíku dvou kružnic u úlohy č.1 vyjdeme z obecných rovnic kružnic k_1 a k_2 (viz rovnice 2.1 a 2.2). Počáteční řešení $\beta_0=(\beta_1;\beta_2)$ jsme získali jako průsečík kružnic k_1 a k_2. Nyní určíme teoretický model měření, který se oproti úloze č.1 změní ve třetí podmínce. Teoretický model měření je:

$$f1(\theta) = \sqrt{(X_1 - \beta_1)^2 + (Y_1 - \beta_2)^2} - z_1 = 0,$$

$$f2(\theta) = \sqrt{(X_2 - \beta_1)^2 + (Y_2 - \beta_2)^2} - z_2 = 0,$$

$$f3(\theta) = arctg\left(\frac{R_{1y}-\beta_2}{R_{1x}-\beta_1}\right) - arctg\left(\frac{R_{2y}-\beta_2}{R_{2x}-\beta_1}\right) - \omega = 0,$$

(2.13)

kde $(X_n; Y_n)$ - souřadnice splňující rovnici n-té kružnice,

$(\beta_1; \beta_2)$ - souřadnice počátečního řešení,

z_n - poloměr n-té kružnice (naměřená hodnota),

ω - naměřený úhel.

Stejně jako v první úloze použijeme metodu nejmenších čtverců, ve které však dochází ke změně kvůli matici chyb. Ta pokud obsahuje na hlavní diagonále stejné hodnoty směrodatné odchylky, není zapotřebí tuto matici započítávat do vzorce (2.10). V našem případě však matice chyb obsahuje dvě různé hodnoty na hlavní diagonále:

$$U_w = \begin{pmatrix} \sigma_d^2 & 0 & 0 \\ 0 & \sigma_d^2 & 0 \\ 0 & 0 & \sigma_u^2 \end{pmatrix},$$

(2.14)

kde σ_d^2 - směrodatná odchylka měření délky,

σ_u^2 - směrodatná odchylka měření úhlu.

Odchylky jsou různé z důvodu, že měření délek je více nepřesné, něž měření úhlů. V praxi proto geodeti dávají přednost odhadu hledaného bodu pomocí měření úhlů, pro zvýšení přesnosti odhadu hledaného bodu β. Vzorec (2.10) se tedy změní následovně[11]:

$$\delta\widehat{\boldsymbol{\beta}} = (\mathrm{F}'.\mathrm{U}_\mathrm{w}^{-1}\mathrm{F})^{-1}.\mathrm{F}'.\mathrm{U}_\mathrm{w}^{-1}.\delta\mathbf{Y},$$

(2.15)

kde F - matice plánu (viz 2.10),

U_w - matice chyb (viz 2.14),

$\delta\mathbf{Y}$ - vektor ze vzorce (2.11).

Po dosazení do vzorce (2.12) získáme stejně jako v první úloze $\hat{\beta}$, což je souřadnice hledaného bodu. Takže nyní máme vektor $\hat{\beta}$, který obsahuje odhad souřadnice hledaného bodu. Konstrukce konfidenční elipsy je naprosto shodná s úlohou č.1, takže netřeba podrobněji popisovat již popsanou činnost. Na této úloze bylo snahou ukázat, jak se změní konfidenční elipsa za použití měření úhlu oproti měření pouze délek stran (viz Obrázek 6).

[11] viz zdroj (10), str. 81

Obrázek 6 - Konfidenční elipsa pro úlohu č.2. Zdroj: vlastní.

Délka os konfidenční elipsy[12] při hladině významnosti $\alpha = 0{,}05$:

$$a_1 = 33{,}14 \text{ m},$$

$$a_2 = 4{,}46 \text{ m}.$$

Z údajů je patrné, že osy konfidenční elipsy druhé úlohy se dostali téměř na poloviční hodnoty oproti první úloze. Důkaz o vyšší přesnosti, při vyrovnávání chyb pomocí úhlu, je tak zřejmý z těchto hodnot.

[12] viz zdroj (10), str. 82

3 Aplikace

Součástí práce je i program, který provádí výpočty a grafické znázornění základních geodetických výpočtů shrnutých v předcházející kapitole. Program byl napsán v programovacím jazyce Java. Jazyk Java byl zvolen hlavně kvůli předešlým zkušenostem z výuky na vysoké škole v tomto jazyce.

3.1 Java

Java je objektově orientovaný programovací jazyk, který je nezávislý na platformě. Byl vyvinut firmou Sun Microsystems a jeho syntaxe vychází z jazyka C++. Oproti jazyku C++ neobsahuje například ukazatele (pointery) a programátor se nemusí starat o správu paměti. O automatickou správu paměti se totiž stará garbage collector. Úkolem garbage collectoru je určit tu část paměti programu, která již není nebo nebude použita a tuto část připravit k dalšímu použití. Místo již zmíněných ukazatelů se v Javě pracuje s odkazy (referencemi), čímž je zamezen přístup do neplatné paměti. Další výhodou jazyka Java je serializace. Serializace umožňuje jednoduchou práci s ukládáním dat do souborů a jejich přenosem po síti. Společnost Sun Microsystems vytvořila jednoduchý slogan: „jednou napiš, spusť kdekoliv" (write once, run everywhere). Tento slogan lze naplnit díky tomu, že zdrojový kód je přeložen do mezikódu (tzv. bytecode), který lze spouštět pomocí virtuálního stroje jazyka Java (JVM). Bytecode je vlastně při spuštění programu převeden pomocí JVM na strojový kód procesoru (s ohledem na operační systém).

Architektura Javy je kombinací čtyř částí:

- Programovací jazyk Java,
- Formát souboru .class,
- Java API,
- JVM.

Při vyvíjení aplikace je kód psán v programovacím jazyce Java. Kód se zkompiluje do souboru s příponou .class. Soubor .class je spuštěn pomocí JVM. Java API je dopředu připravený kód uspořádaný do balíčků a dělí se na tři základní platformy:

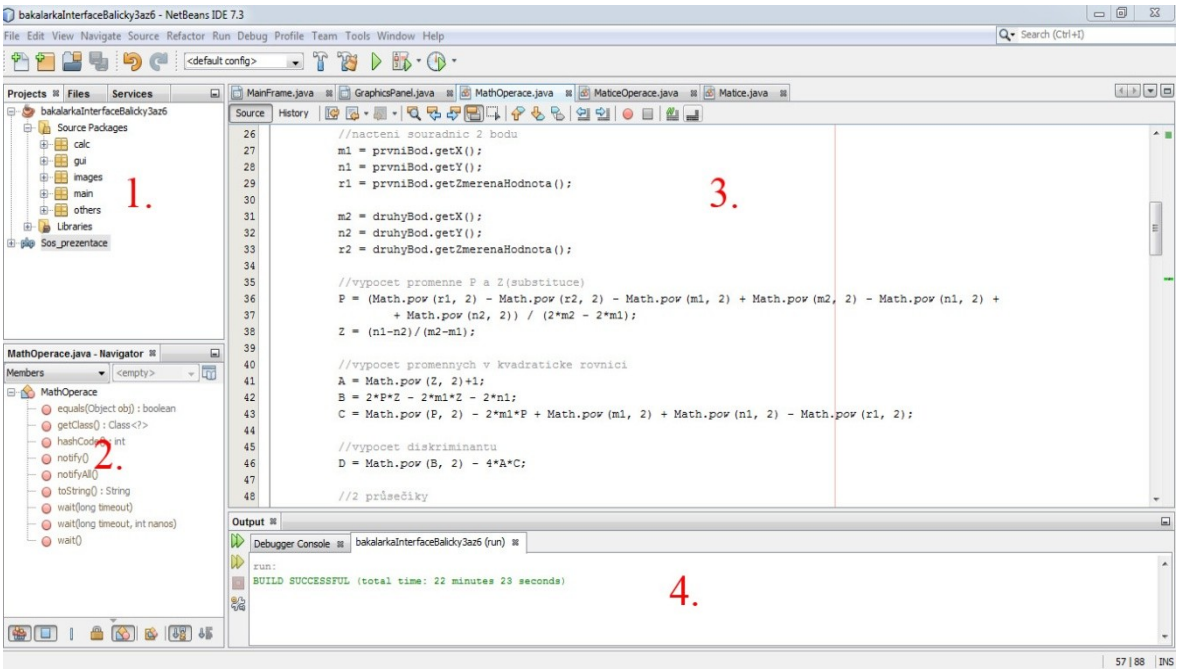
- Java Micro Edition (JME) - vývoj aplikací např. pro mobilní telefony nebo naváděcí systémy motorových vozidel,
- Java Standard Edition (JSE) - základní sada tříd pro tvorbu GUI a standardních aplikací,
- Java Enterprise Edition (JEE) - sada tříd a rozhraní pro vývoj webových aplikací.

Zdroj: (11), (12)

3.2 NetBeans

Pro psaní programu bylo použito open-source vývojového prostředí NetBeans ve verzi 7.3. Toto vývojové prostředí je volně dostupné a primárně bylo určeno pro vývoj aplikací v jazyce Java, ale podporuje i mnoho dalších programovacích jazyků (např. C/C++, PHP, Groovy...). Lze jej spustit na operačních systémech Windows, Linux, Mac OS a Solaris. Vývojové prostředí je přehledně dělené, barevně členěné, s aktivní komunitou uživatelů a rozsáhlou dokumentací. Stejně jako programovací jazyk Java bylo i vývojové prostředí NetBeans 7.3 vybráno na základě předchozích zkušeností z výuky na vysoké škole. Jako alternativu k vývojovému prostředí NetBeans lze zmínit Eclipse, JBuilder nebo IntelliJ IDEA.

Zdroj: (13)



Obrázek 7 - NetBeans IDE 7.3. Zdroj: vlastní.

Obrázek je rozdělen do čtyř očíslovaných částí. Číslo 1 označuje oblast, kde se nachází struktura vytvořených projektů, pro snadnější orientaci ve větších projektech. Číslo 2 symbolizuje Navigatora. Navigator slouží pro zobrazení metod dané třídy v přehledném výpise. Hlavní a nejdůležitější část pro každého uživatele vývojového prostředí NetBeans IDE 7.3 je označeno číslem 3. Zde se nachází prostor pro psaní kódu. Číslo 4 reprezentuje výstup označovaný jako konzole. Je zde mnoho možností, co se na tomto místě může zobrazovat. Velmi užitečná je Debugger Console, která slouží ke krokování. Uživatel si zde definuje, jaké proměnné chce sledovat. V konzoli se mu poté zobrazují aktuální hodnoty definovaných proměnných.

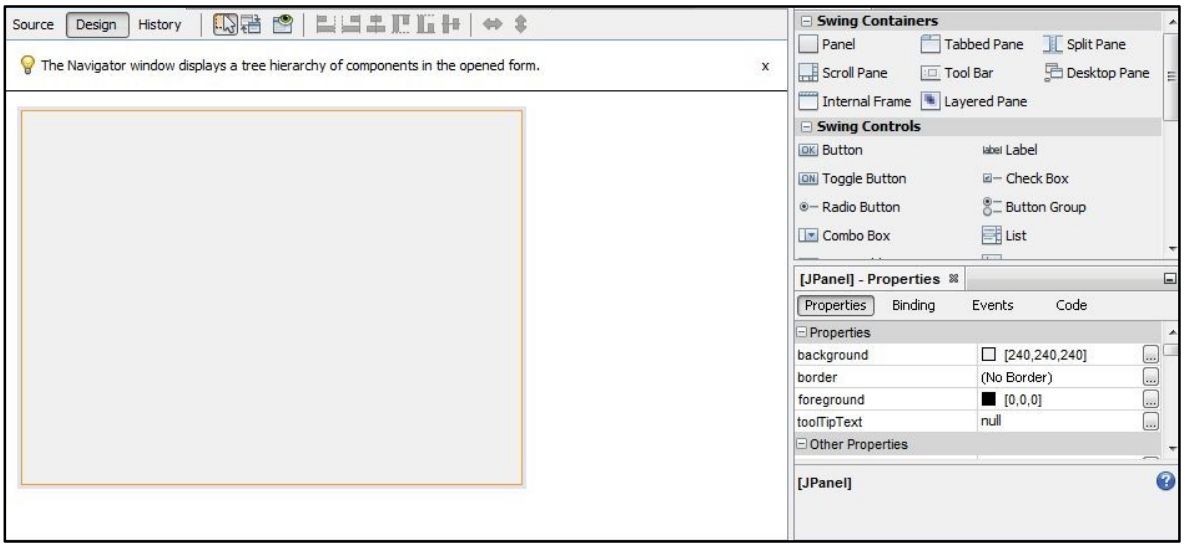
3.3 Swing

K tvorbě grafického uživatelského rozhraní (ovládací prvky, textová pole, vykreslovací panel, atd.) byla použita knihovna uživatelských prvků pro platformu Java. V prostředí NetBeans je implementován návrhář (viz Obrázek 5) pro grafické uživatelské rozhraní. V tomto grafickém prostředí si lze vše snadno sestavit. Kostra kódu se vygeneruje automaticky a vlastní implementaci jednotlivých komponent dopíšeme. V návrháři lze také nastavovat základní věci a pro každou komponentu definovat příslušné metody. Například pro jednoduché tlačítko (Button) lze definovat události (eventy), které se aktivují při určité akci dané komponenty (stisknutí, výběr položky z menu, přejetí myší nad panelem, atd.).

3.4 Použité komponenty knihovny Swing

V celém projektu byly použity následující komponenty knihovny Swing:

- JFrame,
- JPanel,
- JLabel,
- JTextField,
- JComboBox,
- JRadioButton,
- ButtonGroup,
- JButton,
- JOptionPane.



Obrázek 8 - Návrhář v prostředí NetBeans IDE 7.3. Zdroj: vlastní.

V každém projektu, který využívá knihovnu Swing, je třeba jako první vytvořit hlavní třídu (v tomto programu *MainFrame*), jenž dědí od *JFrame*. *JFrame* je vlastně hlavní okno, které se spustí jako první. Právě do hlavního okna se přidávají další komponenty pomocí návrháře (viz výše u výčtu komponent knihovny Swing). Na obrázku 5 je zachycen právě

návrhář. Šedý obdélník v levé části obrázku je *JFrame*, na který lze vkládat komponenty. Komponenty se nacházejí v pravém horním okraji obrazovky. Pod komponentami vidíme záložku properties, sloužící k upřesnění vlastností dané komponenty (např. barva, velikost, ohraničení).

