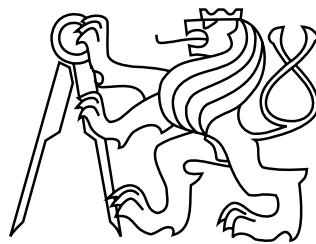


ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE  
FAKULTA STAVEBNÍ  
OBOR GEODÉZIE, KARTOGRAFIE A GEOINFORMATIKA



BAKALÁŘSKÁ PRÁCE  
POSUN LETECKY MĚŘENÝCH BODŮ PO TRAJEKTORII  
V PROSTŘEDÍ QGIS

Vedoucí práce: Ing. Martin Landa, Ph.D.  
Katedra geomatiky

červen 2016

Ondřej PEŠEK



**ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE**


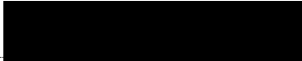
**Fakulta stavební**  
Thákurova 7, 166 29 Praha 6

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE



### I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Pešek	Jméno: Ondřej	Osobní číslo: 423996
Zadávající katedra: Katedra geomatiky		
Studijní program: Geodézie a kartografie		
Studijní obor: Geodézie, kartografie a geoinformatika		

### II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce: Posun letecky měřených bodů po trajektorii v prostředí QGIS	
Název bakalářské práce anglicky: Aerial data move on a trajectory in QGIS software	
Pokyny pro vypracování: Cílem bakalářské práce je návrh softwarového nástroje umožňujícího posun letecky měřených bodů po trajektorii. Takový nástroj je třeba z toho důvodu, že přístroj při leteckých měřeních zapisuje souřadnice s určitým zpožděním. V praktické části práce se počítá s jeho implementací jako tzv. zásuvného modulu do prostředí open source projektu QGIS s využitím grafického frameworku Qt.	
Seznam doporučené literatury: Kurt Menke, G.: Mastering QGIS, Packt Publishing, 2015, ISBN: 9781784390068 Pilgrim, M.: Dive Into Python, Createspace Independent Pub 2009, ISBN: 9781441413024 Summerfield, M.: Rapid GUI Programming With Python and Qt, Prentice Hall, 2015, ISBN: 9780134393339	
Jméno vedoucího bakalářské práce: Ing. Martin Landa, Ph.D.	
Datum zadání bakalářské práce: 22.2.2016	Termín odevzdání bakalářské práce: 22.5.2016
 Podpis vedoucího práce	 Podpis vedoucího katedry

### III. PŘEVZETÍ ZADÁNÍ

<i>Beru na vědomí, že jsem povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je nutně uvést v bakalářské práci a při citování postupovat v souladu s metodickou příručkou ČVUT „Jak psát vysokoškolské závěrečné práce“ a metodickým pokynem ČVUT „O dodržování etických principů při přípravě vysokoškolských závěrečných prací“.</i>	
 Datum převzetí zadání	 Podpis studenta(ky)

## **ABSTRAKT**

Cílem bakalářské práce je návrh softwarového nástroje umožňujícího posun letecky měřených bodů po trajektorii. Takový nástroj je třeba z toho důvodu, že přístroj při leteckých měřeních zapisuje souřadnice s určitým zpožděním. Praktická část práce zahrnuje jeho implementaci jako tzv. zásuvného modulu do prostředí open source projektu QGIS s využitím grafického frameworku Qt. Při vyvíjení zásuvného modulu bylo přihlíženo k požadavkům a připomínkám Státního ústavu radiální ochrany, který tento projekt inicioval. Z důvodu zachování úzu prostředí QGIS byl zásuvný modul vytvářen v jazyku Python.

## **KLÍČOVÁ SLOVA**

GIS, QGIS, zásuvný modul, python, letecká data

## **ABSTRACT**

The object of bachelor thesis is creation of software tool for shifting aerial data by their trajectory (GPS position lag correction). The reason for this tool is that the instrument for aerial data surveying is recording data with some delay. In practical part of bachelor thesis is implementation of this tool as plugin into open source project QGIS using graphical framework Qt. While developing the plugin, comments and requests by National Radiation Protection Institute (initiators of the project) have been taken into consideration. Due to recent habit of QGIS software the plugin will be created in language Python.

## **KEYWORDS**

GIS, QGIS, plugin, python, aerial data

## PROHLÁŠENÍ

Prohlašuji, že bakalářskou práci na téma „Posun letecky měřených bodů po trajektorii v prostředí QGIS“ jsem vypracoval samostatně. Použitou literaturu a podkladové materiály uvádím v seznamu zdrojů.

V Praze dne .....

.....

(podpis autora)

## PODĚKOVÁNÍ

Chtěl bych poděkovat vedoucímu práce, Ing. Martinu Landovi, PhD., za připomínky a pomoc při zpracování této práce. Dále bych chtěl poděkovat Martinu Joskovi za to, že mě poprosil, zda bych mu nepoděkoval.

# Obsah

<b>1</b>	<b>Úvod</b>	<b>9</b>
<b>2</b>	<b>Teoretický základ</b>	<b>11</b>
2.1	Scintilační spektrometrie . . . . .	11
2.1.1	Detektorová část . . . . .	12
2.1.2	Analyzační část . . . . .	14
2.2	Sběr souřadnic a potřeba posunu . . . . .	15
2.3	Posun . . . . .	15
2.4	Problémy . . . . .	17
2.4.1	Elipsoid . . . . .	17
2.4.2	První geodetická úloha . . . . .	17
<b>3</b>	<b>Použité technologie</b>	<b>19</b>
3.1	Python . . . . .	19
3.2	QGIS . . . . .	20
3.3	Qt Project . . . . .	21
<b>4</b>	<b>Zásuvný modul</b>	<b>22</b>
4.1	Obsah CSV . . . . .	22
4.2	Tělo zásuvného modulu . . . . .	23
4.3	Posun o hodnoty . . . . .	24
4.3.1	Posun o kladný počet hodnot . . . . .	25
4.3.2	Posun o záporný počet hodnot . . . . .	25
4.3.3	Posun o nulový počet hodnot . . . . .	26
4.4	Posun o konstantní vzdálenost . . . . .	26
4.4.1	Posun o kladnou vzdálenost . . . . .	27
4.4.2	Posun o zápornou vzdálenost . . . . .	28
4.4.3	Posun o nulovou vzdálenost . . . . .	30
4.4.4	První geodetická úloha . . . . .	30
4.5	Posun o konstantní čas (proměnnou vzdálenost) . . . . .	31
4.5.1	Posun o kladný časový úsek . . . . .	32
4.5.2	Posun o záporný časový úsek . . . . .	34

4.5.3	Posun o nulový časový úsek . . . . .	36
4.6	Licence . . . . .	36
<b>5</b>	<b>Závěr</b>	<b>37</b>
	<b>Seznam zkratek</b>	<b>38</b>
	<b>Literatura</b>	<b>39</b>
<b>A</b>	<b>User guide</b>	<b>40</b>
A.1	Loading of plugin . . . . .	40
A.2	Work with plugin . . . . .	40
A.2.1	Input and output defining . . . . .	41
A.2.2	Styling . . . . .	42
A.2.3	Showing input . . . . .	43
A.2.4	Shift . . . . .	44
A.2.5	Dependencies . . . . .	44

# Seznam obrázků

2.1	Základní schéma elektronického detektoru záření . . . . .	11
2.2	Obecné schéma scintilačního detektoru . . . . .	12
2.3	Různá uspořádání fotonásobičů . . . . .	14
2.4	Posun dat o 1 bod . . . . .	16
2.5	Princip měření . . . . .	16
2.6	Profilové schéma měření . . . . .	16
2.7	Integrační kroky v první geodetické úloze . . . . .	18
3.1	Python logo . . . . .	19
3.2	QGIS logo . . . . .	20
3.3	Qt logo . . . . .	21
4.1	Ukázka dvou typů vstupních souborů . . . . .	23
4.2	Ukázka - Posun o 3 hodnoty . . . . .	24
4.3	Ukázka - Posun o 9.823 metrů . . . . .	26
4.4	Ukázka - Posun o 2.718 sekund . . . . .	31
A.1	GUI . . . . .	40
A.2	GUI . . . . .	41
A.3	Loading input . . . . .	41
A.4	Choosing output directory . . . . .	42
A.5	Choose style . . . . .	43
A.6	Used style . . . . .	43
A.7	Enabled Show button . . . . .	44
A.8	Enabled Solve button . . . . .	45



# 1 Úvod

Scintilační spektrometry měří radioaktivitu, ale nikoli zeměpisné souřadnice, které těmto hodnotám náleží. To zajišťují další přístroje. Tyto přístroje však většinou zapisují souřadnice v momentu, kdy začíná měření dalších dat, nebo v momentu jeho skončení. Pro zpracování takových dat by tedy bylo záhodno mít možnost využití softwarového nástroje, který by tyto posuny alespoň částečně korigoval.

Zájem o vývoj takového nástroje pochází od Státního ústavu radiační ochrany (SÚRO). SÚRO se zabývá mimo jiné vyhledáváním lokalit se zvýšenou koncentrací radonu a vedením centrální databáze takových míst, poskytováním konzultací, prováděním laboratorních expertíz a hodnocením radiační ochrany v oblasti lékařského ozáření.

Ústav se rovněž angažuje v oblasti výzkumných projektů, například „výzkum pokročilých metod detekce, stanovení a následného zvládnutí radioaktivní kontaminace s cílem modernizovat odpovídající části systému zajištění ochrany obyvatel a vybraných kritických infrastruktur ČR v souvislosti s radiologickým útokem nebo velkou radiologickou havárií“ nebo „testování nových systémů hromadného měření radiojódů ve štítné žláze po havárii jaderně energetického zařízení“. [8]

Velkou část činností SÚRO tvoří též terénní měření a následné zpracování těchto dat v prostředí QGIS. Zpracovávaná data jsou však zatížena zmiňovaným zpožděním zápisu.

Optimální by se zdála modifikace zařízení zapisujícího souřadnice. Takovému kroku ale brání nejen nemožnost dostat se k podobným právům u výrobce, nýbrž také empirické zkušenosti SÚRO. Zpracovatelé Ústavu došli ke zjištění, že mnohdy nestačí užít posun, který by se na první pohled zdál ideální (posouvat body o polovinu dráhy k dalšímu bodu); zápis je totiž zatížen dalšími chybami. Zpracovatel pak potřebuje vyzkoušet několik posunů a dle rozložení hodnot usoudit, který z nich vyhovuje skutečnosti nejvíce. Měřená hodnota by neměla mít skokovitý charakter, ale měla by v okolí kritických hodnot nabývat přibližně lineárního rozložení.

Vzhledem k tomu, že SÚRO zpracovává data v prostředí QGIS, nabízí se vývoj rozšiřujícího nástroje právě pro toto prostředí. Zde by měl být implementován jako takzvaný zásuvný modul. S tím by souvisela též veřejná dostupnost zásuvného

modulu - v případě potřeby by jej posléze mohla využívat například i Armáda České republiky, sahající při některých úkonech též po softwaru QGIS.

