Sveučilište u Zagrebu
Prirodoslovno - matematički fakultet
Matematički odjel

Metoda najmanjih kvadrata na splajn funkcijama

Darija Marković

1 Uvod

Osnovni problem aproksimacije je slijedeći: neka su zadani podaci (x_i, y_i), $i = 1, \ldots, m$. Želimo pronaći funkciju $y(x) = y(x; \theta)$ poznatog oblika koja sadrži vektor θ s n nepoznatih parametara takvu da je

$$y(x_i) \approx y_i \tag{1}$$

(u nekom smislu). Kasnije ćemo specificirati uvjet (1) koji ćemo zvati kriterij aproksimacije.

Razlozi i ciljevi fitovanja s krivuljama su različiti(vidi [6]):

- **procjena parametara:** oblik $y(x)$ može biti zadan u kontekstu primjene, pri čemu parametri θ_i, $i = 1, \ldots, n$ najčešće imaju fizikalno značenje. Tada je glavni cilj što točnije procijeniti vektor parametara iz danih podataka. Nadalje ćemo pretpostaviti da funkcionalni oblik $y(x)$ nema fizikalno značenje.

- **zaglađivanje podataka:** ako su dane vrijednosti dovoljno točne, moglo bi biti dovoljno odrediti interpolacijsku funkciju takvu da je $y(x_i) = y_i$, $i = 1, \ldots, m$. No u većini slučajeva podaci y_i su povezani s grešakma mjerenja. Zbog toga se nadamo da će postupkom fitovanja te greške biti "zaglađene", tj. da će $y(x_i)$ biti točniji od y_i. Također, vezano uz to može nam biti bitno da graf funkcije $y(x)$ izgleda dovoljno glatko.

- **funkcionalna reprezentacija:** reprezentacije konačnog skupa podataka (x_i, y_i) pomoću funkcije $y(x)$ ima nekoliko prednosti. Prvenstveno, vrijednost u bilo kojoj točki x u domeni reprezentacije se lako izračuna. Nadalje aproksimacija se može koristiti za određivanje vrijednosti derivacije, određenih integrala i slično.

- **reduciranje podataka:** ako rezultat treba biti sačuvan za kasnije korištenje, može biti bitno da broj parametara θ_i bude puno manji od broja podataka. U tom slučaju govorimo o reduciranju podataka.

Što se tiče oblika funkcije $y(x)$ najčešće se koriste polinomi i splajnovi, pri čemu su splajnovi fleksibilniji (za razliku od polinoma jednako su dobri za zaglađivanje kao i za interpolaciju, bez obzira na broj podataka i njihove pozicije).

Pristup u kojem se aproksimacijski problem reducira na određivanje diskretnog skupa parametara θ_i se često naziva konstruktivni pristup. Uz to postoji i takozvani varijacijski pristup u kojem funkcionalni oblik $y(x)$ nije unaprijed specificiran, ali slijedi iz rješenja varijacijskog problema koji se često može interpretirati kao minimizacija potencijalne energije. I u tom pristupu se splajn funkcije javljaju kao rješenje određenih problema ove vrste.

Ako aproksimaciju funkcije tražimo među splajn funkcijama tada moramo odrediti (ili fiksirati) slijedeće parametre θ_i:

- stupanj k splajna
- broj i poziciju čvorova λ_i, $i = 1, \ldots, g$
- koeficijente c_i u reprezentaciji

$$s(x) = \sum_{i=-k}^{g} c_i B_{i,k+1}(x)$$

Što se tiče stupnja splajna najčešće se predlaže korištenje kubičnog splajna ($k = 3$). On obično čini dobar kompromis između efikasnosti i kvalitete aproksimacije. Dakako, ako nas zanimaju i derivacije višeg reda aproksimacije tada se koristi splajn višeg stupnja pri čemu se obično koriste splajnovi neparnog stupnja.

Za određivanje preostalih parametara postoji nekoliko pristupa koji se međusobno razlikuju po izboru čvorova i, vezano uz to, korištenom aproksimacijskom kriteriju.

Najčešće korišteni kriteriji su slijedeći: kriterij najmanjih kvadrata, kriterij prirodno zaglađujućeg splajna, Powell-ov kriterij i kriterij zaglađivanja (više o njima vidi [6]). Mi ćemo se nadalje baviti problemom aproksimacije primjenjujući kriterij najmanjih kvadrata.

2 Aproksimacija sa splajnom najmanjih kvadrata

Kriterij najmanjih kvadrata dobro je poznat i općenit aproksimacijski kriterij. Primjenjen na splajn funkcije vodi do pronalaženja splajna $s(x)$ za koji izraz

$$\delta := \sum_{i=1}^{m} (w_i(y_i - s(x_i)))^2 \tag{2}$$

postiže minimum.

Brojevi w_i, koje zovemo težinama, omogućavaju da razlikujemo točnost mjerenja y_i. Ako imamo ideju (ili procjenu) o apsolutnoj greški od y_i, pripadajuću težinu bi trebali uzeti tako da bude obrnuto proporcionalna točnosti podatka y_i. No ako ne znamo ništa o greški, tada je najlakše (iako ne i najbolje) odrediti da su sve težine $w_i = 1$, i tada govorimo o netežinskom problemu najmanjih kvadrata (aproksimacija u normi ℓ_2). U slučaju da primjetimo kako određeni podatak ima preveliki (ili premali) utjecaj na rezultirajuću aproksimaciju trebali bi smanjiti (povećati) pripadnu težinu. No, eksperimentiranje s različitim težinama može biti frustrirajuće, i često oduzima puno vremena, te bi određivanje aproksimacije u normi ℓ_1 ili ℓ_∞ bilo dobra alternativa bez obzira na veliku računsku složenost.

Ako su nam uz stupanj k fiksirani i čvorovi splajna, tada je pripadajući problem najmanjih kvadrata linearan; $s(x)$ je linearan po nepoznatim B-splajn koeficijentima $\mathbf{c}$. Oni se tada mogu pronaći kao rješenje najmanjih kvadrata prezadanog sustava linearnih jednadžbi. Vezano uz lokalnost B-splajnova, matrica sustava je rijetka i ima specifičnu vrpčastu strukturu koju možemo iskoristiti. U ovom pristupu trebamo odrediti broj i položaj čvorova što nije trivijalan zadatak, i često se svodi na metodu pokušaja i promašaja. Jedino što može poslužiti kao pravilo pri smještanju čvorova je to da se više čvorova treba smjestiti unutar intervala brze promjene vrijednosti mjerenja ili pak u područje gdje aproksimirajuća krivulja ima kompleksan oblik. Također poziciju čvorova možemo iskoristiti ako želimo smanjiti uvjete neprekidnosti u nekoj točki unutar intervala aproksimacije i to tako da u toj točki smjestimo višestruki čvor.

Brojni autori razmatrali su problem aproksimacije uzevši čvorove kao promjenjive parametre (uz fiksan ili promjenjiv broj čvorova). U tom slučaju dolazimo do nelinearnog problema najmanjih kvadrata. Dodatno, to je ograničeni optimizacijski problem pošto

zahtjevamo da su čvorovi u neopadajućem položaju. Neki teorijski aspekti su diskutirani u [7], što kao rezultat ima teorem u letargiji koji pokazuje da funkcija $\delta(\lambda, \mathbf{c})$ koju minimiziramo ima brojne stacionarne točke. Zbog toga će uspješnost optimizacijskog algoritma jako ovisiti o dobrim početnim pozicijama čvorova.

2.1 Fiksirani čvorovi

Neka je zadan skup podataka (x_i, y_i), $i = 1, \ldots, m$ uz $a \leq x_1 < \ldots < x_i < x_{i+1} < \ldots < x_m \leq b$ i pripadajući skup težina w_i, $i = 1, \ldots, m$. Želimo odrediti splajn $s(x)$ na $[a, b]$ određenog stupnja k s danim čvorovima λ_i, $i = 0, \ldots, g+1$ uz $\lambda_0 = a$ i $\lambda_{g+1} = b$ takve da je izraz (2) minimalan. Dobivamo

$$\delta = \sum_{i=1}^{m} (w_i y_i - \sum_{j=-k}^{g} c_j w_i B_{j,k+1}(x_i))^2$$

koji se može zapisati u matričnom obliku kao

$$\delta = (y - Ec)^\tau (y - Ec) = \|y - Ec\|^2 \tag{3}$$

gdje je

$$E = \begin{bmatrix} w_1 B_{-k,k+1}(x_1) & \cdots & w_1 B_{g,k+1}(x_1) \\ \vdots & & \vdots \\ w_m B_{-k,k+1}(x_m) & \cdots & w_m B_{g,k+1}(x_m) \end{bmatrix},$$

$$y = \begin{bmatrix} w_1 y_1 \\ \vdots \\ w_m y_m \end{bmatrix}, \quad c = \begin{bmatrix} c_{-k} \\ \vdots \\ c_g \end{bmatrix}$$

gdje $\|\cdot\|$ predstavlja standardnu Euklidsku vektorsku normu. Dakle, naš problem se svodi na određivanje B-splajn koeficijenata c_j, $j = -k, \ldots, g$, kao rješenja prezadanog linearnog sustava

$$Ec = y \tag{4}$$

u smislu najmanjih kvadrata.

Nadalje ćemo pretpostaviti da matrica E ima puni rang po stupcima $g + k + 1$, što će biti ispunjeno ako i samo ako postoji podskup $\{u_{-k}, \ldots, u_g\} \subset \{x_1, \ldots, x_m\}$ uz $u_j < u_{j+1}$ takav da je

$$\lambda_j < u_j < \lambda_{j+k+1}, \quad j = -k, \ldots, g,$$

tj. ako uređeni podskup zadovoljava Schoenberg - Whitney-ev uvjet.

2.1.1 Normalne jednadžbe

Rješenje najmanjih kvadrata promatrane jednadžbe (4) možemo pronaći riješavajući normalne jednadžbe

$$Ac = r \tag{5}$$

gdje je

$$A = E^\tau E = \begin{bmatrix} \langle B_{-k}, B_{-k}\rangle & \cdots & \langle B_{-k}, B_g\rangle \\ \vdots & & \vdots \\ \langle B_g, B_{-k}\rangle & \cdots & \langle B_g, B_g\rangle \end{bmatrix},$$

$$r = E^\tau y = \begin{bmatrix} \langle B_{-k}, y\rangle \\ \vdots \\ \langle B_g, y\rangle \end{bmatrix},$$

uz

$$\langle B_j, B_i\rangle = \sum_{r=1}^{m} w_r^2 B_{j,k+1}(x_r) B_{i,k+1}(x_r),$$

$$\langle B_j, y\rangle = \sum_{r=1}^{m} w_r^2 B_{j,k+1}(x_r) y_r.$$

Zaista, raspisivanjem jednadžbe (5) lako se zaključi da je zadovoljen nužan uvjet minimizacije, tj. da je $\frac{\partial \delta}{\partial c_j} = 0$, $j = -k, \ldots, g$.

