TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 Informační technologie
Studijní obor: 1802R007/90-1 Informační technologie

Interaktivní aplikace pro Matlab určená pro tvorbu vzorových geofyzikálních dat

An interactive application in Matlab intended to create samples of the geophysical data

Bakalářská práce

Autor: **Artem Slizkov**
Vedoucí práce: Ing. Lenka Kosková-Třísková

V Liberci 15. 5. 2013

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Abstrakt: Cílem práce je vytvořit interaktivní aplikaci, která je určena pro tvorbu vzorových geofyzikálních dat. Účelem vytvořeného systému je zrychlení fáze modelování dat pro výzkum a možnost jeho automatizace. Zásadou k vypracování aplikace je seznámení s geofyzikálními metodami. Na základě literatury jsou definovány vztahy pro geometrické anomálie, které jsou implementovány jako funkce v Matlabu. Systém je rozdělen podle návrhového vzoru Model-View-Controller. Program využívá výhody jazyka UML pro vytvoření návrhu aplikace.

Software umožňuje modelovat účinky anomálie na určité fyzikální pole, ukládání a načítání dat, detailní postup je uveden v technické zprávě. Grafické rozhraní je vytvořeno v Matlabu GUIDE. Aplikace je rozšiřitelná, proto lze do aplikační logiky přidat nové geofyzikální metody a geometrické tvary pro již existující metody. Návod pro přidávání nových metod je popsán v práci. Software je doplněn o interaktivní část umožňující uživateli pohodlné zadání parametrů pro modelování geofyzikálních anomálií na povrchu.

Klíčová slova: modelování anomálii, vzorová geofyzikální data, geofyzikální metody, aplikace, MVC, UML.

Abstract: The aim of the work is to create an interactive application. The application software is intended to create samples of the geophysical data. The purpose of the system is to speed up the modeling phase of the data for research and the possibility of its automation. A principle for the development of the application is an introduction to geophysical methods. There are defined the relationships between geometric anomalies, based on the literature, which are implemented as the functions in Matlab. The system is divided according to the design pattern Model-View-Controller. The program uses advantages of the UML in order to create design of the system.

The software allows to model the physical effects of anomalies, store and retrieve the data. The detailed procedure is described in the technical documentation. The graphical interface is created in Matlab GUIDE. The application is extensible, so it is possible to add new geophysical methods and geometric shapes to the application logic. The instructions for adding of the new methods are described in the work. The software is complemented by the interactive section, which allows to set parameters for the modeling of geophysical anomalies on the surface.

Keywords: modeling of the anomaly, samples of the geophysical data, geophysical methods, application software, MVC, UML.

Obsah

Úvod

Cílem práce je vytvořit interaktivní aplikaci, která generuje modelová data pro geofyzikální výzkum. Výsledná aplikace je do budoucna rozšiřitelná. To se týká přidaní nových metod pro výpočet geofyzikálních anomálií. Účelem vytvořeného systému je zrychlení fáze modelování vzorových dat pro výzkum, možnost automatizace procesu vypočtu modelů anomálií, které jsou generovány s využitím geofyzikálních metod.

Software je napsaný v jazyku a prostředí Matlab. Aplikace navazuje na výzkumnou část, která je také napsaná v Matlabu. Cílem celé výzkumné části je najít, implementovat, ověřit a do praxe aplikovat algoritmy pro automatické detekce anomálií v geofyzikálním průzkumu.

Aplikace využívá výhody objektového programování a návrhové vzory, protože celý návrh bude do budoucna použitý při přechodu z fáze výzkumu do finální verze. Po ukončení výzkumu bude celá výzkumná část napsána v jazyku Java jako samostatný software a také fungovat na serveru.

Na základě literatury jsou definovány vztahy pro anomálie, které následně implementovány jako funkce v Matlabu. Aplikace pro začátek implementuje gravimetrickou a elektrickou metodu geofyzikálního průzkumu. Uživatel má možnost přidat novou metodu do programu bez změny logiky a již fungujících částí aplikace. Hotové funkce jsou uživateli zpřístupněny s pomocí uživatelského rozhraní navrženého v Matlabu GUIDE.

Software je doplněný o interaktivní část umožňující uživateli pohodlné zadání parametrů pro modelování geofyzikálních anomálií na povrchu, čímž dovoluje uživateli provádět rychlé modelování bez ztráty času na ruční přípravu modelů geofyzikálních polí. Aplikace má možnost ukládat a načítat generovaná data.

1 Teoretický úvod do návrhu implementace aplikace pro geofyzikální průzkum

1.1 Úvod do geofyzikálních procesů

Geofyzika je hraniční vědní obor, který leží na hranici mezi geologií a fyziky. Má souvislosti se seismologií, geotermikou a ostatními geologickými vědními disciplínami, jejichž části využívá nebo doplňuje. Informace je převzatá z knihy [1].

Užitá geofyzika je aplikovaná věda, která se zabývá studiem fyzikálních polí zemského tělesa, pokud to má význam pro řešení geologické stavby zemské kůry a svrchního pláště, pro vyhledávání ložisek surovin nebo pro jinou geologickou činnost. Užitá geofyzika se dělí na jednotlivé metody dle fyzikálního pole, které je prozkoumané. Tíhové pole země měříme a zkoumáme gravimetrickými metodami, magnetické pole magnetometrickými metodami, geoelektrické pole geoelektrickými metodami, pole elastických vln seismickými metodami, radioaktivní pole radiometrickými metodami a metodami jaderné geofyziky, tepelné pole geotermickými metodami.

Geofyzikální anomálie je projev anomálního chování tělesa na zkoumaném povrchu při použití geofyzikálních metod. Anomálie při výzkumu určitého fyzikálního pole má odlišné fyzikální vlastnosti vůči okolnímu prostředí.

Při výzkumu a zpracování geofyzikálních dat používáme pojem přímá a obracená úloha. Přímá úloha znamená hledaní účinku anomálního tělesa v odpovídajícím fyzikálním poli. v daném případě jsou známy velikost, tvar, hloubka, pozice, fyzikální vlastnosti rušivého tělesa. Obrácená úloha řeší hledání odpovídajícího rušivého objektu ve fyzikálním poli podle anomálního chování, jímž daný objekt působí na zkoumaný povrch. Tato úloha není většinou jednoznačná, lze ji však jednoznačně určit kombinováním různých geofyzikálních metod.

Geofyzikální metody lze použít podle různých podmínek a způsobů měření. Každá varianta má své zvláštní vlastnosti. Příkladem jsou měření letecké, pozemní, vrtné, pod hladinou vody.

V geofyzice jsou používané pojmy trojrozměrná a dvojrozměrná tělesa. Trojrozměrné má ve směrech souřadných os x, y, z konečné rozměry. Na rozdíl od

trojrozměrných těles, dvojrozměrná mají ve směru vodorovné osy y nekonečný rozměr, proto při popisu lze použit pouze dva rozměry. Do dvourozměrných objektů lze zařadit objekty, jejich rozměry ve směru osy y jsou třikrát až pětkrát větší než hloubka.

1.1.1 Gravimetrické metody

Gravimetrie je vědní obor, který zkoumá tíhové pole Země na povrchu nebo v okolí jejího povrchu. Zemské tíhové pole dává představu o tvaru a rozměrech Země, rozložení hmot s rozdílnými hustotami v zemské kůře. Informace je převzatá z knihy [1].

Dva Newtonovy zákony jsou základem teorie gravimetrických metod. První je gravitační zákon (1), druhý pohybový zákon (2).

$$F' = \chi \frac{m m_1}{r^2}, \tag{1}$$

kde $\chi = (6{,}670 \pm 0{,}015) 10^{-11} N m^2 kg^{-2}$ je gravitační konstanta, F je síla, se kterou se přitahuji dvě tělesa o hmotnosti m a m_1 , r je jejich vzdálenost.

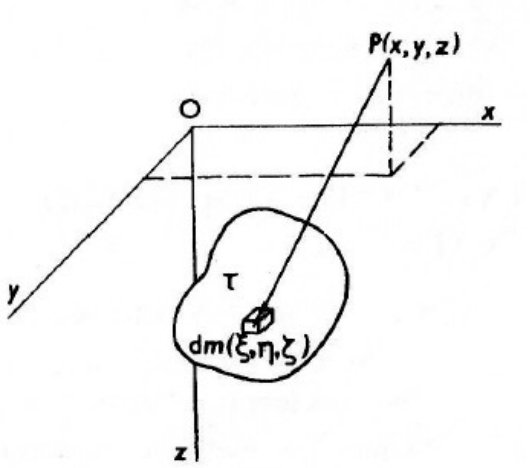
$$F' = m_1 g', \tag{2}$$

kde g' je gravitační zrychlení, které těleso působením síly o hmotnosti m_1 získá.

$$m = \varrho \tau, \tag{3}$$

kde m je hmotnost, τ je objem, ϱ je hustota.

Gravitační účinek libovolného tělesa záleží na jeho hustotě (3). Různé objekty mají různé hustoty, proto se jejich gravitační účinky liší (Obr. 1).



Obr. 1: k odvození tíhového účinku trojrozměrných těles. P(x, y, z) – výpočetní bod; $dm(\xi,\eta,\zeta)=\varrho\, d\tau$ *– hmotný element;* τ *– objem anomálního tělesa;* ϱ *– anomální hustota. Převzato z knihy[1] str.18*

Tíhové zrychlení g se skládá ze všech tíhových účinků hmot zemského tělesa. Geometrický tvar planety, nerovnosti povrchu, složení, geologická stavba způsobují rozdíl ve velikosti tíhového zrychlení v různých místech povrchu Země. v gravimetrii jsou nejdůležitějšími složkami nepravidelně rozložené hmoty uvnitř zemské kůry. Ostatní faktory způsobují šum, který musí být omezen, proto se v užité geofyzice používá rozdíl skutečného tíhového zrychlení a jeho vypočtené teoretické hodnoty.

Gravimetrické anomálie jsou způsobeny nehomogenní strukturou ve vrchních vrstvách planety, aby se dalo správně modelovat měření, jsou potřebné znalosti hustotních poměrů zkoumaného povrchu.

Interpretace anomálního pole je proces, při kterém objevujeme zdroje jednotlivých anomálií - účelem je dát takovým zdrojům význam. Jsou dva druhy interpretace: kvalitativní a kvantitativní.

Kvalitativní interpretace spočívá v tom, že anomálii lze popsat a geologicky vysvětlit, na základě zkušenosti, jaké anomálie může způsobit určitý objekt. Při kvantitativní interpretace zjišťujeme rozměry, hloubku, tvar, hmotnost objektu podle naměřených tíhových veličin. v posledním případě nejsou výsledky přesné, pouze přibližné, protože odvozené hodnoty spolu souvisí.

Pomocí kvalitativní interpretací lze definovat pojmy přímá a obrácená úloha gravimetrie. Pod pojmem přímá úloha se rozumí výpočet tíhového účinku modelu tělesa, při tom veličiny anomálie jsou známé (rozměr, tvar, hmotnost, hustota). Pro homogenní objekty typu koule, válec nebo deska se přímá úloha počítá přesnými metodami. Pro tělesa nepravidelného tvaru existují přibližné integrační postupy. Přímá úloha je jednoznačná.

Naopak cílem obracené úlohy je spočítat charakteristiky anomálního objektu (hloubku, polohu, tvar, rozměr) z obdržených měření tíhových účinků na povrchu. Tato úloha je mnohoznačná, protože stejné anomální tíhové pole může být způsobeno různými kombinacemi prostorového rozložení hmot a hustot anomálního tělesa. Obvykle při interpretaci lze předpokládat určitý tvar objektu, to ale nemusí být pravdivé.

Cílem kvantitativní gravimetrické interpretace je nalézt takové fyzikální modely anomálních objektů, aby naměřené tíhové pole anomálie Δg odpovídalo teoreticky vypočteným účinkům $V_z = \Delta g = \delta V / \delta z$.

Pro výpočet tíhového účinku V_z trojrozměrných těles platí vztah

$$V_z(x, y, z) = \chi\, q \iiint_{\xi\,\eta\,\zeta} \frac{(\zeta - z)}{r^3}\, d\xi\, d\eta\, d\zeta, \tag{4}$$

kde $r = [(\xi - x)^2 + (\eta - y)^2 + (\zeta - z)^2]^{1/2}$ (Obr. 1).

