Precese perihélia Merkuru

F5330 Základní numerické metody
Artem Gorodilov

8. února 2025

1. Abstrakt

V této práci jsem provedl modelování precese dráhy Merkuru pomocí druhého Newtonova zákona s korekcí Obecné Teorie Relativity (OTR). Analyzoval jsem účinnost různých parametrů modelu. Zejména jsem našel optimální hodnotu koeficientu $\beta = 1.1 \times 10^5$ a optimální parametry meze integraci $N_{\mathrm{T}} = 150$ a integračního kroku $N_{\mathrm{dt}} = 150$, při kterých se vypočtená hodnota precese bude minimálně lišit od teoretické a chyba hodnoty bude minimální $\varphi = 43.1(3)$ [arcsec/století], se liší o 0.2(8)% od teoretické hodnoty.

Výpočty byly provedeny pomocí skriptu v Pythonu (viz. PoruchikRzhevsky).

2. Úvod

Jedním z nejvýznamnějších experimentálních potvrzení OTR byla přesná předpověď nadbytečné precesní změny perihelia Merkurovy dráhy, která činí přibližně $\varphi_{\mathrm{teor}} = 43.03(3)$ [arcsec/století] Clemence [1947]. Tento jev byl dlouhou dobu neobjasněným problémem klasické nebeské mechaniky.

Podle Newtonovy gravitační teorie by se oběžné dráhy planet měly chovat jako uzavřené elipsy s ohniskem v místě Slunce. Nicméně vzájemné gravitační působení planet způsobuje malé poruchy, které vedou k postupnému otáčení hlavní osy elipsy v rovině dráhy - tento jev se označuje jako precesní pohyb perihelia. U Merkuru je celková pozorovaná hodnota precesního posunu 9.55 [arcsec/století], z čehož 8.85 [arcsec/století] úhlových minut lze vysvětlit gravitačními interakcemi s ostatními planetami. Zbývající 43.1 [arcsec/století] úhlových vteřin za století však zůstávalo nevysvětleno. OTR tento problém vyřešila zavedením relativistické opravy Newtonova gravitačního zákona.

3. Model

Původní Newtonův pohybový zákon pro gravitační soustavu zní:

$$\vec{F} = m\ddot{\vec{r}}$$

kde $\vec{F}$ je síla působící na těleso, m je hmotnost tělesa, $\vec{r}$ je poloha tělesa a $\ddot{\vec{r}}$ je zrychlení tělesa. Tehdy zrychlení tělesa obíhajícího kolem Slunce lze vyjádřit jako:

$$\ddot{\vec{r}} = -\frac{GM_{\odot}}{r^2}\frac{\vec{r}}{r}$$

kde G je gravitační konstanta, $M_{\odot}$ je hmotnost Slunce a r je vzdálenost tělesa od Slunce. To popisuje dokonalou eliptickou dráhu při keplerovském pohybu. Pokud však uvažujeme další efekty, musíme tuto rovnici upravit tak, aby zahrnovala další členy, které aproximují efekty OTR:

$$\ddot{\vec{r}} = -\frac{GM_\odot}{r^2}\left(1 + \alpha\frac{2GM_\odot}{rc^2} + \beta\frac{L^2}{m_\mathrm{p}^2 r^2 c^2}\right)\frac{\vec{r}}{r} \tag{1}$$

kde α a β jsou koeficienty, L je moment hybnosti tělesa, m_p je hmotnost planery a c je rychlost světla.

Člen

$$\alpha\frac{2GM_\odot}{rc^2}$$

je tzv. Schwarzschildova korekce (relativistická gravitační korekce). Tento člen vyplývá z OTR. Koriguje Newtonovu gravitaci na malých vzdálenostech zohledněním zakřivení časoprostoru. Tato korekce zvyšuje sílu gravitace v blízkosti Slunce, čímž modifikuje oběžnou dráhu. Parametr α řídí sílu této korekce. To způsobuje precesi perihelia Merkuru v čase. Místo toho, aby dráha sledovala dokonalou elipsu, pomalu rotuje v důsledku zakřivení časoprostoru.