Matrica A ima nekoliko bitnih osobina

- ona je simetrična ($A^\tau = A$) i pozitivno definitna matrica ($x^\tau A x > 0 \quad \forall x \neq 0$)
- ako je g dovoljno velik, to je rijetka matrica vrpčaste strukture. Iz svojstva lokalnosti B-splajn baze dobivamo

$$\langle B_j, B_i\rangle = 0 \quad \text{ako je} \quad |i-j| > k$$

i dobivamo

$$\mathrm{BAND}(A) = \min\{g + k + 1, 2k + 1\}$$

Definicija 1 *Neka je $A \in \mathbb{R}^{p\times q}$ $(p \geq q)$ vrpčasta matrica, te neka je sa f_i, odnosno s l_i označen broj stupca u kojem se pojavljuje prvi, odnosno posljednji element i-tog retka matrice A različit od nule. Tada je širina vrpce BAND matrice A definirana s*

$$BAND(A) = \max\{l_i - f_i + 1 | 1 \leq i \leq p\}.$$

Sustav (5) se tada može efikasno riješiti koristeći malo memorije Cholesky metodom za pozitivne vrpčaste matrice. Nenul elementi matrice A i vektora r mogu se efikasno izračunati koristeći slijedeću shemu :

- inicijalizirati $< B_j, B_i >= 0$ i $< B_j, y >= 0$
- za $r = 1, \ldots, m$:
  - odredi l takav da je $\lambda_l \leq x_r < \lambda_{l+1}$ ($l = g$ ako je $x_r = b$)
  - izračunaj nenul vrijednosti B-splajnova $B_{j,k+1}$, $j = l-k, \ldots, l$
  - za $j = l-k, \ldots, l$:
    - $\langle B_j, y \rangle = \langle B_j, y \rangle + w_r^2 B_{j,k+1}(x_r)$
    - za $i = j, \ldots, l$:

$$\langle B_j, B_i \rangle = \langle B_j, B_i \rangle + w_r^2 B_{j,k+1}(x_r) B_{i,k+1}(x_r)$$

2.1.2 Metoda ortogonalizacija

Da bi rješenje najmanjih kvadrata sustava (4) odredili stabilnije možemo koristiti metodu ortogonalizacije. Neka je Q orotogonalna matrica $Q^\tau Q = I$ reda m takva da je

$$E = QR = Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \tag{6}$$

gdje je R_1 gornje trokutasta matrica reda $g + k + 1$. Nadalje, neka je

$$y = Qz = Q \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \tag{7}$$

gdje z_1 ima $g + k + 1$ elemenata. Ako uvrstimo (6) i (7) u (3) dobivamo

$$\delta = (z - Rc)^\tau (z - Rc) = \|z_1 - R_1 c\|^2 + \|z_2\|^2$$

Iz toga slijedi da će δ biti minimaliziran, i to vrijednošću $\|z_2\|^2$, ako je c rješenje trokutastog sustava

$$R_1 c = z_1$$

koje se jednostavno dobije supstitucijom unazad.

Točna dekompozicija (6) može se dobiti na nekoliko načina. Pojasnit ćemo dva koja se najčešće koriste.

Givensove rotacije

Izuzetno efikasna metoda za određivanje $R = Q^\tau E$ i $z = Q^\tau y$ za našu primjenu je korištenje niza Givensovih rotacija. U načelu, Givensove transformacije rotiraju dva vektora

$$0 \cdots 0\ r_i\ r_{i+1}\ r_{i+2} \cdots$$

$$0 \cdots 0\ e_i\ e_{i+1}\ e_{i+2} \cdots$$

i zamjenjuju ih sa

$$0 \cdots 0\ r'_i\ r'_{i+1}\ r'_{i+2} \cdots$$

$$0 \cdots 0\ 0\ e'_{i+1}\ e'_{i+2} \cdots$$

gdje je

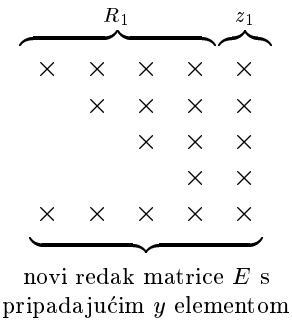
$$\begin{aligned} r'_i &= \sqrt{r_i^2 + e_i^2}, \\ c &= r_i / r'_i, \\ s &= e_i / r'_i, \\ r'_l &= c r_l + s e_l, \quad l = i+1, \ldots, \\ e'_l &= -s r_l + c e_l, \quad l = i+1, \ldots . \end{aligned} \tag{8}$$

Neka su dani vektori u_j -ti i v_j -ti reci matrice E_j. Ako ih zamjenimo s novim vektorima da bi dobili E_{j+1}, gore navedeni postupak može se zapisati u matričnom obliku kao $E_{j+1} = Q_j^\tau E_j$, gdje je

$$Q_j^\tau = \begin{bmatrix} \ddots & & & & & & & \\ & 1 & & & & & & \\ & & c & & & & s & \\ & & & 1 & & & & \\ & & & & \ddots & & & \\ & & & & & 1 & & \\ & & -s & & & & c & \\ & & & & & & & 1 \\ & & & & & & & & \ddots \end{bmatrix} \begin{array}{l} \\ \\ \\ \\ \\ \\ \longleftarrow \text{redak } u_j \\ \\ \longleftarrow \text{redak } v_j \\ \\ \end{array}$$

koja sa od m-dimenzionalne jedinične matrice razlikuje samo u označenim recima. Iz (8) se vidi da je $c^2 + s^2 = 1$ pa slijedi da je $Q_j^\tau Q_j = I$. Pošto je umnožak ortogonalnih matrica ortogonalna matrica možemo zaključiti da niz Givensovih rotacija ($j = 1, \ldots, J$ uz $E_1 = E$) rezultira ortogonalnom transformacijom matrice E ($R = Q^\tau E$ gdje je $Q = Q_1 Q_2 \ldots Q_J$).

Pretpostavimo da imamo gornje trokutastu matricu R_1 s jednim dodatnim retkom



Koristeći (8) možemo matricu svesti na gornje trokutasti oblik rotirajući posljednji redak redom s prvim, drugim, trećim itd. retkom matrice R_1 dok se cijeli redak ne transformira u nulu. Nadalje, iz (8) zaključujemo da rotacija retka s nul-retkom od R_1 (svi $r_l = 0$) rezultira zamjenom redaka sa čime završava proces poništavanja retka. Dakle, ako R_1 i z inicijaliziramo kao nule, gore opisan postupak može se koristiti za svođenje matrice E na trokutasti oblik. Također možemo simultano računati sumu kvadrata $\delta = \|z_2\|^2$ mijenjajući δ (inicijaliziran na nulu) s kvadratom transformiranog elementa vektora y nakon poništavanja retka matrice E.

Ova metoda ima dvije prednosti:

- matricu E promatramo redak po redak te nema potrebe za čuvanje cijele matrice E (niti ortogonalne matrice Q), u memoriji jedino trebamo pohraniti trokutastu matricu R_1 i vektor z_1. Ova nam je činjenica bitna pošto je u većini primjena red m matrica E i Q dosta veći od reda $g + k + 1$ matrice R_1. Dodatni reci koje dobivamo novim promatranjima ili ograničenjima mogu se lako dodati bez potreba za ponovnim provođenjem cijele triangularizacije

- matrica E ima specifičnu vrpčastu strukturu

  (a) iz lokalnosti B-splajn baze slijedi da je najviše $k + 1$ susjednih elemenata u retku različito od nule. Dakle

  $$\mathrm{BAND}(E) = k + 1$$

  (b) Ako su podaci x_i dani u rastućem poretku, matrica E ima standardni oblik, tj. stupčana pozicija prvog nenul elementa u svakom retku je neopadajuća funkcija

broja redaka ili

$$f_i \leq f_{i+1} \quad i = 1, \ldots, m-1$$

gdje je f_i kao u definiciji 1.

Metoda u potpunosti može iskoristiti vrpčasti oblik

- iz (8) možemo vidjeti da se pripadajući nulelementi u određenom stupcu matrice E ne mijenjaju s rotacijom ($r'_l = e'_l$ ako je $r_l = e_l = 0$). Kako posljedicu dobivamo da matrica R_1 također ima vrpčastu strukturu s

$$\text{BAND}(R_1) = k + 1.$$

  Nadalje, transformaciju možemo ograničiti na vektore duljine $k+1$ ($k+2$ ako brojimo i vektor slobodnih koeficijenata), tj. za $l = i+1, \ldots, i+k$

- da bi poništili novi redak od E trebamo najviše $k+1$ rotaciju, tj. ako $\lambda_l \leq x_r < \lambda_{l+1}$ ($l = g$ ako je $x_r = b$) provodimo rotaciju s recima $l+1, \ldots, l+k+1$. Ovo se odnosi samo na vrpčaste matrice standardnog oblika.

Householderove transformacije

Još jedan način pronalaženja QR-dekompozicije matrice E je primjenom Householderovih transformacija. Na početku, definiramo matricu H slijedećom formulom

$$H = H(u) := I - 2uu^\tau, \quad u^\tau u = 1, \quad u \in \mathbb{R}^m, \quad H \in \mathbb{R}^{m \times m} \tag{9}$$

i nju zovemo Householderova matrica. Vektor u nazivamo Householderov vektor. Primjetimo da je Householderova matrica simetrična i ortogonalna.

U cilju svođenja matrice E na gornje trokutasti oblik, njene stupce ćemo promatrati kao vektore iz $\mathbb{R}^m$. Pretpostavimo općenito da je $a \in \mathbb{R}^m$ i da je za $i \geq 1$ barem jedna od komponenti $a_i, \ldots, a_m$ vektora a različita od nule. konstruirat ćemo Householderovu matricu H tako da se prvih $(i-1)$ komponenti vektora $b := Ha$ i vektora a podudaraju, a da se posljednjih $(m-i)$ komponenti vektora b poništavaju. Dakle treba biti

$$(Ha)_j = b_j = \begin{cases} a_j, & j = 1, \ldots, i-1 \\ \pm\gamma, & j = i, \gamma = \sqrt{a_i^2 + \cdots a_m^2} \\ 0, & j = i+1, \ldots, m. \end{cases}$$

Pri tome ćemo predznak i-te komponente vektora b odabrati tako da bude suprotan predznaku od a_i, tj. $b_i = -\text{sgn}(a_i)\gamma$. Očito je $\|a\| = \|b\|$.