Odvozená rovnice pro modelování účinku tíhového pole anomálie, jež má geometrický tvar koule:

$$V_z(x, 0, 0) = \frac{\chi\, M\, h}{(x^2 + h^2)^{3/2}}, \tag{5}$$

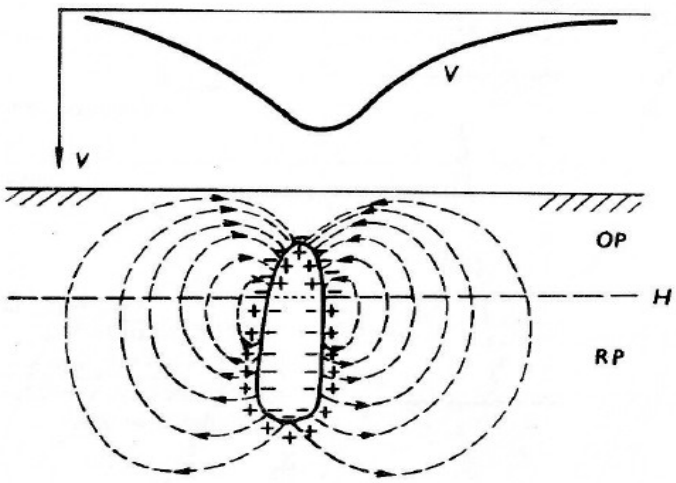
kde χ je gravitační konstanta, M je hmotnost tělesa, h je hloubka, x je vzdálenost od pozice na povrchu.

1.1.2 Metoda spontánní polarizace

Metoda spontánní polarizace patří do elektrochemických metod, které jsou založeny na využití spontánní nebo vyzvané polarizace přirozených vodičů. Anomálie v měřeném poli ukazuje existenci vodiče prvního řádu. Informace je převzatá z knihy [1] a [2].

Metoda je založená na měření a zkoumání přirozených elektrických polí na povrchu země, které samovolně vznikají v důsledku oxidačně-redukčních procesů nad anomáliemi. Do přirozených polí v daném případě patří pole elektrochemického, filtračního a difuzního původu. Podmínkou při vyhledávání a rozpoznání anomálních objektů je jejich elektronová vodivost. Největší význam mají pole elektrochemická, protože vznikají v okolí přirozených vodičů. Intenzita pole závisí na složení a struktuře vodiče a také na gradientu oxidačně redukčních vlastností okolního prostředí ve vertikálním směru.

U přirozených elektrochemických polí se jedno přibližně homogenní těleso nachází v elektrolytu, jehož vlastností se mění ve vertikálním směru. Změny elektrolytu jsou způsobeny přínosem kyslíku z atmosféry. Vody při zemském povrchu pak mají oxidační charakter, ve větších hloubkách redukční (Obr. 2).

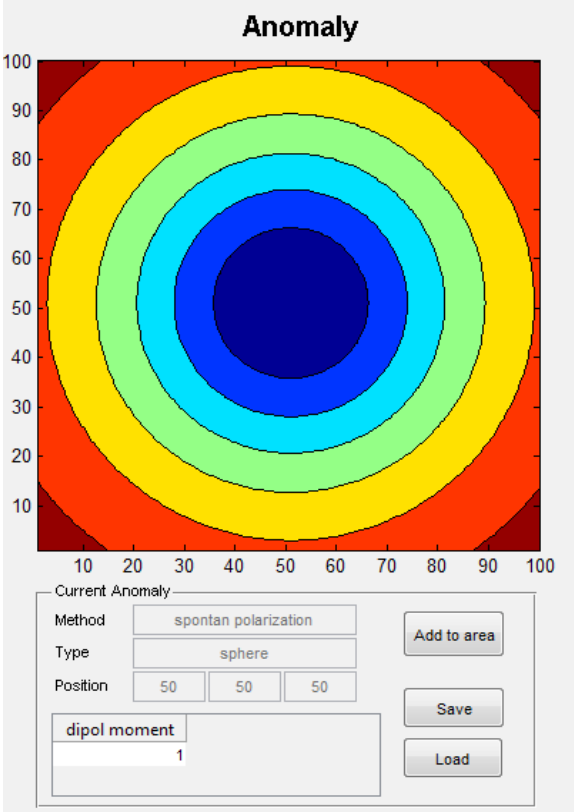


Obr. 2 Vznik přirozených elektrochemických polí v okolí vodiče. OP – oxidační prostředí, RP – redukční prostředí, H – hladina podzemní vody, v – průběh potenciálu na povrchu. Převzato z knihy[1] str.348

Předpokladem pro vznik polí je spojitá vodivost anomálního objektu, nejen přítomnost vodivých části. Důležitým faktorem je také změna ve složení elektrolytu, jenž zaplňuje horniny v okolí zkoumaného tělesa.

Filtrační pole se objevuje při filtraci podzemních vod v horninách. Projevuje se nejvíce v horách a blízko řek. Difuzní pole jsou v místě, kde se míchají podzemní vody s různou koncentrací solí a také s různým složením solí.

Teoretické základy metody spontánní polarizace záleží na povrchu vyhledávaných objektů. Změna potenciálu s hloubkou u isometrických těles je definovaná jako lineární, u objektů deskovitého tvaru lze očekávat nahromadění záporného náboje v horní části a kladného po zbývající části desky.



Obr. 3 Vygenerovaný potenciál na 3D povrchu nad polarizovanou koulí, P(x, y, z) = (50, 50, 50), moment dipólu se rovná 1

Rovnice polarizace nad koulí v pravoúhlých souřadnicích vypadá

$$V = M \frac{-z_0}{\left(x^2 + z_0^2\right)^{3/2}}, \tag{6}$$

kde M je moment dipólu polarizované koule, x je vzdálenost od pozice na povrchu, z_0 je hloubka (Obr. 3).

1.2 Návrhový vzor Model-View-Controller a jeho výhody při tvorbě a údržbě aplikace

Účelem mnoha počítačových systémů je oddělit logiku programu od grafického rozhraní. Důvodem k tomu je častá změna uživatelského rozhraní, přičemž logika

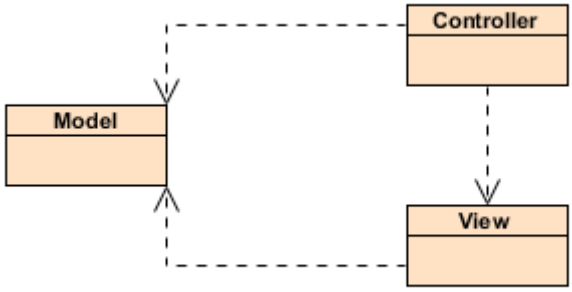
aplikace zůstává po dlouhou dobu stejná. Při přenosu systému na jiný počítač může být zapotřebí změna uživatelské části aplikace při zachování datové části beze změny. Použitá myšlenka je z [5]

Návrhový vzor Model-View-Controller dělí softwarový systém na tři vrstvy, které jsou navzájem oddělené (Obr. 4), což přináší výhody paralelního programování a také jednoduchost při provádění změn, bez potřeby přepisovat každou část nebo celý program. Proto se dají přidat nové funkce a nebo změnit uživatelské rozhraní bez nutného zásahu do celé struktury softwarového systému.

Model je částí návrhového vzoru MVC, která zodpovídá za chování dat v programu – je to aplikační logika. Model dostává požadavky na změnu (obvykle od řadiče) a proto nese zodpovědnost za uchování dat v konzistentním stavu. Po změně stavu předává informace na pohled.

Pohled (View) zodpovídá za zobrazování informací uživateli. Při pokynu ke změně, předá požadavek řadiči. Řadič kontroluje správnost dat obdržených od pohledu a předává dal modelu.

V návrhovém vzoru MVC jsou pohled a řadič závislé na modelu. Model není však provázaný s pohledem a řadičem. Takové rozdělení dovoluje spouštět a testovat model nezávisle od grafického rozhraní, což je jedna z klíčových výhod.



Obr. 4 Struktura tříd MVC

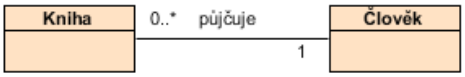
1.3 Jazyk UML pro vizualizace a návrh systému

V softwarovém inženýrství jazyk UML (the Unified Modeling Language) je specifikovaný jazyk pro modelování softwarových systémů. Při tvorbě a návrhu je

důležité konkretně chápat vize budoucího programu. Při velkém počtu lidí v týmu je nezbytné, aby návrhář a vývojář měli stejnou představu aplikace. UML umožňuje vytvořit model systému, tím usnadňuje její předávání. Přičemž se používá skupina symbolů a diagramů. Existují různé druhy diagramů, které slouží k různým účelům. Informace je použitá z knihy [3].

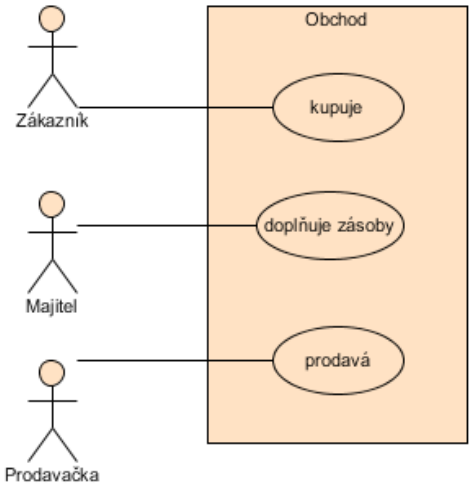
Jazyk UML se skládá z grafických prvků, které se kombinují do diagramů. Kombinace prvků se uskutečňuje podle pevně daných pravidel. Diagramy dovolují vytvořit několik pohledů na systém. Skupina pohledů definuje model, který ukazuje a popisuje co systém má dělat, nikoliv jak proběhne implementace.

Diagram tříd popisuje, jak rozdělit okolní svět na kategorie nebo skupiny, které mají podobné vlastnosti a podobné chování (Obr. 5). Účelem takového rozdělení je, aby software napodoboval nebo simuloval části reálného životu. Při komunikaci s klientem lze získat charakteristiky objektů a rozhodnout, jaké jsou důležité a které naopak se dají vynechat.



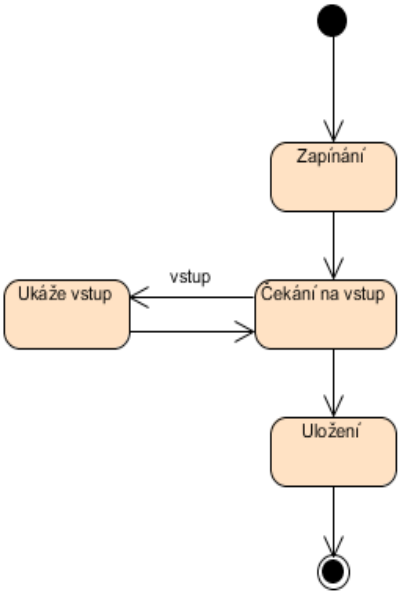
Obr. 5 Příklad diagramu tříd

Diagram případů užití (Obr. 6) specifikuje chování systému z pohledu uživatele (participant). Je to důležitý nástroj pro pochopení, jaké požadavky bude dostávat program od uživatelů, ale také od jiných systémů.



Obr. 6: Příklad diagramu případů užití.
Uživatele obchodu.

Diagram stavu (Obr. 7) ukazuje v jakém stavu bude se nacházet aplikace v určitý okamžik své aktivity.

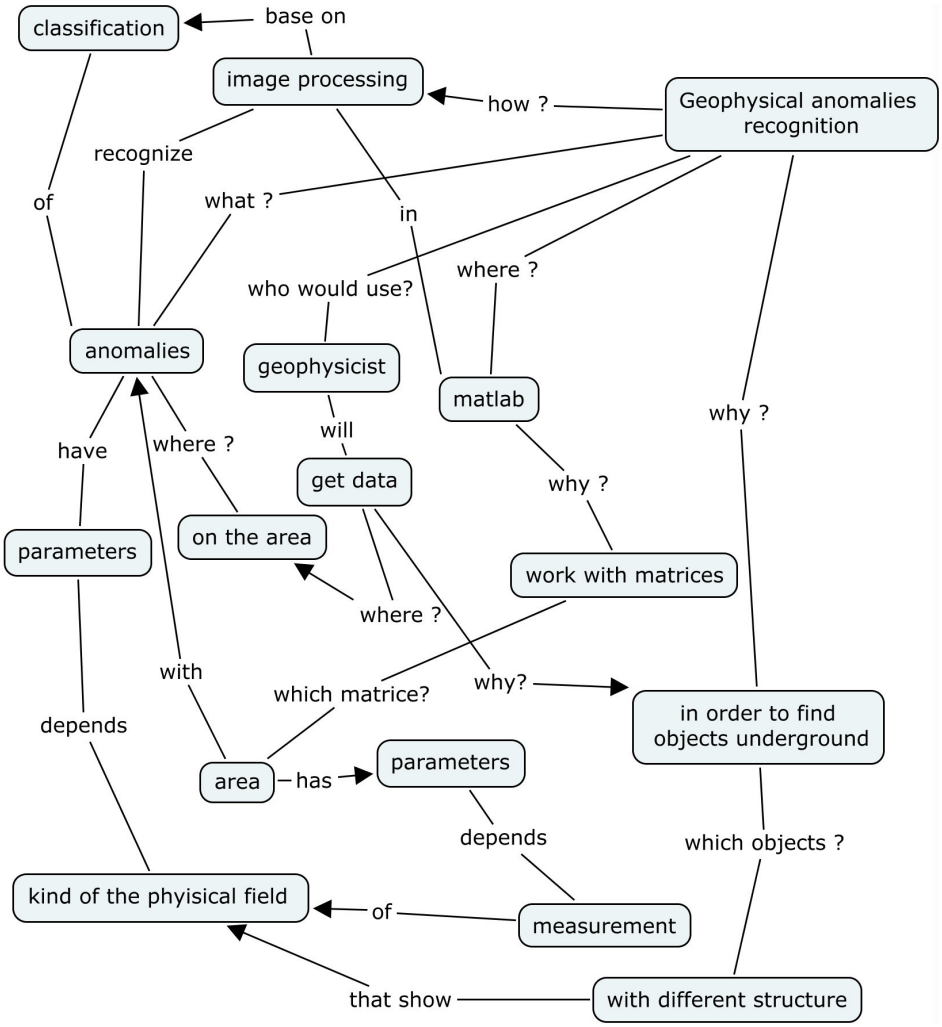


Obr. 7: Příklad diagramu stavu

2 Návrh aplikace pro modelování geofyzikálních procesů

2.1 Sjednocení zadání pomoci geofyziky, analýza, geofyzikální rovnice

Pro správný návrh aplikace, která modeluje geofyzikální metody, jsou nutné znalosti z geofyziky. z knihy [1] jsem zjistil základy metod a procesů, které se používají při průzkumu terénu a určují přítomnost či charakteristiky anomálie. v závislosti na metodě lze odvodit způsoby řešení přímé úlohy geofyziky. v přímé úloze se hledá účinek anomálního tělesa ve fyzikálním poli, jenž závisí na měření odpovídající fyzikální vlastnosti.