Člen

$$\beta\frac{L^2}{m_\mathrm{p}^2 r^2 c^2}$$

je tzv. Frame-dragging korekce (v závislosti na úhlovém momentu hybnosti). Tento člen závisí na úhlovém momentu hybnosti Merkuru L. Zohledňuje, jak pohyb zakřiveným časoprostorem ovlivňuje dráhu. Někdy je spojován s frame-dragging efekty v silných gravitačních polích Pfister [2007]. Moment hybnosty L^2 Merkuru lze vypočítat takto:

$$L^2 = |\vec{r} \times \dot{\vec{r}}|^2 \tag{2}$$

Tento člen modifikuje pohyb v důsledku skutečnosti, že energie a úhlový moment hybnosti se v relativistické soustavě striktně nezachovávají. Přidává další korekci, která mírně mění precesi perihelia. Její velikost se řídí parametrem β.

Koeficienty α a β pro Merkur jsou dány OTR [1]:

$$\alpha = 0, \quad \beta = 3 \tag{3}$$

4. Pipeline

4.1. Počáteční podmínky

K vyřešení problému jsem použil metodu intergování pohybu Merkuru po jeho oběžné dráze. Samozřejmě musíme někde začít. Tímto začátkem je poloha Merkuru v periheliu, daná hodnotou r_per, která byla vypočtena jako:

$$r_\mathrm{per} = a(1 - e) \tag{4}$$

kde a je velká poloosa dráhy a e je excentricita dráhy. Pro Merkur je $a = 57.909 \times 10^9$ m a $e = 0.2056$.

K popisu pohybu je zapotřebí tečnou rychlost $\dot{r}_\mathrm{per}$ a dostředivé zrychlení $\ddot{r}_\mathrm{per}$ v periheliu. Tyto hodnoty byly vypočteny jako:

$$\dot{r}_\mathrm{per} = \sqrt{\frac{GM_\odot}{a}\frac{1+e}{1-e}} \tag{5}$$

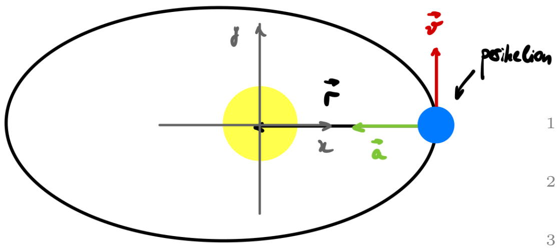
a

$$\ddot{r}_\mathrm{per} = \frac{GM_\odot}{r_\mathrm{per}^2} \tag{6}$$

Získané hodnoty se rovnají:

$$\begin{aligned} r_\mathrm{per} &= 4.6 \times 10^{10}\ \mathrm{m}, \\ \dot{r}_\mathrm{per} &= 5.9 \times 10^4\ \mathrm{m/s}, \\ \ddot{r}_\mathrm{per} &= 0.0627\ \mathrm{m/s^2} \end{aligned}$$

[1] Všechny parametry Merkuru byly převzaty z:: `https://nssdc.gsfc.nasa.gov/planetary/factsheet/mercuryfact.html`



Obrázek (1) Systema Merkur-Slunce a základní parametry.

r_{per} a $\dot{r}_{\mathrm{per}}$ jsem spojil do vektorů:

```
# Initial conditions
r0 = np.array([perihelion_distance, 0, 0])  # Mercury starts at perihelion
v0 = np.array([0, v_perihelion, 0])  # tangential velocity at perihelion
```

Listing (1) Počáteční podmínky

kde `perihelion_distance` je hodnota r_{per} a `v_perihelion` je hodnota $\dot{r}_{\mathrm{per}}$.

Simulace bude vyžadovat také znalost periody oběžné dráhy Merkuru. Tato hodnota bude použita k určení doby integrace. Pro tento účel bude hodnota periody (jeden celý oběh) vynásobena požadovaným počtem oběhů N_{T}. Oběžná perioda Merkuru je rovna:

$$T = 87.969 \text{ d} \tag{7}$$

Integrační krok dt lze dobře aproximovat pomocí vzorce:

$$dt = \frac{2\dot{r}_{\mathrm{per}}}{\ddot{r}_{\mathrm{per}}} \frac{1}{N_{\mathrm{dt}}} \tag{8}$$

kde pomocí N_{dt} lze získat požadovaný počet kroků integrace a tím zvýšit přesnost výpočtu.