Koristeći jednakost $|a_i| = a_i\text{sgn}(a_i)$ dobivamo

$$\|a-b\| = (a_i+\text{sgn}(a_i)\gamma)^2+a_{i+1}^2+\cdots+a_m^2 = a_i^2+2\gamma a_i\text{sgn}(a_i)+\gamma^2+a_{i+1}^2+\cdots+a_m^2 = 2\gamma(\gamma+|a|).$$

Zato je

$$u = \frac{a-b}{\|a-b\|} = \frac{1}{\sqrt{2\gamma(\gamma+|a|)}} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a_i+\text{sgn}(a_i)\gamma \\ a_{i+1} \\ \vdots \\ a_m \end{bmatrix}$$

pripadni Householderov vektor, te je matrica $H = I - 2uu^\tau$ upravo ona za koju je $Ha = b$.

Gore navedeni postupak opisuje jedan korak u procesu triangularizacije, tj. sukcesivnom primjenom Householderovih matrica H_i na matricu E dobivamo gornje trokutastu matricu R

$$R = H_m H_{m-1} \cdots H_2 H_1 E$$

iz čega zbog $H_k^{-1} = H_k^\tau = H_k$ dobivamo

$$E = H_1 H_2 \cdots H_{m-1} H_m R.$$

Kako je produkt ortogonalnih matrica opet ortogonalna matrica na ovaj način dobivamo QR-dekompoziciju matrice E

$$E = QR, \quad \text{gdje je } Q = H_1 H_2 \cdots H_{m-1} H_m.$$

Kao i kod Givensovih rotacija i ovaj način svođenja matrice na gornje trokutasti oblik može iskoristiti specijalni oblik matrice E, tj. njenu vrpčastu strukturu.

Pretpostavimo da su reci matrice E poslagani tako da je E u standardnom obliku. Tada matricu možemo zapisati u blok obliku

$$E = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_q \end{bmatrix} \quad q \leq m$$

pri čemu blok E_i sadrži sve retke matrice E čiji je prvi nenul element u i-tom stupcu, za $i = 1, \ldots, q$. Inicijaliziramo $R = R_0$ gornje trokutasta matrica širine $(k+1)$. Householderov algoritam djeluje u q koraka, $i = 1, \ldots, q$. Nakon prvih $i-1$ koraka prvih $i-1$ blokova je reducirano nizom Householderovih transformacija u gornje trapezoidnu vrpčastu matricu R_{i-1}. U i-tom koraku dodajemo i-ti blok E_i i računamo

$$Q_i^T \begin{bmatrix} R_{i-1} \\ E_i \end{bmatrix} = R_i,$$

gdje je Q_i produkt Householderovih transformacija, a R_i je ponovno gornje trapezopidna. Primjetite da radi toga što blok E_i ima prvi nenul element u stupcu i, kasniji koraci neće mijenjati prvih $i-1$ redaka i stupaca od R_{i-1}, odnosno prvih $i-1$ redaka od R_{i-1} su reci konačne matrice R.

2.2 Algoritmi za dodavanje čvorova

Pretpostavimo da za trenutni skup čvorova λ_j, $j = 1, \ldots, n$, imamo izračunat splajn najmanjih kvadrata $s(x)$. Tada veličina $|e_i| = w_i|y_i - s(x_i)|$, $i = 1, \ldots, m$ daje mjeru blizine krivulje $i-$tom podatku. Dakle imamo merit funkciju u obliku preslikavanja sa skupa podataka u apsolutne vrijednosti težinskog reziduala. Jednostavna strategija za dodatno postavljanje čvorova je slijedeća: odredimo točku x_{MF} u kojoj je merit funkcija najveća i u kojoj još uvijek nije postavljen čvor i postavljamo dodatni čvor u tu točku. Dakle ako počnemo bez unutarnjih čvorova ovaj algoritam generira skup od g čvorova, $g = 1, 2, \ldots$ za koju pripadajuće aproksimacije imaju svojstvo da se poboljšavaju kako se g povećava.

Druga mogućnost je ta da računamo

$$\eta_j = \sum_{x_i \in I_j} e_i^2, \qquad j = 0, 1, \ldots, g$$

i nađemo

$$\eta_u = \max_{j=0,\ldots,g} \eta_j$$

gdje smo s I_j označili interval $[\lambda_j, \lambda_{j+1}\rangle$, $j = 1, \ldots, g-1$ i $[\lambda_g, \lambda_{g+1}]$ za $j = g$. Tada je merit funkcija preslikavanje s intervala čvorova u sumu kvadrata težinskih reziduala na tim intervalima. Sa η_u je određen interval čvorova u kojem je aproksimacija najlošija s obzirom na ovu mjeru. Dodatni čvor se postavlja u točku

$$\lambda' = \sum_{x_i \in I_u} \frac{e_i^2 x_i}{\eta_u}$$

unutar tog intervala, to jest na neku težinsku sredinu apscisa podataka u intervalu I_u (kojem pripada η_u). Primjetimo da se u svakom koraku može dodati jedan ili više dodatnih čvorova.

Još jedna strategija može biti upotrebljena. Neka je $\{x_{p_j}, \ldots, x_{p_j+q_j}\}$ skup vrijednosti apscisa unutar intervala I_j i neka je $s(x)$ splajn baziran na trenutnom skupu čvorova. Računamo

$$T_j = \frac{\sum_{i=p_j}^{p_j+q_j-1} r_i r_{i+1}}{\sum_{i=p_j}^{p_j+q_j} r_i^2}$$

gdje je

$$r_i = y_i - s(x_i).$$

Ukoliko je vrijednost T_j veća od $\frac{1}{\sqrt{q_j}}$ to nam ukazuje da postoji trend među rezidualima unutar intervala I_j. Strategija se temelji na dodavanju dodatnog čvora na sredinu svakog intervala u kojem je pronađen trend. Kompletni algoritam uključuje i neke termine zaglađivanja, no strategija smještanja čvorova je jednako prihvatljiva i u uobičajenoj aproksimaciji podataka metodom najmanjih kvadrata.

Još jedan način za mjerenje trenda vrijednosti reziduala je traženje dijelova na kojima su reziduali istog predznaka. Dugo čuvanje predznaka rezidula interpretiramo kao pokazatelj trenda u tim rezidualima i stoga aproksimaciju u tom dijelu smatramo neprihvatljivom. Postoje dva algoritma koji koriste ovaj način pronalaženja trenda. Prvi dodaje jedan čvor unutar područja s najdužim nizom bez promjene u predznaku i to na slijedeći način: ako su i_1 i i_2 indeksi prvog i posljednjeg podatka koji pripadaju tom nizu tada novi čvor dodajemo na mjesto

$$\lambda' = x_r \quad , \qquad r = \frac{i_1 + i_2}{2}$$

ako je $i_1 + i_2$ paran, ili na

$$\lambda' = \frac{x_{r-\frac{1}{2}} + x_{r+\frac{1}{2}}}{2} \quad , \qquad r = \frac{i_1 + i_2}{2}$$

ako je $i_1 + i_2$ neparan. Drugi algoritam dodaje novi čvor na mjestu gdje je niz predznaka najdulji i unutar svakog niza čija je duljina veća od parametra DULJINA koji zadaje korisnik.

2.3 Slobodni čvorovi

Kvaliteta aproksimacije splajnom najmanjih kvadrata uvelike ovisi o broju i poziciji čvorova. Jasno je da pronalazak "dobre" lokacije čvorova nije trivijalan zadatak. Iz primjena je jasno da ekvidistantna mreža čvorova često daje aproksimaciju loše kvalitete, a već je prije napomenuto da je pronalazak optimalne lokacije čvorova metodom pokušaja težak i naporan zadatak. Zbog toga su brojni autori razmatrali nelinearni problem najmanjih kvadrata u kojem se i pozicija čvorova $\lambda_1, \ldots, \lambda_g$ također optimizira. Većina autora na početku odvajaja linearni i nelinearni aspekt problema, tj. umjesto razmatranja ukupnog problema $\delta(\lambda, \mathbf{c})$, oni pokušavaju minimizirati $\delta(\lambda)$ gdje su za svaki λ pripadajući B-splajn koeficjenti $\mathbf{c}$ oni razmatrani kod problema s fiksiranim čvorovima.

Pošto zahtjevamo da čvorovi budu u neopadajućem poretku optimalni λ moramo tražiti unutar g-dimenzionalnog simpleksa

$$S_g[a,b] = \{\lambda = (\lambda_1, \ldots, \lambda_g) | a < \lambda_1 < \ldots < \lambda_g < b\}.$$

Ozbiljan problem pri tome je postojanje mnogih stacionarnih točaka (lokalni minimumi, sedlaste točke i lokalni maksimumi) $\delta(\lambda)$ na granici $\partial S_g[a,b]$. To intuitivno može biti objašnjeno na sljedeći način: zbog jednostavnosti neka je $g = 2$ i promatramo trokut $S_2[a,b]$ u kojem se nalaze dopušteni parovi čvorova (λ_1, λ_2). Svakoj točki $\lambda^* = (\lambda_1, \lambda_2) \in S_2[a,b]$ pripidružimo točku $\lambda' = (\lambda_2, \lambda_1)$ izvan $S_2[a,b]$. Prema svojstavima B-splajn baze možemo zaključiti da je $\delta(\lambda^*) = \delta(\lambda')$. Kao posljedica simetrije vektor gradijenta $\nabla\delta$ za točke na rubu $\lambda_1 = \lambda_2$ bit će usmjeren duž tog ruba. Pošto je $\delta(a,a) = \delta(b,b)$ vezano za težinski polinom najmanjih kvadrata stupnja k (bez unutarnjih čvorova) i $\delta(c,c)$ za težinski splajn najmanjih kvadrata s dvostrukim čvorom u točki c $(a < c < b)$ također slijedi da je $\delta(a,a) \geq \delta(c,c) \leq \delta(b,b)$. Zbog toga je $\nabla\delta(a,a) \cdot \nabla\delta(b,b) \leq 0$ i možemo zaključiti da δ ima barem jednu stacionarnu točku na rubu $\lambda_1 = \lambda_2$.

Temeljita razmatranja o tom problemu proveo je Jupp. Posljedice teorema o letargiji su sljedeće:

- nekonveksnost $\delta(\lambda)$ za bilo koji skup podataka (x_i, y_i)
- potreba za dobrim početnim pozicijama čvorova λ^0
- gotovo je nemoguće zaključiti da li je postignut globalni optimum

- loše ponašanje minimizacijskog algoritma u blizini granice $S_g[a,b]$

Izbjegavanje situacije s gotovo podudarajućim čvorovima je glavni zadatak svih algoritama koji pokušavaju riješiti problem najmanjih kvadrata sa slobodnim čvorovima.