Pro pokročilejší verzi zásuvného modulu by byla vhodná implementace několika variant posunu a možností stylovat vstupující data a porovnávat posunutá data s daty původními.

## 2 Teoretický základ

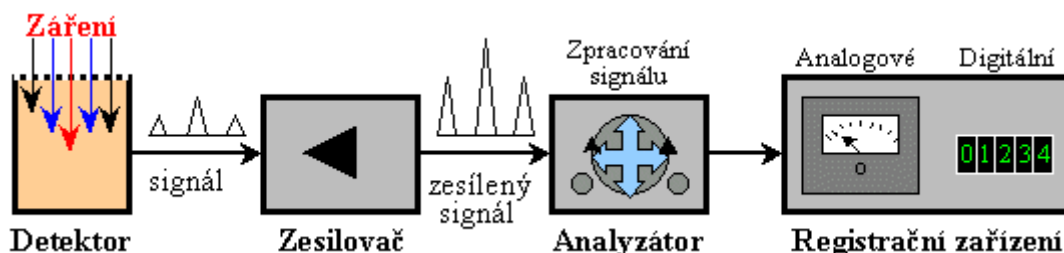
Tato kapitola si klade za cíl seznámit čtenáře s teoretickými základy měření radioaktivity, z nichž plyne potřeba posunu měřených bodů. Dále pak načrtne způsoby, kterými bude možno body posouvat, a popíše nejdůležitější problémy, s nimiž se při takovém posunu lze setkat.

### 2.1 Scintilační spektrometrie

V kapitole o scintilační spektrometrii bylo vycházeno z [4].

Ježto lidský subjekt není sám o sobě schopen detekovat ionizující záření, je třeba si vypomoci příslušnou technikou, tedy přístroji, detektory ionizujícího záření. Takových přístrojů existuje velké a neustále se zvětšující množství.

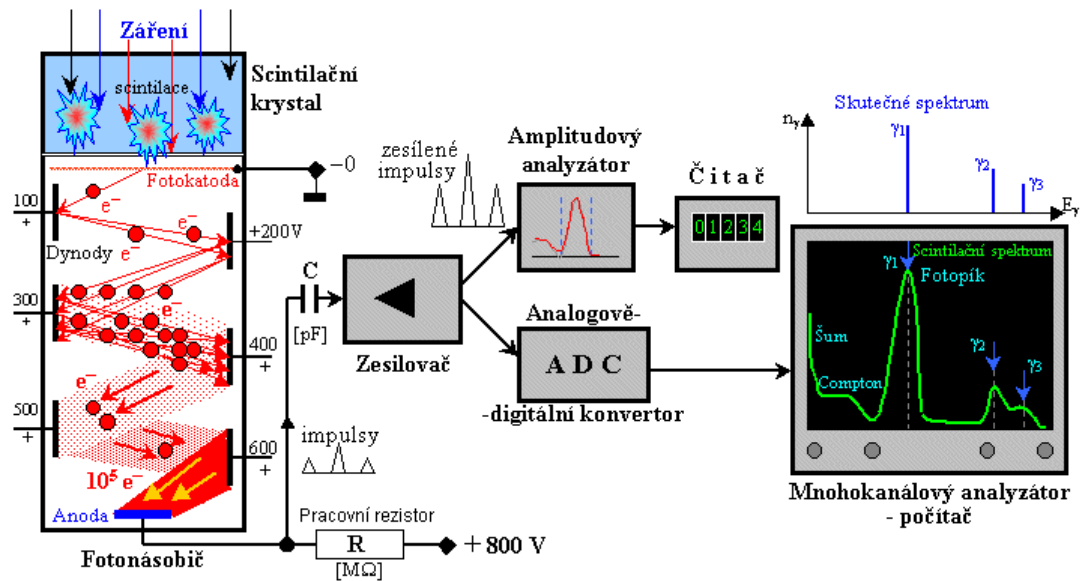
Obecně se detektory dělí buď podle průběhu detekce v závislosti na čase (kumulativní a kontinuální), nebo podle principu detekce (materiálové, fotografické a pro nás nejdůležitější elektronické).



Obrázek 2.1: Základní schéma elektronického detektoru záření (zdroj: [4])

Běžný elektronický detektor ionizujícího záření se skládá ze dvou hlavních bloků - z detektorové a analyzační části. V detektoru dochází k transformaci ionizačního záření na měřitelné elektronické impulsy. Takové impulsy bývají zesilovány buď už v detektoru, nebo v následujícím zesilovači; často se užívá kombinace obého. Signál zesílený na měřitelnou odezvu poté putuje do analyzátoru, který uživateli předává informace buď analogovou formou (například výchylka ručičky), nebo formou digitální (číselný ukazatel, zápis do souboru v počítači).

Následující podkapitoly představí přístroj a metodu současně velice využívané, totiž scintilační spektrometry. Scintilační spektrometry patří do elektronických detektorů a pro naši práci je důležité, že se objevují též v terénním měření SÚRO.



Obrázek 2.2: Obecné schéma scintilačního detektoru (zdroj: [4])

### 2.1.1 Detektorová část

Ve scintilačních detektorech zachycuje ionizační záření takzvaná scintilační sonda. Ta se skládá ze scintilačního krystalu a fotonásobiče umístěných ve světlotěsném pouzdře. Pouzdro nejen že zabraňuje vniknutí nadbytečného světla do detekční jednotky, ale také stíní vnější magnetické pole, které by mohlo ovlivňovat správnou funkci fotonásobiče.

Obecná práce scintilačního detektoru spočívá v tom, že detekuje-li nějaké ionizační záření, dochází ve scintilačním krystalu ke světelným zábleskům. Emise fotonů jsou přímo úměrné absorbovanému záření. Pomocí fotoelektrického jevu je ve vstupní fotokatodě fotonásobiče převedeno fotonové kvantum na kvantum elektronové, načež uvnitř fotonásobiče dojde k zněkolikanásobení elektronů.

#### Scintilační krystal

Existují látky, takzvané scintilátory, které na pohlcení kvant ionizujícího záření reagují scintilacemi (z latinského scintilla - jiskra<sup>1</sup>, scintilace tedy představují světelné záblesky). Byť se mohou objevovat i scintilátory kapalně či plynně, zde se budeme zabývat scintilátory nejběžněji užívanými - krystaly.

<sup>1</sup>zdroj: FÜRST, Kamil a MEISSNER, Josef: *Slovník latinsko-český a česko-latinský*, Praha: Nakladatelé Kvasnička a Hampl, 1941

Krystaly bývají planárního válcového tvaru, případně pro speciálnější účely (vysokoenergetická záření, detekce měkkého záření gama) velkoobjemové, studnové či tenké. V současné době jsou krystaly nejčastěji vyráběny z jodidu sodného aktivovaného thaliem ( $NaI(Tl)$ ), dříve se též používal sirník zinečnatý aktivovaný stříbrem nebo kyanid platino-barnatý.

Pokud bychom scintilátor jen přivařili ke vstupnímu okénku fotonásobiče, docházelo by ve vzduchové vrstvě mezi výstupním okénkem scintilátoru a vstupním okénkem fotonásobiče k nechtěné ztrátě fotonů. Tento prostor tedy bývá zaplňován látkou s vysokou světelnou vodivostí, například silikonovou vazelinou. V případě většího průřezu bývají oba elementy spojeny světlovodičem či optickými vlákny.

### Fotonásobič

Fotonásobič představuje optoelektronickou součástku, která na základě fotoelektrického jevu převádí světelný tok na elektrické signály. Jeho výhoda tkví ve vysoké citlivosti způsobené několikanásobným zvětšením počtu elektronů pomocí dynod, protože lze i z jediného fotonu vyprodukovat dobře detekovatelný elektronický impuls. Fotonásobič tak navzdory svému názvu nenásobí fotony, nýbrž elektrony vyvolané prvotním dopadem fotonů na fotokatodu.

Vstupním elementem fotonásobiče je tedy fotokatoda. Jedná se o tenkou vrstvu napařenou na vstupním okénku, tvořenou zejména antimonidy alkalických prvků (nejčastěji cesia a antimonu), ona převádí dopadající fotony fotoelektrickým jevem na elektrony. Tenkost tvoří důležitou součást její výroby; v opačném případě by mohlo docházet k pohlcování emitovaných elektronů v materiálu fotokatomy.

Za fotokatodou (mezi fotokatodou a první dynodou) se může nacházet kladně nabitá mřížka. Její kladné napětí udává emitovaným elektronům vyšší rychlost a usměrňuje je na první z dynod.

Následně tyto vyloučené elektrony násobí sekundární emise na dynodách. Dynody představují soustavu zvláštních elektrod, kde se na každou následující přivádí vyšší (zpravidla o 100 V nebo dvojnásobné) kladné napětí, protože přitahují emitované elektrony. Tento systém funguje jako elektronový zesilovač. Každá z dynod pracuje tím způsobem, že po nárazu jednoho elektronu emituje další elektrony. Mezi počtem vyražených elektronů a kinetickou energií dopadajícího elektronu platí přímá úměra. Obvykle se počet elektronů na každé dynodě zdvojnásobí. Několikerým



Spektrometrická analýza vyžaduje transformaci impulsů na bitové informace; k té dochází v analogově-digitálním převodníku. V paměti počítače se každé amplitudě záření gama střídacím algoritmem přidělí buňka. Při detekci takového kvanta se její hodnota zvyšuje o 1. Takovým postupem získáme scintilační spektrum, jehož grafické znázornění je na obrázku 2.2.

## 2.2 Sběr souřadnic a potřeba posunu

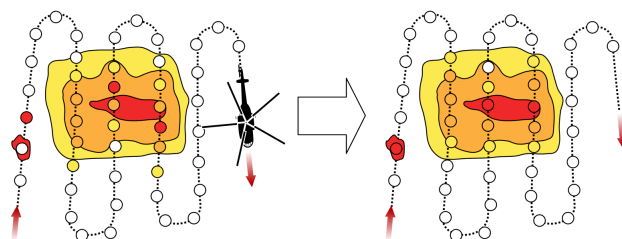
Sběr dat probíhá letecky. Scintilační spektrometr je uložen ve vrtulníku, který přelétá nad zkoumaným územím; při přeletu se v spektrometru registrují hodnoty záření. Tato data jsou registrována v podstatě neustále, ale zapisují se až po jistém časovém úseku (běžně po vteřině) jako průměr hodnot za onen časový úsek měřených. V okamžiku zápisu hodnot radionuklidů se rovněž tak zapisují zeměpisné souřadnice (a další parametry). Ty se však nijak neprůměrují, zapisují se momentální hodnoty.

Z popsaného plyne, že se registrované zeměpisné souřadnice od souřadnic geometrického průměru snímaného pole liší; jedná se o souřadnice začátku nebo konce měření v dané epoše (záleží na použitém přístroji). Vzniká tedy potřeba korekce zpoždění/předbíhání záznamu souřadnic oproti měřeným hodnotám.