4.2. Simulace

Integrace se provádí pomocí funkce `simulate_orbit()`, která jako vstupní parametry přijímá α, β, $\mathrm{N_T}$ a $\mathrm{N_{dt}}$. Tato funkce vrací vektor polohy Merkuru v průběhu času $\vec{r}$.

```
def simulate_orbit(alpha, beta, N_T, N_dt):
    T, dt, steps = time_parameters(N_T, N_dt)

    r = np.zeros((steps, 3))
    v = np.zeros((steps, 3))

    r[0] = r0
    v[0] = v0

    for i in range(steps - 1):
        a = acceleration(r[i], v[i], alpha, beta)

        v[i+1] = v[i] + a * dt * 86400  # days to seconds
        r[i+1] = r[i] + v[i+1] * dt * 86400

    return r
```

Listing (2) Simulace oběhu

Funkce vypočítá celkovou dobu integrace T a integrační krok dt pomocí funkce `time_parameters()`, která jako vstup přijme $\mathrm{N_T}$ a $\mathrm{N_{dt}}$ a vrátí T, dt a počet integračních kroků *steps*.

```
def time_parameters(N_T, N_dt):
    T = N_T * T_mercury # iteractions in days
    dt = (2 * v_perihelion / a_perihelion) / 86400 / N_dt  # days

    steps = int(T / dt)

    return T, dt, steps
```

Listing (3) Parametry integrace

Funkce `simulate_orbit()` vytvoří pole `r` a `v` pro uložení vektorových dat polohy a rychlosti. Jako počáteční vektory bereme jejich hodnoty v perehelu vypočtené dříve. Funkce pak pomocí cyklu `acceleration()`

vypočítá zrychlení pro každou polohu planety při jejím oběhu.

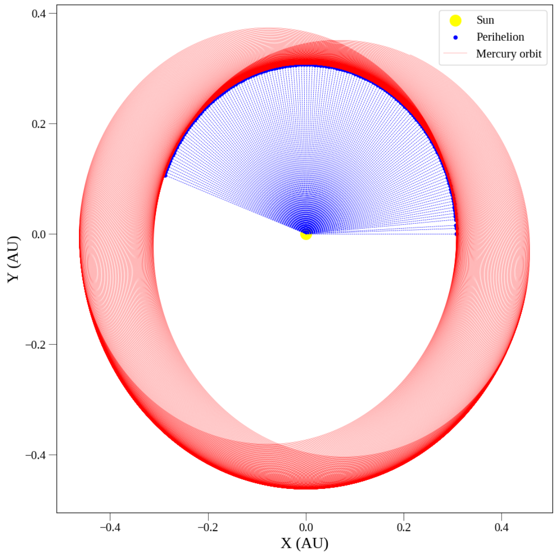
Tato funkce je nejdůležitější, protože je zodpovědná za precesi Merkurova perihelia. Funkce přijímá jako vstup vektor polohy planety $\vec{r}$, rychlost $\dot{\vec{r}}$ a koeficienty α a β. Funkce provede vektorové transformace pro výpočet momentu hybnosti L a vypočítá jej podle vzorce (2). Dále funkce vypočítá korekční člen v závorkách v rovnici (1). Poté funkce vypočítá vektor zrychlení $\mathbf{a}$ podle vzorce (1) a vrátí jeho hodnotu.

```
def acceleration(r, v, alpha, beta):
    r_mag = np.linalg.norm(r)  # calculating the magnitude of the position vector
    r_unit = r / r_mag  # unit vector in the direction of r

    # Calculating angular momentum squared term: (r vec * r vec dot)^2 / c^2
    L_squared = np.linalg.norm(np.cross(r, v))**2 / c**2

    # Calculating the relativistic correction factors
    correction_term = 1 + (alpha * 2 * G * M_sun) / (r_mag * c**2) + beta * L_squared / r_mag**2

    # Calculating the acceleration vector
    a = -G * M_sun / r_mag**2 * correction_term * r_unit

    return a
```

