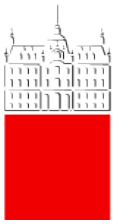
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Luščenje modelskih parametrov in razdelčni modeli

6. naloga pri Modelski Analizi 1

Avtor: Marko Urbanč (28232019)
Predavatelj: prof. dr. Simon Širca
Asistent: doc. dr. Miha Mihovilovič

15.11.2023

Kazalo

1 Uvod

V tej nalogi si pogledamo zelo praktično uporabo nekaterih metod, ki smo jih spoznali. Osnoven princip fizikalnega modeliranja je ta, da si izberemo nek model, ki ga lahko opišemo z nekaj parametri. Te parametre lahko izmerimo ali pa jih izračunamo iz nekih osnovnih fizikalnih zakonov. Vendar pa se v praksi pogosto zgodi, da imamo nek model, ki ga želimo uporabiti, vendar pa ne poznamo parametrov, ki jih potrebujemo, ker jih ne znamo drugače izmeriti. Tu v igro pride princip luščenja parametrov, ki nam omogoča, da iz nekih meritev določimo parametre modela. V tej nalogi si bomo pogledali, kako se to počne na treh praktičnih primerih. Metode, ki jih uporabimo lahko za to so različne, ampak v glavnem se vse vrtijo okoli minimizacije neke funkcije, ki je odvisna od parametrov modela.

1.1 Farmakologija

V farmakologiji merijo odziv tkiv na različne reagente, recimo na neko zdravilo. Za večina teh pojavov lahko privzamemo, da gre za reakcijo, kjer spremljamo vezavo molekul reagenta X na receptorje Y v tkivu, kar opišemo kot:

$$X + Y \rightleftharpoons Y^* \,. \tag{1}$$

V stacionarnem stanju, dobimo zvezo:

$$y = \frac{y_0 x}{x + a} \,, \tag{2}$$

kjer je y_0 nasičeni odziv tkiva in a enaka koncentraciji, potrebni za odziv, ki je enak polovici nasičenega odziva. Model lahko še razširimo na nelinearni model oblike:

$$y = \frac{y_0 x^p}{x^p + a^p} \,. \tag{3}$$

1.2 Čistilnost ledvic

V tej nalogi si bomo pogledali, kakšna je čistilnost ledvic. To je količina krvi, ki jo ledvice očistijo v določeni časovni enoti. To pomeni, da problem opisuje enačba oblike:

$$\frac{\mathrm{d}c}{\mathrm{d}t} = -\frac{\varepsilon\phi}{V} = -\lambda c \,, \tag{4}$$

kjer je ε čistilna sposobnost ledvic, ϕ pretok krvi skozi ledvice in V volumen krvi v telesu. To vse lahko zapišemo kot en parameter $\lambda = \frac{\varepsilon\phi}{V}$, ki ga želimo določiti. V praksi se to meri tako, da se v telo vbrizga nek radioaktivni izotop, najpogosteje hipuran, ki se izloča skozi ledvice. Merimo količino radioaktivnosti v urinu in krvi in iz tega izračunamo čistilnost ledvic. Za to potrebujemo nek model, ki nam pove, kako se radioaktivnost spreminja v času. V našem primeru bodo to razdelčni modeli. Pogledali si bomo 4 modele. Prvo eno-razdelčna modela brez aktivnosti ozadja in z aktivnostjo ozadja:

$$c(t) = ae^{-bt}\,, \tag{5}$$

$$c(t) = ae^{-bt} + c\,, \tag{6}$$

kjer je b zdaj parameter, ki nas najbolj zanima, a pa je neka skalirna konstanta. V primeru drugega modela prestavlja c aktivnost ozadja. Pogledali si bomo še dva dvo-razdelčna modela, ki sta oblike:

$$c(t) = ae^{-bt} + ce^{-dt}\,, \tag{7}$$

$$c(t) = ae^{-bt} + ce^{-dt} + e\,, \tag{8}$$

kjer sta b in d parametra, ki nas zanimata, a in c pa sta skalirni konstanti. V primeru drugega modela predstavlja e aktivnost ozadja. Tu sem mogoče malo nerodno izbral ime za parameter, ker je e tudi Eulerjeva konstanta, ampak upam, da ne bo prišlo do zmede.