## 2.3 Posun

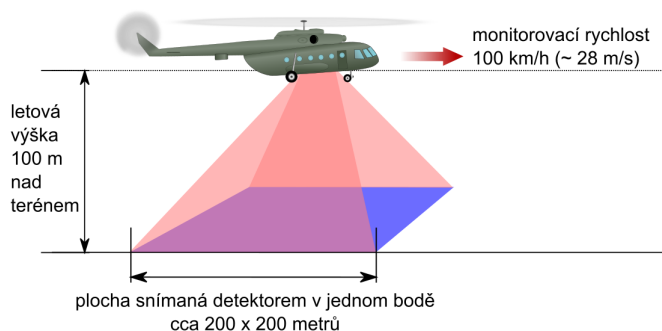
Bohužel není k dispozici záznam výšek vrtulníku během letu, ani přesný záznam jeho rychlosti, tudíž není možnost dokonalé souřadnicové korekce, různými algoritmy se ale ideální hodnotě můžeme alespoň přiblížit. Zásuvný modul umožňuje tři základní posuny měřených dat.

Nejjednodušším způsobem je posun o hodnoty. Hodnotě každého bodu přiřadíme souřadnice bodu předcházejícího či následujícího (s libovolným krokem). Takový posun ovšem nijak neposouvá jednotlivé body, alebrž jen vzájemně vyměňuje jejich souřadnice. Před vývojem zásuvného modulu byl však tento *posun* tím zdaleka nejvyužívanějším, téměř bezkonkurenčně, pro svou jednoduchost - stačilo posunout sloupce s potřebnými daty o zamýšlený počet hodnot.

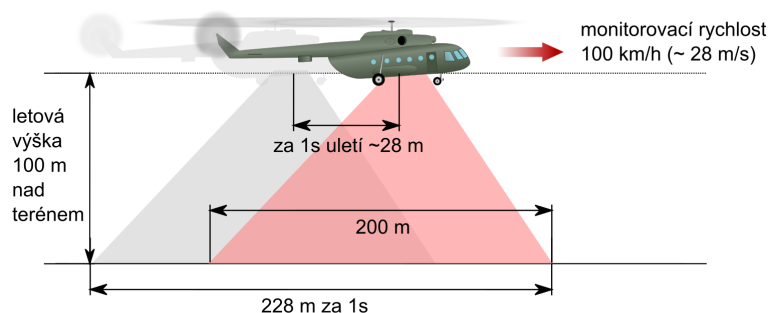


Obrázek 2.4: Posun dat o 1 bod (zdroj: HELEBRANT, Jan a GRÝC, Lubomír: *GPS position lag correction* [interní dokument SÚRO, v.v.i.]

Nápaditěji působí posun o konstantní vzdálenost. Uživatel dostane možnost posunout všechny body po trajektorii letu o jím volenou vzdálenost. Tento princip představuje již skutečný, realitu lépe vystihující posun. V ideálním případě popsaném na následujících obrázcích by posun o 14 metrů přesunul body na jejich náležité místo.



Obrázek 2.5: Princip měření (zdroj: HELEBRANT, Jan a GRÝC, Lubomír: *GPS position lag correction* [interní dokument SÚRO, v.v.i.]



Obrázek 2.6: Profilové schéma měření (zdroj: HELEBRANT, Jan a GRÝC, Lubomír: *GPS position lag correction* [interní dokument SÚRO, v.v.i.]



Skutečnosti nejméně odpovídá posun o sekundy. V dřeni se jedná o posun o vzdálenost, ale tentokrát nikoli konstantní.

Ze známých zeměpisných souřadnic se mezi každou dvojicí bodů vypočte vzdálenost. Neboť známe pro každý z bodů časovou značku, lehce (rozdíl dvou hodnot) vypočteme dobu, za niž vrtulník tuto vzdálenost urazil. Tím získáme veškeré potřebné veličiny pro výpočet rychlosti mezi dvěma body. Následuje postup opačný: Na základě uživatelem zadané doby, o niž se mají body posunout, se vypočte vzdálenost, kterou by vrtulník za danou dobu urazil. Vztáhneme-li tuto vzdálenost opět na trajektorii pohybu, obdržíme souřadnice posunutého bodu.

## 2.4 Problémy

Při tvorbě softwarového nástroje, který by jmenované operace umožňoval, může dojít k mnoha nepříjemnostem a problémům, s nimiž je třeba se vypořádat. Jmenujme alespoň dva nejdůležitější:

### 2.4.1 Elipsoid

Ačkoli různé typy spektrometrů produkují odlišné výstupy a některé mohou obsahovat i zeměpisné souřadnice na jiném referenčním tělese, zpravidla obsahují (také) souřadnice na elipsoidu WGS84.

Přítomnost elipsoidických souřadnic namísto rovinných vnáší do výpočtu záludnou překážku, na niž musí tvůrce algoritmu dbát zvýšeného zřetele. Od výpočtů na referenční kouli (nerci v rovině) se zásadně liší nejen délka, ale také azimut. Nejkratší spojnice (takzvaná geodetická křivka), která reprezentuje také trajektorii mezi sousedními body, pak samozřejmě v rovinném zobrazení nemá tvar úsečky. Majoritu těchto problémů vyřešíme užitím první základní geodetické úlohy na elipsoidu.

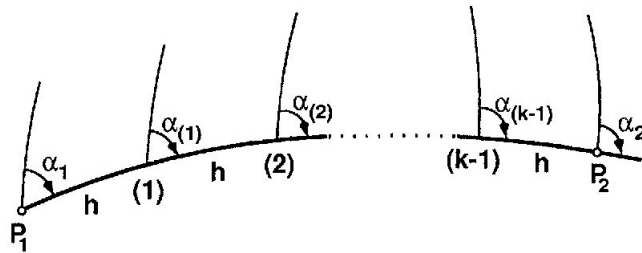
### 2.4.2 První geodetická úloha

První geodetická úloha představuje hledání (výpočet) parametrů bodu na konci křivky, v případě představovaného softwarového nástroje na referenčním elipsoidu. Známými parametry jsou souřadnice počátečního bodu křivky a její azimut v tomto bodě. Hledanými parametry zeměpisné souřadnice koncového bodu. Zkoumaná křivka reprezentuje geodetickou křivku.

Základní geodetické úlohy se dají řešit nejedním způsobem a v minulosti vždy znamenaly mnohé potíže a dlouhé výpočty. S rozvojem informačních technologií došlo k značnému urychlení výpočtu. Presentovaný princip vychází z principu popisovaného v [2] a vycházejícího z postupu Ing. Jana Douši.

Postup spočívá v rozdělení křivky na integrační kroky. Křivku, resp. její úsek  $h$  aproximujeme polynomem čtvrtého stupně, takzvanou Runge-Kuttovou metodou řádu 4. Potíž spočívá v tom, na kolik úseků máme danou křivku rozdělit. Přesnost totiž ovlivňuje nejen délka křivky, ale také její poloha na elipsoidu.

Daný problém řeší postupné iterace. Křivka se postupně půlí, následně se půlí její polovina atd. (jde tedy o  $1/n$  násobky původní délky, kde  $n$  představuje mocniny dvou), než dosáhneme požadované přesnosti. Ta je kontrolována pomocí podmínky cyklu: Pokud se od sebe všechny vypočtené atributy koncového bodu křivky ve dvou po sobě jdoucích iteracích liší o hodnotu menší než danou (v našem případě 0.000000001 úhlového stupně), jsou poslední vypočtené atributy brány jako výsledné.



Obrázek 2.7: Integrační kroky v první geodetické úloze (zdroj: [2])

## 3 Použité technologie

Třetí kapitola se zabývá užitými technologiemi, stručně představuje jejich historii a zaměření. Podrobnější informace o jednotlivých projektech lze nalézt na oficiálních internetových stránkách.

### 3.1 Python



Obrázek 3.1: Python logo (zdroj: Wikimedia Commons)

Python je open-source multiplatformní programovací interpretovaný jazyk pojmenovaný podle kultovního britského komediálního seriálu Monty Pythonův létající cirkus. Momentálně jsou vyvíjeny verze třetí řady (3.X).

První verzi, vydanou roku 1991, navrhl nizozemský programátor a inženýr Guido van Rossum za užití kódu mnoha dalších programátorů. Silnou inspiraci van Rossum našel v jazyku ABC. Odtud také plyne přívětivost jazyka, pro niž je často vyučován a doporučován jako ideální pro rychlý (a přesto kvalitní) vývoj aplikací.

Zásadní byl pro vývoj rok 2001, kdy vznikla nezisková organizace Python Software Foundation (PSF). PSF zaštiťuje další vývoj Pythonu, vlastní jeho *intelektuální jmění* a pořádá a podporuje konference ohledně Pythonu. Sám van Rossum bývá pro svůj dohled a svou neomezenou možnost zasahovat do vývoje nazýván *benevolentní doživotní diktátor*, španělská inkvizice jazyka Python.

Python se na první pohled vyznačuje odsazováním kódu. Toho se ve většině jazyků používá jen jako pomůcky pro přehlednost, v Pythonu je však odsazování povinné.

Mezi další specifikace jazyka patří například:

- Chování funkce: Ta se uchovává jako objekt. Python je obecně orientován na objektové programování.

- Proměnné: Není třeba deklarovat jejich typ. To může ušetřit značné množství práce v psaní kódu, ale je třeba si na tuto vlastnost dát pozor (celé číslo dělené celým číslem je vždy celé číslo, i kdyby správným výsledkem mělo být číslo desetinné).
- Proměnné uvnitř objektu: Není třeba je udávat při vytváření objektu. Lze je založit později.
- Dokončení zápisu: V interaktivním režimu provádíme dokončení zápisu prázdným řádkem.
- Kompilace: Python si automaticky kompiluje moduly do souborů *.pyc* či *.pyd*. Zkompilovaný modul je platformně závislý.

## 3.2 QGIS



Obrázek 3.2: QGIS logo (zdroj: Wikimedia Commons)

QGIS (zkratka dříve užívaného názvu Quantum GIS) představuje na poli geografických informačních systémů (GIS) open-source alternativu k proprietárním geografickým informačním systémům typu ArcGIS.

Na začátku 21. století už byly geografické informační systémy běžnou praxí. Zrodila se tedy potřeba volby svobodného multiplatformního systému s širokou podporou formátů geodat. Roku 2002 začal Quantum GIS vyvíjet Gary Sherman, k němuž se později připojovalo čím dál více dobrovolníků. Časem projekt zaštitilo též Open Source Geospatial Foundation (OSGeo) – organizace pro podporu a vývoj otevřených geoinformačních technologií a dat vzniklá roku 2006. Verze 1.0 byla uveřejněna v lednu 2009.

První verze systému QGIS byly pojmenovávány podle psů, později se přešlo na jména měsíců Jupiteru a Saturnu. Momentálně nesou nejnovější verze názvy měst.

QGIS se ve svém zaměření příliš neliší od běžných geografických informačních systémů. Uživatel v něm má možnost prohlížení, zpracování, tvorby a editace geodat, nad nimiž může provádět například SQL dotazy. Data v prostředí QGIS mohou být jak rastrová, tak také vektorová. Co se funkcionality týče, běžnému uživateli dostačuje, ale možností komerčně rozšířených gigantů typu ArcGIS nedosahuje.