Listing (4) Výpočet zrychlení

4.3. Vypočet precese

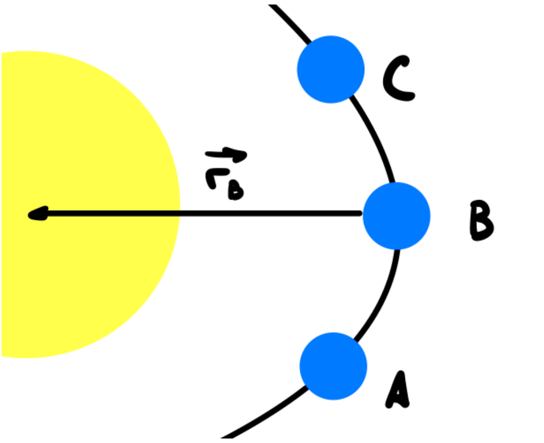
Získáním vektorů polohy planety v čase můžeme ověřit, že dráha Merkuru je skutečně precesní. To lze provést vizuální analýzou dráhy, kterou planeta urazila během doby integrace. Výsledek je znázorněn na obrázku (2).



Obrázek (2) Oběžná dráha Merkuru s precesí perihélia. Parametry: $\alpha = 0$, $\beta = 1.1 \times 10^5$, $N_\mathrm{T} = 150$, $N_\mathrm{dt} = 150$.

Abychom mohli vypočítat precesi perihelia Merkuru, musíme zjistit všechny polohy perihelia Merkuru a poté zjistit úhlový posun perihelia mezi dvěma po sobě jdoucími polohami. K tomu jsem použil funkci `find_perihelion_positions()`, která přijímá vektor poloh Merkuru $\mathbf{r}$ a vrací pole poloh Merkuru v periheliu a jejich chyby. Funkce funguje následovně: cyklem prochází všechny polohy planety a hledá tři polohy A, B a C, které následují (viz obrázek 3). Bod r_B bude perehelem, pokud $r_\mathrm{B} < r_\mathrm{A}$ a $r_\mathrm{B} < r_\mathrm{C}$. Pokud je tato podmínka splněna, je poloha r_B přidána do pole poloh perihelia. Chyba polohy perihelia se aproximuje jako střední hodnota rozdílu mezi r_A a r_C.

```
def find_perihelion_positions(r):
    r_magnitude = np.linalg.norm(r, axis=1)
    perihelion_positions = [r[0]]  # starting with the initial perihelion
    uncertainties = []
```



Obrázek (3) Schéma hledání polohy perihelia.

```
    for i in range(1, len(r_magnitude) - 1):
        if r_magnitude[i] < r_magnitude[i - 1] and r_magnitude[i] < r_magnitude[i + 1]:
            perihelion_positions.append(r[i])
            uncertainty = abs(r_magnitude[i+1] - r_magnitude[i-1]) / 2  # approximation of positional uncertainty
            uncertainties.append(uncertainty)

    first_uncertainty = np.mean(uncertainties)
    uncertainties.insert(0, first_uncertainty)

    return np.array(perihelion_positions), np.array(uncertainties)
```

Listing (5) Výpočet poloh perihelia

Samotná precese je definována jako úhlový posun perihelia mezi dvěma po sobě následujícími polohami $\vec{r_1}$ a $\vec{r_2}$. Úhel mezi těmito dvěma vektory je definován jako:

$$\cos\theta = \frac{\vec{r_1} \cdot \vec{r_2}}{|\vec{r_1}||\vec{r_2}|} \tag{9}$$

Dot Product vektorů $\vec{r_1}$ a $\vec{r_2}$ je definován jako:

$$\vec{r_1} \cdot \vec{r_2} = x_1 x_2 + y_1 y_2 \tag{10}$$

Velikosti vektorů $\vec{r_1}$ a $\vec{r_2}$ jsou definovány jako:

$$|\vec{r_1}| = \sqrt{x_1^2 + y_1^2},\ |\vec{r_2}| = \sqrt{x_2^2 + y_2^2} \tag{11}$$