2.3.1 Nepenalizirana procjena

U de Boor-u je predstavljen slijedeći rezultat: pretpostavimo da $S_\lambda^{k+1}[x_1, x_m]$ predstavlja prostor splajnova reda $(k+1)$ definiranog na $[x_1, x_m]$ s g specificiranih unutarnjih čvorova. Ako funkcija f ima $(k+1)$ neprekidnih derivacija na $[x_1, x_m]$, tada postoji splajn s iz S_λ^{k+1} takav da za svaki $j = 0, \ldots, g$

$$\|f - s\|_{[\lambda_j, \lambda_{j+1}]} \leq K_{k+1}(\lambda_{j+1} - \lambda_j)^{k+1} \|f^{(k+1)}\|_{[\lambda_j, \lambda_{j+1}]} \tag{10}$$

gdje $\|\cdot\|_{[\lambda_j, \lambda_{j+1}]}$ predstavlja uniformnu ili max normu na intervalu $[\lambda_j, \lambda_{j+1}]$, a K_{k+1} je pozitivna konstanta koja ovisi o redu $(k+1)$. Primjetite da ako definiramo

$$h = \max_{j=0,\ldots,g} (\lambda_{j+1} - \lambda_j)$$

tada iz (10) imamo

$$\|f - s\| \leq K_{k+1} h^{k+1} \|f^{(k+1)}\| \tag{11}$$

gdje $\|\cdot\|$ predstavlja max normu na $[x_1, x_m]$. Dakle, (10) i (11) daju nam lokalnu, odnosno globalnu ocjenu o udaljenosti f od linearnog potprostora S_λ^{k+1}. Algoritam opisan u de Boor-u temelji se na pronalaženju čvorova za koje se postiže

$$\min(\max_{j=0,\ldots,g} K_{k+1}(\lambda_{j+1} - \lambda_j)^{k+1} \|f^{(k+1)}\|_{[\lambda_j, \lambda_{j+1}]}) \tag{12}$$

i dakle osigurava da ocjene (10) i (11) budu što manje.

Pretpostavimo da smo definirali

$$E_j(\lambda_j, \lambda_{j+1}) = K_{k+1}(\lambda_{j+1} - \lambda_j)^{k+1} \|f^{(k+1)}\|_{[\lambda_j, \lambda_{j+1}]}$$

za $\lambda_{j+1} \geq \lambda_j$ i $j = 0, \ldots, g$. E_j je očito strogo rastuće funkcija s obzirom na λ_{j+1} i strogo padajuće s obzirom na λ_j. Nadalje, ako je $f^{(k+1)}$ neprekidna tada je i E_j neprekidna s obzirom na oba argumenta. Tada možemo postići (12) izabirući čvorove λ_j $(j = 0, \ldots, g)$ tako da je

$$E_j(\lambda_j, \lambda_{j+1}) = \text{const.} \tag{13}$$

Tada, svaki E_j, $j = 0, \ldots, g$ poprima maksimalnu vrijednost. Da bismo vidjeli zašto (13) povlači (12) pretpostavimo da čvorovi λ_j za $j = 0, \ldots, g$ zadovoljavaju (13) za dani $(k+1)$ i funkciju f (takav skup čvorova postoji zbog neprekidnosti $E_j - ta$). Nadalje, pretpostavimo da perturbiramo unutarnje čvorove, recimo λ_{j_0} u $\lambda^*_{j_0}$ gdje, bez smanjenja općenitosti uzmemo $\lambda^*_{j_0} > \lambda_{j_0}$. Tada zbog monotonosti funkcije E_j imamo

$$E_{j_0-1}(\lambda_{j_0-1}, \lambda_{j_0}) < E_{j_0-1}(\lambda_{j_0-1}, \lambda^*_{j_0}) \tag{14}$$

i

$$E_{j_0}(\lambda_{j_0}, \lambda_{j_0+1}) < E_{j_0}(\lambda^*_{j_0}, \lambda_{j_0+1})$$

zapravo, (14) kaže da je naša lokalna ocjena na j_0-tom intervalu povećana kada premjestimo λ_{j_0} u $\lambda^*_{j_0}$. Dakle, maksimum lokalnih ocjena uzetih na intervalima čvorova je povećan. Svi skupovi čvorova mogu se dobiti iz čvorova koji zadovoljavaju (13) na opisan način i zbog toga takvi skupovi čvorova moraju povećati maksimum lokalnih ocjena. Dakle, zadovoljavanje jednadžbe (13) povlači postizanje minimalne lokalne ocjene, odnosno (12). Zanimljivo je u ovom stadiju interpretirati što (12) i (13) zapravo postižu. Cilj nam je odabrati distribuciju unutarnjih čvorova koja minimizira udaljenost između funkcije f i prostora aproksimacijskih funkcija S^{k+1}_λ. Ako fiksiramo ovaj linearni prostor, tada pronalazimo funkciju koja aproksimira f po našem kriteriju fitovanja - odnosno u našem slučaju kriteriju najmanjih kvadrata. Ne tvrdimo da ta aproksimacija funkcije zadovoljava ocjene (10) ili (11).

Pronalaženje čvorova koji zadovoljavaju (13) za fiksiran red splajna je ekvivalentno pronalaženju čvorova koji zadovoljavaju

$$(\|f^{(k+1)}\|_{[\lambda_j, \lambda_{j+1}]})^{1/(k+1)}(\lambda_{j+1} - \lambda_j) = \text{const} \tag{15}$$

za svaki $j = 0, \ldots, g$. no, još uvijek nije lako riješiti (15). Osnovni problem je taj što nam je f nepoznata, pa je stoga i $f^{(k+1)}$ nepoznata. Dakle, (15) možemo primjeniti kada poznajemo aproksimaciju $f^{(k+1)}$ dobivenu na neki način. Npr. ako konstruiramo splajn reda $(k+2)$ za podatke koristeći neki trenutni skup čvorova i deriviramo ga $(k+1)$ puta, dobivamo po dijelovima konstantnu aproksimaciju za $f^{(k+1)}$ (splajn prvog reda). Ali čak i kada znamo $f^{(k+1)}$ vjerojatno je (15) moguće aproksimativno zadovoljiti, a to bi moglo biti računski skupo postići. No, (15) postiže raspored čvorova asimptotski jednak, kada

$g \to \infty$ i $(\lambda_{j+1} - \lambda_j) \to 0$, rasporedu dobivenom uz uvijet da λ_j zadovoljava

$$\int_{\lambda_j}^{\lambda_{j+1}} |f^{(k+1)}(x)|^{1/(k+1)} \mathrm{d}x = \text{const} = \frac{1}{g+1} \int_{x_1}^{x_m} |f^{(k+1)}(x)|^{1/(k+1)} \mathrm{d}x. \qquad (16)$$

Ako imamo po dijelovima konstantnu aproksimaciju $f^{(k+1)}$ to daje problem koji je relativno lako riješiti i to je osnova de Boor-ovog algoritma. Vrijedi primjetiti da je jedna prednost (16) nad (15) to što je konstanta na desnoj strani dobro definirana te imamo jednu nepoznanicu manje nego u (15). Još bitnije, (16) je moguće zadovoljiti s po dijelovima konstantnom aproksimacijom $f^{(k+1)}$ što ne mora biti slučaj u (15) - pretpostavili smo da je $f^{(k+1)}$neprekidna.

Konačno spomenili smo da de Boor-ova metoda u računanju koristi po dijelovima konstantnu aproksimaciju, recimo $h(x)$ za $|f^{(k+1)}|$. Umjesto da koristi splajn aproksimaciju od f reda $(k+2)$ s obzirom na trenutni skup čvorova, kako je prije predloženo, aproksimacija za $|f^{(k+1)}|$ se izvodi iz splajna reda $(k+1)$.

Pretpostavimo da je $s(x)$ splajn aproksimacija f-a $(k+1)$-og reda s unutarnjim čvorovima λ_j $(j = 1, \ldots, g)$. Tada je $s(x)$ po dijelovima polinom stupnja k koji nakon što je deriviran k puta daje po dijelovima konstantu

$$s^{(k)}(x) = \alpha_0 + \sum_{j=1}^{g} \alpha_j (x - \lambda_j)_+^0$$

što je aproksimacija za $f^{(k)}$. Ovdje de Boor koristi bazu krnjih potencija za reprezentaciju $s^{(k)}$, radije nego B-splajnove, koje su definirane sa

$$(x-\tau)_+^r = \begin{cases} (x-\tau)^r & , \ x \geq \tau \\ 0 & , \ x < \tau \end{cases}$$

Tada je

$$H(x) = \int_{x_1}^{x} |s^{(k+1)}(t)| \mathrm{d}t = \sum_{j=1}^{g} |\alpha_j| (x - \lambda_j)_+^0$$

po dijelovima konstantna aproksimacija za

$$\int_{x_1}^{x} |f^{(k+1)}(t)| \mathrm{d}t,$$

traženi integral. Da bi pronašli funkciju $h(x)$ prvog reda koja je jednaka derivaciji funkcije H s obzirom na x, moramo H zamijeniti sa splajnom višeg reda. Zapravo, reprezentiramo lokalno ponašanje funkcije H s parabolom koja H interpolira u točkama $\lambda_{\frac{j-1}{2}}$, $\lambda_{\frac{j+1}{2}}$ i $\lambda_{\frac{1+3}{2}}$

gdje uzimamo $\lambda_{\frac{j+1}{2}} = \frac{\lambda_j + \lambda_{j+1}}{2}$. Tada je, $h(x)$, naša aproksimacija $|f^{(k+1)}|$ definirana na $[\lambda_j, \lambda_{j+1}]$ gradijent u $\lambda_{\frac{j+1}{2}}$ ove lokalne parabole.

Cox, Harris i Jones predstavili su slijedeći način poboljšanja prije navedenog matematičkog algoritma. Ideja je $f^{(k+1)}$ aproksimirati s po dijelovima linearnom funkcijom (odnosno splajnom drugog reda), umjesto s po dijelovima konstantom. Ovo se čini razumno ne samo zbog toga što bi aproksimacija $f^{(k+1)}$ bila bolja, nego i zbog toga što smo na početku zahtjevali neprekidnost $f^{(k+1)}$. Pojasnit ćemo algoritam koji eksplicitno računa čvorove koji zadovoljavaju (16) koristeći takvu aproksimaciju. Naravno, moramo očekivati više rada nego u ranije opisanom algoritmu.

Pretpostavimo da možemo konstruirati po dijelovima linearnu aproksimaciju $l(x)$ za $f^{(k+1)}$ s obzirom na trenutni skup čvorova. Tada po dijelovima linearna funkcija $L(x) = |l(x)|$ može biti uzeta kao aproksimacija za $|f^{(k+1)}|$. Općenito, pošto $l(x)$ obično ima nule u $\langle x_1, x_m \rangle$, $L(x)$ će imati "prošireni" skup čvorova dan s unijom skupa trenutnih čvorova i tih nula.

Elemente proširenog skupa čvorova označit ćemo s τ_r, $r = 1, \ldots, R$ i definiramo vanjske čvorove

$$\tau_{-1} = \tau_0 = x_1 \qquad \tau_{R+1} = \tau_{R+2} = x_m$$

gdje pretpostavljamo da su τ_r, $(r = 0, 1, \ldots, R+1)$ različiti.