Obr. 8 Konceptuální mapa, která stručně popisuje chování systému

Pro modelování byly zvoleny metody gravimetrické a spontánní polarizace. Ze zadání také zní, že aplikace má být snadno rozšířitelná, proto je důležité najít společné body, které charakterizuje anomálie různých geofyzikálních metod.

Zapsal jsem stručně popis výpočtu veličin fyzikálního pole (Obr. 8). Výpočet závisí na metodě, která se používá, ale na druhou stranu všechny metody mají nějaké společné rysy. Vyčlenil jsem objekty a jejich vlastnosti, které jsou nezbytné pro vytvoření modelu povrchu s anomáliemi na něm přítomnými.

Stručný popis procesů, řešení přímé úlohy: měření fyzikálního pole se provádí na povrchu. Při měření se může vyskytnout anomálie. Pole obsahuje součet účinků každé anomálie, proto může být počet anomálních objektů větší než jedna. Každé těleso způsobující anomálii, má své zvláštní a stejné parametry. Parametry jsou určené druhem metody (hmotnost pro gravimetrii). Společná vlastnost pro každé anomální těleso je jeho poloha vůči povrchu.

Zkoumaný povrch a hledané objekty jsou na sobě navzájem nezávislé, povrch může mít větší množství anomálních objektů, ale nemusí. Anomálie může být zobrazená na různých profilech. To znamená, že profil a těleso jsou objekty, které lze vytvořit zvlášť od sebe a určitým způsobem je potom provázat. Proto řešení přímé úlohy bude spočívat v modelování fyzikálního pole povrchu, na který lze zobrazit různý počet anomálií, jejichž charakteristiky budou uvedeny v uživatelském rozhraní a uživatel je bude ručně zadávat nebo generovat náhodně.

Při modelování geometrických těles využitím gravimetrické metody jsem zvolil modelování jednoduchá tělesa tvaru koule a válec. Rovnice pro výpočet tíhového účinku takových těles lze najít v knize [1]. Rovnice koule potom vypadá (5), kde χ je gravitační konstanta, M je hmotnost tělesa, h je hloubka, x je vzdálenost od určité pozice na povrchu. z toho jsem zvolil vstupní parametry pro gravimetrickou metodu, typ tělesa – koule.

Modelování geoelektrických metod se liší parametricky od gravimetrie. Rovnice potenciálů nad polarizovanou kouli v pravoúhlých souřadnicích vypadá (6), kde M je moment dipólu polarizované koule, x je vzdálenost od určité pozice na povrchu, z_0 je hloubka. Parametry rovnice jsem zvolil jako vstup při modelování metody spontánní polarizace, typ tělesa – koule.

Pro vytvoření profilu, na který se potom zobrazí anomální objekty, jsem zvolil jako parametry délku, šířku, krok po délce a šířce. Takové vlastnosti mi dovolily vytvořit matici fyzikálního pole, do které se potom započítávají účinky modelovaných těles.

Výsledkem zkoumaní geofyziky je teoretický základ pro vytvoření diagramů tříd, případů užití, stavů v UML. Rozdělil jsem program také na moduly a udělal návrh uživatelského rozhraní z ohledem na informace zjištěné z knihy [1]. Studium geofyzikálních základů je nezbytné pro pochopení procesu modelování geofyzikálních dat.

2.2 Třídy a diagramy v jazyku UML

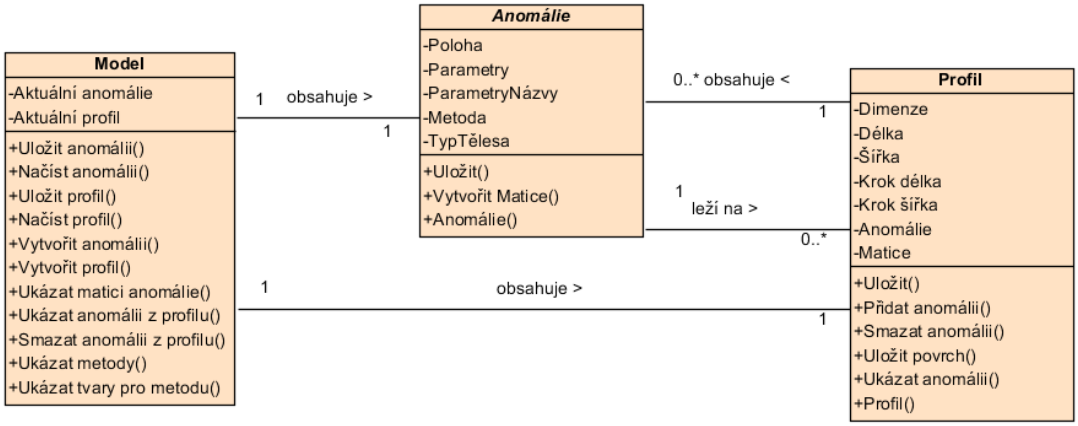
Vytvoření diagramů v jazyku UML pomáhají pochopit podstatu a funkce budoucí aplikace. Proto jsem provedl analýzu projektu. z geofyzikálního základu jsem vydělil nejdůležitější prvky a nepodstatné věci se snažil minimalizovat.

Abych vytvořil diagramy zopakoval jsem vlastnosti a cíle, které musí splňovat aplikace.

1. Modelovaní vzorových geofyzikálních dat. Takový proces patří do řešení přímé úlohy geofyziky, výpočtu fyzikálního pole v závislosti na profilu a anomálních objektů.
2. Aplikace musí být rozšiřitelná, proto přidání nové metody modelování musí být jednoduché a nezasahovat do již vytvořených částí programu.

Diagram tříd znázorňuje rozdělení modelu aplikace na třídy. z diagramu je vidět, že třída „Anomálie“ má proměnné, které jsou společné pro každou anomálii. Jsou to název geofyzikální metody, poloha v prostoru, parametry, podle kterých se počítá anomální účinek na daném profilu. v Matlabu je volání metod objektů mnohonásobně pomalejší než volání funkce (Tabulka 1).

Při manipulaci s velkými daty by program fungoval pomalu. Proto jsem se rozhodl volat metodu „Vytvoř Matice“ z funkce, která je definovaná pro každou geofyzikální metodu a tvar objektu zvlášť.



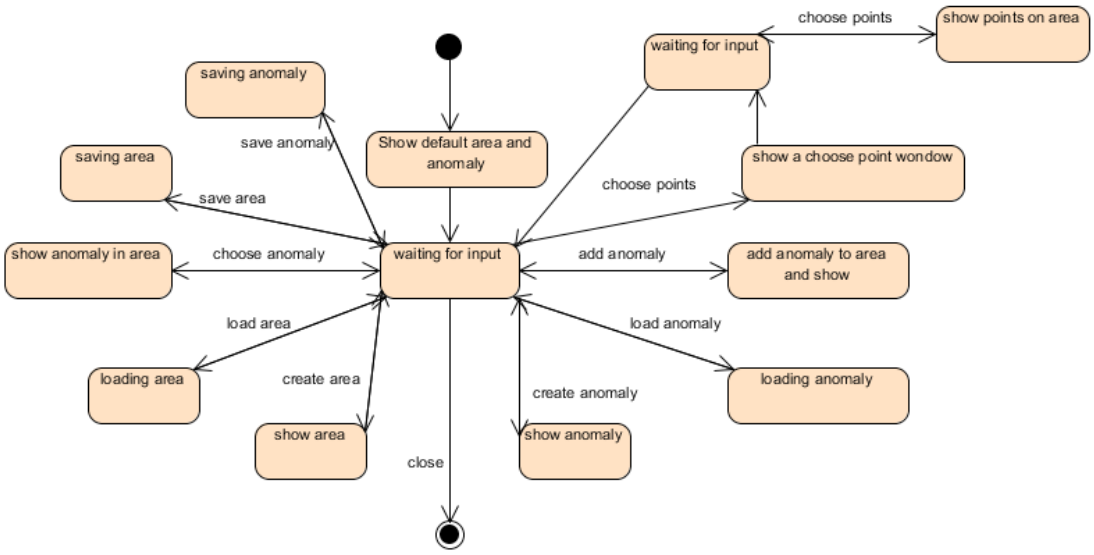
Obr. 9: Diagram tříd aplikace

Třída „Profil“ definuje povrch, na který budou jednotlivé anomálie přidány (Obr. 9). Podle zadání třídy také obsahuje matice každé anomálie zvlášť a celou matici povrchu se všemi tělesy. „Profil“ je definován délkou, šířkou a krokem v každém směru. Třída „Model“ znázorňuje, že najednou bude uživatel pracovat s jednou anomálií a povrchem. Lze také měnit instance objektů, uložit a načíst je.

Diagram způsobů užití ukazuje, jaké požadavky dostane aplikace od uživatele (Obr. 10). z diagramu je patrné, že uživatel může vytvořit nové, přidat do existujícího profilu, uložit nebo načíst anomálie. Profil může být také uložený, načtený a změněn. Uživatel vybírá parametry objektů, jenž následně budou vytvoření.

Obr. 10 Diagram způsobů užití

Diagram stavu (Obr. 11) ukazuje stav aplikace v každém úseku času. Po zapnutí programu uživatele se zobrazí předefinované objekty a jejich grafická reprezentace. Dál systém čeká na požadavky zvenčí, po jejich obdržení aktualizuje programová data a následně grafické rozhraní.



Obr. 11: Diagram stavu

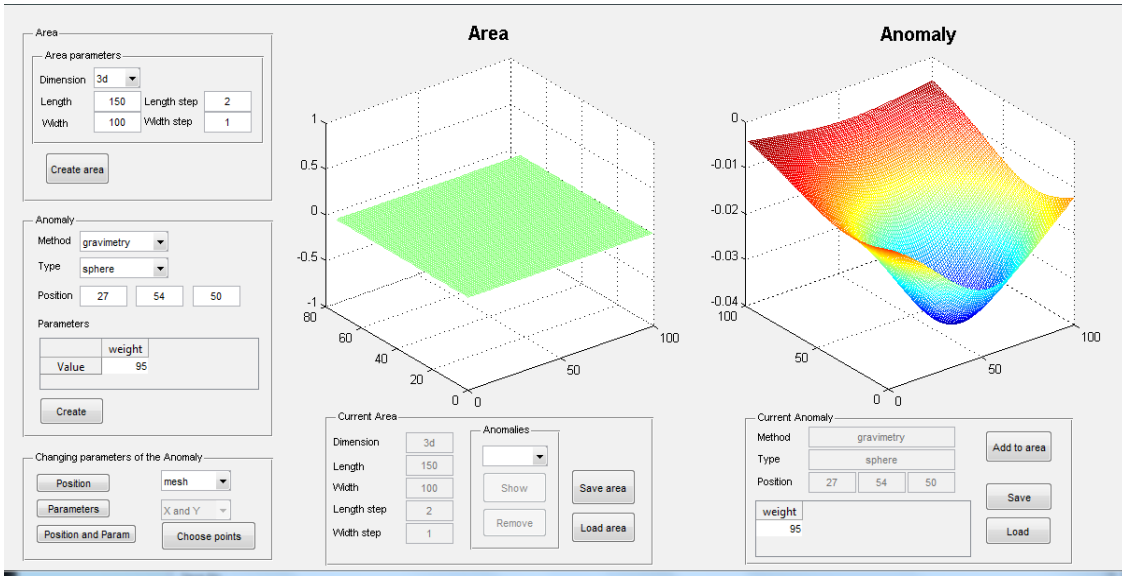
3 Implementace a popis využiti systému pro generování geofyzikálních anomálie

3.1 Rozdělení aplikace podle návrhového vzoru MVC

Zvolil jsem návrhový vzor MVC, abych řešil přenositelnost aplikační logiky pro různé grafické rozhraní. Do modelu jsem zahrnul všechno, co se týká modelování anomálií na profilu, jejich uložení na disk, načítání. Což mi dovolilo testovat aplikace bez grafických prvků.