JPanel se používá jako kontejner pro ostatní komponenty. Na jeden *JPanel* lze neomezeně přikládat další *JPanely* a jiné komponenty. V této bakalářské práci byl použit jeden z mnoha *JPanelů* k vykreslování bodů, kružnic, souřadnic a dalších potřebných věcí.

JButton je tlačítko, které slouží ke spuštění specifické činnosti pomocí události. Pokud se tlačítko stiskne, spustí se metoda s názvem identifikátoru tlačítka (pro určení, se kterým tlačítkem se pracuje) a typem události *ActionPerformed* (stisknutí tlačítka).

JTextField slouží k získávání hodnot od uživatele. Jedná se o vstupní formulář (textové pole), do kterého lze zadávat libovolné znaky. Vstup tedy musíme ošetřit proti zadání nechtěných hodnot (např. součet dvou čísel, z formuláře očekáváme celá čísla, ale uživatel zadá do formuláře libovolné písmeno). Zde je vstup ošetřen pomocí bloku *try-catch* a v případě, že se jedná o špatné vstupní hodnoty, je vyvolán *JOptionPane* (dialogové okno), který uživatele upozorní, na špatně zadané hodnoty. Blok *try-catch* nedovolí, aby program pracoval se špatnými hodnotami, protože pokud se v oblasti *try* vyskytne výjimka (zde *NumberFormatException*), běh programu se přesune do bloku *catch*, kde se výjimka ošetří, aby nezpůsobila nechtěné chování programu.

K identifikaci textových polí pro uživatele jsou použity *JLabely* (štítky). *JLabel* se využívá k zobrazení textu, ale lze ho i za běhu programu měnit a zapisovat do něho libovolné hodnoty (např. výstupy programu).

JRadioButton slouží k označování, takže lze pracovat se specifičtější volbou od uživatele. Často se používá v kombinaci s *ButtonGroup* (skupina). *ButtonGroup* umožňuje přidávat *JRadioButtony* do celku a v takto definovaném celku lze vždy označit jen jeden *JRadioButton* z celé *ButtonGroup*. Tato implementace je žádoucí v případě, kdy má uživatel vybrat jen jednu volbu z více možných. *JRadioButton* se automaticky odškrtne, pokud se označí jiný ze skupiny. *JCheckbox* (zaškrtávací pole) se zdá prakticky totožný s *JRadioButtonem*. Nelze s ním ale pracovat v kombinaci s *ButtonGroup*, což je asi jediný rozdíl (pokud nebereme v potaz rozdílný vzhled).

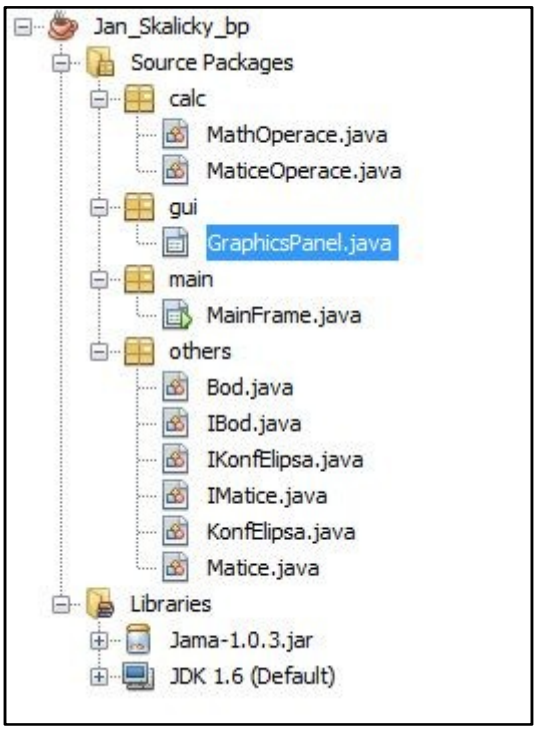
Posledním z řady swingových komponent zde máme *JComboBox* (výběrové pole). *JComboBox* umožňuje uživateli, aby si vybral z definovaných možností právě jednu. Je to vlastně to samé, jako mít více komponent *JRadioButton*, který by byli v jedné *ButtonGroup*. *JComboBox* poskytuje elegantnější řešení a úsporu prostoru na daném *JPanelu*, protože zobrazuje jen jedinou položku, kterou lze změnit vysouvacím seznamem, jenž se zobrazí po kliknutí na šipku daného *JComboBoxu*. Existuje samozřejmě více komponent, které lze použít, ale v tomto projektu byli nepotřebné.

3.5 Knihovna JAMA

Knihovna JAMA byla použita, vzhledem k nutnosti získáni vlastních čísel a vlastních vektorů z kovarianční matice, pro vykreslení konfidenční elipsy. Z této knihovny byli použity dvě třídy. Třída *Matrix*, která reprezentuje libovolnou matici (v našem případě matici 2x2), a třída *EigenValueDecomposition*. *EigenValueDecomposition* umožňuje z libovolné instance třídy Matrix získat vlastní čísla a vlastní vektory. Po vytvoření instance třídy *Matrix* s parametry počet řádků a počet sloupců, lze pomocí metody *set* nastavit libovolné prvky dané matice. Poté je možné, pomocí konstruktoru třídy *EigenValueDecomposition* s parametrem *Matrix* (čili parametrem konstruktoru je instance třídy *Matrix*), vytvořit instanci třídy *EigenValueDecomposition*, jenž umožňuje pomocí metod *getRealEigenvalues* (vlastní čísla matice) a *getV* (vlastní vektory) získat potřebné hodnoty. Návratovými hodnotami jsou dvouprvkové pole typu double (vlastní čísla) a instance třídy *Matrix* (vlastní vektory).

Zdroj: (15)

3.6 Struktura balíčků a tříd



Obrázek 9 - Struktura balíčků a tříd. Zdroj: vlastní.

Celý projekt je rozdělen do čtyř balíčků. Balíčky jsou zde hlavně z důvodu přehlednosti a logického členění všech tříd tohoto projektu. Ve složce *libraries* je vidět výše zmíněná knihovna JAMA (verze 1.0.3).

3.6.1 Balíček calc

Balíček *calc* obsahuje třídy *MathOperace* a *MaticeOperace*. Třída *MathOperace* slouží ke složitějším matematickým výpočtům. Obsahuje metody pro výpočet průsečíků kružnic, určení správného průsečíku ze dvou bodů, výpočet středu N-úhelníku a získání funkční hodnoty v daném bodě (pro rovnice z teoretického modelu měření uvedené v kapitole 2, konkrétně se jedná o rovnice z teoretických modelů 2.9 a 2.13). *MaticeOperace* je třída, která provádí vybrané maticové operace. Nechybí zde metody pro násobení dvou matic, transponování matice, získání inverzní matice, tvorbu matice plánu (matice derivací), výpočet determinantu matice, výpočet vlastních čísel matice a vlastních vektorů. Právě výše zmíněná knihovna JAMA je použita v této třídě. Ještě je velmi důležité zmínit, že všechny výpočetní metody tříd *MathOperace* a *MaticeOperace* jsou statické. Statické jsou zde z důvodu, aby mohli být metody těchto tříd volány, aniž by byla vytvořena instance dané třídy obsahující tyto metody. V praxi tak stačí pouze zavolat danou metodu, která provede výpočet a vrátí potřebné hodnoty.