Dosazením (10) a (11) do (9) a řešením pro θ získáme úhlové posunutí mezi oběma vektory:

$$\theta = \arccos\left(\frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2}\sqrt{x_2^2 + y_2^2}}\right) \tag{12}$$

K výpočtu se používá funkce `calculating_perihelion_angles()`, která bere vektory polohy planety a jejich chyby a vrací průměrný úhlový posun perihelia a jeho chybu.

```
def calculating_perihelion_angles(positions , uncertainties):
    angles_val = []
    angles_err = []
    for i in range(1, len(positions)):
        position_x_1 = ufloat(positions[i-1][0], uncertainties[i-1])
        position_y_1 = ufloat(positions[i-1][1], uncertainties[i-1])

        position_x_2 = ufloat(positions[i][0], uncertainties[i])
        position_y_2 = ufloat(positions[i][1], uncertainties[i])

```

```
        phi = um.acos((position_x_1
            * position_x_2 +
            position_y_1 *
            position_y_2) / (um.sqrt
            (position_x_1**2 +
            position_y_1**2) * um.
            sqrt(position_x_2**2 +
            position_y_2**2)))
        angles_val.append(um.degrees
            (phi).nominal_value)
        angles_err.append(um.degrees
            (phi).std_dev)

    average_precession = ufloat(np.
        mean(angles_val), uncert(
        angles_val, np.mean(
        angles_err)))

    return average_precession
```

Listing (6) Výpočet úhlového posunu perihelia

Pro určení úhlu posunu perehelia za století v akresekundách je třeba určit poměr pozemského roku k merkurovskému roku:

$$T_{\text{rel}} = \frac{T_{\text{Earth}}}{T_{\text{Mercury}}} = \frac{365.256}{87.969} = 4.152$$

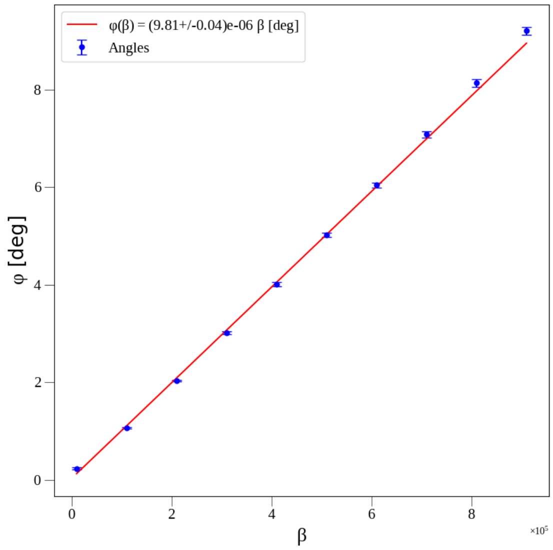
Pak úhel posunu perehelia za století v akresekundách je roven:

$$\varphi = \varphi_{\text{deg}} \times 3600 \frac{100}{T_{\text{rel}}} \frac{(k_\beta, k_\alpha)}{(\alpha, \beta)} \tag{13}$$

Poslední násobitel v rovnici (13) je nutný pro škálování výsledku na koeficienty α a β. Z (3) vyplývá, že $k_\alpha = 0$ a $k_\beta = 3$. Hodnoty α a β budou vysvětleny a vypočteny v následující části. Rovnice (13) bude vypadat takto:

$$\varphi = \varphi_{\text{deg}} \times 3600 \frac{100}{T_{\text{rel}}} \frac{3}{\beta} \tag{14}$$

K výpočtu veličin a jejich nejistot byla použita knihovna Uncertinties pro Python.



Obrázek (4) Závislost úhlu posunu perehelia na hodnotě koeficientu β při $\alpha = 0$.