1.3 Rekonstrukcija koordinat pri poskusih s spektrometrom

V tej nalogi pa si bomo pogledali, kako lahko določimo neznano zvezo med koordinatami na goriščni ravnini detektorja in kotom na tarči. Dobili smo meritve θ_{tg} in $(\theta_{fp}\,, x_{fp})$, med katerimi moramo najti čim bolj varčno zvezo. Za ta del naloge mi je nekoliko zmanjkalo časa, tako da sem naredil zelo osnoven model, ki je oblike:

$$\theta_{tg}(\theta_{fp}\,, x_{fp}) = V\vec{x} \cdot \vec{\theta}^{\intercal}\,, \tag{9}$$

kjer je V matrika parametrov, ki jo želimo določiti, torej:

$$V = \begin{bmatrix} v_0 & v_1 & v_2 \\ v_3 & v_4 & v_5 \\ v_6 & v_7 & v_8 \\ v_9 & v_{10} & v_{11} \end{bmatrix}\,.$$

Vektorja $\vec{x}$ in $\vec{\theta}$ pa sta oblike:

$$\vec{x} = \begin{bmatrix} 1 \\ x_{fp} \\ x_{fp}^2 \end{bmatrix}\,, \qquad \vec{\theta} = \begin{bmatrix} 1 \\ \theta_{fp} \\ \theta_{fp}^2 \\ \theta_{fp}^3 \end{bmatrix}\,.$$

2 Naloga

Naloga želi, da za dane modele izluščimo parametre, ki jih potrebujemo. Za vse primere so dani podatki.

3 Opis reševanja

Reševanja sem se lotil klasično kot se lotim nalog za Modelsko Analizo, tako da sem šel pisati svojo implementacijo parllel-capable SVD algoritma, kjer sem se blazno zapletel in sem na koncu pravzaprav samo izgubil veliko časa. Time better spent bi bil, da bi dejansko zamenjal ta čas z kakim delom ža tretjo nalogo, ki sem jo naredil zelo na hitro rešil. Pa še tako velike ideje sem imel za 3. nalogo. Recimo podatki kar kličejo po ML modelu, za katerega na koncu nikakor ni bilo časa.

Sicer pa je, kot vedno, reševanje potekalo v Pythonu z standardnim naborom knjižnic za numerično računanje. Torej `numpy`, `scipy` in `matplotlib`. Največ sem se posluževal funkcije `curve_fit` iz `scipy.optimize`. Sem pa tudi napisal svojo funkcijo, ki skuša na grobo minimizirati razdaljo med modelom in podatki. Za dodatek pa sem si pogledal še uporabo `scipy.optimize.minimize`. Kjer se je dalo, sem narisal še prostor parametrov, da lahko vidimo vizualno, kako dobro smo zadeli hipotetčni minimum pri danem modelu.