3.6.2 Balíček gui

Balíček *gui* obsahuje pouze třídu *GraphicsPanel*. Tato třída je však nejobsáhlejší třídou celého projektu, takže musíme blíže popsat její funkčnost. *GraphicsPanel* velmi využívá ke své činnosti tříd *MathOperace* a *MaticeOperace* (pro potřebné výpočty) a stará se o vykreslování všech výsledků. Vykresluje zadané body od uživatele, kružnice reprezentující naměřené hodnoty z daných bodů, průsečíky těchto kružnic, počáteční řešení, řešení pomocí vyrovnání chyb a konfidenční elipsu. V režimu bez načtené mapy je zde možnost přibližovat a oddalovat celé plátno pomocí skrolovacího kolečka myši (o načtení mapy viz kapitola 3.7). Konstruktor se zdá být velmi obsáhlý, ale opodstatněním je přímá práce s danými komponenty. Například při pohybu myší je třeba ihned měnit hodnotu souřadnice v textovém poli, které se nachází ve třídě *MainFrame*, proto nutnost přímé vazby při vytvoření instance třídy *GraphicsPanel* ve třídě *MainFrame*.

Plátno využívá privátních metod, díky kterým může program pracovat s libovolným souřadným systémem, který se nadefinuje. Tím je myšleno, že zde máme čtyři privátní atributy (*realMaxX*, *realMaxY*, *realMinX*, *realMinY*), které jsou použity pro práci s libovolným souřadným systémem. Plátno v Javě totiž má defaultní souřadný systém jiný, než je tomu obyčejně zvykem. Počáteční bod (rozuměj bod [0; 0]) se nachází v levém horním rohu, nikoliv jak jsme zvyklí v levém dolním rohu (osa y je tedy obráceně). Proto potřebujeme privátní atributy pro přepočet, do kterých si načteme právě potřebný souřadný systém a v případě, že má být cokoliv vykresleno do souřadného systému zařízení (v našem případě plátno v Javě), musí se každá vykreslovaná hodnota převést z reálného (našeho) souřadného systému do souřadného systému zařízení (plátna). K tomu slouží privátní metody *pixToReal* a *realToPix* pro danou souřadnici, kde *pix* reprezentuje souřadný systém zařízení a *real* reprezentuje souřadný systém uživatele. *GraphicsPanel* využívá dvou kontejnerů *ArrayList* pro ukládání průsečíků kružnic a zadaných bodů od uživatele. Pomocí metody *paint* se provádí veškeré vykreslování (kromě třídy konfidenční elipsy, která obsahuje také metodu *paint*), na panelu. Tato metoda je volána

při prvotním vytvoření a poté ji lze libovolně volat metodou *repaint*, která metodu *paint* zavolá a celé plátno se překreslí.

3.6.3 Balíček main

V tomto balíčku je třída *MainFrame*, která má jediný atribut *GraphicsPanel*. *MainFrame* je volána při prvotním spuštění programu. Zobrazí hlavní okno definované v návrháři a poté volá jednotlivé metody *GraphicsPanelu* pro konkrétní činnosti (např. uložení bodu od uživatele, překreslení plátna nebo změnu hodnot souřadného systému). Obsahuje i privátní metody (*UhelVisible, BodVisible*), které slouží k zobrazení nebo skrytí dané komponenty v *MainFramu*.

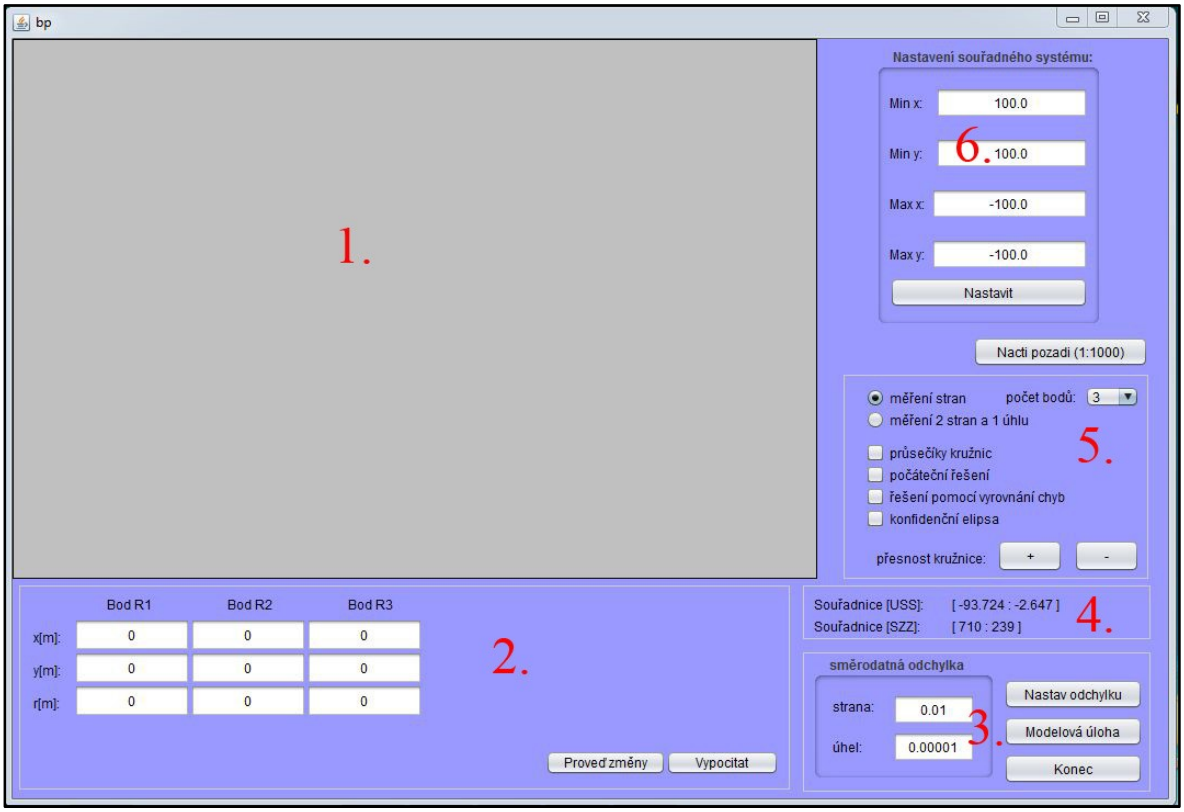
3.6.4 Balíček others

Tento balíček obsahuje tři třídy a k těmto třídám tři rozhraní (*interface*). Rozhraní obsahuje seznam metod, které může implementovat libovolná třída. V rozhraní můžeme deklarovat všechny potřebné metody, aniž by bylo nutné je konkrétně implementovat. Pokud třída implementuje dané rozhraní, musí implementovat i dané metody tohoto rozhraní. Stručně řečeno je rozhraní jakýmsi předpisem pro třídu, která jej implementuje.