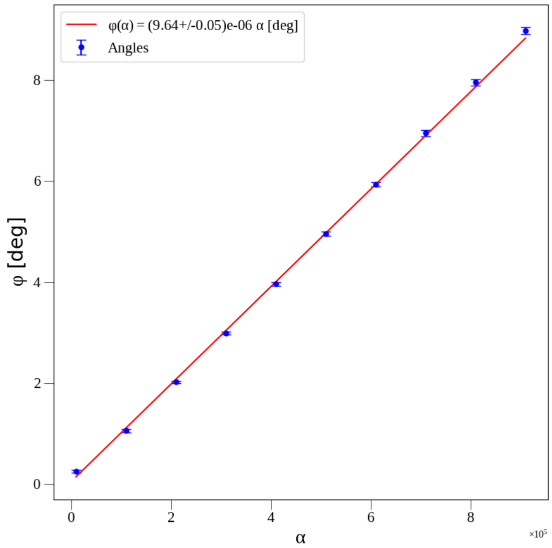
Chyby byly rozšířeny o Studentův koeficient (2-Tail Confidence Level) s ohledem na stupně volnosti pro každou hodnotu, pro interval spolehlivosti 68.27%.

5. Analýza výsledků

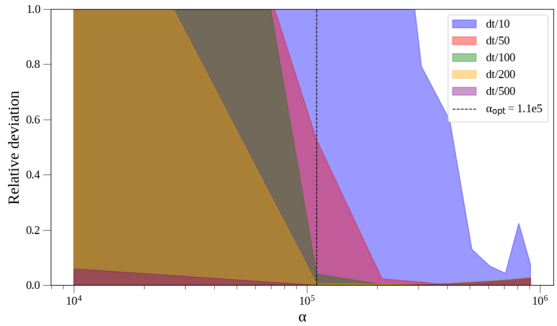
5.1. Optimální hodnoty koeficientů α a β

Pro získání odpovídající hodnoty úhlu posunu perehelia je třeba zvolit potřebné hodnoty koeficientů α a β. Z (3) vidíme, že hodnoty α a β pro Merkur jsou resp. 0 a 3. Pokud však provedeme simulace s těmito hodnotami, posun perehelia nepozorujeme. To je dobře vidět na obrázku (4), který ukazuje lineární závislost úhlu posunu perehelia φ na hodnotě koeficientu β při pevné hodnotě $\alpha = 0$. Jak je vidět, posun se stává znatelným při hodnotách $\beta \approx \times 10^5$.

Totéž se stane, pokud stanovíme $\beta = 0$ a změníme α. Jak je patrné z obrázku (5), úhlový posun perihelia je patrný při hod-



Obrázek (5) Závislost úhlu posunu perehelia na hodnotě koeficientu α při $\beta = 0$.



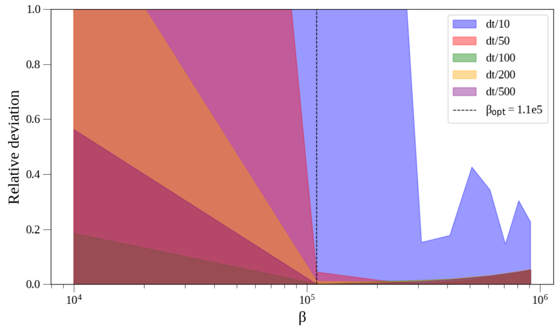
Obrázek (6) Závislost odchylky úhlu posunu perehelia od teoretické hodnoty na hodnotě koeficientu α při $\beta = 0$.

notách $\alpha \approx \times 10^5$.

Vyvstává otázka, jak zvolit optimální hodnoty koeficientů α a β, aby se průměrná hodnota úhlu posunutí perehelia co nejméně lišila od teoretické hodnoty? Za tímto účelem jsem provedl sérii simulací s různými hodnotami β při pevných hodnotách $\alpha = 0$ a α při pevných hodnotách $\beta = 0$. Simulace byly provedeny pro různé hodnoty integračních kroků dt/N_{dt}. Integrační interval byl ve všech situacích stejný $N_{\mathrm{T}} = 150$.

Výsledky jsou uvedeny na obrázcích (6) a (7). Výpočty byly provedeny pomocí funkce `getting_angle_data()`, která vypočítá hodnoty úhlů perehelia pro různé hodnoty α a β, načež funkce `plotting_difference()` vypočítá relativní odchylku od teoretické hodnoty a vykreslí závislost této odchylky na α a β.