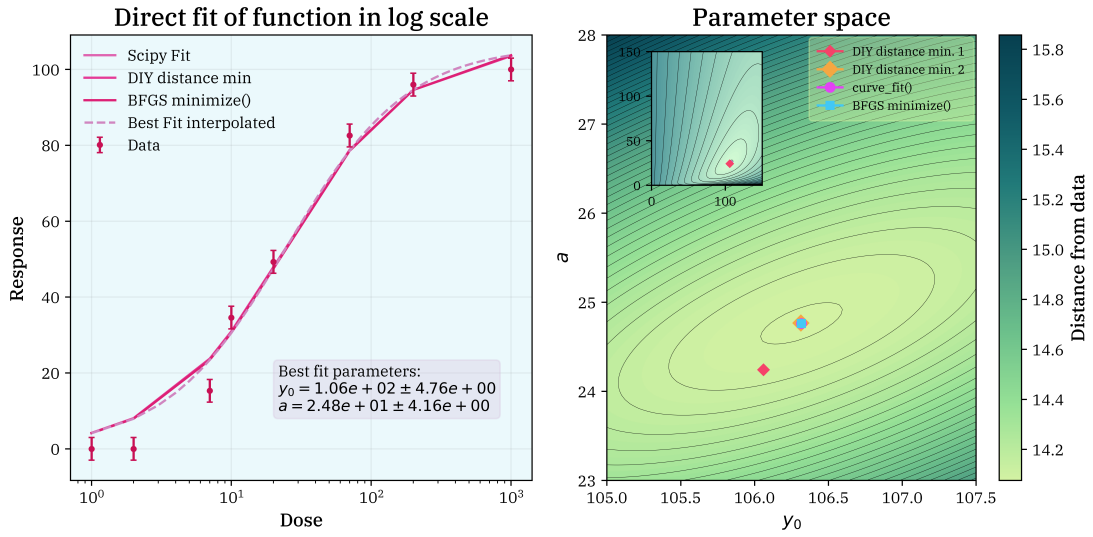
4 Rezultati

Se mi zdi, da se nima smisla predolgo razpisovati o samem reševanju, ker je vse skupaj zelo podobno kot pri prejšnjih nalogah. Pojdimo na rezultate.

4.1 Farmakologija

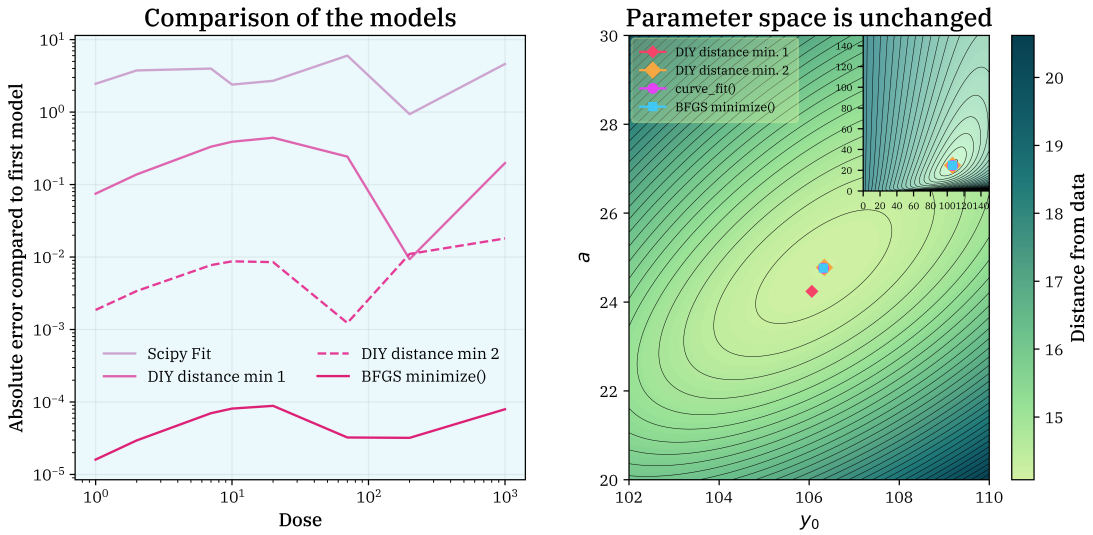
Najprej si poglejmo, kako se obnese linearni model. Tu je rezultat prikazan na sliki (1). Ugotovil sem, da dobim najboljše prileganje k podatkom, če postavim y skalo na logaritmično. Z vsemi metodami dobimo približno enako krivuljo, ki se razmeroma dobro prilega podatkom. Malenkost manjka v prvih nekaj točkah. Vem da bi v svoji funkciji lahko to poskusil popraviti tako, da bi dal prvim nekaj točkam večjo težo. Nevem pa kako bi to naredil v `curve_fit`. Vidimo sicer, da moja domača metoda v prvi iteraciji zgreši čisti minimum, ampak ob ponovnem zagonu na skrčenem območju že najde pravi minimum. To je zaradi končne natančnosti mreže po kateri išče. V osnovi je to samo grid search algoritem. Najboljši fit nam tako da parametre:

$$y_0 = 106 \pm 4.76\,, \qquad a = 24.8 \pm 4.16\,.$$



Slika 1: Prileganje linearnega modela.