Síla QGIS vedle volnosti užití tkví především ve značném množství veřejně přístupných zásuvných modulů – tzv. pluginů. Ty byly zprvu psány především v jazyku C++ (stejně jako základní tělo programu), nyní se čím dál častěji přechází k jazyku Python, přičemž především pro grafické uživatelské rozhraní se využívá Qt knihoven.

### 3.3 Qt Project



Obrázek 3.3: Qt logo (zdroj: Wikimedia Commons)

Qt představuje uživatelsky přívětivé multiplatformní vývojové prostředí s důrazem na grafické uživatelské rozhraní (GUI).

Vývoj Qt započali na úsvitu devadesátých let dva programátoři - Haavard Nord a Eirik Chambe-Eng. Původně dali své společnosti jméno Quasar technologies, ale brzy společnost přejmenovali na Trolltech. Pod touto hlavičkou působili až do roku 2008, kdy firmu odkoupila Nokia. V letech 2011 a 2012 pak probíhal další přesun, tentokrát pod podnik Digia. Digia nadále spolupracuje s Qt Company.

Qt není programovacím jazykem. Qt je framework psaný v C++, umožňující ale práci i v dalších jazycích včetně jazyka python (PyQt). Qt klade zesílený důraz na GUI a práci s ním. Mezi jednotlivými objekty se dá komunikovat a vykonávat nad nimi operace pomocí tzv. signálů a slotů. GUI se ukládá do souboru s příponou *.ui*. Pro náš zásuvný modul je důležitá právě tato odnož Qt poskytovaná samostatně jako Qt Designer.

## 4 Zásuvný modul

Čtvrtá kapitola popisuje vývoj zásuvného modulu a rozebírá důležité části kódu. Pro názornější popis jsou texty doplněny takzvanými pseudokódy, schematickými přehledy nejdůležitějších částí kódového zpracování; v textové části se na takové části odkazuje označením příslušného pseudokódu a řádku v závorkách. Při samotné tvorbě kódu bylo vycházeno z příručky [6].

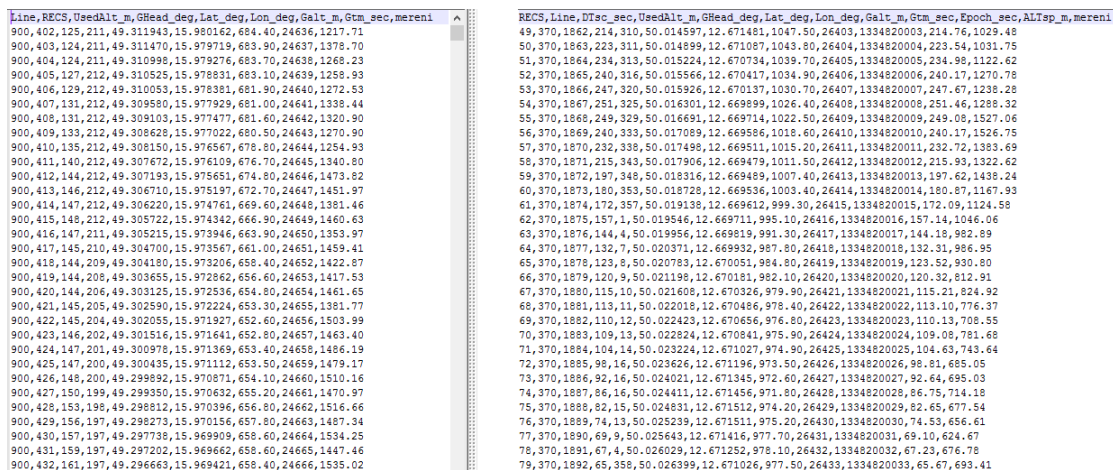
### 4.1 Obsah CSV

Vstupní soubor s měřeními daty musí být tzv. CSV (*comma-separated values* - hodnoty oddělené čárkami) soubor. Oddělování čárkami představuje pro čtení souboru zásuvným modulem zásadní podmínku. Software v měření radioaktivity (a mnohdy i jiných atributů) běžně používaný k zápisu obvykle produkuje právě soubory s koncovkou .CSV s nezbytnými i doplňujícími atributy. Na pořadí atributů nezáleží.

Povinné části a jejich označení v hlavičce souboru:

- **Lat\_deg**: Sloupec `Lat_deg` obsahuje zeměpisnou šířku snímaného bodu na elipsoidu. Za referenční elipsoid, na němž se úloha vypočítává, byl zvolen nejběžněji užívaný elipsoid WGS84.
- **Lon\_deg**: Sloupec `Lon_deg` obsahuje zeměpisnou délku snímaného bodu na elipsoidu. Za referenční elipsoid, na němž se úloha vypočítává, byl zvolen nejběžněji užívaný elipsoid WGS84.
- **Gtm\_sec**: Čas měření ve vteřinách. Není důležité, jaký typ udávání času daný soubor podporuje (zda například počítá vteřiny od nějakého roku, nebo od začátku měření), počítá se s rozestupy mezi jednotlivými údaji.
- **mereni**: `mereni` obsahuje měřenou hodnotu (v prezentovaném případě hodnotu radioaktivity, ale sloupec by mohl obsahovat jakoukoli jinou, i nečíselnou hodnotu). Zásuvný modul posune body i bez těchto hodnot, ale v primárním určení, měření radioaktivity, by bez této hodnoty postrádal smysl.
- **RECS**: Pod označením `RECS` se skrývá číslo bodu. Tato položka nezbytná není, avšak číslování bodů zajišťuje lepší čitelnost dat, protože je doporučováno užívat jej vždy.

Soubor bývá doplněn o data pro zásuvný modul zbytná, například číslo linie, souřadnice na jiném referenčním tělese, nadmořskou výšku, jiný systém času nebo datace.



Obrázek 4.1: Ukázka dvou typů vstupních souborů

## 4.2 Tělo zásuvného modulu

Základ zásuvného modulu byl vytvořen ve volně šiřitelném modulu nazvaném Plugin Builder<sup>3</sup>. Jedná se o zásuvný modul vytvořený pro QGIS především (ale nikoli pouze) suitou kolem GeoApt LLC, skupiny zaměřující se na volně šiřitelný GIS. Plugin Builder na základě uživatelem zadaných parametrů vytvoří základní skelet zásuvného modulu - ikonku, holobyt grafického rozhraní, metadata, základní funkce (zapnout, vypnout) a vazby zásuvného modulu.

Fundamentální tělo zajišťující základní funkcionalitu zásuvného modulu je uloženo v několika vzájemně provázaných souborech:

- `__init__.py`: Základní inicializace zásuvného modulu.
- `plugin_upload.py`: Zajišťuje všeobecnou dostupnost zásuvného modulu.
- `position_correction.py`: Implementuje zásuvný modul do prostředí QGIS. Načítá jeho ikonu, jmenuje a aktivuje jej a v případě vypínání se stará o jeho destrukci. Funkce `run` jej váže s `position_correction_dockwidget.py`.
- `position_correction_dockwidget.py`: Nejdůležitější ze základních souborů zásuvného modulu. Volání souboru `position_correction_dockwidget_base.ui`

<sup>3</sup>Dostupný na <https://plugins.qgis.org/plugins/pluginbuilder/>

vytvořeného v prostředí Qt Designer zde vyvolává grafické uživatelské rozhraní. Zajišťuje také funkčnost veškerých jeho částí a celkové dodržování vnitřního řádu, od načítání cest vstupních a výstupních dat přes jejich zobrazování a stylování (volá `show_as_layer.py`) až po samotné provedení posunu (`move.py`). Na pozadí dochází ke kontrole vstupujících údajů, v případě nesrovnalostí bude uživatel upozorněn chybovým hlášením. Aby se zamezilo zbrklým nehodám, byly ke zpřístupnění tlačítek zabudovány podmínky - ke zpřístupnění tlačítka na zobrazení vstupního souboru musí být k tomuto souboru definována cesta, taktéž k samotnému posunu je třeba mít vyplněny všechny potřebné parametry.

- `show_as_layer.py`: Zobrazí soubor, an je předán jako jeden ze vstupních parametrů. Druhým vstupním parametrem je styl, jenž bude při zobrazování použit. Aby se zamezilo gravitě kódu a obecné duplicitě kódu v podhoubí prostředí QGIS, byly k zobrazení užity interní objekty `QgsVectorLayer` a `QgsMapLayerRegistry`.

Toliko k hrubému náčrtu fungování zásuvného modulu. Některé dílčí složky byly při představování zanedbány, poněvadž nejsou k pochopení fungování modulu nezbytné. Nyní se zvláště podívejme na soubory související se zkoumanou problematikou, tedy obsah `move.py`.

### 4.3 Posun o hodnoty



Obrázek 4.2: Ukázka - Posun o 3 hodnoty (červeně vyznačeny posunuté body)

Posun o hodnoty se skrývá pod metodou `by_points` třídy `Move`. Zde nejprve proběhne analýza vstupních dat. Analýza spočívá v prohledání hlavičky souboru, je třeba zjistit polohu hledaných hodnot - sloupce `Lat_deg` a `Lon_deg`.



Následně se čte vstupní soubor řádek po řádku a postup posunu se liší na základě vstupní hodnoty posunu.

### 4.3.1 Posun o kladný počet hodnot

Jedná-li se o číslo kladné, připojujeme k informacím bodu  $n$  souřadnice bodu  $n+x$ , kde  $x$  představuje hodnotu, o niž body posouváme. Toho dosáhneme tím, že v každé smyčce cyklu postupně načteme  $x$  řádků, přičemž první z těchto řádků vložíme do proměnné (zbylé se nikam neukládají a v paměti tak nezůstávají - zůstává vždy pouze poslední čtený); v paměti tedy zůstane první a poslední z čtených řádků (pseudokód 1-1, 4). Hodnoty ze sloupců souřadnic posledního načteného řádku vložíme do příslušných sloupců prvního čteného řádku (pseudokód 1-5). Modifikovaný první řádek se zapíše do výstupního souboru (pseudokód 1-6). Načte se poloha konce prvního řádku (pseudokód 1-7) a cyklus se může opakovat.

---

#### Pseudokód 1 Posun o kladné hodnoty

---

- 1: posunovanyradek=přečti první řádek
  - 2: **while** existuje posunovanyradek **do**
  - 3:   pozice=ulož pozici ve vstupním souboru
  - 4:   přečti  $x$  řádků
  - 5:   vlož souřadnice posledního čteného řádku do posunovanyradek
  - 6:   zapiš posunovanyradek
  - 7:   seek(pozice)
  - 8:   posunovanyradek=přečti řádek
  - 9: **end while**
- 

### 4.3.2 Posun o záporný počet hodnot

Jedná-li se o číslo záporné, ještě před cyklem se provede odečtení  $x$  řádků, kde  $x$  je hodnota posunu; tyto řádky jsou zapsány do seznamu hodnot (pseudokód 2-1).