Z obrázků je patrné, že velikost relativní chyby závisí na velikosti integračního kroku. Například při $dt/10$ je relativní chyba vysoká v celém rozsahu hod-



Obrázek (7) Závislost odchylky úhlu posunu perehelia od teoretické hodnoty na hodnotě koeficientu β při $\alpha = 0$.

not α a β. Se zmenšováním integračního kroku je zřejmé, že nejmenší relativní chyba je dosažena při hodnotách $\beta \approx 1.1 \times 10^5$ $\alpha \approx 1.1 \times 10^5$.

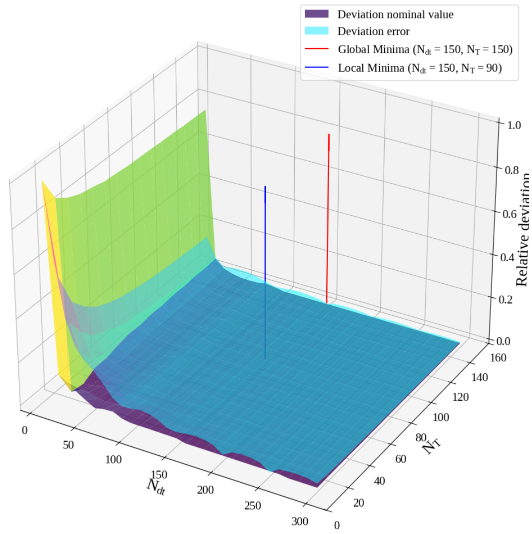
Právě tato hodnota by měla být použita při simulaci a později bude použita jako hodnota β ve vzorci (14) pro výpočet úhlu posunu perehelia za století.

5.2. Optimální hodnoty integračních kroků a intervalu integrace

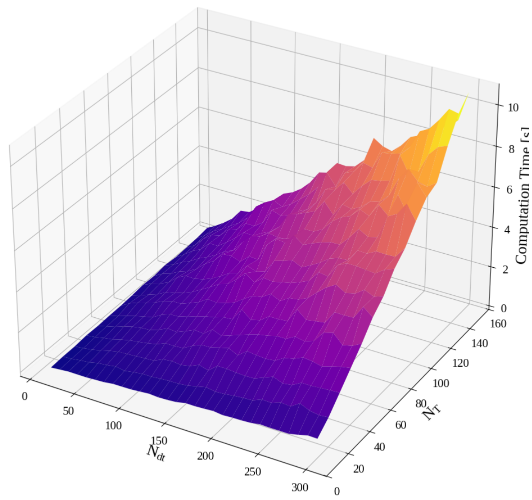
Je velmi důležité správně definovat integrační krok a integrační interval. Oba tyto parametry ovlivňují přesnost a dobu výpočtu. Minimalizace těchto parametrů vyžaduje analýzu funkce dvou proměnných $\sigma(\mathrm{N_T}, \mathrm{N_{dt}})$, kde σ je relativní odchylka precesního úhlu od teoretické hodnoty, σ_{err} je chyba odchylky a $t_{\mathrm{compute}}(\mathrm{N_T}, \mathrm{N_{dt}})$ je doba výpočtu.

Abych zjistil optimální hodnoty těchto parametrů, provedl jsem test efektivity pomocí funkce `efficiency_test()`. Tato funkce přijímá jako vstup hodnoty α a β a pole hodnot $\mathrm{N_T}$ a $\mathrm{N_{dt}}$ v následujících rozmezích: $10 < \mathrm{N_T} < 200$ a $10 < \mathrm{N_T} < 500$ v přírůstcích po 10 bodech. Funkce vrací pole hodnot σ, σ_{err} a t_{compute} pro každou kombinaci $\mathrm{N_T}$ a $\mathrm{N_{dt}}$. Vysledky jsou zobrazeny na obrázcích (8) a (9).