Zdaj poglejmo nelinearni model, kjer fiksiramo potenco $p = 1$. To je sanity check, ki bi moral preveriti ali naša koda pravilno funkcionira. Dobiti bi morali iste oz. vsaj podobne rezultate kot pri linearnem modelu. Rezultat je prikazan na sliki (2). Vidimo, da je ujemanje **zelo** odvisno od metode. Zanimivo mi je, da je `curve_fit` najslabši, čeprav je to v osnovi najboljša metoda. Heck še moj grid search algoritem v dveh iteracijah jo razbije za 4 velikostne rede absolutne napake. To je zelo zanimivo. Najboljši rezultat pa da `minimize`, ki minimizira polje, ki ga vrne moja domača funkcija. Torej je to v osnovi ista metoda, ampak z mnogo boljšim iskanjem minimuma.

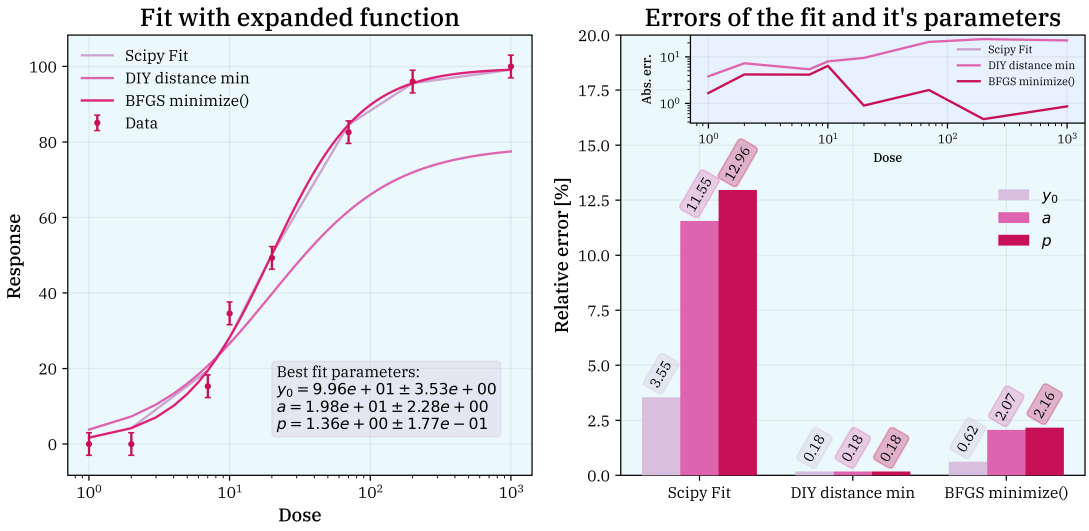


Slika 2: Prileganje nelinearnega modela z $p = 1$.

Tako je torej upravičeno, da si pogledamo tudi nelinearni model z p kot prostim parametrom. Rezultat je prikazan na sliki (3). Tu najboljši rezultat da `minimize`, kar mi je mogoče malenkost zanimivo oz. vsaj kaže na to, da je moja metoda iskanja minimumov preko grid search-a preveč groba, ker moja domača metoda tu kompletno pogrne in vrne napačen rezultat. Tudi `curve_fit` je tu mnogo mnogo boljši. Najboljši fit nam da parametre:

$$y_0 = 99.6 \pm 3.5\,, \qquad a = 19.8 \pm 2.3\,, \qquad p = 1.36 \pm 0.18\,.$$

Narisal sem potem še, kar bi lahko ljubkovalno rekel, Dunning-Krugerjev efekt DIY metode. To elaborate: Vsaka metoda ima tudi določeno negotovost pri določanju parametrov in zanimivo je videti, da je moja metoda v tem primeru najbolj samozavestna, čeprav je najslabša. Je pa res, da je zopet metoda izračuna napak spet domača skovanka. Gre za bootstrap metodo, kjer vzorčim podatke in nato izračunam napako na parametrih. Problem je v tem, da zaradi programerskega overhead-a ni bilo možno multiprocesiranje grid searcha, kar pomeni, da je bootstrap že za majhno število vzorcev zelo počasen. Pravzaprav je celoten grid za grid search tu zelo zelo majhen. Če me spomin ne vara samo $10 \times 10 \times 10$ točk. Človek bi želel te 10 spremenit v kake 100 ali 1000.