Funkce modelu:

- vytvořit, uložit, načíst, změnit anomálie;
- vytvořit, uložit, načíst, změnit povrch;
- přidání aktuálního anomálního objektu do aktuálního povrchu;
- vrátit matice do modelovaného fyzikálního pole;
- vrátit matici účinků anomálního tělesa na aktuální povrch;
- vrátit nebo smazat anomálie z povrchu podle jeho pořadového čísla;
- vrátit parametry aktuální anomálie a jejich pojmenování;
- vrátit všechny geofyzikální metody, jež používá program;
- vrátit pojmenování geometrických tvarů podle zvolené metody.



Obr. 12 Hlavní okno aplikace

Podle zadání jsem vytvořil grafické rozhraní (Obr. 12) v GUIDE Matlab [4]. Uživateli jsou zobrazené aktuální modely anomálie a profilu. Vedle jsou parametry, které popisují dané objekty. Informace o objektech se nachází v datové části aplikace.

Uživatel může nastavit vlastnosti profilu anebo vybrat již existující z disku a načíst ho do aplikace, stejné platí pro vlastnosti modelované anomálií. Lze také vybrat možnost pseudonáhodné volby parametrů pro vzorové objekty. Implementoval jsem také zvláštní okno, ve kterém lze vybrat pozice bodů na modelu profilu pro určitý geometrický tvar a geofyzikální metodu.

V aplikaci jsem zařadil posluchače (listereners), které mění grafické rozhraní při změně proměnných v části modelu. Program je napsaný v jazyku a vývojovém prostředí Matlab, který obsahuje „obervable“ vlastnost proměn ve třídě, což dovolilo informovat pohled o změně dat v modelu. Informace o Matlabu lze najít je v [4].

V řadiči jsem provázal komunikaci mezi modelem a pohledem. Řadič řadí operace a požadavky, které dostává od grafického rozhraní a volá metody logické vrstvy na změnu dat.

Provedl jsem testování rychlosti objektového programování v Matlabu. Napsal jsem jednoduchou třídu, která obsahuje prázdnou metodu. Vytvořil jsem také prázdnou funkce. Při srovnání rychlosti metoda objektu byla přibližně 100 krát pomalejší než funkce (Tabulka 1). Proto při návrhu řešení jsem použil objektový přístup, ale časové náročné výpočty provádím ve funkcích.

Typ	100000 operace	1 operace
Prázdná funkce	0.02342 sec	0.23 usec
Prázdná metoda	0.24342 sec	2.43 usec

Tabulka 1: Popisuje rozdíl mezi rychlostí volání prázdné funkce a metody v Matlabu

3.2 Postup pro modelování anomálie na povrchu

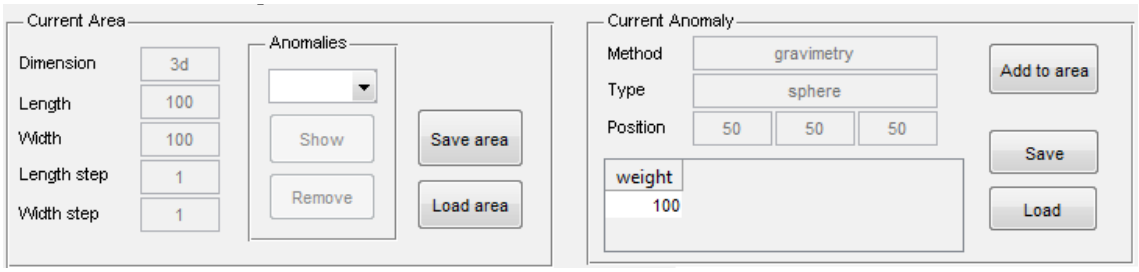
Při spuštění aplikace se uživateli objeví okno, ve kterém jsou zobrazené modele standardního profilu a anomálie. Parametry standardního modelovaného anomálního objektu jsou (Obr. 13):

- geofyzikální metoda: gravimetrie,

- geometrický tvar: koule,
- pozice na určitém povrchu: souřadnice po x – 50 , y – 50, z – 50 metrických jednotek,
- parametry pro zvolenou metodu: hmotnost modelované koule.

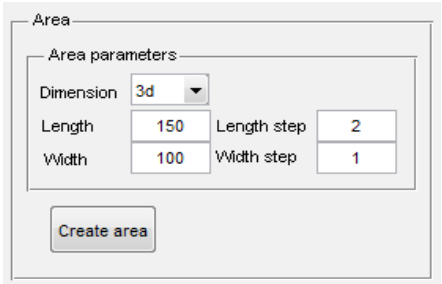
Parametry standardního profilu (Obr. 13), modelovaného při startu aplikace:

- dimenze modelovaného povrchu: 3D,
- délka a šířka: 100 metrických jednotek,
- krok po délce a šířce: 1 metrická jednotka,
- základní profil neobsahuje žádnou anomálii, proto seznam anomálních těles je prázdný.



Obr. 13: Parametry pro standardní profil a anomálii

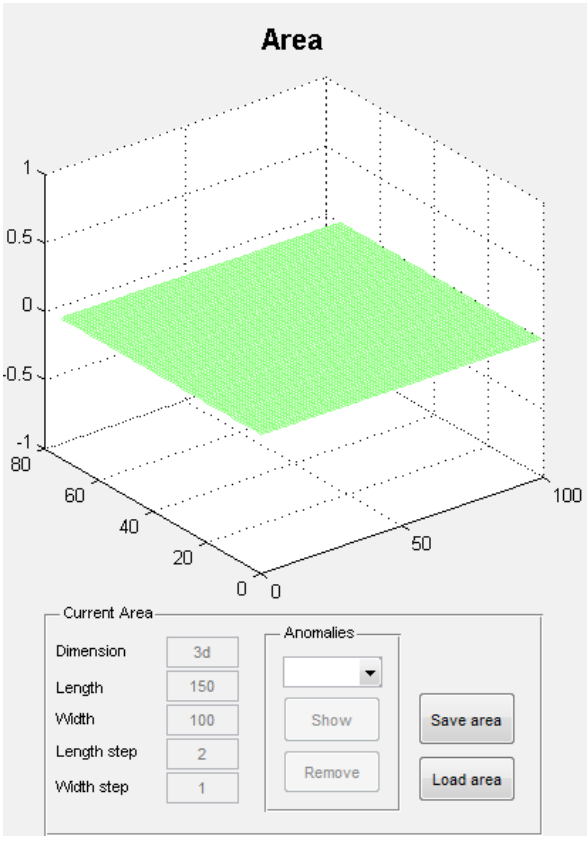
Pro vytvoření nového modelu povrchu jsem implementoval panel pro uživatelské vstupy. v panelu „Area parameters“ jsou ukázané parametry. Podle nich lze vytvořit nový profil (Obr. 14). Uživatel musí zvolit délku, šířku, rozměr, krok po šířce a délce, jenž určuje, kolik vzorků fyzikální hodnoty bude modelovaný povrch ukazovat.



Obr. 14: Panel pro vytvoření nového profilu

Při volbě dvourozměrného profilu, nelze zvolit šířku a krok po šířce. Pro modelovaní povrchu v 2D stačí pouze výběr délky a její vzorkovací hodnota, šířka je přitom nastavená automaticky programem a se rovná 1 (povrch je 2D křivkou).

Po stisku tlačítka „Create area“ program vymodeluje profil z parametrů, ukázaných v panelu „Area parameters“ (Obr. 15). Výsledný model se zobrazí v rámci „Area“. Zobrazení profilu závisí na tom, zda body vytvářejí křivku (2D) nebo plošnou matici (3D). Pod zobrazením modelovaného profilu je panel „Current area“, jenž obsahuje informace o typu a hodnotách aktuálního povrchu. To jsou hodnoty, které uživatel zadává při vytvoření modelu profilu.

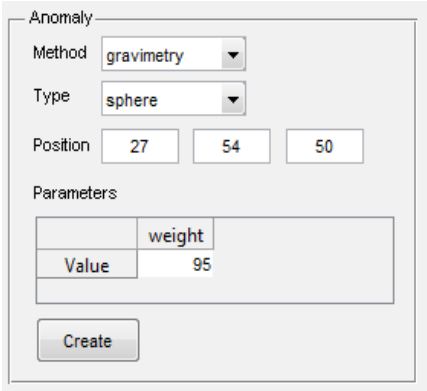


Obr. 15 Vygenerovaný profil podle uživatelských parametrů

Fáze modelování anomálního účinku na určité modely povrchu:

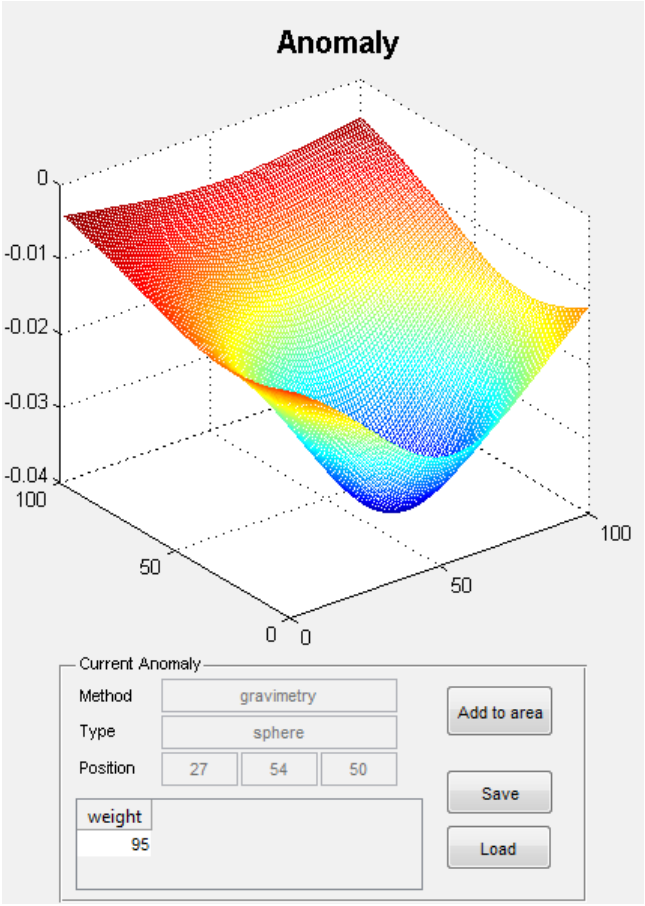
1. V panelu „Anomaly“ (Obr. 16) si uživatel zvolí geofyzikální metodu, jež určuje typ fyzikálního pole, které bude dál modelované. v programu podle zadání jsem naprogramoval gravimetrickou metodu a metodu spontánní polarizace. v dalších kapitolách bude vysvětleno, jak lze přidat nové geofyzikální metody nebo druhy geometrických tvarů.

2. Ke každé metodě je seznam geometrických tvarů, jejichž anomální účinky může program namodelovat. Je to povinná volba, protože určuje vlastnosti, které pří vytvoření vzorového objektu jsou nezbytné.
3. Další vlastnosti, které uživatel vybírá, je pozice anomálie na povrchu. Jsou to tři vstupy pro x, y, z souřadnice. Je-li pro modelování tělesa je potřeba více než jediného bodu, ostatní body se uvádí v dalších parametrech. Není-li pozice pro výpočet anomálního účinku není nutná, lze nechat standardní hodnoty.
4. Poslední informace, které musí být uživatelem uvedené, jsou hodnoty v tabulce „Parameters“. Jsou to veličiny, které jsou specifikované zvlášť pro každou geofyzikální metodu a geometrický tvar tělesa. Pro gravimetrickou metodu se sférickým tvarem tělesa je důležitou vlastností hmotnost tělesa, kterou lze zvolit. Dále jsou uvedené doplňující body pozice objektu, příkladem je tvar válce, jehož polohu lze popsat jedině dvěma body.



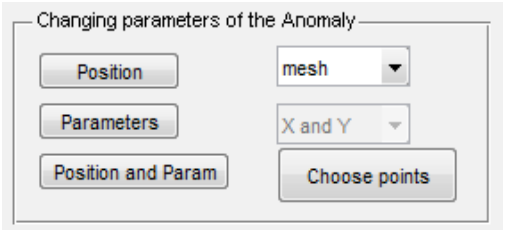
Obr. 16 Panel pro zadání parametrů anomálie

Při stisku tlačítka „Create“ se z parametrů vypočtou vlastnosti anomálie a v rámci „Anomaly“ (Obr. 17) se zobrazí model fyzikálního účinku objektu na aktuálním profilu, ukázaného v rámci „Area“. Pod rámcem se obnoví informace v panelu „Current anomaly“, kde jsou ukázané vlastnosti aktuální anomálie: metoda, tvar, pozice, parametry.