Jak je vidět, s rostoucí dobou integrace a klesajícím integračním krokem se lineárně počítá doba výpočtu t_{compute}. Což se dalo očekávat. Pro výpočet optimálních hodnot $\mathrm{N_T}$ a $\mathrm{N_{dt}}$ budu analyzovat pouze funkce $\sigma(\mathrm{N_T}, \mathrm{N_{dt}})$ a $\sigma_{\mathrm{err}}(\mathrm{N_T}, \mathrm{N_{dt}})$. Za tímto účelem jsem obě funkce normalizoval a sečetl do jedné funkce $\sigma_{\mathrm{total}}(\mathrm{N_T}, \mathrm{N_{dt}})$. Poté jsem našel globální minimum této funkce (červená šipka na obrázku 8). Ukázalo se, že je to hodnota:



Obrázek (8) Závislost odchylky úhlu posunu perehelia od teoretické hodnoty na hodnotách $\mathrm{N_T}$ a $\mathrm{N_{dt}}$.



Obrázek (9) Závislost doby výpočtu na hodnotách $\mathrm{N_T}$ a $\mathrm{N_{dt}}$.

$$\mathrm{N_T} = 150, \mathrm{N_{dt}} = 150$$

Provedením simulace s těmito hodnotami byla získána hodnota $\varphi = 43.1(3)$ [arcsec/století] která se liší od teoretické hodnoty o $0.2(8)\%$

Jsem se rozhodl najít lokální minimum v oblasti hodnot $\mathrm{N_T} < 100$ a $\mathrm{N_{dt}} < 200$ (modra šipka na obrázku 8). Ukázalo se, že tato hodnota je rovna:

$$\mathrm{N_T} = 90, \mathrm{N_{dt}} = 150$$

Provedením simulací s těmito hodnotami jsem získal hodnotu $\varphi = 43.4(5)$ [arcsec/století], která se od teoretické hodnoty liší o $(0.9 \pm 1.3)\%$.

Je tedy vidět, že hodnoty $\mathrm{N_T}$ a $\mathrm{N_{dt}}$ nalezené v globálním minimu jsou tedy pro tento problém optimální.

6. Závěr

Výsledky simulace ukazují uspokojivé výsledky, které se shodují s teoretickými údaji z literatury $\varphi = 43.1(3)$ [arcsec/století]. Pro tento problém jsem také našel optimální hodnoty parametrů α a β: $\alpha = 0$, $\beta = 1.1 \times 10^5$. Také jsem našel optimální hodnoty $\mathrm{N_T} = 150$, $\mathrm{N_{dt}} = 150$. To vše nás vede k závěru, že tento model je pro danou úlohu vyhovující.

Mezi věci, které lze zlepšit, patří doba výpočtu. Při hledání optimálních parametrů v části 5.2. bylo globální minimum nalezeno v suboptimálních mezích z hlediska výpočetního času. Zároveň se ukázalo, že hodnoty na výstupu jsou poměrně přesné. Možná, že změna některých kroků v procesu integrace by mohla proces optimalizovat lepším způsobem.

Reference

G. M. Clemence. The Relativity Effect in Planetary Motions. *Reviews of Modern Physics*, 19(4):361–364, Oct. 1947. doi: 10.1103/RevModPhys.19.361.

C. Körber, I. Hammer, J.-L. Wynen, J. Heuer, C. Müller, and C. Hanhart. A primer to numerical simulations: the perihelion motion of mercury. *Physics Education*, 53(5):055007, jun 2018. doi: 10.1088/1361-6552/aac487. URL `https://dx.doi.org/10.1088/1361-6552/aac487`.

H. Pfister. On the history of the so-called Lense-Thirring effect. *General Relativity and Gravitation*, 39(11):1735–1748, Nov. 2007. doi: 10.1007/s10714-007-0521-4.

PoruchikRzhevsky. mercury_perihelion_precession. `https://github.com/PoruchikRzhevsky/mercury_perihelion_precession`.

Poděkování

Tato práce byla inspirována článkem Körber et al. [2018]. Jedná se o skvělou práci, a to jak z pedagogického, tak praktického hlediska. Převzal jsem z něj to nejlepší a některé tipy použil ve své analýze.