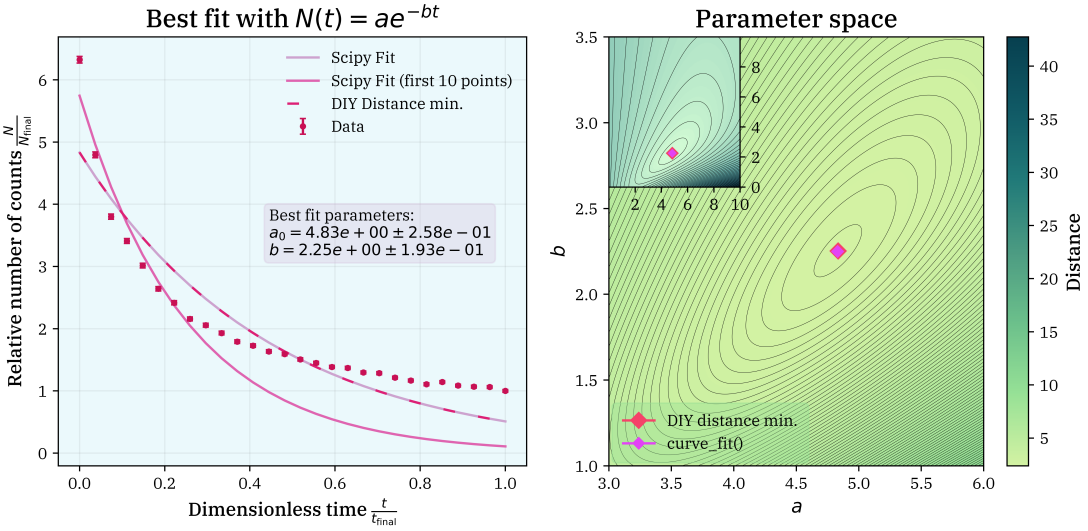
Slika 3: Prileganje nelinearnega modela z p kot prostim parametrom.

4.2 Čistilnost ledvic

Onwards na čistilnost ledvic. Poglejmo kar takoj naš eno-razdelčni model brez ozadja. Rezultat je prikazan na sliki (4). Tu sta rezultata, ki ju dobimo z `curve_fit` in domačo metodo zelo podobna, ampak vidimo, da noben ni zares dober, kar pomeni, da je model pomanjkljiv. Poskusil sem še, kaj se zgodi, če vzamem za podatke samo prvih 10 točk in zdi se mi, da je oblika mogoče boljša, ampak skala je vseeno precej napačna. Najboljši fit nam da parametre:

$$a = 4.83 \pm 0.26\,, \qquad b = 2.25 \pm 0.193\,.$$

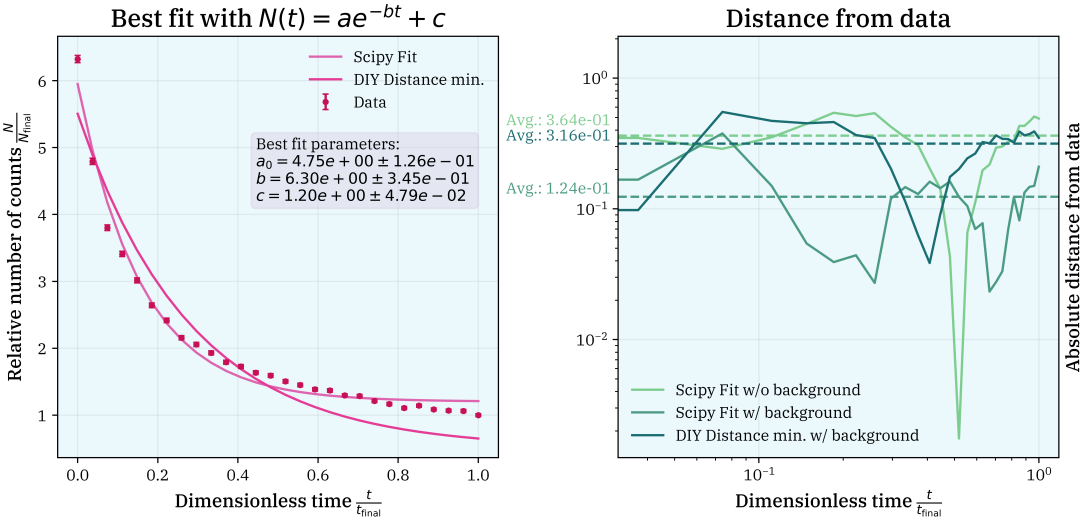
Tu se splača zdaj dati **disclaimer**. Moje osi so v brezdimenzijski obliki, kar pomeni, da so moje parametri transformirani. Reskalirana verzija pride na koncu.