Obr. 17 Vygenerovaná anomálie podle uživatelských parametrů na aktuálním profilu

V panelu „Changing parameters of the Anomaly“ (Obr. 18) lze změnit modelované vlastnosti anomálie. Někdy je potřeba vygenerovat nový anomální objekt bez nutnosti uvádět všechny jeho parametry, proto jsem implementoval pseudonáhodný výběr hodnoty pro pozice a parametry.

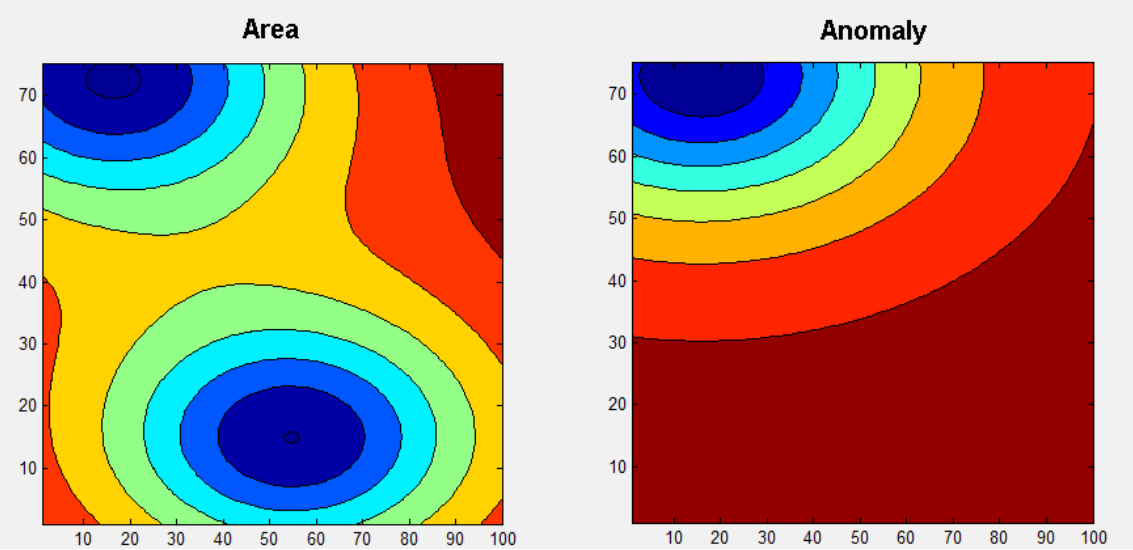


Obr. 18 Panel pro výběr parametrů anomálie

Tlačítko „Position“ mění pozice anomálie v rozsahu šířky a délky aktuálního profilu. Změny jsou vidět v panelu „Anomaly“ v části „Position“. Tlačítko „Parameters“ mění pseudonáhodně hodnoty v tabulce „Parameters“ v panelu „Anomaly“. Při stisku

tlačítka „Position and parameters“ je výsledek stejný, jako u tlačítka „Parameters“ a „Position“ dohromady, tj. se mění pozice a zároveň parametry pro generovaní anomálního tělesa.

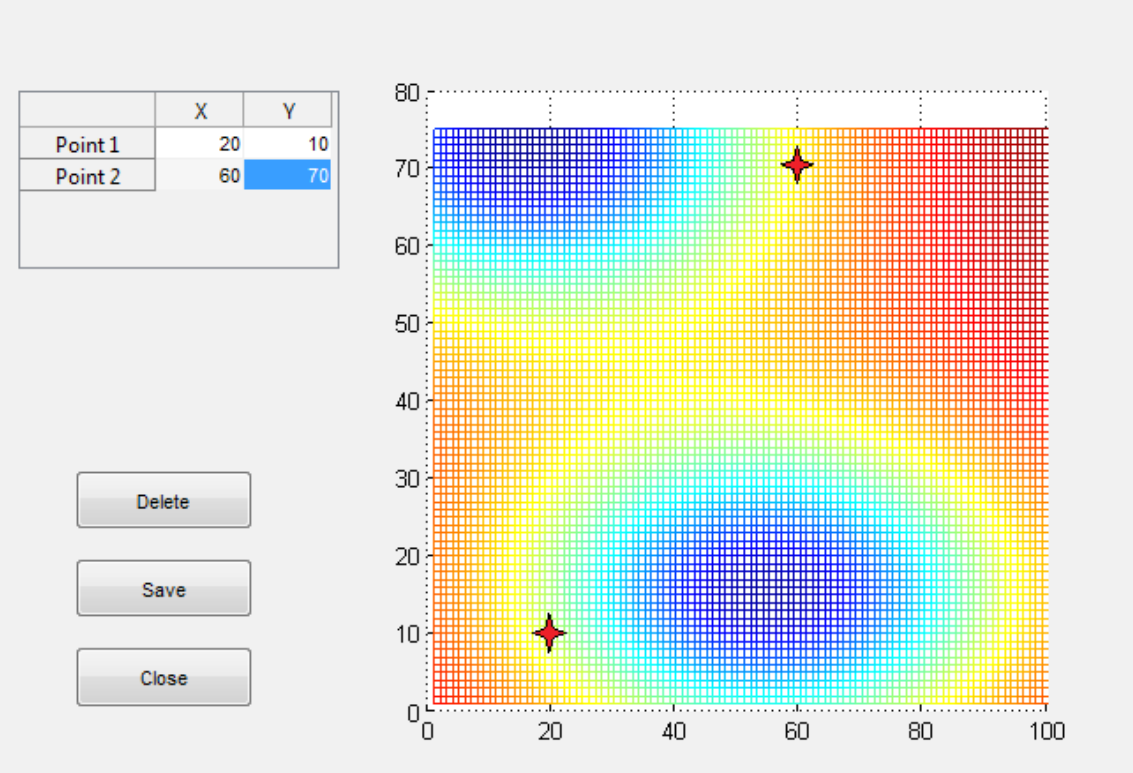
Panel „Changing parameters of the Anomaly“ také obsahuje výběr pohledu rámců „Area“ a „Anomaly“, které jsou aktuálně zobrazeny. Uživatel může vybrat jeden druh znázorňovaní profilu a anomálie ze třech: „mesh 3D“, „contourf“ (Obr. 19) a „mesh 2D“.



Obr. 19 Profil a anomálie jsou vykresleny pomocí „contourf“

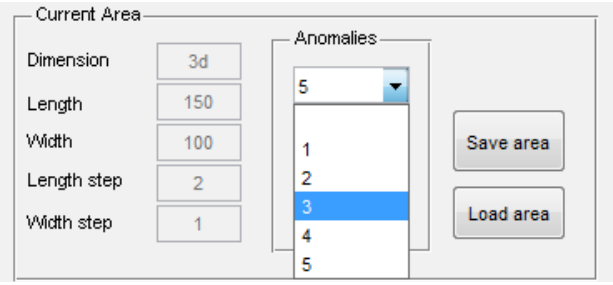
Při volbě způsobu vykreslování „mesh 2D“ a při stisku v rámci „Area“ uživatel může ručně nastavit x a y souřadnice v panelu „Anomaly“, položka „Position“.

Uživatel také může zvolil ruční nastavení bodů pozice anomálie. Tlačítko „Choose points“ otevře nové okno (Obr. 20), kde bude zobrazená matice povrchu. Po pravé straně je pak vidět všechny souřadnice bodů, které určují pozice modelovaného objektu. Klikáním v rámci „Area“ si uživatel může zvolit x a y souřadnice zvoleného bodu. Okno „Choose points“ obsahuje také tři tlačítka: „Delete“, „Save“, „Close“. „Delete“ smaže vybraný bod pozice, standardně je vybraný poslední. „Save“ zavře okno a předá zvolené souřadnice do panelu modelování anomálie v panelu „Anomaly“. „Close“ zavře okno „Choose points“ a neuloží výsledek výběru do hlavního okna.



Obr. 20 Okno pro výběr bodů na povrchu

Již vytvořenou anomálii lze vidět v rámci „Anomaly“. Ve skutečnosti do profilu anomálie zatím není přidaná. Pro přidaní anomálního účinku modelovaného tělesa do fyzikálního pole je potřeba stisknout tlačítko „Add to area“. v rámci „Area“ se zobrazí celkový model povrchu včetně všech dříve přidaných objektů.



Obr. 21 Aktuální profil, který obsahuje 5 anomálií

V panelu „Current Area“ lze vidět subpanel „Anomalies“, jež obsahuje seznam výběru a dvě tlačítka (Obr. 21). v seznamu se při každém procesu přidání anomálie na profil objeví nová položka. Je to pořadové číslo, které charakterizuje objekty. Každý objekt je instance třídy „Anomaly“. Celkový počet přítomných těles vyjadřuje počet anomálních těles, jejichž účinky se sečtou v matici fyzikálního pole v rámci „Area“.

Tlačítko „Show“ v subpanelu „Anomalies“ odpovídá za zobrazení vybraného objektu ze seznamu. Při stisku se poté objeví v rámci „Anomaly“ příslušná matice fyzikálního pole jednotlivého tělesa na aktuální profil. Také na panelu, jenž charakterizuje vlastnosti aktuální anomálii, se změní parametry.

Tlačítko „Remove“ smaže vybranou instanci třídy „Anomaly“ z aktuálního profilu. Přivede to k odečtení anomálního účinku od existující matice povrchu a následně ji obnoví v rámci „Area“. Při smazání libovolné položky seznamu se pořadová čísla aktualizují. Vybraná položka po odstranění je poslední anomální těleso na povrchu. v případě, že profil obsahuje jenom jediný anomální objekt, po odečtení anomálie z povrchu pak bude seznam prázdný .

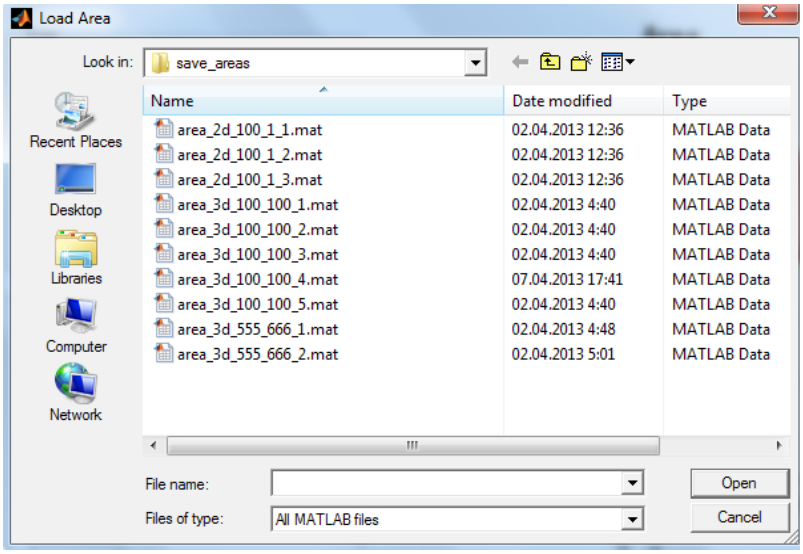
Čas na vytváření nové instance třídy „Area“ a výpočet odpovídajícího účinku anomálie je závislý na velikosti matice aktuálního profilu. Naprogramoval jsem algoritmus, který počítá fyzikální účinek pro každý prvek matice. Prvek je bod na povrchu, na kterém se modeluje možné terénní měření. Výpočet takového modelu je relevantně složitá operace. Časová složitost modelování je $O(n^2)$, kde n je počet prvků matice. Ten je dán uživatelem a rovná se součinu vektorů šířky a délky profilu s příslušným vzorkovacím krokem. Standardní profil v aplikaci má šířku a délku 100 metrických jednotek s krokem 1. Při procesu modelování na něm má výpočet anomálie časovou složitost $O(100\,000\,000)$.

Implementoval jsem algoritmu vkládání a mazaní anomálie na povrch. Jeho složitost je $O(n)$, kde n je počet prvku matice profilu. Při operaci vkládaní se do každého prvku matice se přečte anomální účinek vkládaného objektu. Naopak při operaci mazaní se ten účinek odečte. Při standardních nastaveních se časová náročnost algoritmu rovná $O(10\,000)$ (šířka a délka jsou rovné 100 metrickým jednotkám z krokem 1).

3.3 Ukládání a načítání dat

Aplikace umožňuje uživateli modelování vzorových geofyzikálních dat. Naprogramoval jsem také možnost načítání a ukládání dat, aby bylo možné modelované anomálie a profily nahrávat na disk.

Data jsou uložená ve formátu mat-souboru, který využívá Matlab (Obr. 22). Anomálie se ukládá jako objekt - má položky: metoda, tvar objektu, poloha, hodnoty parametrů a jejích textová reprezentace, ze které lze zjistit názvy odpovídajících hodnot v poli parametrů.



Obr. 22 Souborový prohlížeč. Je otevřená složka pro ukládání profilů

Soubor pro data charakterizující profil obsahují hodnoty délky, šířky, jejích vzorkovací periodu, matici fyzikálního pole. Také v uloženém modelu povrchu je pole pro anomálie, jejichž účinky má zobrazené na vzorové matici. Každé anomální těleso má poté příslušnou matici, ve které znázorňuje způsobené fyzikální odchylky na fyzikálním pole profilu.