Syžet cyklu následně probíhá tím způsobem, že se z tohoto seznamu odebírání první hodnota. Z té jsou odečteny informace ze sloupců příslušejících zeměpisným souřadnicím. Do dočasné proměnné je vložena poslední hodnota seznamu (pseudokód 2-3) a její souřadnice jsou přepsány souřadnicemi výše zmiňovanými (pseudokód 2-4). Modifikovaný řádek se zapíše do výstupního souboru. Nyní je třeba doplnit seznam

opět na  $x$  hodnot, toho se dosáhne čtením dalšího řádku ze vstupního souboru a jeho vložení na konec seznamu (pseudokód 2-8). Cyklus se může opakovat.

---

**Pseudokód 2** Posun o záporné hodnoty

---

- 1: seznam=přečti  $x$  řádků
  - 2: **while** existuje hodnota seznam[ $x$ ] **do**
  - 3:   posunovanyradek=seznam[ $x$ ]
  - 4:   vlož souřadnice seznam[0] do posunovanyradek
  - 5:   zapiš posunovanyradek
  - 6:   vymaž seznam[0]
  - 7:   seek(pozice)
  - 8:   načti do seznamu další řádek
  - 9: **end while**
- 

### 4.3.3 Posun o nulový počet hodnot

Ve speciálním případě, kdy vstupuje do posunu o hodnoty číslice 0, dochází k pouhému kopírování vstupního souboru do souboru výstupního. Ovšem na žádný pád nemá v takovém případě použití zásuvného modulu odůvodnění.

## 4.4 Posun o konstantní vzdálenost



Obrázek 4.3: Ukázka - Posun o 9.823 metrů (červeně vyznačeny posunuté body)

Posun o konstantní vzdálenost je obsažen v `Move.by_distance`.

Na začátek funkce je třeba opět základní analýza vstupujícího souboru; její průběh je shodný s analýzou při posunu o hodnoty (viz 4.3).

Následně je třeba připravit si pole pro výpočty na elipsoidu. Redundance kódu není v ničem zájmu, pro potřeby elipsoidických operací tedy užíváme knihovnu prostředí QGIS `QgsDistanceArea`. Zde nastavíme užitý referenční elipsoid WGS84.

Výpočet se opět liší dle povahy vstupující hodnoty posunu (kladná vzdálenost, záporná, 0). Posledním krokem před rozlišením, kterou větev výpočtu volání funkce vyvolá, je nastavení parametrů elipsoidu pro výpočet první geodetické úlohy (ta v `QgsDistanceArea` implemetována není, a je třeba ji řešit v samotném kódu). Pro její obecný popis viz 2.4.2, kódové zpracování bude popsáno později v 4.4.4

#### 4.4.1 Posun o kladnou vzdálenost

Posunujeme-li body o kladnou vzdálenost, na první pohled se nezdá, že by se v algoritmu skrývaly nějaké záludnosti. Na pohled druhý ano.

Nejzásadnější problém se vynoří, snaží-li se uživatel posunout body o vzdálenost větší, než jaká se rozpíná mezi dvěma body. V takovém případě je třeba v momentě překročení zmiňované vzdálenosti přepočítat trajektorii na novou dvojici bodů a o tuto hodnotu požadovanou délku posunu snížit.

Z tohoto důvodu proběhne v každém cyklu čtení  $x$  řádků ve vnořeném cyklu *while* (pseudokód 3-4 až 11). Tentokrát není  $x$  daná hodnota, její velikost závisí na rozestupy mezi jednotlivými body. V každé smyčce vnitřního *while* cyklu dojde k výpočtu vzdálenosti mezi dvěma momentálně načtenými body. Je-li vypočtená vzdálenost větší než délka posunu, cyklus skončí (pseudokód 3-4). Avšak není-li, délka posunu se zkrátí o vzdálenost mezi těmito body (pseudokód 3-5) a bodem 2 se nahradí bod 1 (pseudokód 3-6). Vzápětí se čte další řádek představující nový bod 2 (pseudokód 3-7) a cyklus se opakuje. V momentě, kdy skončí, máme tedy novou dvojici bodů a zbytek z posunu náležející na tento úsek. Výpočet první geodetické úlohy tedy probíhá až v takto předpřipraveném prostředí.

---

#### Pseudokód 3 Posun o kladnou vzdálenost, vnitřní cyklus

---

```

1: radek2=radek
2: vzdalenost=vzdálenost mezi posunovanyradek a radek
3: delkaposunu=uživatелеm volená hodnota posunu
4: while delkaposunu>vzdalenost do
5:   delkaposunu=delkaposunu-vzdalenost
6:   radek1=radek2
7:   radek2=přečti řádek
8:   if existuje radek2 then

```

---

---

```

9:     vzdalenost=vzdálenost mezi radek1 a radek2
10:  end if
11: end while

```

---

Kromě zeměpisných souřadnic je ale třeba veškeré ostatní informace převzít z prvního čteného řádku (jediný, který kromě momentálního páru bodů zůstává v paměti). Takto vytvořený řádek se zapíše do výstupního souboru. Načte se poloha konce prvního čteného řádku a vnější cyklus se může opakovat.

---

#### **Pseudokód 4** Posun o kladnou vzdálenost, vnější cyklus

---

```

1: posunovanyradek=přečti řádek
2: pozice=ulož pozici ve vstupním souboru
3: while existuje posunovanyradek do
4:   seek(pozice)
5:   radek=přečti řádek
6:   pozice=ulož pozici ve vstupním souboru
7:   if existuje radek then
8:     vnitřní cyklus, viz pseudokód 3
9:     výpočet první geodetické úlohy
10:    vlož vypočtené souřadnice do posunovanyradek
11:    zapiš posunovanyradek
12:    posunovanyradek=radek
13:   end if
14: end while

```

---

### 4.4.2 Posun o zápornou vzdálenost

Vstupuje-li do posunu o konstantní vzdálenost záporná hodnota, nabízí se možnost využití kódu pro kladnou hodnotu, pouze s inverzně vyměněnými body. Opět bychom ale narazili ve chvíli, kdy posun překračuje vzdálenost mezi dvěma body.

Zde je zapotřebí jiného způsobu řešení problému. Tento úkol ztěžuje ještě fakt, že nechceme zahltit paměť načítáním celého vstupního souboru, snažíme se tedy pracovat jen s nutně potřebnou dávkou řádků.

Opět si vypomůžeme seznamem hodnot (řádků). Ponejprve byl vytvořen vstupní seznam na stejném principu, jako v předchozím případě (porovnávání *uražené*

vzdálenosti se vzdáleností mezi body), s tím rozdílem, že tentokrát již použité body vkládáme do seznamu hodnot (pseudokód 5-5 až 9).

---

**Pseudokód 5** Posun o zápornou vzdálenost, vytvoření vstupního seznamu

---

```

1: seznam=přečti 2 řádky
2: x=1
3: delkaposunu=algolutní hodnota uživatelem volené hodnoty posunu
4: urazenavzdalenost=vzdálenost mezi seznam[0] a seznam[1]
5: while delkaposunu>urazenavzdalenost do
6:   načti do seznamu další řádek
7:   x=x+1
8:   urazenavzdalenost=urazenavzdalenost+délka mezi seznam[x-1] a seznam[x]
9: end while
  
```

---

Základní cyklus, jenž by se poznovu dal nazvat cyklem vnějším, se skládá ze dvou na sebe navazujících cyklů vnitřních.

První vnitřní cyklus slouží k revizi, zda není seznam vstupující do výpočtu zbytečně obsáhlý. Zjistí-li, že posun nepřekročí více než  $x$  prvků (pseudokód 6-5), smaže redundantní prvky z počátku seznamu (posouváme bod z konce seznamu). Na každý pád vstupuje do výpočtu seznam o rozsahu právě onoho  $x$ .

Po dobrozdání výpočet vstupuje do cyklu spočívajícího ve výpočtu první geodetické úlohy. Nadto do tohoto cyklu vstupují hodnoty kvůli posunu zpět v reversním pořádku. Měl-li by posun překročit vzdálenost mezi dvěma body, kvůli úspoře se výpočet přeskočí a do pomocných proměnných se zapíší přímo souřadnice druhého z bodů.

Po smyčce, při kteréž již dojde k výpočtu první geodetické úlohy (dojde k ní mezi prvními dvěma body seznamu - pseudokód 6-10), je proveden zápis do výstupního souboru - zapíše se poslední řádek ze seznamu se zeměpisnými souřadnicemi přepsanými souřadnicemi vypočtenými (pseudokód 6-11).

---

**Pseudokód 6** Posun o zápornou vzdálenost, hlavní cyklus

---

```

1: while poslední hodnota seznamu není prázdný řádek do
2:   urazenavzdalenost=0
  
```

---

---

```

3:  for i=obrácená řada od 1 do [velikost seznam] do
4:      urazenavzdalenost=urazenavzdalenost+délka mezi seznam[i-1] a seznam[i]
5:      if delkaposunu<=urazenavzdalenost then
6:          smaž body na pozici menší než i
7:      end if
8:  end for
9:  posun=delkaposunu-(součet vzdáleností mezi body seznamu[1 až i])
10: první geodetická úloha mezi seznam[1] a seznam[0], vzdálenost=posun
11: vložíme vypočtené hodnoty do seznam[i] a zapíšeme jej
12: načteme na konec seznamu další řádek
13: end while

```

---

### 4.4.3 Posun o nulovou vzdálenost

Ježto by měl zásuvný modul fungovat i při operacích, které kterakolivěk postrádají důvod k jeho využití, byl do jeho vnitřností implementován i posun o nulovou vzdálenost. V takovém případě se však vážně nejedná o posun, nýbrž o pouhé přepsání neupravených hodnot ze vstupního souboru do souboru výstupního.

### 4.4.4 První geodetická úloha

Velký problém při výpočtech na elipsoidu představuje první geodetická úloha. Ta byla popsána již výše v kapitole 2.4.2, nyní tedy pouze několika poznámek k jejímu kódovému zpracování.

Samotný kód posunů se pro své značné množství nutných podmínek a cyklů nevyznačuje elegantní přehledností, pročež byl alespoň výpočet první geodetické úlohy vytržen jako zvláštní funkce.

Výpočet probíhá na základě iterací. Anžto byl jejich postup již výše popsán, jen zdůrazníme některé části. Každá z iterací zpřesňuje výpočet a tyto iterace jsou prováděny do doby, kdy přesnost dosáhne 0.0000000001 radiánu.