Slika 4: Prileganje eno-razdelčnega modela brez ozadja.

Poglejmo če se rezultat kaj izboljša, če dodamo ozadje. Rezultat je prikazan na sliki (5). Res se, ampak ni še popolno, je pa opazno boljše. Torej je ozadje očitno pomembno. Najboljši fit nam da parametre:

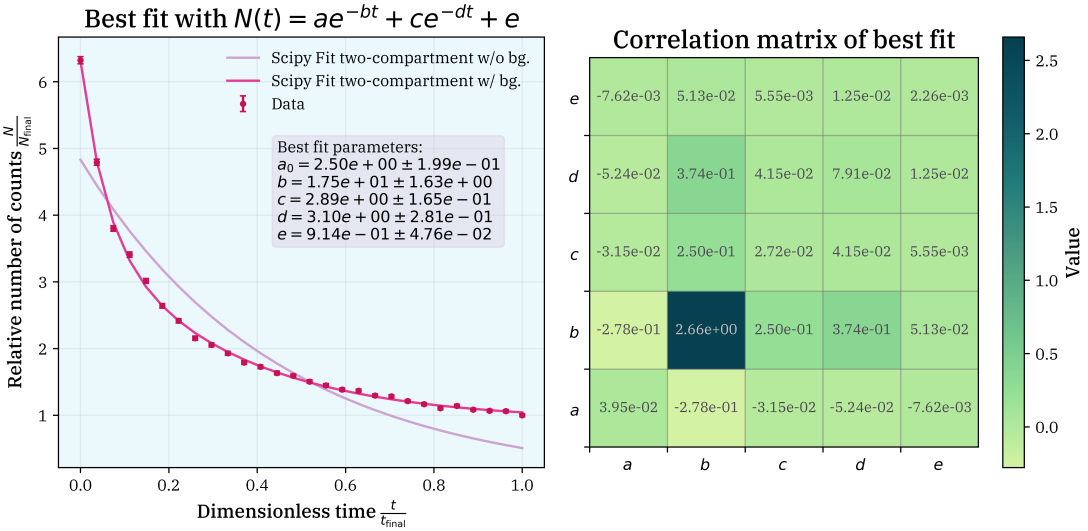
$$a = 4.75 \pm 0.13\,, \qquad b = 6.30 \pm 0.35\,, \qquad c = 1.20 \pm 0.05\,.$$



Slika 5: Prileganje eno-razdelčnega modela z ozadjem.

Poglejmo si še dvo-razdelčni model z in brez ozadja. Rezultat je prikazan na sliki (6). Tu dobimo res dokaz, kako pomembno je ozadje, ker dobimo spet mnogo slabše prileganje. Vidimo, da za model z ozadjem pa dobimo zelo dobro ujemanje, torej je to najbolj osnoven model, ki dobro opiše naše podatke Najboljši fit nam da parametre:

$$a = 2.50 \pm 0.2\,, \qquad b = 17.5 \pm 1.63\,, \qquad c = 2.89 \pm 0.17\,,$$
$$d = 3.10 \pm 0.28\,, \qquad e = 0.91 \pm 0.05\,.$$