Pro uložení aktuálního profilu v aplikaci je nutné vytvořit vzorový povrch podle postupu, který je vysvětlený v předchozí kapitole. Lze také uložit standardní profil, který generuje program při spuštění. Při stisku tlačítka „Save“, které se nachází v panelu „Current anomaly“, se otevře souborový prohlížeč. Podle standardního nastavení primární složka pro ukládaní povrchu je „save_areas“. Nachází se v subsložce „Model“, kde je uložený program.

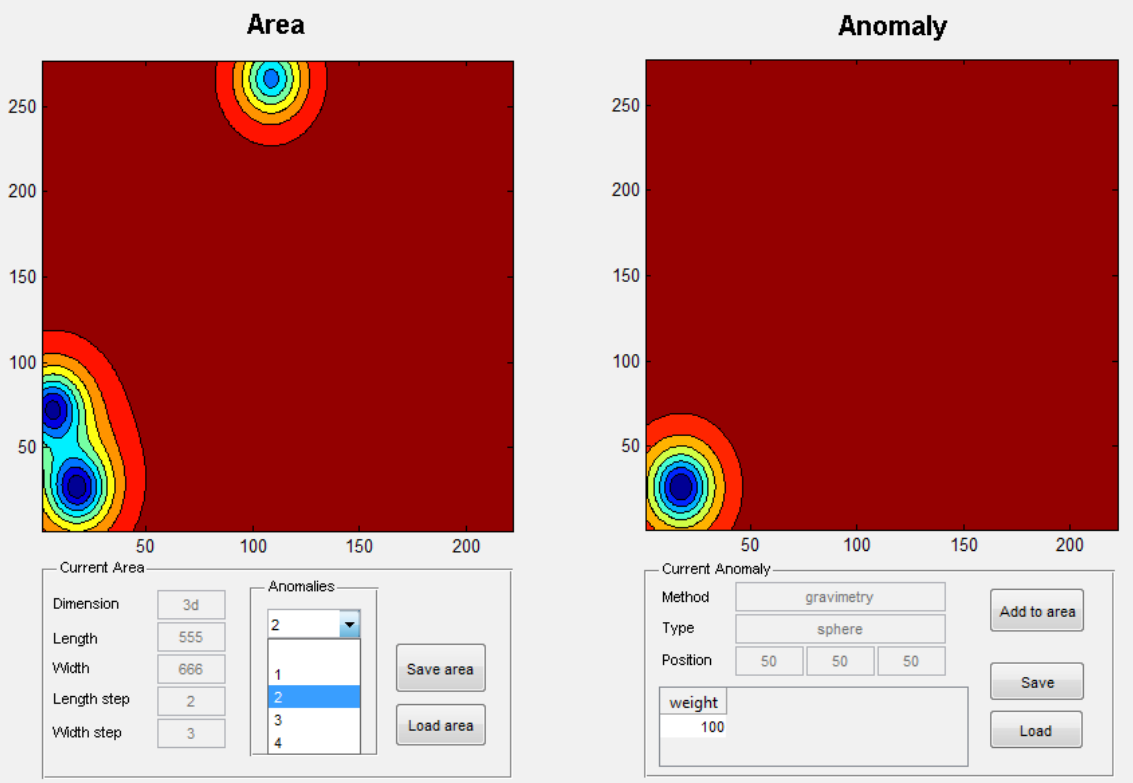
V souborovém prohlížeči si uživatel může zvolit název mat-souboru, do kterého následně uloží aktuální profil. Základní nastavení umožňuje generovaní názvů podle charakteru dat. Standardní povrch s parametry:

- délka a šířka: 100 metrických jednotek,

- vzorkování po délce a šířce: každá 1 metrická jednotka,
- dimenze: 3D,
- matice vzorového fyzikálního pole: 100x100,
- prázdné poli s anomáliemi,

program pojmenuje *„area_3d_100_100_1.mat“* (Obr. 22), kde *area* znamená profil, 3*d* je dimenze, *100* – délka a šířka profilu, *1* – pořadové číslo. Jestli název pro datový soubor již existuje, program inkrementuje pořadové číslo o jednu a název bude vypadat: *„area_3d_100_100_2.mat“*.

Načítání již existujícího profilu do aplikace se provádí pomoci tlačítka „Load area“, jež se nachází v panelu „Current area“. Po stisku se otevře souborový prohlížeč, základní složka pro uložené profily je „save_areas“. Dále si uživatel vybere potřebný mat-soubor a potvrdí výběr. Data se nahrají do aplikace a lze je spatřit v panelu „Current anomaly“ a v rámci „Area“, kde se zobrazí model příslušného profilu. Jestli uložený povrch má na sobě anomálie, tak uživatel je může podle potřeby vidět v subpanelu „Anomalies“ panelu „Current area“ (Obr. 23).



Obr. 23 Otevřený profil se zobrazí v rámci „Area“. Anomálie, které obsahuje, lze vybrat v subpanelu „Anomalies“ a zobrazit v rámci „Anomaly“

Modelované anomálie jsou uložené stejně jako povrch v mat-souborech. Po stisku tlačítka „Save“ v panelu „Current anomaly“ se otevře souborový prohlížeč. Standardní složka pro uložené anomálie je „save_anomalies“ a nachází se ve složce, kde je uložená aplikace, subsložka „Model“.

Uživatel si v okně souborového prohlížeči zvolí název souboru. Implementoval jsem funkci, která vytváří název odpovídající vlastnostem anomálního objektu. Například, pro standardní anomálie bude vytvořený název: *„anomaly_gravimetry_sphere_1“,* kde *anomaly* ukazuje, že datový soubor je určen pro uložení anomálie, *gravimetry* je geofyzikální metoda, *sphere* je geometrický tvar modelované anomálie, *1* je pořadové číslo. Při ukládaní anomálního tělesa, které má stejné parametry proměn *metoda* a *tvar,* program vygeneruje název a inkrementuje pořadové číslo: *„anomaly_gravimetry_sphere_2“*. Potvrzením názvu se anomálie uloží do zvolené složky.

Načítání existujícího anomálního objektu se provádí ve dvou etapách. První spočívá v otevření souborového prohlížeče a v případném ukládaní předchozí anomálie, protože při otevření nové anomálie do aplikace se předchozí instance třídy „Anomaly“ smaže. Po stisku tlačítka „Load“ v panelu „Current anomaly“ se objeví okno souborového prohlížeče. Podle základního nastavení je primární složka „save_anomalies“.

Druhá etapa je nalezení odpovídajícího datového mat-souboru, který uchovává data. Data, která se nahrají do aplikace, lze pohlédnout v panelu „Current anomaly“, kde se ukážou vlastnosti načtené anomálie. Poté lze danou anomálii přidat na aktuální povrch.

3.4 Přidávání nových metod modelování anomálie

Účelem výsledné aplikace je zrychlení fáze modelování geofyzikálních anomálií. Proto jsem implementoval způsob, jak lze do aplikace přidat nové geofyzikální metody pro modelování účinku anomálního tělesa.

Aplikace generuje jednoznačné řešení pro přímou úlohu geofyziky. Základní nastavení dovoluje vypočítat účinky anomálních těles pro gravimetrickou metodu a metodu spontánní polarizace. Tato kapitola obsahuje postup, jak lze přidat jednotlivé

metody. Celý systém je nezávislý na počtu modelovaných geofyzikálních metod, proto přidaní nové metody žádným způsobem neovlivňuje již naprogramované části.

Do aplikační logiky lze přípojit nejen nové metody, ale také výpočet anomálního účinku jednotlivých geometrických forem. Příkladem jsou vzorky pro výpočet vlivu horizontálního nekonečného válce, stupně, vodorovné desky.

Fáze postupu přidáni nové geofyzikální metody do aplikace:

1. Definovat metodu, která se následně do programu začlení. Každá geofyzikální metoda zkoumá určité prostředí. Je potřeba proto přesně definovat a určit, jaké fyzikální veličiny budou modelovány. Metodu, jež je zvolená podle druhu fyzikálního pole, potom lze detailně prozkoumat a určit její podstatu.
2. Klasifikovat, jaké druhy geometrických tvarů budou modelované systémem. Různé geometrické tvary mají své zvláštnosti při procesu vytvoření jejich vzorů.
3. Podle metody a tvaru lze určit vzorec výpočtu anomálního účinku na zkoumané fyzikální pole v každém bodě povrchu, který bude modelovaný. Pro přidaní nové metody je nutné přesně definovat parametry, jež bude program používat. Tyto parametry jsou části vzorce pro výpočet nestandardního účinku a lze je získat analýzou vlastností příslušného geofyzikálního pole.

Například, parametry pro modelování metody spontánní polarizace s anomálním objektem tvaru koule jsou: M je moment dipólu polarizované koule, x je vzdálenost od pozice na povrchu, z_0 je hloubka. Parametry jsou určené ze vzorce pro polarizovaný vodič ve tvaru koule (6).

Pro názornost uvedu postupně kroky pro přidání způsobu modelování gravimetrické metody s objektem ve tvaru nekonečného válce.

1. Gravimetrická metoda je geofyzikální metoda, která má své základy ve vědním oboru gravimetrie. Měřená hodnota je tíhové pole určitého profilu. Pro řešení přímé úlohy je potřeba vypočítat tíhový účinek modelu tělesa. Parametry anomálie jsou známé předem, což znamená, že na vstupu uživatel musí je definovat.

2. Kruhový válec má v dvourozměrné dimenzi podobnou anomálii jako koule, ale na ploše vypadá jako rovnoběžné přímky. Hloubka a hmotnost tělesa tvaru válce se určuje obdobně jako u koule.
3. Když je definovaná metoda a tvar objektu, lze najít rovnici, která popíše gravitační účinek anomálie na gravitační pole na ploše. Rovnice pro výpočet tíhového účinku válce:

$$V_z(x,0,0)=\frac{2\chi M h}{x^2+h^2}, \tag{7}$$

kde χ je gravitační konstanta, M je hmotnost tělesa, h je hloubka, x je vzdálenost od pozice na povrchu. Parametry rovnice budou vstupem od uživatele.

Na daném kroku jsou zjištěné parametry budoucího modelu. Aby se dalo přidat modelování nové metody do aplikace, je nutné:

1. Vytvořit složku s názvem geofyzikální metody, která se musí nacházet ve složce, kde je uložená aplikace. Cesta je *„\Model\formula\název_metody\“*. Pro gravimetrickou metodu název cesty lze definovat tak: *„\Model\formula\gravimetry\“*. Jestliže stejná metoda již existuje, dá se přejít ke kroku číslo 2.
2. Ve vytvořené složce je nutné vytvořit složku s názvem geometrického tvaru. Příkladem je *„\Model\formula\gravimetry\cylinder“*.
3. Dále je nutné vytvořit mat-soubor, který se nazývá *„parametersString.mat“*. Uvnitř se zadávají parametry, které jsou ve vzorci pro výpočet anomálního účinku. Pro gravimetrickou metodu, tvar objektu je válec, parametry jsou druhá pozice válce (první bod je definovaný v standardních nastaveních a také hloubka) a hmotnost tělesa. Obsah souboru má vypadat: *„weight;x2;y2.“*. Takový formát zápisu je povinný, každý nový název se odděluje středníkem, na konci je tečka. Popis souřadnice má vždycky být *„x nebo y souřadnice + pořadové číslo bodu v prostotu“*. První bod je standardně uveden, proto pořadové číslo musí začínat od čísla dvě. Konstanty se v parametrech neuvádí (viz. Krok 4).

4. Další povinný soubor je m-file, jenž právě počítá matice modelovaného fyzikálního pole. v souboru je nutně postupovat podle šablony (Obr.24). Za prvé zavolat get metody objektů na vstupu a vytvořit lokální proměny do kterých přidáme vlastnosti objektů. Na šabloně jsou znázorněné všechny potřebné proměnné: délka a šířka povrchu, pozice a parametry anomálie. Dále v cyklech je nutně napsat kód pro výpočet účinku anomálie pro každý prvek matice.

```
1.    function [ matrix ] = creatingMatrix( anomaly, area )
2.    %Template of the function to count gravi anomaly
3.    %with the cylindrical shape
4.
5.    %in the first part difine all parameters,
6.    %which are used in the modeling of anomaly
7.        X = (0:get(area, 'lengthAreaStep'):get(area, 'lengthArea'));
8.        Y = (0:get(area, 'widthStep'):get(area, 'width'));
9.        matrix = zeros(length(X)-1, length(Y)-1);
10.       positionX = anomaly.position(1);
11.       positionY = anomaly.position(2);
12.       positionZ = anomaly.position(3);
13.       parameters = get(anomaly, 'parameters');
14.       weight = parameters(1);
15.       x2 = parameters(2);
16.       y2 = parameters(3);
17.       CKappa=6.670E-11;
18.   %second part count the matrice of influence of the anomaly
   object
19.       for i = 1:length(X) - 1
20.           for j = 1:length(Y) - 1
21.               matrix(i, j) =  ...;
22.           end
23.       end
24.
25.   end
```

Obr.24 Šablona funkce pro přidávaní nových metod

Po zaplnění šablony lze spustit aplikace a nová metoda modelovaní vzorových dat bude zpřístupněná v seznamu metod v panelu „Anomaly“. Dále se pracuje s novou metodou stejně jako bylo uváděno v kapitole *„Postup pro modelování anomálie na povrchu“*.