Tražimo novi skup čvorova λ_j takav da, za $j = 0, 1, \ldots, g$ vrijedi

$$\int_{\lambda_j}^{\lambda_{j+1}} (L(x))^{1/(k+1)} \mathrm{d}x = \frac{1}{g+1} \int_{x_1}^{x_m} (L(x))^{1/(k+1)} \mathrm{d}x. \tag{17}$$

Na svakom intervalu $[\tau_r, \tau_{r+1}]$ za $r = 0, \ldots, R$ pišemo

$$L(x) = a_r + b_r(x - \tau_r).$$

Naravno, $L(x)$ je splajn drugog reda. Ako je njegova B-splajn reprezentacija

$$L(x) = \sum_{i=-1}^{R} c_i B_{i,2}(\tau, x)$$

gdje je $\tau = (\tau_{-1}, \tau_0, \ldots, \tau_{R+2})^\tau$, tada je

$$L(\tau_r) = c_r.$$

Odavde imamo

$$a_r = c_r; \qquad b_r = \frac{c_{r+1} - c_r}{\tau_{r+1} - \tau_r}. \tag{18}$$

Za svaki $r = 0, 1, \ldots, R$ definiramo

$$J_r = \int_{\tau_r}^{\tau_{r+1}} (L(x))^{1/(k+1)} \mathrm{d}x = \int_{\tau_r}^{\tau_{r+1}} (a_r + b_r(x - \tau_r))^{1/(k+1)} \mathrm{d}x.$$

Koristeći (18) dobivamo

$$J_r = \begin{cases} \frac{k+1}{k+2}(\tau_{r+1} - \tau_r)\frac{c_{r+1}^{(k+2)/(k+1)} - c_r^{(k+1)/(k+1)}}{c_{r+1} - c_r}, & c_r \neq c_{r+1} \\ (\tau_{r+1} - \tau_r)c_r^{1/(k+1)}, & c_r = c_{r+1} \end{cases} \tag{19}$$

Svaki J_r možemo eksplicitno izračunati. Nadalje, konstantni izraz na desnoj strani u (17) jednak je

$$\frac{1}{g+1} \sum_{r=0}^{R} J_r$$

što ćemo označavati s Δ. Dakle, prema (17) tražimo λ_j koji zadovoljava

$$\int_{\lambda_j}^{\lambda_{j+1}} (L(x))^{1/(k+1)} \mathrm{d}x = \Delta.$$

Sada ćemo predstaviti jedan korak algoritma koji određuje λ_{j+1} ako smo ranije izračunali λ_j.

Pretpostavimo da je p jedinstveni prirodan broj takav da je $\lambda_j \in \langle \tau_p, \tau_{p+1}]$. Definiramo

$$\Delta_j^L = \int_{\tau_p}^{\lambda_j} (L(x))^{1/(k+1)} \mathrm{d}x,$$

$$\Delta_j^R = \int_{\lambda_j}^{\tau_{p+1}} (L(x))^{1/(k+1)} \mathrm{d}x.$$

Nadalje, pretpostavimo da je $q (\geq p)$ najmanji broj za koji je

$$\Delta_j^R + \sum_{r=p+1}^{q} J_r \geq \Delta \tag{20}$$

Jasno je da ako je $q = p$ tada suma nema nikakav utjecaj. Ako vrijedi jednakost u (20) tada imamo

$$\int_{\lambda_j}^{\tau_{q+1}} (L(x))^{1/(k+1)} \mathrm{d}x = \Delta$$

i tada je $\lambda_{j+1} = \tau_{q+1}$. Tada uz

$$\Delta_{j+1}^{L} = \int_{\tau_q}^{\lambda_{j+1}} (L(x))^{1/(k+1)} \mathrm{d}x = J_q$$

i

$$\Delta_{j+1}^{R} = \int_{\lambda_{j+1}}^{\tau_{q+1}} (L(x))^{1/(k+1)} \mathrm{d}x = 0$$

možemo nastaviti s pronalaženjem slijedećeg čvora.

Ako jednakost ne stoji, tada je $\lambda_{j+1} \in \langle \tau_q, \tau_{q+1} \rangle$ i razlikujemo dva slučaja

(a) $q = p$.
Tada, iz (20) $\Delta_j^R > \Delta$ pa λ_j i λ_{j+1} leže u istom intervalu $\langle \tau_q, \tau_{q+1} \rangle = \langle \tau_p, \tau_{p+1} \rangle$. Računamo vrijednost integrala

$$\Delta_{j+1}^{L} = \int_{\tau_q}^{\lambda_{j+1}} (L(x))^{1/(k+1)} \mathrm{d}x = \Delta_j^L + \Delta. \tag{21}$$

(b) $q > p$.
Tada λ_j i λ_{j+1} leže u različitim intervalima $\langle \tau_p, \tau_{p+1}]$ i $\langle \tau_q, \tau_{q+1} \rangle$. Računamo vrijednost integrala

$$\Delta_{j+1}^{L} = \int_{\tau_q}^{\lambda_{j+1}} (L(x))^{1/(k+1)} \mathrm{d}x = \Delta - (\Delta_j^R + \sum_{r=p+1}^{q-1} J_r). \tag{22}$$

U oba slučaja, također možemo računati

$$\Delta_{j+1}^{R} = \int_{\lambda_{j+1}}^{\tau_{q+1}} (L(x))^{1/(k+1)} \mathrm{d}x = J_q - \Delta_{j+1}^L. \tag{23}$$

Sada eksplicitno integrirajući možemo riješiti (21), (22) ili (23) za nepoznati λ_{j+1}. Na primjer, iz (21), (22) imamo

$$\int_{\tau_q}^{\lambda_{j+1}} (a_q + b_q(x - \tau_q))^{1/(k+1)} \mathrm{d}x = \Delta_{j+1}^L$$

i dakle

$$\lambda_{j+1} = \begin{cases} \tau_q + \dfrac{\left(\frac{(k+2) b_q \Delta_{j+1}^L}{k+1} + a_q^{(k+2)/(k+1)} \right)^{(k+1)/(k+2)} - a_q}{b_q}, & b_q \neq 0, \\ \tau_q + \dfrac{\Delta_{j+1}^L}{a_q^{1/(k+1)}}, & b_q = 0. \end{cases} \tag{24}$$

Primjetite da nije moguće da su oba $a_q = 0$ i $b_q = 0$ u (24). Inače, ako bi oba bili nula, $L(x) \equiv 0$ na $[\tau_q, \tau_{q+1}]$ i dakle $J_q = 0$. Dakle, ne bi imali slučaj da je q najmanji broj za koji (20) stoji.

Ako inicijaliziramo

$$p = -1, \quad \lambda_0 = x_1, \quad \Delta_0^L = \Delta_0^R = 0$$

tada dana analiza vodi do pronalaženja λ_1. Nadalje sukcesivno računamo λ_j za $j = 2, 3, \ldots, g$.

Ako imamo način za računanje po dijelovima linearne $L(x)$, tada imamo sve što nam je potrebno za algoritam. Nadalje, ako $L(x)$ temeljimo na posljednjem skupu čvorova, te ju mijenjamo kada pronađemo novi skup čvorova, tada imamo osnove strategije za adaptivni razmještaj čvorova.

Razvoj algoritma

(1) stabilna evaluacija formule unutar algoritma

U implementaciji opisanog algoritma moramo računati integrale J_r $(r = 0, \ldots, R)$ i pozicije novih čvorova. Za prvu od ovih operacija možemo koristiti formulu (19), a za drugu ćemo primjeniti (24). Primjetite da za svako računanje imamo par formula ovisno o tome da li je $L(x)$ na promatranom intervalu konstantan ili ne. Ako je $b_r = 0$ (odnosno $c_r = c_{r+1}$), tada prvi dijelovi formula daju vrijednosti koje nisu definirane i moramo koristiti druge dijelove formula (19) i (24). Ako je c_r "blizu" c_{r+1}, tada se u formulama (19) i (24) mogu pojaviti greške vezano uz kraćenje. Zato trebamo razmotriti kako to računanje možemo izvesti na stabilni način.

Prvo primjetimo da za bilo koje brojeve s i t vrijedi lako dokaziv identitet

$$\frac{1 - x^t}{1 - x^{t/s}} = \sum_{i=0}^{s-1} x^{it/s}.$$

No pošto je

$$1 - x^t = (1 - x) \sum_{i=0}^{t-1} x^i$$

imamo

$$\frac{1 - x^{t/s}}{1 - x} = \frac{\sum_{i=0}^{t-1} x^i}{\sum_{i=0}^{s-1} x^{it/s}}. \tag{25}$$

Da bi dobili rezultat x ograničimo da prima nenegativne vrijednosti, kada računanje desne strane u (25) jedino uključuje zbrajanje nenegativnih vrijednosti pa se efekt skraćivanja kada je x blizu jedinice odstranjuje. Nadalje, desna strana daje vrijednosti za sve nenegativne x-eve uključujući $x = 1$. Koristeći L'Hopitalovo pravilo, pronalazimo da je vrijednost

limesa kada $x \to 1$ l/s što je vrijednost desne strane za $x = 1$. Primjetite da za svaki broj x, pošto je

$$\sum_{i=0}^{k} x^i = ((\dots(((x+1)x+1)x+1)\dots)x+1) \tag{26}$$

gdje ima k zagrada, imamo stabilan način za računanje desne strane (25).

Koristimo (25) i (26) da bi postigli stabilno računanje formule na slijedeći način:

(a) računanje J_r :
Ako je $c_r = c_{r+1} = 0$ stavimo $J_r = 0$. Inače, iz (19) i pretpostavljajući bez smanjenja općenitosti da je $c_{r+1} \geq c_r$, imamo

$$J_r = \frac{k+1}{k+1}(\tau_{r+1} - \tau_r)c_{r+1}^{1/(k+1)} \frac{1 - \left(\frac{c_r}{c_{r+1}}\right)^{(k+2)/(k+1)}}{1 - \frac{c_r}{c_{r+1}}}.$$

Sada možemo primjeniti (25) i (26) uz $x = \frac{c_r}{c_{r+1}}$, $t = k+2$ i $s = k+1$. Pošto su c_r i c_{r+1} nenegativni kao vrijednosti nenegativne funkcije $L(x)$ u τ_r, odnosno τ_{r+1}, pa je $x \geq 0$. Dakle, nema problema s kraćenjem kada je c_r blizu c_{r+1}, a primjenjujemo (19) kada je $c_r = c_{r+1}$.