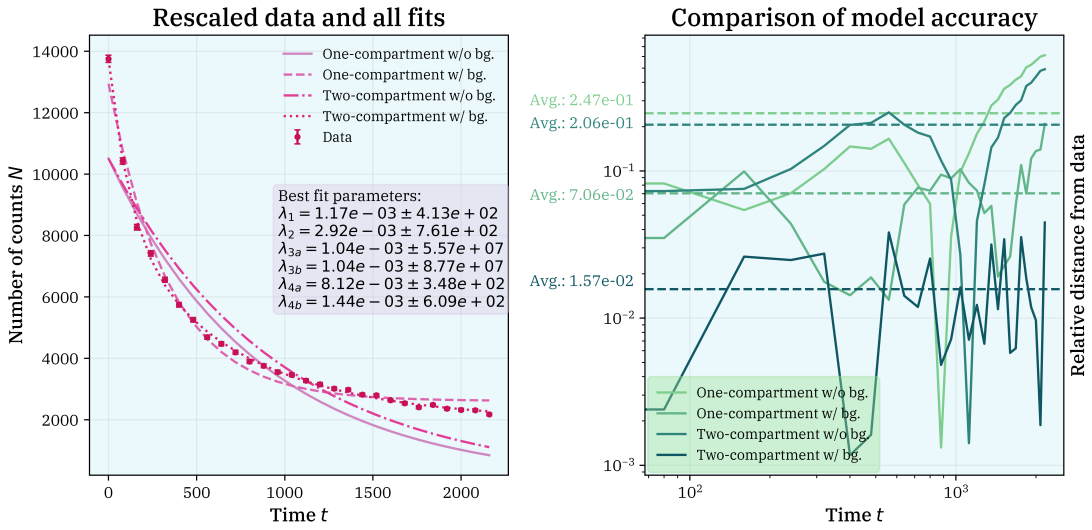
Slika 6: Prileganje dvo-razdelčnega modela z in brez ozadja.

Zraven sem narisal še korelacijsko matriko za najboljši fit (torej 4. model). Jaz ne bi rekel, da je neka znatna korelacija. Parametri (b, c) in (b, d) sta malenkost bolj korelirana kot ostali, ampak zdi se mi, da to ni nič takega.

Da dobimo nazaj prave parametre, moramo vse parametre, ki niso v eksponentih deliti z N_{final}, vse v eksponentih pa pomnožiti z t_{final}. Torej:

$$\bar{a} = \frac{a}{N_{\text{final}}}\,, \quad \bar{c} = \frac{c}{N_{\text{final}}}\,, \quad \bar{e} = \frac{e}{N_{\text{final}}}\,,$$
$$\bar{b} = b\,t_{\text{final}}\,, \quad \bar{d} = d\,t_{\text{final}}\,.$$

Tako dobimo reskaliran graf, kjer so osi v pravih enotah. Zraven pa je še primerjava relativnega odstopanja od podatkov v log-log skali. To je prikazano na sliki (7). Vidimo, da je prileganje res zelo dobro in iz grafa relativnega odstopanja je zelo očitno da je 4. model najboljši.



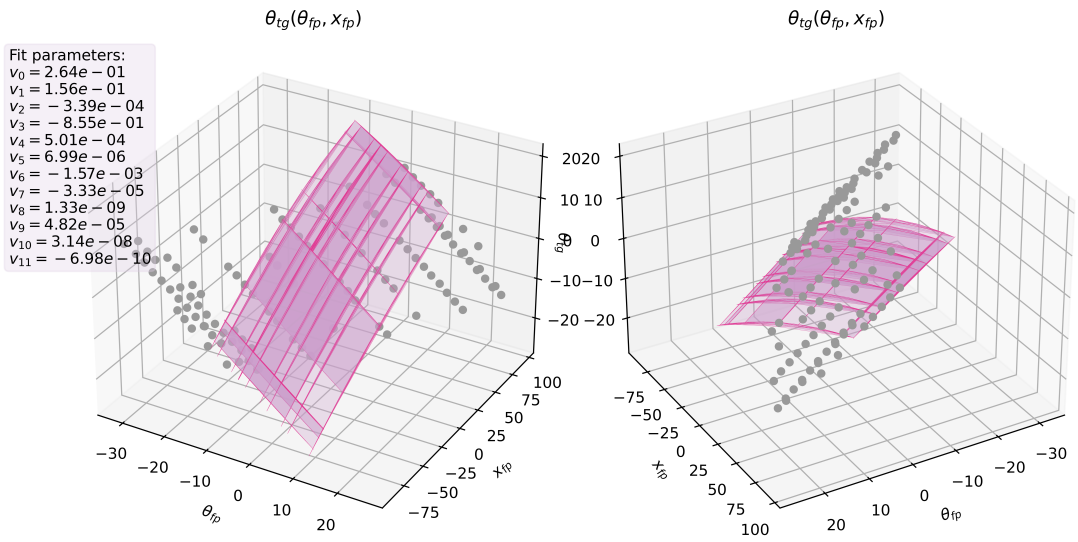
Slika 7: Reskalirani graf in primerjava relativnega odstopanja od podatkov.