V balíčku *others* jsou rozhraní *IBod* (implementuje třída *Bod*), *IMatice* (implementuje třída *Matice*) a *IKonfElipsa* (implementuje třída *KonfElipsa*). Třída *Bod* reprezentuje jednu souřadnici (x, y) v souřadném systému, změřenou hodnotu (pro výpočty) a název bodu (pro identifikaci). Dále obsahuje konstruktor a metody *get* a *set* pro získání případně nastavení všech zmíněných atributů. Třída *Matice* reprezentuje matici o libovolném počtu řádků a sloupců. Prvky matice jsou datového typu double.

Třída *Matice* tedy obsahuje atributy *pocetRadku*, *pocetSloupcu* a *prvkyMatice*, což je dvourozměrné pole typu double. Nechybí zde konstruktor a metody *setPrvekMatice* (pro nastavení prvku na zvoleném indexu), *getPrvekMatice* (pro získání prvku na daném indexu), *naplnMatici* (nastaví na danou hodnotu všechny prvky matice) a *naplnMaticiSOdchylky* (nastaví na hlavní diagonále matice danou hodnotu, zbylé prvky na hodnotu 0). Metoda *vypisMatici* slouží spíše ke kontrolnímu výpisu. Poslední třída je *KonfElipsa*. Ta obsahuje konstruktor a dvě veřejné a tři privátní metody. První z veřejných metod se jmenuje *nastavSS*. Metoda *nastavSS* umožňuje nastavení souřadného systému pro privátní převodné metody *realToPixX* a *realToPixY* (převod souřadnice z uživatelského souřadného systému do souřadného systému zařízení). Veřejná metoda *paint* slouží k vykreslení konfidenční elipsy za pomoci privátní metody *kresliUsecku* a dvou výše zmíněných převodních metod.

3.7 Grafické rozhraní a funkčnost aplikace



Obrázek 10 - Grafické rozhraní. Zdroj: vlastní.

Obrázek 10 zachycuje grafické rozhraní celé aplikace. Toto rozhraní bylo navrženo pomocí návrháře (viz kapitola 3.3). Nyní k vysvětlení, co se na obrázku vyskytuje. Celé uživatelské rozhraní je vlastně *MainFrame* (viz kapitola 3.6.3). Jednotlivá čísla reprezentují specifické panely.

Pod číslem 1 se skrývá *GraphicsPanel*, který slouží ke kreslení. Číslo 2 reprezentuje *PanelPlatnoHodnoty*. Zde se nachází prostor pro vstupní data od uživatele. Uživatel má možnost naměřené hodnoty zapsat přímo do textových polí nebo je myší označit na *GraphicsPanelu*. Na panelu plátno je využito metody *setVisible*. Tuto metodu má každá komponenta z knihovny swing. *SetVisible* slouží k zobrazení nebo skrytí dané komponenty. Jednoduše do argumentu metody zadáme *true*, pro zobrazení komponenty a *false* pro skrytí. Na obrázku 10 je mnoho komponent skryto, protože je defaultně nastaveno, že uživatel bude pracovat pouze s naměřenými hodnotami stran (nikoliv úhlu) a počet naměřených bodů je na hodnotě 3. Pokud uživatel zvolí v panelu *RozsireneMoznosti* (číslo 5 na obrázku 10) jinou hodnotu z comboboxu *pocetBodu*, zobrazí se mu více či méně textových polí na panelu *PanelPlatnoHodnoty*. V případě, že vybere možnost *měření 2 stran a 1 úhlu* zaškrtnutím příslušného radiobuttonu. Na panelu *PanelPlatnoHodnoty* se zobrazí pouze textové pole pro dva body (čili 6 textových polí) a navíc se zobrazí možnost pro zadání úhlu (viz Obrázek 11). Tlačítko s otazníkem je zde

pro nápovědu. Pro uživatele se zobrazí jednoduché dialogové okno s nápovědou k zadávání úhlu.

Obrázek 11 - Změna panelu při volbě úhlu. Zdroj: vlastní.

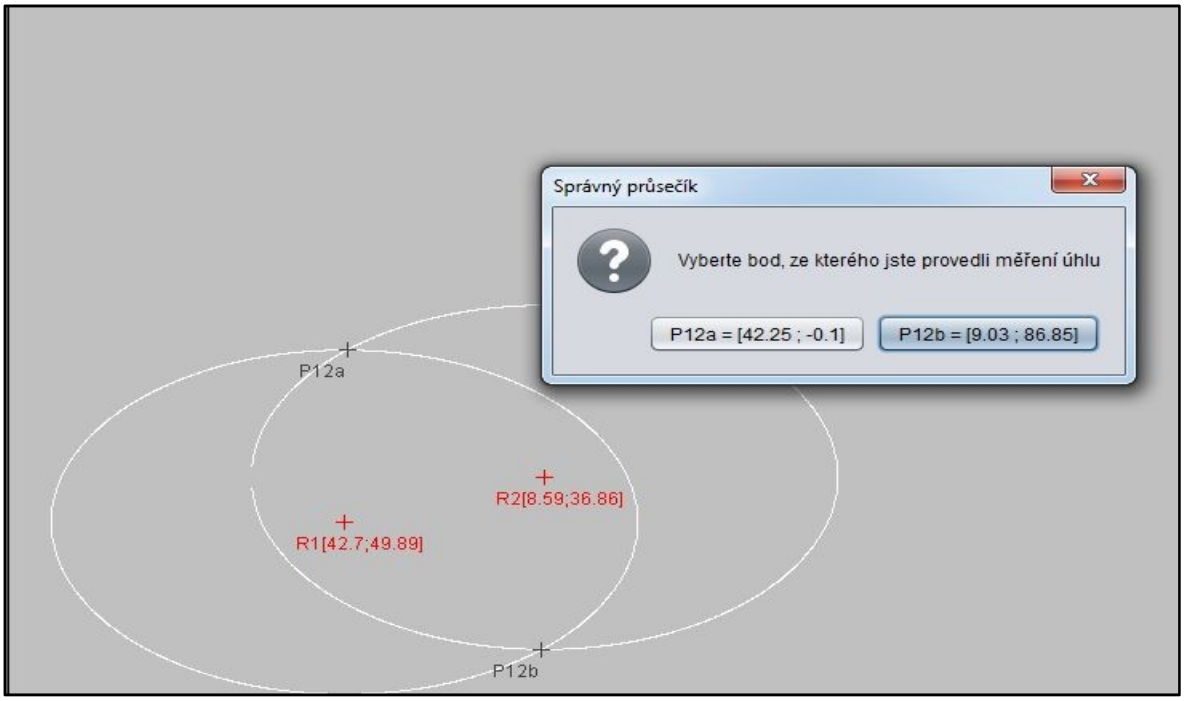
Panel označen číslem 3 reprezentuje *PanelMoznosti*. Na tomto panelu je možné nastavit směrodatnou odchylku na libovolnou (číselnou) hodnotu. Po stisknutí tlačítka *Nastav odchylku* se provede potřebná změna díky metodám *GraphicsPanelu* (konkrétně jsou volány metody *setSmOdStr* (nastav směrodatnou odchylku měření strany) a analogicky *setSmOdUhl* (pro směrodatnou odchylku měření úhlu). Tlačítko *konec* slouží k ukončení programu a tlačítko *Modelová úloha* nabízí možnost načíst modelové úlohy řešené v kapitole 2. Pokud je zaškrtnut radiobutton pro *měření stran*, je načtena úloha č.1, pokud označí uživatel radiobutton *měření 2 stran a 1 úhlu*, načte se úloha č.2. Pod načtením se rozumí, že se nastaví souřadný systém *GraphicsPanelu* na definované hodnoty, dále jsou vloženy *Body* do *ArrayListu seznamBodu*, který uchovává data od uživatele, nastaví se počet bodů, se kterými bude *GraphicsPanel* pracovat, a směrodatné odchylky. Poté pokud uživatel stiskne tlačítko *Vypocitat*, bude proveden kompletní výpočet potřebných hodnot. Na panelu *panelRozsireneMoznosti* (č.5 na Obrázku 10) si může uživatel zvolit, jaké hodnoty se mají vykreslovat. Po provedení činností, po stisknutí tlačítka *Vypocitat*, jsou totiž veškeré hodnoty spočítány, ale nejsou vykresleny. *GraphicsPanel* vykresluje defaultně zadané body a kružnice o naměřených hodnotách. Je zde však možnost zaškrtnutí jednoho či více checkboxů z panelu *panelRozsireneMoznosti*. Každý checkbox má nastavený listener (posluchač), ohledně změny stavu. To znamená, že pokud se provede zaškrtnutí libovolného checkboxu, je vyvolána příslušná metoda, která zavolá metodu *repaint. Repaint* (neboli překreslení) se postará o nové volání metody *paint* ze třídy *GraphicsPanel*. V metodě *paint* se poté testuje, zda je libovolný checkbox zaškrtnut. V případě, že je zaškrtnut checkbox průsečíky se vykreslí spočítané průsečíky daných kružnic, když se zaškrtne checkbox konfidenční elipsa, je volána metoda *paint*, která se vyskytuje ve třídě konfidenční elipsa a následně je vykreslena. Takto metoda *paint* pracuje se všemi checkboxy použitými v aplikaci.