(b) računanje λ_{j+1}:
Iz (24), pretpostavimo da definiramo

$$\lambda_{j+1} = \tau_q + \chi^L$$

gdje je

$$\chi^L = \frac{\left(\frac{(k+2)b_q\Delta_{j+1}^L}{k+1} + a_q^{(k+2)/(k+1)}\right)^{(k+1)/(k+2)} - a_q}{b_q}.$$

Ako je $a_q = 0$ ovo daje

$$\lambda_{j+1} = \tau_q + \left(\frac{(k+2)\Delta_{j+1}^L}{k+1}\right)^{(k+1)/(k+2)} b_q^{-1/(k+2)}$$

i prisjetimo se da garantiramo $b_q \neq 0$. Pretpostavljajući da je $a_q \neq 0$, izraz za χ^L zapišemo na slijedeći način:

$$\chi^L = \frac{a_q}{b_q}\left[\left(1 + \frac{k+2}{k+1}\Delta_{j+1}^L a_q^{-\frac{k+2}{k+1}b_q}\right)^{\frac{k+1}{k+2}} - 1\right] = \beta a_q \frac{(1+\alpha)^{\frac{k+1}{k+2}} - 1}{\alpha},$$

gdje je

$$\beta = \frac{(k+2)\Delta_{j+1}^{L} a_q^{-\frac{k+2}{k+1}}}{k+1},$$

$$\alpha = \beta b_q.$$

Sada postavljajući $x = 1 + \alpha$ daje

$$\lambda_{j+1} = \tau_q + \beta a_q \frac{1 - x^{\frac{k+1}{k+2}}}{1 - x}.$$

Ovo je sada formula prikladna za primjenu (25) i (26). Kada je $x = 1$ (tj. $b_q = 0$), primjenjujemo (24). No, iako za dani x možemo stabilno izračunati gornju formulu, mogli bi izgubiti točnost u računanju x ako je α blizu -1. Zato ćemo koristiti provedenu analizu ako je $\alpha \geq 0$, tj. kada je $b_q \geq 0$.

Kada je $b_q < 0$ koristimo alternativnu formulu za λ_{j+1}. Umjesto računanja λ_{j+1} za (21) ili (22), koristimo (23) i pišemo $L(x)$ za $x \in [\tau_q, \tau_{q+1}]$ u alternativnom obliku

$$L(x) = a_{q+1} + \overline{b}_q(\tau_{q+1} - x),$$

gdje je

$$a_{q+1} = c_{q+1}; \qquad \overline{b}_q = \frac{c_q - c_{q+1}}{\tau_{q+1} - \tau_q} > 0.$$

Tada je λ_{j+1} određen iz

$$\int_{\lambda_{j+1}}^{\tau_{q+1}} \left(a_{q+1} + \overline{b}_q(\tau_{q+1} - x)\right)^{1/(k+1)} \mathrm{d}x = \Delta_{j+1}^{R}$$

i integrirajući

$$\lambda_{j+1} = \begin{cases} \tau_{q+1} + \dfrac{\left(\frac{(k+2)\overline{b}_q \Delta_{j+1}^{R}}{k+1} + a_{q+1}^{(k+2)/(k+1)}\right)^{(k+1)/(k+2)} - a_{q+1}}{\overline{b}_q}, & \overline{b}_q \neq 0, \\ \tau_{q+1} + \dfrac{\Delta_{j+1}^{R}}{a_{q+1}^{1/(k+1)}}, & \overline{b}_q = 0. \end{cases} \tag{27}$$

Sada je (27) u obliku analognom (24). Uz $\overline{b}_q > 0$, možemo računati (27) na stabilan način opisan za (24).

Primjetite da će zbrajanje korištenjem (26) biti najefikasnije u najpraktičnijem slučaju, tj. kada je $(k+1)$ mali.

(2) računanje po dijelovima linearne $L(x)$

U ranije predstavljenom matematičkom algoritmu imamo paradoksalnu situaciju potrebe

za aproksimacijom $(k+1)$-ve derivacije nepoznate funkcije u svrhu pronalaženja skupa čvorova koji daju dobru aproksimaciju same funkcije. Prisjetimo se da $L(x)$ predstavlja po dijelovima linearnu aproksimaciju $|f^{(k+1)}(x)|$ gdje je f nepoznata funkcija. $L(x)$ možemo izračunati iz po dijelovima linearne aproksimacije $l(x)$ od $f^{(k+1)}(x)$. Jasno, efikasnost algoritma ovisi o tome koliko dobro možemo aproksimairati $f^{(k+1)}(x)$ u svakoj iteraciji.

Jednostavan i očit način za konstrukciju $l(x)$ je slijedeći: pronaći splajn $(k+3)$-eg reda za dane podatke (koji je C^{k+1} neprekidna funkcija ako su čvorovi jednostruki) i deriviramo ju $(k+1)$ puta. Ovo je strategija koja je na kraju iskorištena.

Čini se razumnim da temeljimo trenutnu aproksimaciju $l(x)$ u svakoj iteraciji s obzirom na trenutni skup čvorova i da promjenimo $l(x)$ u narednoj iteraciji. No, skup čvorova koji generiramo je izabran tako da poboljša aproksimaciju $(k+1)$-og reda, te ne mora biti da će aproksimacija $(k+3)$-eg reda, iz koje računamo $l(x)$, biti poboljšana. Zato bi, prema de Boorovom algoritmu, računanje $l(x)$ iz splajna nižeg reda, u ovom slučaju aproksimacije $(k+1)$-og reda, moglo biti prikladno. Zato razmatramo slijedeću strategiju:

(a) aproksimiramo podatke sa splajnom reda $(k+1)$ s obzirom na trenutne čvorove i jednom deriviramo

(b) aproksimiramo rezultirajući splajn reda k sa splajnom reda $(k+1)$ koristeći skup podataka dobiven iz vrijednosti splajna u krajevima, čvorovima i njihovim polovištima

(c) ponavljamo (a) i (b) još jednom. Rezultat je splajn aproksimacija reda $(k+1)$ druge derivacije nepoznate funkcije

(d) sada deriviramo $(k-1)$ puta da bi dobili $l(x)$

No u praksi, niti ova niti slične strategije nisu dovele do poboljšanja u ponašanju osnovnog algoritma.

Analiza pokazuje da su greške derivacije na krajevima intervala najveće, što i nije tako iznenađujuće pošto je diferenciranje tipično nestabilan proces. Da bi riješili ovaj problem mogli bi "utjecati" na aproksimaciju derivacije u krajevima preko ponašanja u blizini; to znači zamijeniti vrijednosti u krajevima s vrijednostima izračunatim iz susjednih čvorova. Na primjer, da bi odredili aproksimaciju derivacije u x_1 mogli bi:

(a) interpolirati vrijednosti u λ_1, λ_2 i λ_3 s kvadratnom funkcijom i ekstrapolirati s x_1

(b) interpolirati vrijednosti u λ_1 i λ_2 s pravcem i ekstrapolirati u x_1

(c) interpolirati vrijednosti u λ_1 s konstantom i ekstrapolirati u x_1

(d) derivaciju u x_1 izjednačiti s nulom.

Nakon što aproksimiramo vrijednost u x_1, definiramo $l(x)$ kao linearnu na $[x_1, \lambda_1]$.

(3) strategija uklanjanja čvorova
Za sada je algoritam dizajniran kao strategija za pomicanje fiksiranog broja čvorova u svrhu poboljšanja splajn aproksimacije za dane podatke. No bitan aspekt korištenja splajn funkcija za fitovanje podataka je to što imamo slobodu u biranju kako pozicije tako i broja čvorova. U predstavljenom algoritmu ta dodatna sloboda nije iskorištena. Zaista, kao rezultat toga, ili korisnik treba eksperimentirati s brojem čvorova, ili mora unaprijed znati potreban broj čvorova. Ako je specificirani broj čvorova premal, tada ni optimalna pozicija čvorova neće dati zadovoljavajuću aproksimaciju. Ako kažemo da je broj čvorova prevelik mislimo na to da je moguće preaproksimirati podatke s tim brojem čvorova u nekom položaju. U slučaju kada je količina prostora potrebna za pohranjivanje parametara aproksimacijskog splajna bitna, tada bi fitovanje s minimalnim brojem čvorova bilo poželjno.

Uobičajeno, prikladan broj čvorova nije poznat unaprijed i zato bi bila prednost kada bi automatski, kroz algoritam, odabrali broj čvorova.

Ranije razmatrane strategije temeljile su se na dodavanju čvorova. Ako ih kombiniramo ih s algoritmom za premještanje čvorova javljaju se neki problemi. Na primjer, moramo odlučiti koliko puta ćemo premjestiti fiksirani broj čvorova prije nego dodamo još jedan čvor. Generalnije, koji kriterij mora biti zadovoljen da bi uzrokovao povećenje broja čvorova?

Kombiniranje dodavanja čvorova s premještanjem ne treba odbaciti, ali ovdje želimo slijediti filozofiju uklanjanja čvorova. Startajući s g_1 inicijalnih čvorova, pokušavamo naći skup od g čvorova ($g \leq g_1$) koji daju "dobar" aproksimacijski splajn iz skupa splajnova s najviše g_1 čvorova, pri čemu želimo da je g što manji. Dakle, g_1 je izabran tako da bude "prevelik" i mogao bi biti povezan s veličinom problema, recimo s brojem podataka.

Korištena je vrlo jednostavna strategije uklanjanja čvorova. Osiguravamo da svaki interval čvorova sadrži barem $(k+1)$ različitih točaka podataka s nenul težinama, gdje

je $(k+1)$ red splajna korišten za aproksimaciju podataka. Kada interval čvorova sadrži manje od $(k+1)$ takvih točaka, desni krajnji čvor intervala se uklanja iz skupa unutarnjih čvorova, osim u posljednjem intervalu gdje uklanjamo lijevi čvor. Kao rezultat, Schoenberg - Whitney-jev uvjet je automatski zadovoljen za splajn bilo kojeg reda (primjetite da bi i slabiji uvjet povezan s točkama i čvorovima bio dovoljan). Motivacija za ovu strategiju je dvostruka. Prvenstveno, ako zadovoljiumo Schoenberg - Whitneyj-ev uvjet, uvijek generiramo jedinstvenu splajn aproksimaciju. Kao drugo, spriječavamo gomilanje unutarnjih čvorova i s time modeliranje greške. Dakle puno je manja vjerojatnost preaproksimiranje. Naravno, postoje i skupovi podataka koji zahtjevaju bliske čvorove u nekim dijelovima za dobivanje prihvatljive aproksimacije, te u tom slučaju ne treba koristiti ovu strategiju uklanjanja.

Kompletan algoritam ima slijedeći oblik:

1. Definirati g_1 inicijalnih čvorova;
2. Izračunati po dijelovima linearnu aproksimaciju $f^{(k+1)}$ koristeći aproksimaciju $(k+3)$-eg reda podataka i trenutnih čvorova;
3. Ograničiti aproksimaciju na krajevima;
4. Premjestiti čvorove;
5. Ukloniti čvorove ako je potrebno;
6. Ponavljati 2. do 6. dok ne postignemo zadovoljavajuću aproksimaciju.