4.3 Rekonstrukcija koordinat pri poskusih s spektrometrom

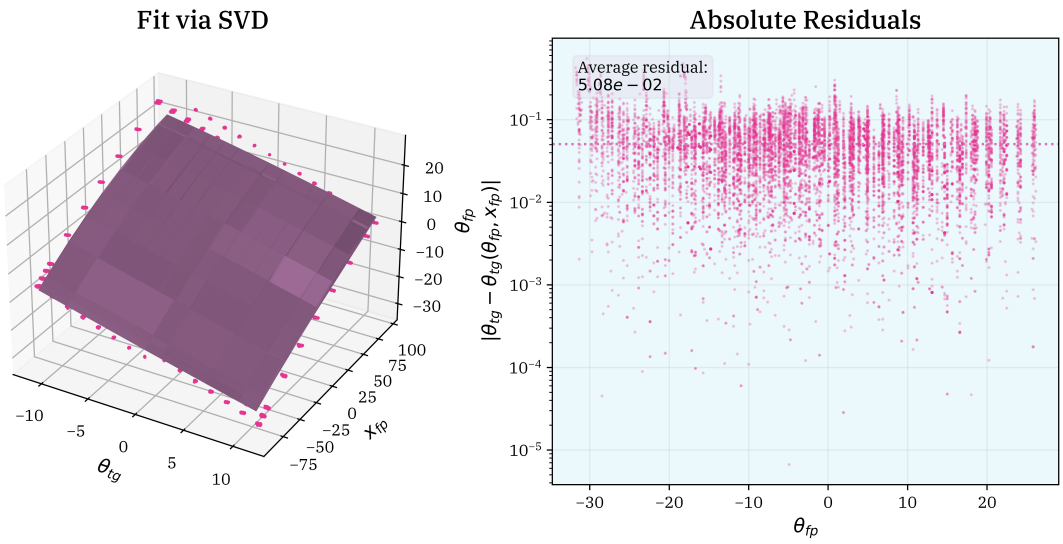
Ta del naloge sem moral žal rešiti ekspresno hitro. Imel sem želje, da bi minimiziral funkcijo z komercialnim solverjem `Gurobi`, kjer bi lahko navil število parametrov če bi želel. Dodatno so podatki podani v tako sugestivnem načinu, da se mi zdi, da prav kličejo po strojnem učenju, čeprav to ni nujno praktično, če želimo dobiti dejansko zvezo, ki opisuje naše podatke. Machine learning je žal pogosto črna škatla, ki nam da rezultat, ki ga ne moremo razložiti.

Vseeno sem pa naredil 12 parametričen model in ga minimiziral z `Gurobi`, nato pa sem ga še rešil preko `SVD`. Rezultata sta prikazana na slikah (8) in (9). Vidimo, da je rezultat zelo podoben, ampak je `SVD` nekoliko boljši. To je zanimivo, ker je `Gurobi` v osnovi boljši solver, ampak ne vem kje sem ga polomil pri izdelavi modela v njegovem sistemu. Nekako uspem opisati naklon ravnivne v $3D$ prostoru, ampak orientacija pa se mi ne zdi pravilna. Torej yes, ta del naloge res leaves something to be desired.

Čeprav glede na to, da sta profesorja rekla, da jim za 100 parametričen model optimizira nekaj mescev if memory serves right je potem napaka 10^{-2} acceptable za model, ki ima 12 parametrov in se reši v nanosekundah z `Gurobi`.



Slika 8: Rezultat minimizacije z `Gurobi`.



Slika 9: Rezultat minimizacije z `SVD`.

5 Komentarji in izboljšave

Tokrat sem se naloge lotil res takoj takoj ko smo jo dobili in nekako sem še vedno padel v zmoto, da sem imel prevelik cilj za par dni dela. Zato sem se moral zadnji dan kar malo panično lotiti tretje naloge, ki sem jo na koncu naredil zelo na hitro in mi je za to prav škoda. Mogoče se bom res moral končno naučiti kako razpolagati s svojim časom ultra učinkovito, ker takole ne bo šlo več naprej I'm afraid.