Závěr

V práci je uplatněný postup pro modelování geofyzikálních anomálií. Hlavní úspěchy jsou zrychlení fáze modelování a možnost automatizace procesu při vytvoření modelu vlivu anomálního tělesa na určité fyzikální pole. Vzorová fyzikální pole jsou definovaná pomocí geofyzikálních rovnic a následně implementováná v programu.

Výsledný software je rozšířitelný. v dokumentaci je vysvětlen postup pro přidaní nové geofyzikální anomálie na modelovaný povrch bez změny již vytvořených částí systému. Aplikace řeší přímou úlohu geofyziky pro aplikované metody a geometrické tvary. Řešení je jednoznačné.

Podařilo se aplikovat návrhový vzor model-view-conroller, jehož použitím jsem oddělil aplikační logiku a uživatelské rozhraní. Přínosem je lepší testovatelnost aplikace a také přenositelnost modelu pro různá grafická rozhraní. Pro rozdělení systému na logické celky byl použit jazyk UML, který dovolil lepší znázornění vztahů mezi třídami a způsobů rozšířitelnosti.

Před vypracováním modelu výsledné aplikace jsem se seznámil s principy geofyziky. Do základních modelovaných metod ve vytvořeném programu patří metody gravimetrické a elektrické. Na základě literatury jsem definoval vztahy pro geometrické anomálie v tíhovém a potenciálovém poli. Aplikace je implementována v Matlabu. Geofyzikální vztahy jsem aplikoval jako funkce, výsledkem čehož je zpracování dat rychlejší než při využití objektových metod.

Navrhl jsem grafické uživatelské rozhraní v Matlabu GUIDE, kde uživatel může využívat naprogramované funkce pro výpočet přímé geofyzikální úlohy. Podle parametrů, které zadává uživatel, program vytvoří model fyzikálního pole s anomáliemi. V aplikaci jsem implementoval možnosti uložit a načíst generovaná data, což umožňuje použiti modelovaných objektů při dalším zpracování.

Implementovaný software se následně začlení do výzkumné části, která bude po ukončení fáze výzkumu přepsaná do jazyka JAVA. Proto pro účel přenositelnosti návrhu jsem využil výhody návrhového vzoru MVC a jazyka UML.

Seznam použité literatury

[1] MAREŠ, Stanislav. *Úvod do užité geofyziky*. 2. přeprac. vyd. Praha: SNTL - Nakladatelství technické literatury, 1990, 677 s. ISBN 80-030-0427-6.

[2] *MOŽNOSTI POUŽITÍ GEOFYZIKÁLNÍCH METOD při ověřování nejasných strukturně geologických, popř. jiných vztahů na lokalitách při průzkumu a nápravě starých ekologických zátěží*. Praha: Ministerstvo životního prostředí ČR, 1999, VII, 1/99. ISSN 1210- 4124.

[3] SCHMULLER, Joseph. *Myslíme v jazyku UML*. Vyd. 1. Praha: Grada, 2001, 360 s. ISBN 80-247-0029-8.

[4] The MathWorks, Inc. *MathWorks* [online]. 1994-2013 [cit. 2013-05-16]. Dostupné z: http://www.mathworks.com/products/matlab/

[5] Model-View-Controller. *Microsoft msdn* [online]. 2013 [cit. 2013-05-16]. Dostupné z: http://msdn.microsoft.com/ru-ru/library/ff649643.aspx

Příloha A – obsah CD

Přiložené CD obsahuji zdrojový kód aplikace.

Příloha B – main funkce a třída „Model“

```
function main()
%MAIN function add to the path all paths of the program,
%create Model and Conroller, implements MVC
    addpath(genpath('model/'));
    addpath(genpath('controller_view/'));
    model = Model();
    controller = Controller(model);

end
     classdef Model < hgsetget
         %Model has properties and methods, that controll all behavior of the
         %program. Model has one current anomaly and area.
         properties(SetObservable)

             area;% current area
             anomaly; % current anomaly
             currentAnomalyMatrix; % current anomaly matrix(it can be not in the area yet)
         end

         methods
             %constructor MOdel with default area and anomaly
             function obj = Model()
                 obj.area = Area();
                 obj.anomaly = Anomaly();
             end
             % save anomaly from the Model to the .mat file
             function saveAnomaly(obj, location, name)
                 obj.anomaly.saveAnomaly(location, name);
             end
             % load anomaly from the location to this model
             function loadAnomaly(obj, location)
                 newAnomaly = load(location);
                 obj.anomaly = newAnomaly.obj;
             end
             % save area from the Model to the .mat file
             function saveArea(obj, location, name)
                 obj.area.saveArea(location, name);
             end
             % load area from the location to this model
             function loadArea(obj, location)
                 newArea = load(location);
                 obj.area = newArea.obj;
             end
             % create an anomaly and change the Model's anomaly
             function createAnomaly(obj, newMethod, newType, newPosition, newParameters)
                 obj.anomaly = Anomaly(newMethod, newType, newPosition, newParameters);
             end
              % create an area and change the Model's anomaly
             function createArea(obj, lengthArea, width, lengthAreaStep, widthStep, dimension)
                 obj.area = Area(lengthArea, width, lengthAreaStep, widthStep, dimension);
```

```
51.             end
52.             %add model's anomaly to the model's area
53.             function addAnomalyToArea(obj)
54.                 obj.area.addAnomaly(obj.anomaly);
55.             end
56.             % return the matrix of the area
57.             function matrix = showAreaMatrix(obj)
58.                 matrix = obj.area.matrix;
59.             end
60.             % return the last added anomaly's matrix
61.             function matrix = showAnomalyMatrix(obj)
62.                 if(isempty(obj.area.anomalies))
63.                     matrix = [];
64.                 else
65.                     matrix =
    obj.area.matrixAnomalies(length(obj.area.matrixAnomalies)).matri
    x;
66.                 end
67.             end
68.             %return matrix of the anomaly, but which is not
    added yet to the area
69.             function matrix = showAnomalyExampleMatrix(obj)
70.                 matrix = obj.anomaly.createMatrix(obj.area);
71.             end
72.             % return matrix of the anomaly by ID from area
73.             function [anomaly matrix] = showAnomaly(obj, id)
74.                 if(length(obj.area.anomalies) < id)
75.                     disp('anomaly with such number does not
    exist');
76.                 else
77.                     [anomaly matrix] =
    obj.area.retAnomaly(id);
78.                     obj.anomaly = anomaly;
79.                 end
80.             end
81.             %return count of the anomalies in the area
82.             function count = countAnomalies(obj)
83.                 count = length(obj.area.anomalies);
84.             end
85.             % remove anomaly from the area by ID
86.             function deleteAnomaly(obj, id)
87.                 if(length(obj.area.anomalies) < id)
88.                     disp('anomaly with such number does not
    exist');
89.                 else
90.                     obj.area.deleteAnomaly(id);
91.                 end
92.             end
93.             %return parameters of the anomaly as string names
94.             function listP = getParametersStringList(obj)
95.                 listP = obj.anomaly.getParametersStringList();
96.             end
97.             %return parameters of the anomaly as values
98.             function listValueP = getValueParameters(obj)
99.                 listValueP = obj.anomaly.parameters;
100.            end
101.            %create name to the area in order to save it
102.            function areaName = getAreaName(obj)
103.
```

```
104.                 areaName = ['model/save_areas/area_'
   obj.area.dimension '_' num2str(obj.area.lengthArea) '_'
   num2str(obj.area.width) '_' ];
105.                 orderNum = 1;
106.                 while(exist([areaName num2str(orderNum)
   '.mat'], 'file'))
107.                     orderNum = orderNum + 1;
108.                 end
109.                 areaName = [areaName num2str(orderNum)
   '.mat'];
110.             end
111.             %create name to the anomaly in order to save it
112.             function anomalyName = getAnomalyName(obj)
113.                 anomalyName = ['model/save_anomalies/anomaly_'
   obj.anomaly.method '_' obj.anomaly.type '_' ];
114.                 orderNum = 1;
115.                 while(exist([anomalyName num2str(orderNum)
   '.mat'], 'file'))
116.                     orderNum = orderNum + 1;
117.                 end
118.                 anomalyName = [anomalyName num2str(orderNum)
   '.mat'];
119.             end
120.
121.         end
122.         methods (Static)
123.             %return list of methods user can use
124.             function list = listMethod(~)
125.                listMethod = dir('model/formula');
126.                list{1} = '';
127.                for i = 3:length(listMethod)
128.                    list{(i-2)} = listMethod(i).name;
129.
130.                end
131.             end
132.             %return list of types in each method
133.             function list = listType(method)
134.                listMethod = dir(['model/formula/' method]);
135.                list{1} = '';
136.                for i = 3:length(listMethod)
137.                    list{(i-2)} = listMethod(i).name;
138.
139.                end
140.             end
141.             %return parameters for method and  type as string
   list
142.             function parameters = getParam(method, type)
143.                 parameters = getParamFun(method, type);
144.                  listP{1} = '';
145.                 i = 1;
146.                 j = 1;
147.                 param = '';
148.                 len = length(parameters);
149.                 for i = 1:len
150.
151.                     if (strcmp(parameters(i), ';') ||
   strcmp(parameters(i), '.'))
152.                         listP{j} = param;
153.                         param = '';
```

```
154.                         j = j + 1;
155.
156.                     else
157.                         param = [param parameters(i)];
158.
159.                     end
160.
161.                 end
162.                 parameters = listP;
163.             end
164.
165.         end
166.
167.     end
```