Panel *panelSouradniceAktualni* (č.4 na obrázku 10) slouží k zobrazení aktuální pozice (souřadnice) myši. Jsou zde dvě hodnoty. Souřadnice SSZ a souřadnice USS. Souřadnice SSZ slouží ke zobrazení souřadného systému zařízení. Je to tedy přesná hodnota v pixelech (obrázkových bodech). Oproti tomu souřadnice USS reprezentuje Uživatelský souřadný

systém, tedy uživatelem definovaný systém s počátkem v levém dolním rohu. Souřadné systémy byli vysvětleny v kapitole 3.6.2. Důležité je, že ke zobrazení je využito události *MouseMoved*. To znamená, že při vytvoření činnosti reagující na tuto událost (v návrháři vybereme, že chceme odchytit tuto událost) se nám vytvoří metoda *formMouseMoved* s parametrem *evt*. Parametr *evt* obsahuje metody *getX* a *getY* pro zachycení hodnoty dané souřadnice. Poté se tato souřadnice (SSZ) vypíše právě do panelu *panelSouradniceAktualni* a po přepočtu i souřadnice USS.

Poslední panel, označen číslem 6, slouží ke změně souřadného systému. Uživatel má možnost vyplnit čtveřici textových polí, pro změnu příslušných hodnot. Souřadnice *Min x* a *Min y* se nachází v levém dolním rohu plátna, oproti tomu souřadnice *Max x* a *Max y* leží v pravém horním rohu plátna. Hodnoty lze psát jako celá i jako desetinná čísla. Pokud se v textovém poli vyskytne například písmeno, čili zadaná hodnota je špatná, objeví se uživateli dialogové okno, které ho informuje o skutečnosti, že jedna ze zadaných hodnot (mohou to být samozřejmě všechny) je zadána špatně. Změny se nevykonají a tak není možné program nějakým způsobem narušit. Změny se totiž provedou jen v případě, že se v textových polích vyskytují validní hodnoty. Tento postup je proveden i u textových polí pro směrodatné odchylky, úhel, i pro všechna textová pole reprezentující souřadnice a naměřené hodnoty z dané souřadnice. Právě tlačítka *Proveď změny*, *Nastavit odchylku* a *Nastavit* (u nastavení souřadného systému) obsahují bloky *try-catch* pro zachycení nechtěných vstupních dat z výše zmíněných textových polí.

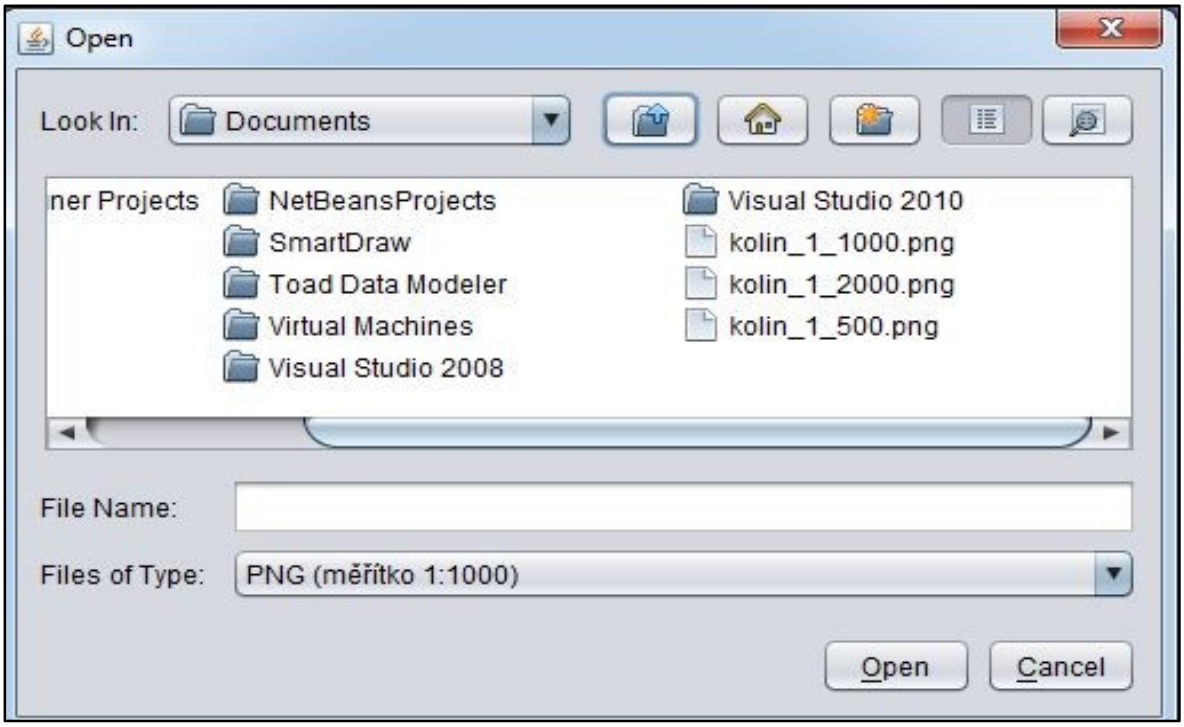
V kapitole č.2 bylo u druhé úlohy prohlášeno, že pokud se provádí měření pomocí dvou bodů a k nim příslušných dvou stran, třetí měřenou hodnotou je úhel. Pro zjištění průsečíku ale nešlo jako v první úloze provést výpočet přímo, ale zjistit od uživatele, z kterého místa přibližně prováděl měření. V tomto programu je k určení správného průsečíku použito dialogového okna (viz Obrázek 10). Uživateli se zobrazí oba možné průsečíky na grafickém panelu a také se mu zobrazí dialogové okno, ze kterého si vybere ten správný průsečík. Princip je následující. Oba průsečíky se spočítají stejně jako v úloze, kde se nevyskytuje úhel. Tam se pro určení dosazují výsledné hodnoty do podmínky z teoretického modelu. Ten průsečík, který má po dosazení do podmínky bližší hodnotu nule, je správný a vloží se do *ArrayListu* s průsečíky. V případě druhé úlohy, kde je nutné si vybrat správný průsečík, se oba průsečíky spočítají. Poté nejsou dosazeny do podmínky teoretického modelu, ale jsou zobrazeny v dialogovém okně a na grafickém panelu. Uživatel vybere správný průsečík, který se uloží do *ArrayListu*, druhý průsečík je smazán a dále se s ním nepracuje.