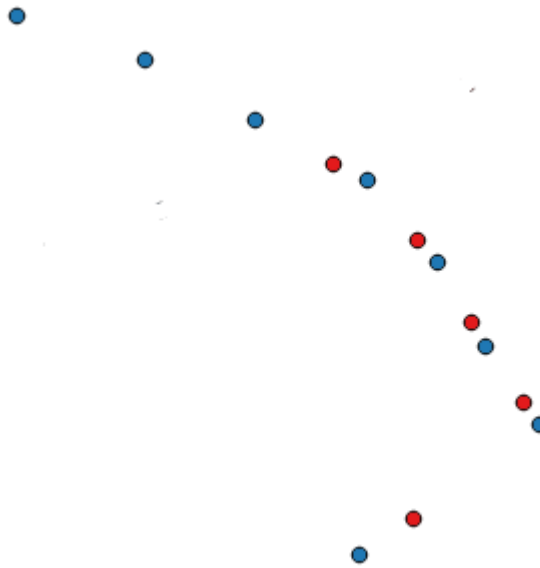
Důležitým prvkem, který by neměl zůstat opomenut, jest vstupující azimut. Zde se musí pamatovat na to, že azimut na elipsoidu se od azimutu na kouli či v rovině liší. Z tohoto důvodu byla využita funkce `computeDistanceBearing` knihovny `QgsDistanceArea`.

Nyní by bylo vhodné zdůvodnit právě tento druh výpočtu první geodetické úlohy a nepoužití složitější, ačť přesnější metody interpolace.

Chtěli-li bychom vytvořit kód, který vykresluje trajektorii pravděpodobněji, musel by do výpočtu vstupovat kompletní vstupní soubor; a vzhledem k tomu, že není ničím neobvyklým soubor čítající mnoho přes 10000 záznamů o více než desítky sloupců, rozdíl v systémové zátěži by byl obrovský.

Přesto byl tento návrh zpracování na poradě se Státním ústavem radiační ochrany vznesen. Po projednání bylo ze strany SÚRO uvedeno, že vyhovuje námi momentálně používaný způsob výpočtu; a to nikolivěk pouze z důvodu úspory systémových nároků, nýbrž především z toho důvodu, že výsledné zpřesnění souřadnic už by mělo na výsledky jejich práce prakticky nulový vliv.

## 4.5 Posun o konstantní čas (proměnnou vzdálenost)



Obrázek 4.4: Ukázka - Posun o 2.718 sekund (červeně vyznačeny posunuté body)

Velká část popisu kódu v kapitole 4.4 se týká též posunu o konstantní čas - o vzdálenost s uvážením rychlosti. Odkazujeme se zde na základní přípravu hrací plochy, analýzu, parametry elipsoidu a vyvolání knihovny `QgsDistanceArea`. Též výpočet první geodetické úlohy se shoduje s výše popsány postupy. Co se problémů jednotlivých segmentů tyče, dá se říci, že se dědí problémy posunu o konstantní vzdálenost, pouze se k nim přidávají některé nové.

Též základní fungování částí funkce se od těch popsaných výše příliš neliší; popsány tedy budou pouze případné zásadnější odlišnosti.

Nejočividnější rozdíl je pro veškeré posuny o čas stejný - výpočet vzdálenosti, o níž se bod posouvá. Nejdříve je zapotřebí funkcí `computeDistanceBearing` vypočítat vzdálenost mezi dvojicí bodů. Vydělíme-li ji časovým úsekem, za který tuto vzdálenost měřicí zařízení urazilo, získáme momentální rychlost. K požadované délce posunu se dostaneme již pouhým vynásobením této rychlosti uživatelem zadaným časovým úsekem.

Z uvedeného vyplývá, že je mezi dvojicí bodů předpokládán lineární pohyb. Taková idea vzešla již od Státního ústavu radiační ochrany na základě toho, že vrtulník začíná měřit až po jisté uražené vzdálenosti. V tomto momentě již nabyt rychlost, kterou se snaží během celého procesu měření konstantně udržovat.

Zásadním problémem, který se při zpracování vynořil, je nejednotnost měření. Ačkolivěk bývá zvykem měřit data po jedné sekundě, nedá se na takový postup spoléhat. Používal-li by zásuvný modul cizinec, není vyloučené, že by v jeho zemi bylo území měřit data po vteřinách třech nebo pěti. Je tedy třeba pracovat s časovými značkami ze sloupce `Gtm_sec` a jejich rozdíly. Tím dochází k zesložitému kódu, ale rovněž k potřebnému podchycení případných nesrovnalostí.

Též je třeba ověřovat body vstupující do výpočtu první geodetické úlohy. Má-li měřicí zařízení omezenou přesnost, může se stát, že budou mít dva po sobě měřené body zapsány stejné souřadnice. Pokud by k takové situaci došlo, je třeba se vyhnout výpočtu geodetické úlohy. Problém ani tak nespočívá v azimutu, jenž by mohl nabývat jakékoli hodnoty, nýbrž ve vzdálenosti. Dělení nulou by vyvolávalo ve výpočtech chybu. Anžto vzdálenost mezi body jest nulová, nastane zápis souřadnic jednoho z bodů.

#### 4.5.1 Posun o kladný časový úsek

Při posunu o kladný časový úsek je třeba vytvořit si pomocnou proměnnou, nazvěme ji zde  $x$ , představující čas, o který momentální bod posouváme. Ve chvíli vytvoření tedy bude  $x$  rovno uživatelem zadané hodnotě posunu.

Před základním cyklem dojde k odečtení a uložení prvního řádku. Ten představuje posouváný bod. V základním cyklu pak algoritmus čte další řádek. Ten představuje bod, k němuž směřuje trajektorie, po níž první bod posouváme.



V tomto vnějším cyklu opět pracuje cyklus vnitřní, v kterémžto dochází k porovnávání našeho  $x$  s dobou, za kterou měřicí zařízení urazilo vzdálenost mezi právě načtenou dvojicí bodů (pseudokód 7-10 až 15).

Je-li  $x$  menší než tato doba, proběhne výpočet první geodetické úlohy na trajektorii a s uvážením rychlosti mezi těmito dvěma body (pseudokód 7-11).

Pokud se tyto hodnoty rovnají, jedná se o podmínky pro výpočet ještě příznivější. Dojde k pouhému zápisu souřadnic druhého bodu do příslušných proměnných. V takovém případě by se posun dal označit za posun o hodnoty.

Zákeřněji působí možnost, při níž je hodnota  $x$  větší než doba mezi měřeními dvou bodů v momentální paměti. Pak je zapotřebí dobu posunu  $x$  zmenšit o dobu, kterou vrtulník mezi dvěma body urazil (pseudokód 7-12), druhý bod vložit do bodu prvního (pseudokód 7-13) a načíst nový druhý bod (pseudokód 7-14). Následně se vnitřní cyklus zabývající se dobrozdáním ohledně hodnoty  $x$  a hodnoty rozdílu časových značek mezi dvěma body v paměti může opakovat.

Poté, co jednou z možných cest skončí vnitřní cyklus, dojde k vložení nových souřadnic (ať už vypočtených, nebo jen vyjmutých) do odpovídajících sloupců prvního čteného řádku. Ten se zapíše do výstupního souboru (pseudokód 7-16), do proměnné  $x$  se opět načte uživatelem volená hodnota posunu, ve čtení se vyvolá pozice před započítáním vnitřního cyklu, a vnější cyklus se opakuje s řádky o jeden posunutými (pseudokód 7-4 až 19).

---

#### **Pseudokód 7** Posun o kladný čas

---

```

1: posunovanyradek=přečti řádek
2: radek1=posunovanyradek
3: pozice=ulož pozici ve vstupním souboru
4: while existuje posunovanyradek do
5:   x=uživatelem zadaná hodnota posunu
6:   seek(pozice)
7:   nasledujiciradek=přečti řádek
8:   radek2=nasledujiciradek
9:   pozice=ulož pozici ve vstupním souboru
10: while x>0 do
11:   vypočti první úlohu mezi radek1 a radek2
12:   x=x-(doba uplynulá mezi měřeními radek1 a radek2)

```

---

---

```

13:   radek1=radek2
14:   radek2=přečti řádek
15:   end while
16:   vlož vypočtené souřadnice do posunovanyradek a zapiš jej
17:   posunovanyradek=nasledujiciradek
18:   radek1=posunovanyradek
19: end while
  
```

---

### 4.5.2 Posun o záporný časový úsek

Pro posun o záporný časový úsek se jako ideální řešení opět zdálo využití seznamu řádků. Tento seznam byl načten před samotným započítáním vnějšího cyklu a hodnoty do něj byly přidávány do té doby, dokud součet rozdílů časových značek mezi jednotlivými body nepřekročí uživatelem požadovanou hodnotu posunu (lépe řečeno její absolutní hodnotu, poněvadž uživatel volí hodnotu zápornou). Tvoření seznamu naznačuje pseudokód 7-5 až 9).

---

#### Pseudokód 8 Posun o záporný čas, vytvoření vstupního seznamu

---

```

1: seznam=přečti 2 řádky
2: x=1
3: casuzivatel=absolutní hodnota uživatelem volené hodnoty posunu
4: ubehlycas=rozdíl časových značek seznam[0] a seznam[1]
5: while casuzivatel>ubehlycas do
6:   načti do seznamu další řádek
7:   x=x+1
8:   ubehlycas=ubehlycas+rozdíl časových značek seznam[x] a seznam[x-1]
9: end while
  
```

---

Seznam již vstupuje do vnějšího cyklu, jehož cílem je postupně projít celý vstupní soubor a posunout veškeré body. Tento cyklus se skládá ze dvou cyklů vnitřních.

První vnitřní cyklus reviduje vstupující seznam hodnot. Zjišťuje, zda není příliš obsáhlý. Kontrolu provádí na základě porovnávání součtu rozdílů časových značek s uživatelem voleným časovým posunem (pseudokód 9-3 až 8). Narazí-li algoritmus na hodnoty, jež by nebyly při výpočtu využity (pseudokód 9-5 až 7), ze seznamu je kvůli úspoře paměti umaže (protože se nové hodnoty - posouvající se

body - přidávají na konec seznamu, nadbytečné hodnoty se nachází na jeho začátku). Tyto řádky totiž nebudou třeba k žádné z následujících smyček.

Druhý z vnitřních cyklů se příliš neliší od toho popsaného v kapitole 4.4.2. Čte v reversním pořadí seznam hodnot a spočívá v analýze vstupujících dat. Analýza probíhá s pomocnou proměnnou (v textu si ji označme  $x$ ), která představuje čas posunu od momentálního bodu.

Pokud  $x$  převyšuje dobu, za niž měřicí zařízení urazilo vzdálenost mezi dvěma načtenými body, pouze se posuneme v seznamu o úroveň níže (bod 1 se stane bodem 2 a do nového bodu 1 se vloží další řádek seznamu);  $x$  se zmenší o velikost zmiňovaného časového úseku.

Rovnají-li se tyto dvě hodnoty, do paměti se uloží souřadnice bodu 1 (bodů, k němuž se počítá trajektorie).

V případě, že je absolutní hodnota  $x$  menší než rozdíl časových značek, dochází k výpočtu první geodetické úlohy. Pseudokódy jsou jen schematické, poslední kroky tedy shrnuje pseudokód (pseudokód 9-9 a 10)

Následuje chvíle pro již typickou koncovku vnějšího cyklu. Souřadnice z paměti se vloží do odpovídajících sloupců řádku s posunovaným bodem a ten se zapíše do výstupního souboru (pseudokód 9-11). Následně se na konec seznamu hodnot načte další řádek (nový posunovaný bod) a  $x$  se nastaví na uživatelem volenou hodnotu posunu. Cyklus pokračuje další smyčkou.