Na kraju spomenimo problem koji proizlazi iz automatskog postavljanja čvorova i općenitije, iz postupka fitovanja podataka. To je način mjerenja "dobre" ili "prihvatljive" aproksimacije koji vodi do zaustavljanja postupka premještanja čvorova. U većini slučajeva to ovisi o primjeni (ili čak korisniku). Da bi automatizirali postupak zaustavljanja trebamo definirati što je "dobra" aproksimacija. Ako koristimo gore opisani algoritam problem je još kompliciraniji pošto možda ne želimo zaustaviti postupak čim je zadovoljen prvi kriterij jer je možda moguće postići uvjete i s manjim brojem čvorova. Zato je pri korištenju algoritma, još uvijek, potrebna korisnikova interakcija.

(4) strategije izbora početnih čvorova

Najjednostavnija strategija koju možemo upotrijebiti je takozvana BIKE rutina. Odabir čvorova provodi se na slijedeći način: neka su zadane vrijednosti x_i s pripadajućim težinama w_i i broj g traženih unutarnjih čvorova, te red splajna $(k+1)$. Unutarnji čvorovi postavljaju se tako da je otprilike jednak broj različitih x_i-ova s pozitivnim težinama w_i u svakom podintervalu određenom čvorovima osim u krajnjim intervalima gdje ih ima $(k+1)/2$ puta toliko.

Za neke primjene ovakva distribucija čvorova može biti zadovoljavajuća i može se koristiti da bi pronašli bolju poziciju čvorova (ali ručno), ako pogledamo kakve bi kvalitete bila aproksimacija s tako raspoređenim unutarnjim čvorovima. No možemo i bolje odrediti početne pozicije s alternativnim strategijama.

Ideja je generirati određeni broj ekvidistantnih čvorova, ali umjesto da mjerimo duž osi apscise, mjerimo udaljenost na krivulji. No, krivulja nam još nije poznata pa udaljenost duž krivulje aproksimiramo mjereći udaljenosti točaka podataka. Točke podataka smatramo vrhovima poligona i s d_i označavamo kumulativnu Euklidsku udaljenost od prve do i-te točke, odnosno:

$$\begin{aligned} d_1 &= 0 \\ d_i &= d_{i-1} + \delta d_{i-1}, \quad i = 2, 3, \ldots, m \end{aligned}$$

gdje je

$$\delta d_{i-1} = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}.$$

Parametarska vrijednost

$$t_i = \frac{d_i}{d_m}, \quad i = 1, \ldots, m$$

daje skalarnu mjeru udaljenosti od prve do i-te točke. Skaliranje je napravljeno tako da je udaljenost od prve do posljednje točke jednaka 1. Tada g unutarnjih čvorova postavljamo tako da je udaljenost među susjednim čvorovima mjerena duž nastalog poligona jednaka $\frac{1}{g+1}$. Dakle da bi locirali λ_j trebamo pronaći prvi parametar t_p takav da je $t_p > \frac{j}{g+1}$, jer tada λ_j leži između x_{p-1} i x_p i linearno interpolirajući između tih vrijednosti imamo

$$\lambda_j = x_{p-1} + \frac{\frac{j}{g+1} - t_{p-1}}{t_p - t_{p-1}}(x_p - x_{p-1}).$$

Još jedna strategija za inicijalno postavljanje čvorova koju možemo upotrijebiti temelji se na korištenju lokalne polinomijalne aproksimacije. Pokazat ćemo kako svakom "dovoljno

unutarnjem" podatku možemo pridružiti lokalni skup podataka i lokalnu polinomijalnu aproksimaciju reda $(k+1)$.

Razmatrajmo i-tu točku (x_i, y_i). Prvo konstruiramo podskup susjednih točaka uzetih iz kompletnog skupa podataka, s indeksima postavljenim centralno s obzirom na i koji sadrži $(k+2)$ točku ako je $(k+1)$ paran, odnosno $(k+1)$ točaka za neparan $(k+1)$. Ovaj skup zovemo lokalni skup i-toj točki. Pretpostavljeno je da ove točke mogu biti adekvatno aproksimirane s jednostavnim polimomom reda $(k+1)$. Ovo će svakako biti istina u slučaju kada je $(k+1)$ neparan pošto možemo interpolirati lokalne podatke s takvim polinomom. Sada je jasno što mislimo s time da je točka "dovoljno unutarnja" - to je točka za koju se ovaj inicijalni skup podataka može konstruirati.

Slijedeći korak je povećavanje skupa dodajući susjedne točke na svakom kraju ako je moguće. Polinomijalna aproksimacija reda $(k+1)$ novih lokalnih podataka se izračuna i koristi se Powellov test trenda da procjenimo prihvatljivost aproksimacije. Najveći tako nastao podskup za koji polinomijalna aproksimacija tih podataka zadovoljava test je uzeta za lokalni skup podataka pridružen i-toj točki. Postupak se ponavlja za sve dovoljno unutarnje točke.

Pri korištenju opisanog algoritma, može se javiti problem, posebno za velike skupove podataka, u vezi vremena potrebnog za njegovo provođenje. To je zbog toga što možda trebamo računati veliki broj polinomijalnih aproksimacija (i izračunati vrijednost u svakoj točki pripadajućeg lokalnog skupa podataka za korištenje u trend testu) za svaku od velikog broja točki.

Da bi reducirali količinu posla koristimo veličinu konačnog lokalnog skupa podataka pridruženog $(i-1)$-om podatku kao početnu vrijednost veličine lokalnog skupa podataka za i-tu točku. Ovaj skup se tada povećava ili smanjuje ovisno o tome možemo li ga adekvatno aproksimirati s polinomom u opisanom smislu. Nije jasno da li možemo zaključiti da ako aproksimacija tog skupa podataka zadovoljava trend test, da tada aproksimacija svih manjih skupova mora zadovoljavati test. Ali također, ako uvijek krećemo od najmanjeg podskupa i tražimo prvi lokalni skup podataka koji ne zadovoljava trend test, nije jasno da svi veći skupovi također neće zadovoljiti test. Dakle, vjerujemo da reduciranje posla na ovaj način neće bitno utjecati na rezultatat.

Vrijme izvedbe ovisi i o načinu na koji pronalazimo polinomijalnu aproksimaciju. Tradicionalno, metode korištene za generiranje polinomijalne aproksimacije najmanjih kvadrata su Forsythe i Forsythe-Clenshaw.

Pretpostavimo da smo pronašli lokalne skupove podataka pridružene svim dovoljno unutarnjim točkama koristeći polinomijalnu aproksimaciju reda $(k+1)$. Neka prvi i posljednji podatak u svakom lokalnom skupu označavaju donju i gornju granicu skupa. Tada postavljamo čvor u svakoj točki koja je donja (ili gornja) granica za više od dva lokalna skupa podataka pridruženim uzastopnim podacima. Na ovaj način ne pronalazimo samo procjenu za g nego i inicijalne pozicije čvorova λ_j, $j = 1, \ldots, g$.

Ako je lokalni skup podataka za i-tu točku određen nezavisno od onog za druge točke, tada ova strategija daje vektor λ čiji elementi ne ovise o redu u kojem su točke obrađivane. No, primjetili smo da zbog efikasnosti koristimo veličinu konačnog lokalnog skupa podataka za $(i-1)$-vu točku kao početnu vrijednost veličine lokalnog skupa podataka za i-tu točku. Dakle, točke su nužno procesuirane slijeva na desno i postoji zavisnost između lokalnih skupova uzastopnih točaka. Bolji kriterij je da lociramo čvor u granicu lokalnog skupa podataka koji je veći ili jednak granici više od dva slijedeća uzastopna lokalna skupa podataka. Slično, ako su točke procesuirane sdesna na lijevo koristili bi test "manji ili jednak".

Nakon toga koristimo prije opisanu strategiju odbacivanja čvorova da bi uklonili nakupine čvorova, te pokrećemo algoritam za adaptirano postavljanje čvorova.

Osim do sada spomenutih metoda koje su temeljene na posebnim algoritmima za naš problem, možemo koristiti i neke uobičajene metode za rješavanje problema najmanjih kvadrata i to ili na način da problem tretiramo direktno, ali pazeći na ograničenja ili da nekom transformacijom problem svedemo na neograničeni. Opisat ćemo algoritam koji koristi generaliziranu Gauss-Newtonovu metodu.

Ova metoda, opisana u [10], može pronaći dobar položaj podskupa ili cijelog skupa unutarnjih čvorova, dok sve ostale metode u procesu optimizacije razmještaju sve unutarnje čvorove. Pošto ne razmatramo problem gdje bi neki čvorovi trebali biti fiksirani opisat ćemo algoritam uz pretpostavku da dopuštamo premještanje svih unutarnjih čvorova.

Neka je $\lambda = (\lambda_1, \ldots, \lambda_g)^\tau$ vektor unutanjih čvorova. Optimizacijski problem koji trebamo riješiti je

$$\min\{ \|y - E(\lambda)c\|^2 \ : \ c \in \mathbb{R}^{g+k+1} \ , \lambda \in \mathbb{R}^g \ \} \tag{28}$$

Pošto zahtjevamo da su čvorovi postavljeni u rastućem poretku imamo dodatno uvjet

$$\lambda_l - \lambda_{l-1} \geq \varepsilon, \quad l = 1, \ldots, g+1 \tag{29}$$

gdje je ε dovoljno mali parametar separacije, $0 < \varepsilon \ll 1$. Matrični zapis dan je s $C_1\lambda \geq h$ gdje je $C_1 \in \mathbb{R}^{g+1 \times g}$, $h \in \mathbb{R}^{g+1}$

$$C_1 = \begin{bmatrix} 1 & & & & & \\ -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 1 & \\ & & & & -1 & \end{bmatrix}, h = \begin{bmatrix} \varepsilon + \lambda_0 \\ \varepsilon \\ \vdots \\ \varepsilon \\ \varepsilon - \lambda_{g+1} \end{bmatrix}.$$

U (29) parametar separacije ε predstavlja donju granicu apsolutne udaljenosti između susjednih čvorova. No, ukoliko je raspored čvorova vrlo različit od ekvidistantnog, tada bi umjesto apsolutne trebali ograničiti relativnu udaljenost, tj.

$$\lambda_l \geq \lambda_{l-1} + \varepsilon(\lambda_{j+1} - \lambda_{j-1}), \lambda_l \leq \lambda_{l+1} - \varepsilon(\lambda_{j+1} - \lambda_{j-1}), \quad l = 1, \ldots, g+1 \tag{30}$$

Ovo ograničenje su uveli de Boor i Rice koji su koristili vrijednost $\varepsilon = 0.0625$. U ovom slučaju matrični zapis je dan s $C_1\lambda \geq h$ gdje je $C_1 \in \mathbb{R}^{2g \times g}$, $h \in \mathbb{R}^{2g}$

$$C_1 = \begin{bmatrix} 1 & -\varepsilon & & & & & \\ -1 & 1-\varepsilon & & & & & \\ \varepsilon - 1 & 1 & -\varepsilon & & & & \\ \varepsilon & -1 & 1-\varepsilon & & & & \\ & \varepsilon - 1 & 1 & -\varepsilon & & & \\ & \varepsilon & -1 & 1-\varepsilon & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \varepsilon - 1 & 1 & -\varepsilon & \\ & & & \varepsilon & -1 & 1-\varepsilon & \\ & & & & \varepsilon - 1 & 1 & \\ & & & & \varepsilon & -1 & \end{bmatrix}, h = \begin{bmatrix} \lambda_0(1-\varepsilon) \\ -\lambda_0\varepsilon \\ 0 \\ \vdots \\ 0 \\ \lambda_{g+1}\varepsilon \\ \lambda_{g+1}(\varepsilon - 1) \end{bmatrix}.$$