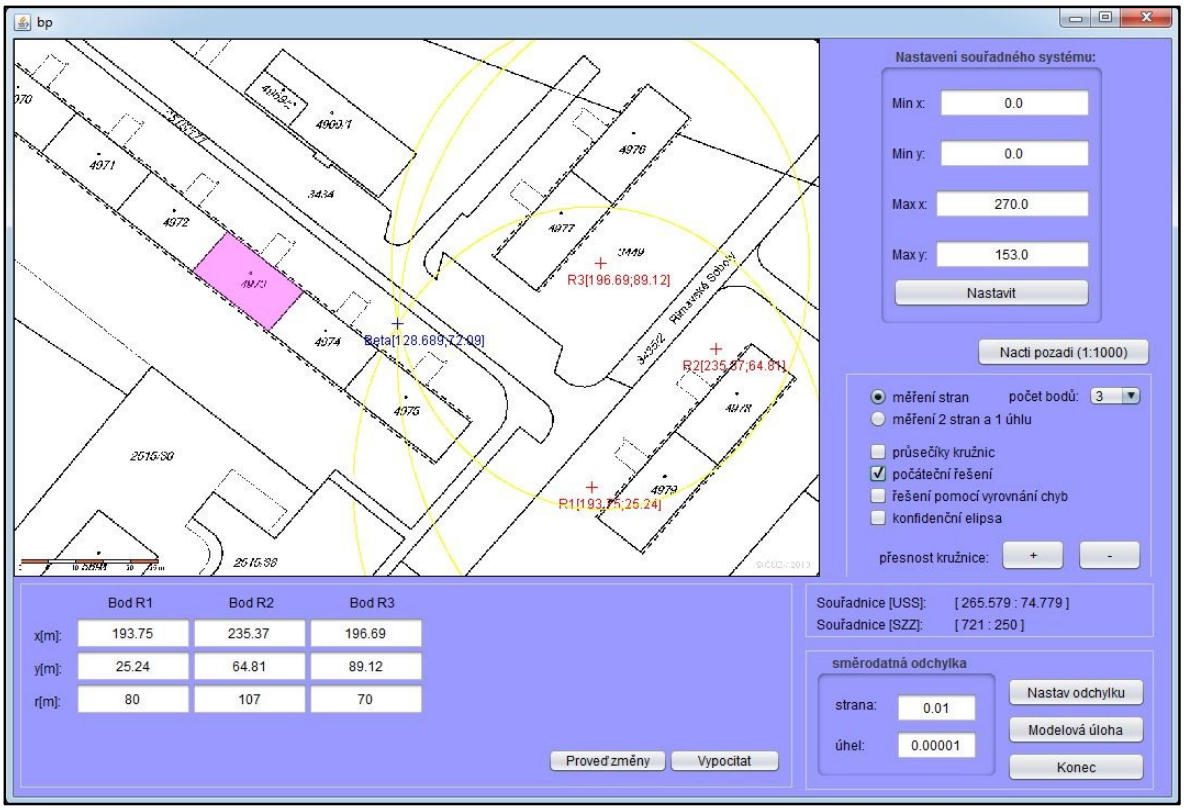
Obrázek 12 - Dialog pro výběr bodu ze kterého bylo provedeno měření. Zdroj: vlastní.

3.7.1 Načtení mapy

Při načítání mapy v měřítku 1:1000 je použit blok *try-catch*. Po stisknutí tlačítka *Načtení mapy (1:1000)* se objeví dialog, s možností výběru obrázku, který bude použit jako pozadí celého *GraphicsPanelu* (viz Obrázek 13). Na tomto obrázku je vidět právě dialogové okno s možností výběru obrázku pouze ve formátu .png. Tato konstrukce byla použita z důvodu, že mapy je možné stahovat ze stránky www.cuzk.cz (stránky českého úřadu zeměměřického a katastrálního). V případě, že by bylo možné získat kompletní mapové podklady České republiky, bylo by možné vytvořit program interaktivnější. V tomto případě se však jedná spíše o ukázku pro použití v praxi. Geodet nebo jiný uživatel aplikace by si načetl libovolnou mapu a v ní by prováděl měření vzhledem ke své poloze (vytvořil by si vlastně svůj souřadný systém). Mapy na stánkách Českého úřadu zeměměřického a katastrálního jsou dostupné zdarma, a tak není sebemenší problém si libovolnou mapu stáhnout a použít v této aplikaci. Pokud je v měřítku 1:1000, souřadný systém, který se po načtení mapy nastaví, je ve správném formátu. Pro načtení jiného měřítka by si uživatel musel zjistit souřadný systém dané mapy a nastavit pomocí panelu pro nastavení souřadného systému.



Obrázek 13 - Načtení pozadí. Zdroj: vlastní.



Obrázek 14 - Ukázka práce s mapou. Zdroj: vlastní.

Na obrázku 14 je vidět, že práce při použití mapového podkladu (dostupného na www.cuzk.cz) není odlišná od použití bez mapového podkladu. Jediné rozdíly jsou v barevném provedení, kdy je zvolena jiná barva kružnice pro vykreslení naměřené hodnoty, a v zamezení možnosti přibližování a oddalování. V případě, že není načtena mapa, lze přibližovat a oddalovat plátno dle libosti. V případě načtení mapy se ale jedná o statickou mapu, ke které nejsou ani přiblížené, ani oddálené podklady. Nelze tedy mapu libovolně přibližovat z tohoto důvodu. Souřadný systém je nastaven s počátkem v bodě o souřadnici [0; 0] a maximem v bodě [270; 153]. Tyto hodnoty jsou opět převzaty ze stránek www.cuzk.cz pro mapy v měřítku 1:1000. Na obrázku 14 je vidět, že v případě zaškrtnutí checkboxu *počáteční řešení*, je zobrazena pouze souřadnice počátečního řešení. Nic jiného ze spočítaných hodnot se uživateli nezobrazí. I v tomto případě lze zadávat od tří do šesti bodů pro měření pouze délek stran nebo zadání dvou bodů a naměřeného úhlu.