---

#### **Pseudokód 9** Pseudokód - Posun o záporný čas, hlavní cyklus

---

```

1: while poslední hodnota seznamu není prázdný řádek do
2:   ubehlycas=0
3:   for i=obrácená řada od 1 do [velikost seznam] do
4:     ubehlycas=rozdíl časových značek seznam[0] a seznam[1]
5:     if casuzivatel<=ubehlycas then
6:       smaž body na pozici menší než i
7:     end if
8:   end for
9:   casposunu=casuzivatel-(rozdíl časových značek seznam[i] a seznam[1])

```

---

---

```

10: první geodetická úloha mezi seznam[1] a seznam[0], vzdálenost vypočtená z
    casposunu
11: vložíme vypočtené hodnoty do seznam[velikost seznam-1] a zapíšeme jej
12: načteme na konec seznamu další řádek
13: end while
  
```

---

### 4.5.3 Posun o nulový časový úsek

Byť živou věcí neexistuje důvod k použití zásuvného modulu za účelem takové operace, dovede si zásuvný modul poradit i s případem posunu o 0 sekund. Abychom se vyhnuli naprosto zbytečné zátěži ve formě počítání nulových posunů, poznovu tato část kódu funguje jen jako kopírování vstupního souboru do souboru výstupního.

## 4.6 Licence

Nebtě byl zásuvný modul vytvořen pro QGIS a využívá jeho knihoven, dědí jeho licenci: GNU General Public License (GPL).

Jde o *copyleftovou* (odvozené dílo musí být dostupné pod stejnou licencí) licenci. Ve své podstatě zde nejde o omezování svobody, alebrž právě o její dodržování - zajišťuje, aby prvotně svobodný software tuto svobodu modifikacemi či přidáváním neztratil.

## 5 Závěr

Cílem této bakalářské práce byla tvorba softwarového nástroje umožňujícího posun letecky měřených bodů po trajektorii a jeho implementace do prostředí QGIS jako tzv. zásuvného modulu.

Celý projekt se zakládal na pracích Státního ústavu radiační ochrany (SÚRO). Ten dosud prováděl posun tak, že každému bodu pouze přiřadil souřadnice bodu předcházejícího či následujícího (s libovolným krokem). Přihlédneme-li k bouchoři, kterým podobný posun o hodnoty jest, můžeme říci, že bylo zadání nejen splněno, ale na základě návrhů SÚRO dovedeno ještě dále.

V současnosti zásuvný modul umožňuje posun dat nikoli jen o hodnoty, nýbrž také o konstantní vzdálenost či o čas (tj. o vzdálenost s uvážením rychlosti). V grafickém rozhraní si může uživatel též zvolit styl, který by chtěl pro zobrazení vstupních a výstupních dat použít.

V čase budoucím se nabízejí některá rozšíření stávající funkcionality. Počítá se s implementací více stylů běžně užívaných při zpracování. Bylo by rovněž vhodné dát uživateli možnost zvolit referenční elipsoid ad libitum; momentálně probíhají veškeré výpočty na elipsoidu WGS84. Veškerá dostupná data sice obsahují zeměpisné souřadnice právě na tomto elipsoidu, nelze ale vyloučit, že se v cizině vyskytují i nějaké adventivní případy.

Zásuvný modul byl, zatím pod hlavičkou *experimentální*, uveřejněn v repositáři OSGeoREL (<http://geo.fsv.cvut.cz/osgeorel/qgis-plugins.xml>). V SÚRO probíhá již od dubna 2016 jeho testování.

## Seznam zkratek

SÚRO	Státní ústav radiační ochrany
PSF	Python Software Foundation
GIS	Geografický informační systém
OSGeo	Open Source Geospatial Foundation
SQL	Structured Query Language
GUI	Grafické uživatelské rozhraní (Graphical user interface)
CSV	Hodnoty oddělené čárkami (Comma-separated values)
WGS84	Světový geodetický systém 1984 (World Geodetic System 1984)
GPL	Všeobecná veřejná licence (General Public License)

## Literatura

- [1] MENKE, Kurt et al. *Mastering QGIS*. Packt Publishing, 2015. ISBN 978-1-78439-868-2.
- [2] MERVART, Leoš a CIMBÁLNÍK, Miloš. *Vyšší geodézie 1*. Praha: Vydavatelství ČVUT, 2002. ISBN 80-01-02527-6.
- [3] PILGRIM, Mark. *Dive Into Python*. Apress, 2004. ISBN 978-1-59059-356-1.
- [4] ULLMANN, Vojtěch. *Detekce a spektrometrie ionizujícího záření* [online]. [cit. 2016-04-08]. Dostupné z: <http://astronuklfyzika.cz/DetekceSpektrometrie.htm>.
- [5] ŠVEC, Jan. Učebnice jazyka Python (aneb Létaující cirkus), 2002. Dostupné z: <http://www.root.cz/knihy/ucebnice-jazyka-python/>. Česká verze knihy Python tutorial od Guida van Rossuma a Freda L. Drakea.
- [6] *PyQGIS Developer Cookbook* [online]. [cit. 2016-04-08]. Dostupné z: [http://docs.qgis.org/testing/en/docs/pyqgis\\_developer\\_cookbook/](http://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/).
- [7] *Státní ústav radiační ochrany, v.v.i.: Odborná činnost ústavu* [online]. [cit. 2016-04-08]. Dostupné z: <http://www.suro.cz/cz/suro/odborna-cinnost-ustavu>.
- [8] *Státní ústav radiační ochrany, v.v.i.: Výzkum v SÚRO a jeho hlavní orientace* [online]. [cit. 2016-04-08]. Dostupné z: <http://www.suro.cz/cz/vyzkum>.

# A User guide

This user guide is written for plugin using in QGIS 2.12. There is a possibility that usage of the plugin can change across other versions of QGIS.

## A.1 Loading of plugin

There are many ways to install the plugin. The easiest method is to install it from the QGIS Plugin repository.

Firstly, you have to open the plugins dialog – select *Manage and install plugins* from *Plugins* tab. Plugin is now registered as experimental, to see it select *Settings* tab and tick the checkbox *Show also experimental plugins*.

Search for *GPS Position Lag Correction* plugin in the list on the *All* or *Not installed* tab. Select it and press the *Install* button.

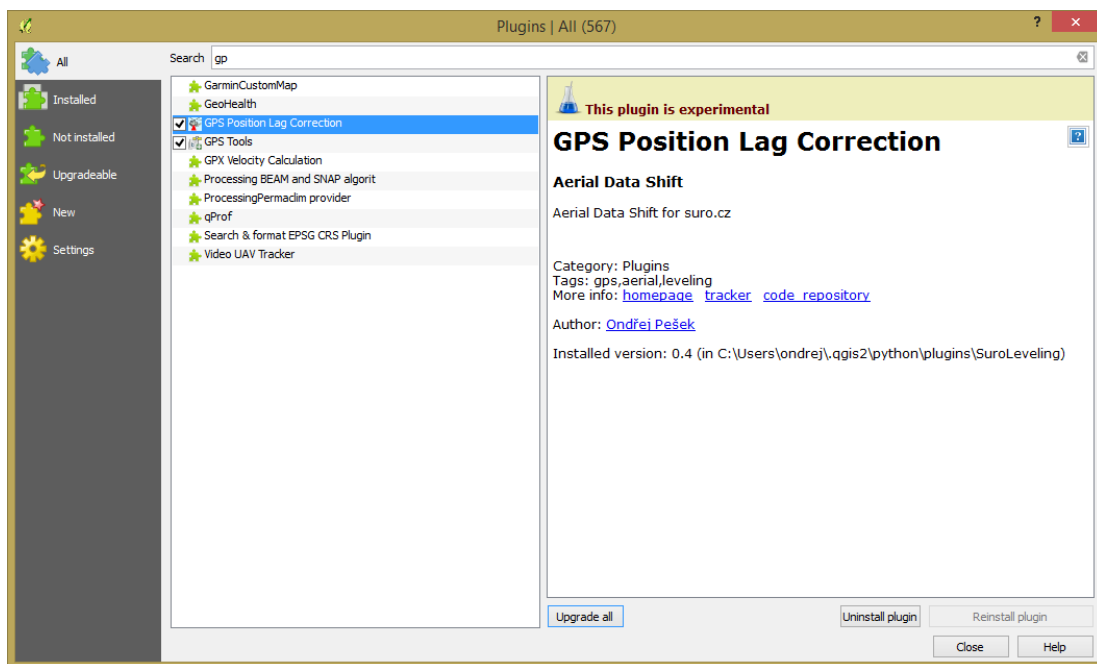


Figure A.1: Plugins dialog

## A.2 Work with plugin

In this section will be described graphical user interface and primary functionality of the plugin.



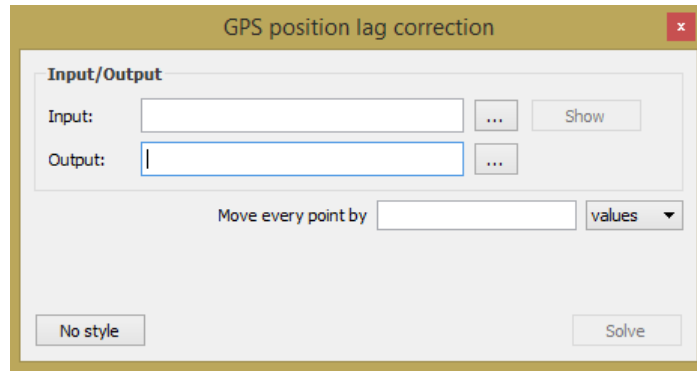


Figure A.2: GUI of the plugin

### A.2.1 Input and output defining

As input, you have to define the file you want to work with. You can write the path to this file manually or there is the button with . By clicking on this button, you will get the dialog intended to choose your file. Click on OK will insert this path to the basic interface. Click on Cancel will interrupt the choosing dialog, and the input path will not be changed.