Příloha C – Třída „Area“, „Anomaly“

```
1. classdef Area < hgsetget
2.     % Area class contains the main properties of every area
3.
4.     properties (SetObservable)
5.         lengthArea; % length of the area
6.         width; % width of the area
7.         lengthAreaStep; % step by length
8.         widthStep; % step by width
9.         dimension; % dimension of area
10.              anomalies; % anomalies in the area
11.              matrix; % matrix contains all anomalies in area
12.              matrixAnomalies; % structure with every anomalie's
   matrix in area
13.          end
14.
15.          methods
16.              % constructor create default area without
   parameters and user area
17.              % with parameters
18.              function obj = Area(lengthArea, width,
   lengthAreaStep, widthStep, dimension)
19.                  if nargin == 0
20.                      lengthArea = 100;
21.                      width = 100;
22.                      lengthAreaStep = 1;
23.                      widthStep = 1;
24.                      dimension = '3d';
25.
26.                      obj.lengthArea = lengthArea;
27.                      obj.width = width;
28.                      obj.lengthAreaStep = lengthAreaStep;
29.                      obj.widthStep = widthStep;
30.                      obj.dimension = dimension;
31.                      X = (0:lengthAreaStep: lengthArea);
32.                      Y = (0:widthStep: width);
33.                      obj.matrix = zeros(length(X)-1, length(Y)-
   1);
34.                  end
35.                  if nargin > 0
36.                      if(strcmp(dimension, '2d'))
37.                          widthStep = 1;
38.                          width = 1;
39.                      end
40.                      obj.lengthArea = lengthArea;
41.                      obj.width = width;
42.                      obj.lengthAreaStep = lengthAreaStep;
43.                      obj.widthStep = widthStep;
44.                      obj.dimension = dimension;
45.                      X = (0:lengthAreaStep: lengthArea);
46.                      Y = (0:widthStep: width);
47.                      obj.matrix = zeros(length(X)-1, length(Y)-
   1);
48.
49.
50.                  end
51.              end
```

```
52.             % set and get lengthArea
53.             function set.lengthArea(obj, newLengthArea)
54.                 if(newLengthArea > 0)
55.                     obj.lengthArea = newLengthArea;
56.                 else
57.                     error('length of area must be number > 0')
58.                 end
59.             end
60.             function lengthArea = get.lengthArea(obj)
61.                 lengthArea = obj.lengthArea;
62.             end
63.             % set and get width
64.             function set.width(obj, newWidth)
65.                 if(newWidth > 0)
66.                     obj.width = newWidth;
67.                 else
68.                     error('width of area must be number > 0')
69.                 end
70.
71.             end
72.             function width = get.width(obj)
73.                 width = obj.width;
74.             end
75.             % set and get lengthAreaStep
76.             function set.lengthAreaStep(obj,
   newlengthAreaStep)
77.                 obj.lengthAreaStep = newlengthAreaStep;
78.
79.             end
80.             function lengthAreaStep = get.lengthAreaStep(obj)
81.                 lengthAreaStep = obj.lengthAreaStep;
82.             end
83.             % set and get widthStep
84.             function set.widthStep(obj, newWidthStep)
85.                 obj.widthStep = newWidthStep;
86.             end
87.             function widthStep = get.widthStep(obj)
88.                 widthStep = obj.widthStep;
89.             end
90.             % set and get dimension
91.             function set.dimension(obj, newDimension)
92.                 obj.dimension = newDimension;
93.             end
94.             function dimension = get.dimension(obj)
95.                 dimension = obj.dimension;
96.             end
97.             %add an anomaly to the Areae
98.             function addAnomaly(obj, anomaly)
99.
100.                obj.anomalies = [obj.anomalies anomaly];
101.                newMatrix = anomaly.createMatrix(obj);
102.
103.               % obj.matrixAnomalies(length(obj.anomalies)).id
   = (length(obj.anomalies));
104.
   obj.matrixAnomalies((length(obj.anomalies))).matrix = newMatrix;
105.
106.                obj.matrix = obj.matrix + newMatrix;
107.
```

```
108.
109.
110.            end
111.            %return the anomaly and its matrix by id
112.            function [anomaly matrix] = retAnomaly(obj, id)
113.                if length(obj.anomalies) < id
114.                else
115.                anomaly = obj.anomalies(id);
116.                matrix = obj.matrixAnomalies(id).matrix;
117.                end
118.            end
119.            % set and get matrix
120.            function matrix = get.matrix(obj)
121.                matrix = obj.matrix;
122.            end
123.
124.            function deleteAnomaly(obj, id)
125.                if length(obj.anomalies) < id
126.                elseif length(obj.anomalies) == 1 && id == 1
127.                    obj.anomalies(id) = [];
128.
129.                    obj.matrix = obj.matrix * 0;
130.                    obj.matrixAnomalies(id) = [];
131.                else
132.                    obj.anomalies(id) = [];
133.
134.                    obj.matrix = obj.matrix -
    obj.matrixAnomalies(id).matrix;
135.                    obj.matrixAnomalies(id) = [];
136.                end
137.            end
138.            function saveArea(obj, location, name)
139.                location = [location name];
140.                save(location, 'obj*');
141.            end
142.        end
143.
144.    end
145.
146.
147.    classdef Anomaly < hgsetget
148.
149.
150.        properties (SetObservable)
151.
152.            position;  %position of the anomaly x, y, z
153.            parameters;  %array of the anomaly parameters
154.            parametersString; %array of the string name of the
    parameters
155.            method; %family name of the anomaly(gravimetry,
    magnet...)
156.            type; %type of the object (sphere, cylender ...)
157.        end
158.
159.        methods
160.            function obj = Anomaly( newMethod, newType,
    newPosition, newParameters)
161.                if nargin == 0
162.                    newPosition = [50 50 50];
```

```
163.                     newMethod = 'gravimetry';
164.                     newType = 'sphere';
165.                     newParameters = 100;
166.
167.                     obj.position = newPosition;
168.                     obj.parametersString =
   getParamFun(newMethod, newType);
169.                     obj.method = newMethod;
170.                     obj.type = newType;
171.                     obj.parameters = newParameters;
172.                 end
173.                 if nargin > 0
174.                     obj.position = newPosition;
175.                     obj.parametersString =
   getParamFun(newMethod, newType);
176.
177.                     obj.parameters = newParameters;
178.                     obj.method = newMethod;
179.                     obj.type = newType;
180.
181.                 end
182.             end
183.             % set and get position
184.             function set.position(obj, newPosition)
185.                 if(length(newPosition) == 3)
186.                     obj.position = newPosition;
187.                 else
188.                     error('position needs 3 parameters')
189.                 end
190.             end
191.             function position = get.position(obj)
192.                 position = obj.position;
193.             end
194.             % set and get parameters
195.             function set.parameters(obj, newParameters)
196.                 obj.parameters = newParameters;
197.             end
198.             function parameters = get.parameters(obj)
199.                 parameters = obj.parameters;
200.             end
201.             % set and get parametesString
202.             function set.parametersString(obj,
   newParametersString)
203.
204.                 obj.parametersString = newParametersString;
205.             end
206.             function parametersString =
   get.parametersString(obj)
207.                 parametersString = obj.parametersString;
208.             end
209.             function parametersStringList =
   getParametersStringList(obj)
210.
211.                 listP{1} = '';
212.                 i = 1;
213.                 j = 1;
214.                 param = '';
215.                 len = length(obj.parametersString);
216.                 for i = 1:len
```

```
217.
218.                     if (strcmp(obj.parametersString(i), ';')
   || strcmp(obj.parametersString(i), '.'))
219.                         listP{j} = param;
220.                         param = '';
221.                         j = j + 1;
222.
223.                     else
224.                         param = [param
   obj.parametersString(i)];
225.
226.                     end
227.
228.                 end
229.                 parametersStringList = listP;
230.                 end
231.             % set and get method
232.             function set.method(obj, newMethod)
233.                 obj.method = newMethod;
234.             end
235.             function method = get.method(obj)
236.                 method = obj.method;
237.             end
238.             % set and get type
239.             function set.type(obj, newType)
240.
241.                 obj.type = newType;
242.             end
243.             function type = get.type(obj)
244.                 type = obj.type;
245.             end
246.             %get the matrix one the area
247.             function matrix = createMatrix(obj, area)
248.                 [matrix] = createMatrixFunction(obj, area);
249.             end
250.             function saveAnomaly(obj, location, name)
251.                 location = [location name];
252.                 save(location, 'obj*');
253.             end
254.
255.         end
256.
257.     end
```

Příloha D – Funkce CreateMatrix pro válec

```
1. function [ matrix ] = creatingMatrix( anomaly, area )
2. %Template of the function to count gravi anomaly
3. %with the cylindrical shape
4.
5. %in the first part difine all parameters,
6. %which are used in the modeling of anomaly
7.     X = (0:get(area, 'lengthAreaStep'):get(area, 'lengthArea'));
8.     Y = (0:get(area, 'widthStep'):get(area, 'width'));
9.     matrix = zeros(length(X)-1, length(Y)-1);
10.          positionX = anomaly.position(1);
11.          positionY = anomaly.position(2);
12.          positionZ = anomaly.position(3);
13.          parameters = get(anomaly, 'parameters');
14.          weight = parameters(1);
15.          x2 = parameters(2);
16.          y2 = parameters(3);
17.          CKappa=6.670E-11;
18.      %second part count the matrice of influence of the anomaly
   object
19.          for i = 1:length(X) - 1
20.              for j = 1:length(Y) - 1
21.                  u = [x2 y2] - [positionX positionY];
22.                  n = [-1*u(2) u(1)];
23.                  c = -1 * (n(1) * positionX + n(2) *
   positionY);
24.                  r = abs(n(1) * X(i) + n(2) * Y(j) + c) /...
25.                      sqrt(power(n(1),2) + power(n(2),2));
26.                  matrix(i, j) = 2 * CKappa * weight * (-
   positionZ) / ...
27.                      power((r*r + positionZ * positionZ), 1);
28.              end
29.          end
30.
31.      end
```

Příloha E – obsahuje zdrojový kód Controller, View

```
1. classdef Controller < handle
2.     properties
3.         model
4.         view
5.     end
6.
7.     methods
8.         function obj = Controller(model)
9.             obj.model = model;
10.                 obj.view = View(obj);
11.             end
12.             function saveAnomaly(obj, location, name)
13.                 obj.model.saveAnomaly(location, name);
14.             end
15.             function loadAnomaly(obj, location)
16.                 obj.model.loadAnomaly(location);
17.             end
18.             function saveArea(obj, location, name)
19.                 obj.model.saveArea(location, name);
20.             end
21.             function loadArea(obj, location)
22.                 obj.model.loadArea(location);
23.             end
24.             function createAnomaly(obj, newMethod, newType,
    newPosition, newParameters)
25.                 obj.model.createAnomaly(newMethod, newType,
    newPosition, newParameters);
26.             end
27.             function createArea(obj, lengthArea, width,
    lengthAreaStep, widthStep, dimension)
28.                 obj.model.createArea(lengthArea, width,
    lengthAreaStep, widthStep, dimension);
29.             end
30.             function addAnomalyToArea(obj)
31.                 obj.model.addAnomalyToArea();
32.             end
33.             function matrix = showAreaMatrix(obj)
34.                 matrix = obj.model.showAreaMatrix();
35.             end
36.             function matrix = showAnomalyMatrix(obj)
37.                 matrix = obj.model.showAnomalyMatrix();
38.             end
39.             function [anomaly matrix] = showAnomaly(obj, id)
40.                 [anomaly matrix] = obj.model.showAnomaly(id);
41.             end
42.             function count = countAnomalies(obj)
43.                 count = obj.model.countAnomalies();
44.             end
45.             function deleteAnomaly(obj, id)
46.                 obj.model.deleteAnomaly(id)
47.             end
48.             function listP = getParametersStringList(obj)
49.                 listP = obj.model.getParametersStringList();
50.             end
51.             function listValueP = getValueParameters(obj)
52.                 listValueP = obj.model.getValueParameters();
```

```
53.             end
54.             function areaName = getAreaName(obj)
55.                 areaName = obj.model.getAreaName();
56.
57.             end
58.             function anomalyName = getAnomalyName(obj)
59.                 anomalyName = obj.model.getAnomalyName();
60.             end
61.             function matrix = showAnomalyExampleMatrix(obj)
62.                 matrix = obj.model.showAnomalyExampleMatrix();
63.             end
64.             function setCurrentAnomalyMatrix(obj, matrix)
65.                 obj.model.currentAnomalyMatrix = matrix;
66.             end
67.             function matrix = getCurrentAnomalyMatrix(obj)
68.                 matrix = obj.model.currentAnomalyMatrix;
69.             end
70.         end
71.     end
72.
73.
74.     classdef View < handle
75.         properties
76.             gui;
77.             model;
78.             controller;
79.             area;
80.             anomaly;
81.
82.         end
83.
84.         methods
85.             function obj = View(controller)
86.                 obj.controller = controller;
87.                 obj.model = controller.model;
88.                 obj.gui =
    areaGUI('controller',obj.controller);
89.
90.
91.
    addlistener(obj.model,'area','PostSet',@(src,evnt)View.handlePro
    pEvents(obj,src,evnt));
92.
    addlistener(obj.model,'anomaly','PostSet',@(src,evnt)View.handle
    PropEvents(obj,src,evnt));
93.
94.             end
95.
96.
97.         end
98.
99.         methods (Static)
100.             function handlePropEvents(obj,src,evnt)
101.                 evntobj = evnt.AffectedObject;
102.                 handles = guidata(obj.gui);
103.
104.                 switch src.Name
105.
106.                     case 'area'
```

```
107.                            set(handles.length, 'String',
    evntobj.area.lengthArea);
108.                            set(handles.lengthStep, 'String',
    evntobj.area.lengthAreaStep);
109.                            set(handles.dimension, 'String',
    evntobj.area.dimension);
110.                            set(handles.width, 'String',
    evntobj.area.width);
111.                            set(handles.widthStep, 'String',
    evntobj.area.widthStep);
112.                            if(strcmp(evntobj.area.dimension,
    '2d'))
113.                                plot( handles.axesArea,
    evntobj.area.matrix);
114.
    set(handles.popupmenuPaint,'Enable', 'off');
115.                            else
116.                                mesh( handles.axesArea,
    evntobj.area.matrix);
117.
    set(handles.popupmenuPaint,'Enable', 'on');
118.
119.                            end
120.
121.                            numberAnomaly =
    length(evntobj.area.anomalies);
122.
123.                            if(numberAnomaly)
124.                                listAnomalies{1} = '';
125.                                for i = 2:numberAnomaly+1
126.                                    listAnomalies{i} = num2str(i-
    1);
127.                                end
128.
    set(handles.popupmenuAreaAnomalies, 'String', listAnomalies);
129.
    set(handles.popupmenuAreaAnomalies, 'Value', 1);
130.                            else
131.                                listAnomalies{1} = '';
132.
133.
    set(handles.popupmenuAreaAnomalies, 'String', listAnomalies);
134.
    set(handles.popupmenuAreaAnomalies, 'Value', 1);
135.                            end
136.                            obj.area = evntobj.area;
137.                        case 'anomaly'
138.                            set(handles.method, 'String',
    evntobj.anomaly.method);
139.                            set(handles.type, 'String',
    evntobj.anomaly.type);
140.                            set(handles.posX, 'String',
    evntobj.anomaly.position(1));
141.                            set(handles.posY, 'String',
    evntobj.anomaly.position(2));
142.                            set(handles.posZ, 'String',
    evntobj.anomaly.position(3));
143.                            set(handles.tableAnomaly,
    'ColumnName',evntobj.anomaly.getParametersStringList());
```

```
144.                         set(handles.tableAnomaly, 'RowName',
   []);
145.                         set(handles.tableAnomaly, 'Data',
   evntobj.anomaly.parameters);
146.                         set(handles.tableAnomaly,
   'ColumnWidth',{50})
147.                         obj.anomaly = evntobj.anomaly;
148.                     end
149.                 end
150.             end
151.     end
```