3.7.2 Zhodnocení programu

Celý program provádí výpočty velmi rychle. Největším zpomalením je zvýšení přesnosti kružnice, což nastavuje přesnější výpočet jednotlivých bodů. Body kružnice jsou totiž vypočítávány pomocí dosazení do obecné rovnice kružnice a následně kreslením spojnice mezi jedním (nejprve prvním) a následujícím bodem. Výpočet se provádí v cyklu *for*, kde se nastaví počáteční hodnota x, na hodnotu středu kružnice-poloměr. Poté je v každém kroku pro dané x spočítána hodnota y. Zvýšení přesnosti má za následek snížení velikosti kroku (inkrementace x) v cyklu *for*. Při největší přesnosti se může stát, že vykreslení není okamžité, ale samotný výpočet tímto problémem netrpí. Opoždění programu tedy může být zapříčiněno složitějším grafickým výstupem. Při testování se nikdy nestalo, že by se program zasekl nebo nechtěně ukončil. Všechny možné vstupy od uživatele, jsou ošetřeny výjimkami. Výjimky jsou odchyceny a zpracovány, takže je nemožné, aby program pracoval s nevalidními daty.

4 Závěr

Cílem práce bylo vytvořit aplikaci, která by dokázala řešit základní geodetické výpočty. Aplikace měla zobrazovat grafický výstup, který poskytuje názorný příklad dané problematiky.

Cíle práce byli splněny ve všech bodech. V teoretické části byl přiblížen obecný úvod do geodézie, její stručná historie a matematický aparát vyrovnávací počet. Dále byly popsány a vysvětleny dvě úlohy z této oblasti, které sloužily k názorné ukázce celé problematiky. V praktické části byla vytvořena aplikace pro řešení zadaných úloh, konkrétně úloh popsaných v kapitole Řešení typických úloh. Aplikace umožňuje nastavit vstupní údaje od uživatele pomocí textových polí nebo přímým označením souřadnic na vykreslovacím plátně. Jsou zde možnosti pro nastavení libovolného souřadného systému, nastavení možností vykreslení, jenž usnadňuje orientaci ve vypočítaných hodnotách a načtení map, které jsou volně dostupné na stránkách Českého úřadu zeměměřického a katastrálního. Grafické uživatelské rozhraní bylo navrženo co nejjednodušeji, aby každý potenciální uživatel dokázal jednoduše a efektivně pracovat s touto aplikací.

Nezbytně nutnou částí tvorby aplikace bylo její testování. Veškeré chyby, které se objevily, byly ošetřeny. Zejména vstupní údaje z textových formulářů, u kterých je velké riziko vzniknutí nežádoucí chyby, jenž by mohla ovlivnit běh celé aplikace.

Pro budoucí rozvoj by bylo nezbytné, sehnat kompletní mapové podklady Českého úřadu zeměměřického a katastrálního, které jsou zpoplatněny. Pro běh aplikace by to zajistilo lepší funkcionalitu v praxi. Celkově by se dalo s programem pracovat rychleji a s kompletním mapovým podkladem by bylo možné udělat aplikaci interaktivnější v případě načtení rozsáhlejší mapy.

5 Literatura

1. **BOHM, Josef, RADOUCH, Vladimír a HAMPACHER, Miroslav. 1990.** *Teorie chyb a vyrovnávací počet*. Praha: Geodetický a kartografický podnik Praha, 1990, 416 s. ISBN 80-7011-056-2.

2. **VITÁSEK, Josef a NEVOSÁD, Zdeněk. 1999.** *Geodezie*. Brno: CERM, 87 s. ISBN 80-214-1152-X.

3. České stavby: portál o stavbě, zahradě a bydlení. [online] 25. 11. 2011. [cit. 2013-04-03]. Dostupné z: http://www.ceskestavby.cz/clanky/geodezie-ma-zajimavou-a-dobrodruznou-historii-20468.html

4. **ČADA, Václav.** Přednáškové texty z Geodézie. [online]17. 5. 2007. [cit. 2013-05-03].Dostupné z: http://gis.zcu.cz/studium/gen1/html/ch01s03.html

5. **BUREŠ, Jiří.** Converter: převody jednotek. [online] [cit. 2013-05-04].Dostupné z: http://www.converter.cz/prevody/historie-metr.htm

6. Zeměměřič: časopis a server o geodézii, katastru nemovitostí a kartografii. [online] [cit. 2013-09-04]. Dostupné z: http://www.zememeric.cz/filatelie.html

7. **SOUKENÍK, Marek.** Prevod.cz: převod fyzikálních jednotek. [online] 2000. [cit. 2013-09-04]. Dostupné z: http://www.prevod.cz/popis.php?str=240&parent=y

8. **ČUŘÍKOVÁ.** Jak jsme měřili obvod Země. *Metodický portál: Články* [online]. 12. 05. 2008.[cit. 2013-04-09]. Dostupné z: http://clanky.rvp.cz/clanek/c/ZFAAAA/2287/JAK-JSME-MERILI-OBVOD-ZEME.html. ISSN 1802-4785.

9. **Chlubný, Jiří a Bustová Milena.** Eratosthenés z Kyrény a měření zemského obvodu. *Antika: Články* [online]. 17. 05. 2007.[cit. 2013-04-09] Dostupné z: http://antika.avonet.cz/article.php?ID=4640

10. **WIMMER, Gejza, Rudolf PALENČÁR a Viktor WITKOVSKÝ**. *Stochastické modely merania*. Bratislava: Grafické štúdio Ing. Peter Juriga, 2001, 115 s. ISBN 80-968-4492-X.

11. **BOSÁK, Tomáš.** Úvod do programovacieho jazyka Java. [online] 24. 04. 2006.[cit. 2013 -04-10]. Dostupné z: http://programujte.com/clanek/2006041804-uvod-do-programovacieho-jazyka-java

12. **SEMECKÝ, Jiří.** Naučte se Javu - úvod. [online] 26. 04. 2002.[cit. 2013-04-10]. Dostupné z: http://interval.cz/clanky/naucte-se-javu-uvod

13. NetBeans IDE 7.3 Release Information. [online] [cit. 2013-04-11]. Dostupné z: https://netbeans.org/community/releases/73/index.html

14. **ROBINSON, Andrew**. *Jak se měří svět: příběhy z dějin měření*. Praha: Euromedia Group - Knižní klub, 2008, 224 s. Universum (Knižní klub). ISBN 978-80-242-2187-8.

15. JAMA: A Java Matrix Package. [online] [cit. 2013-04-24]. Dostupné z: http://math.nist.gov/javanumerics/jama

16. **KVAPIL, Jiří**. *Kosmický segment GPS a jeho budoucnost*. [online] [cit. 2013-04-30]. Dostupné z: http://www.aldebaran.cz/bulletin/2005_02_gps.php