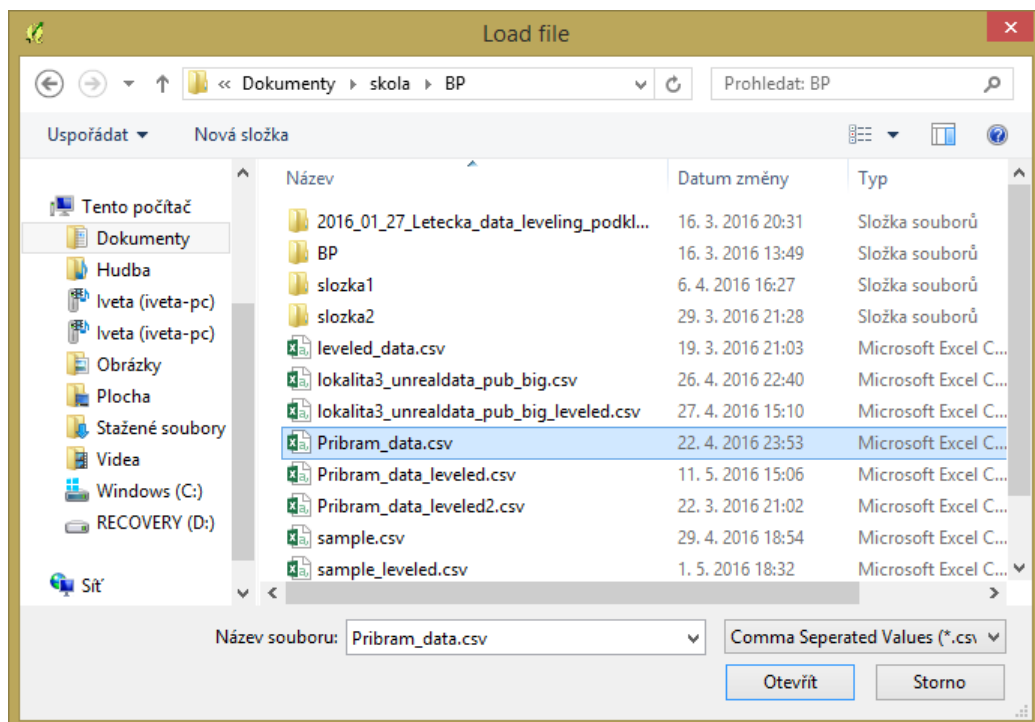


Figure A.3: Loading input

The same procedure is with output, but there you define the path where the new file will be created. The choosing dialog for output is intended to choose folder, not file.

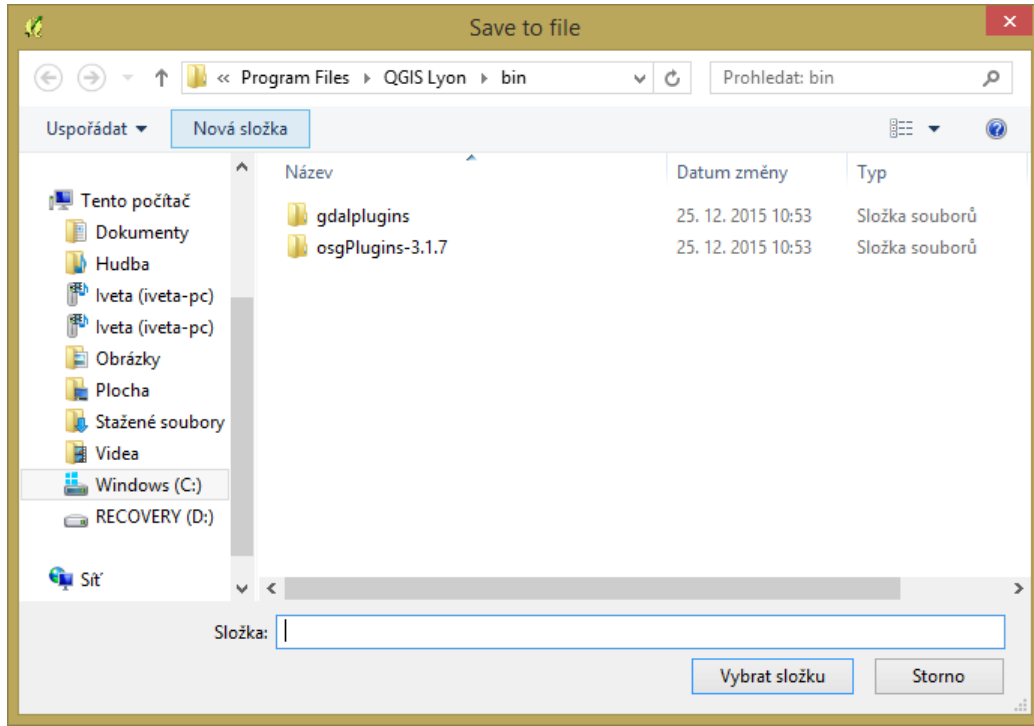


Figure A.4: Choosing output directory

### A.2.2 Styling

There is also optional possibility of styling your points for better visualisation.

By default, you have not defined any style and points will be created in default QGIS style (on Style button is written *No style*) If you want your own style, you have to click on this button. You will get the browsing dialog. You are automatically directed to folder *styles* in the plugin folder; here you can choose qml file and click OK. Click on Cancel interrupts the dialog and set again *No style*.

If you choose your own style, you will see its name on the style button.

In the default version of plugin there are two presetted styles. Styling in those styles is based on column *mereni* – one style for higher and one style for lower values.

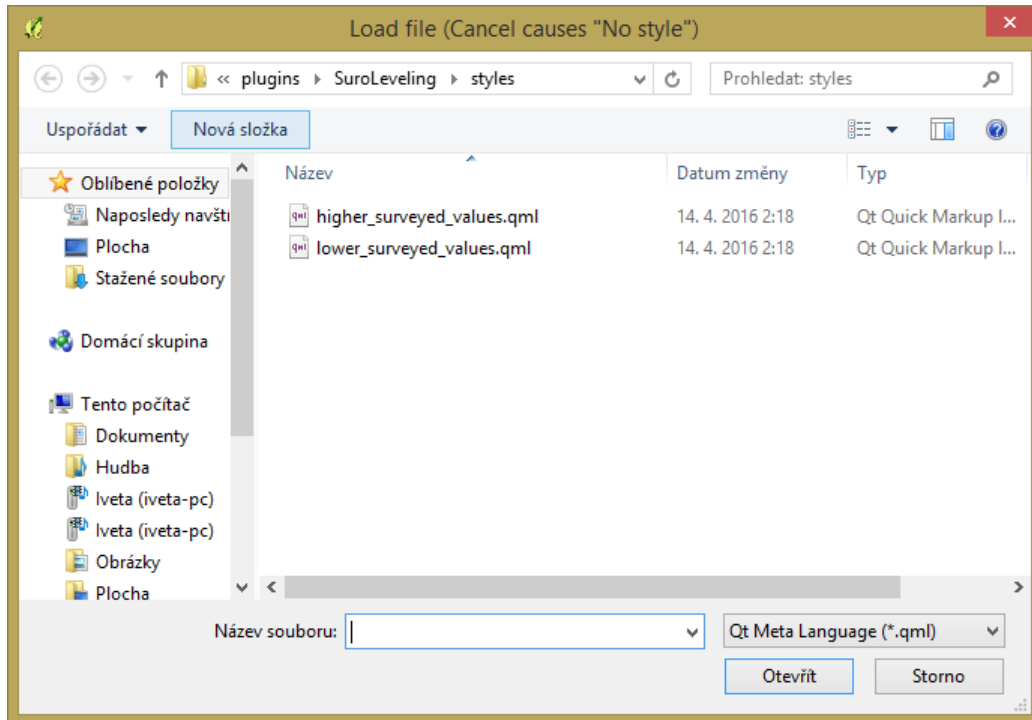


Figure A.5: Choose style

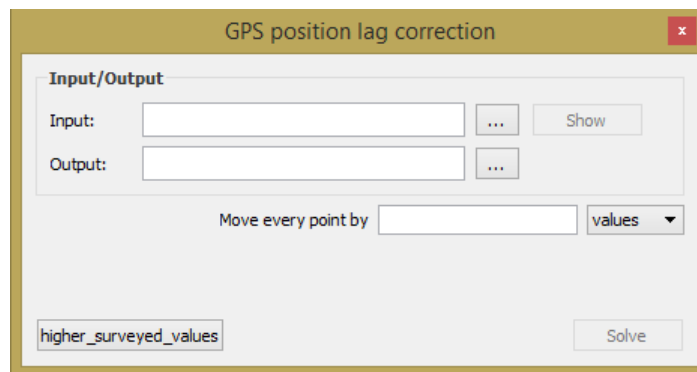


Figure A.6: Used style

### A.2.3 Showing input

Next to input is the button *Show*.

Show button is just to visualise the input file as layer in QGIS. It has no effect on the run of plugin and it is not necessary for it, but it allows you to visualise the input file in the same style as the output file. It gives you also the opportunity to compare the shifted values with the original ones.

This button was created on request by users who prefer to visualise both the input and the output from one dialog.

If input file does not exist, the plugin raises the error message.

### A.2.4 Shift

Shift will be done by clicking on button *Solve*. Before shifting you have to define some proprieties.

In combo box (drop-down list) you can choose the units of shift – each presents other type of shift; values mean shift by values, meters mean shift by constant distance and seconds mean shift by constant time/variable distance considering current velocity.

In the appropriate line text editor you have to insert value of move. For shift by values it should be integer (it does not make sense to shift point by 1.5 values because nothing like 1.5 value does not exist), for other shifts it should be integer or float.

For wrong input, plugin raise the error message.

### A.2.5 Dependencies

To avoid some unnecessary errors, there are some dependencies in the plugin graphical nterface.

The first is *Show* button. This button is disabled until you have wrote anything into input.

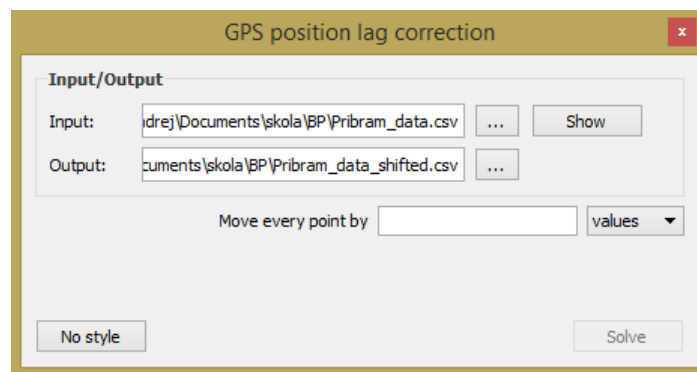


Figure A.7: Enabled Show button

The second is button *Solve*. You can't solve anything until you have defined input, output and value of shift.

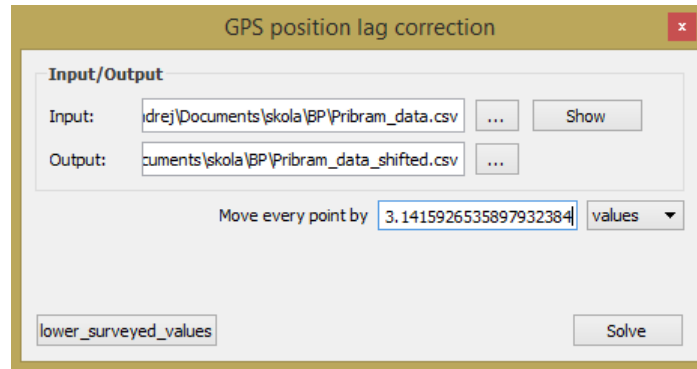


Figure A.8: Enabled Solve button

There is also implemented a shortcut for defining output directory. Many users want to save output file to the directory from which was read the input file; so when you choose input file, the directory will be automatically copied into output, just with added *\_shifted* into filename.