Dakle, problem zajedno s ograničenjem glasi

$$\min\{ \|y - E(\lambda)c\|^2 \ : \ C_1\lambda \geq h \ , \lambda \in \mathbb{R}^g \ \ c \in \mathbb{R}^{g+k+1} \ \}. \tag{31}$$

Problem (31) je seprabilni nelinearni problem najmanjih kvadrata, i možemo ga riješiti na slijedeći način: za fiksirani $\lambda \in \mathbb{R}^g$ pronađemo jedinstveno rješenje najmanjih kvadrata c_{opt} dano s

$$c_{opt} = E(\lambda)^+ y,$$

gdje je s E^+ označen generalizirani inverz. Mijenjajući c s "optimalnom" vrijednošću dobivamo reducirani problem

$$\min\{\|(I - E(\lambda)E(\lambda)^+)y\|^2 \ : \ C_1\lambda \geq h, \lambda \in \mathbb{R}^g\} = \min\{\|G(\lambda)\|^2 \ : \ C_1\lambda \geq h, \lambda \in \mathbb{R}^g\}. \quad (32)$$

Uz pretpostavku konstantnog ranga matrice E, funkcija $G(\lambda)$ je diferencijabilna. Ako uz to ima puni rang, što ćemo nadalje pretpostavljati, Jacobijan od G je dan s

$$G'(\lambda)p = \{A(\lambda)[p] + (A(\lambda)[p])^\tau\}y,$$

$$A(\lambda)[p] = -[I - E(\lambda)E(\lambda)^+][E'(\lambda)[p]]E(\lambda)^+, \quad \forall p \in \mathbb{R}^g.$$

Ovaj problem sada možemo riješiti koristeći Gauss-Newtonovu metodu. Pri tome možemo koristiti dva modela

$$\mu_{GP} = \|G + G'p\|^2 \approx \|G(\lambda + p)\|^2$$

koji su uveli Golub i Pereyra ili

$$\mu_K = \|G + Kp\|^2 \approx \|G(\lambda + p)\|^2, \text{ gdje je}$$

$$Kp = A(\lambda)[p]y = -[I - E(\lambda)E(\lambda)^+][E'(\lambda)[p]]E(\lambda)^+ y$$

nazvan po Kaufmanu.

U $j-$tom koraku generalizirane Gauss-Newtonove metode trebamo riješiti potproblem

$$\min\{\|G + Jp\|^2 \ : \ C_1 p \geq h - C_1\lambda^j \ , \ p \in \mathbb{R}^g\},$$

gdje je J ili G' ukoliko koristimo Golub-Pereyra model ili K ako koristimo Kaufmanov model, a λ^j trenutni vektor čvorova. Ukoliko J ima puni rang tada imamo jedinstveno rješenje p^j (vektor smjera silaska), te dobivamo

$$\lambda^{j+1} = \lambda^j + \alpha^j p^j$$

gdje je $\alpha^j \in (0, 1]$ duljina koraka koju trebamo odrediti tako da je zadovoljeno

$$\mu(\lambda^j) \geq \mu(\lambda^{j+1}).$$

Kada koristimo Golub-Pereyra model Jacobijan $G'(\lambda^j)$ aproksimiramo po stupcima na slijedeći način

$$G'(\lambda)e^l \approx \frac{G(\lambda + h_l e^l) - G(\lambda)}{h_l}, \quad l = 1, \ldots, g$$

gdje je $h_l = \varepsilon_1(|\lambda_l| + \varepsilon_2), \quad \varepsilon_1 \approx \sqrt{\text{macheps}}$.

Ukoliko koristimo Kaufmanov model, tada su stupci od K definirani s

$$Ke^l = -[I - E(\lambda)E(\lambda)^+][E'(\lambda)[e^l]]E(\lambda)^+ y,$$

$$(E'(\lambda)[e^l])_{i,j} = \frac{\partial B_{j-k-1,k+1}(x_i, \lambda)}{\partial \lambda_l}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, g + k + 1, \quad l = 1, \ldots, g.$$

Gore navedene derivacije B-splajnova s obzirom na čvorove aproksimiramo s

$$(E'(\lambda)[e^l])_{i,j} \approx \frac{B_{j,k+1}(x_i, \lambda + h_l e^l) - B_{j,k+1}(x_i, \lambda)}{h_l}.$$

Potproblemi koji se javljaju u svakom koraku Gauss-Newtonove metode su linearni problemi najmanjih kvadrata s linearnim ograničenjem, koji se ukoliko J ima puni rang mogu transformirati u problem najmanje udaljenosti (kao što je opisano u [8]). Algoritam za riješavanje problema najmanje udaljenosti preuzet je iz [8].

Cijeli algoritam ima slijedeći oblik:

1. Izaberi početnu poziciju čvorova $\lambda^0 \in \mathbb{R}^g$
   j:=0

2. Ponavljaj

   2.1. $G = G(\lambda^j) = [I - EE^+]y$
   $J \approx G'(\lambda^j)$

   2.2. Izračunaj vektor silaska p^j iz

   $$\min\{\|G + Jp\| \ : C_1 p \geq h - C_1\lambda^j, \ p \in \mathbb{R}^g\}$$

   2.3. Osiguraj smanjenje vrijednosti funkcije
   $\alpha^j = 1$
   Dok je $\mu(\lambda^j) - \mu(\lambda^j + \alpha^j p^j) < \alpha^j \|Jp^j\|_2^2$
   $\alpha^j = \alpha^j/2$

2.4. $\lambda^{j+1} = \lambda^j + \alpha^j p^j$

$j = j + 1$

dok je je $j > jmax$ ili konvergencije

Algoritam zaustavljamo ukoliko je zadovoljen barem jedan od slijedećih uvjeta

$$\begin{gathered} \|G\| \leq \varepsilon_0 \\ \|\lambda^{j+1} - \lambda^j\| \leq \varepsilon_1(\|\lambda^j\| + \varepsilon_2) \\ \|J^\tau G\| \leq \varepsilon_3 \\ |G^\tau J p| \leq \varepsilon_4 \\ j > jmax \end{gathered}$$

2.3.2 Penalizirana procjena

Jedan od načina za rješavanje ograničenja je korištenje penalne funkcije (eng. *penalty function*) $P(\lambda)$. Tada umjesto $\delta(\lambda)$ minimiziramo funkciju

$$\xi(\lambda) = \delta(\lambda) + pP(\lambda) \tag{33}$$

pri čemu $P(\lambda)$ optimalni položaj čvorova λ^* drži podalje od $\partial S_g[a,b]$, a parametar p kontrolira odnos između $\delta(\lambda)$ i $P(\lambda)$.

U [6] uzeto je

$$P(\lambda) = \sum_{i=0}^{g} (\lambda_{i+1} - \lambda_i)^{-1} \tag{34}$$

i

$$p = \varepsilon_1 \frac{\delta(\lambda^0)}{P(\lambda^e)} \tag{35}$$

pri čemu je ε_1 točnost s kojom želimo izračunati $\xi(\lambda^*)$, a

$$P(\lambda^*) = \frac{(g+1)^2}{b-a}$$

je vrijednost P-a za ekvidistantnu distribuciju čvorova λ^e. Izbor (35) implicira dvije bitne stvari:

- ako $\delta(\lambda)$ postiže minimum u točki bez podudarajućih čvorova, vrijednost dodatnog izraza $pP(\lambda)$ u toj točki će biti reda $\varepsilon_1\delta(\lambda^0)$. Dakle, ako je početna pozicija čvorova λ^0 razumna, penalna funkcija će imati mali utjecaj na optimalnu poziciju

- što točnije želimo odrediti rješenje (tj. što je ε_1 manji), utjecaj penalne funkcije će biti manji, te će se čvorovi eventualno moći približiti.

Nadalje, ako minimiziramo (33) krenuvši od lokacije $\lambda^0 \in S_g[a,b]$ možemo biti sigurni da će dobar optimizacijski algoritam ostati unutar simpleksa. Konačno, očito je da uvođenje penalne funkcije nema utjecaj na računanje koeficijenata B-splajna.

Postoje odlični algoritmi za minimizaciju nelinearnih funkcija više varijabli. Naš problem se može svesti na nelinaerni problem najmanjih kvadrata, tj. $\xi(\lambda)$ se iz (33), (34) i (2) može zapisati u obliku $\sum_{r=1}^{m+g+1} \xi_r(\lambda)^2$. Ovaj minimizacijski problem ima posebne karakteristike koje bi posebni algoritam mogao iskoristiti. Naš glavni cilj je zapravo premjestiti čvorove na takav način da se rezultirajuća suma kvadratnog reziduala $\delta(\lambda)$ znatno smanji. Optimalna pozicija kao takva nema nikakvo fizikalno značenje, pa stoga nema smisla pokušavati odrediti poziciju čvorova jako točno ako to ima mali utjecaj na rezultirajuću krivulju. Nadalje ne smijemo zaboraviti da će svako izračunavanje funkcije $\xi(\lambda)$ sadržavati rješenje prezadanog sustava jednadžbi i zato moramo pokušati ograničiti broj iteracija. Naravno da ne možem oočekivati da ćemo uvijek imati dobru početnu poziciju čvorova λ^0. Dakle, odabrani minimizacijski algoritam mora biti dovoljno robusan da bi dao prihvatljiv rezultat u svim situacijama (s dobrim ili lošim λ^0) i mora biti dovoljno jednostavan da bi se lako prilagodio našim posebnim potrebama. Visoka asimptotska brzina konvergencije nam ne mora biti prioritetna uzimajući u obzir to da ćemo biti zadovoljni i s malom točnosti.

Konjugirano gradijentna metoda Fletchera i Reevesa zadovoljava naše zahtjeve. Ona reducira problem na nekoliko jednodimenzionalnih optimizacijskih problema (nazvanih potraga pravcem (eng. *line search*)) i ima teorijsko svojstvo postizanja minimuma u točno g koraka ako je $\xi(\lambda)$ kvadratna funkcija. Ako ovaj uvijet nije ispunjen (kao u našem slučaju) i/ili ako se svaka potraga pravcem provodi samo aproksimativno, proces mora biti ponovno pokrenut nakon g koraka i mora postojati neki kriterij zaustavljanja da bi okončali algoritam. Točnije, počevši s proizvoljnom točkom λ^0 on generira skup aproksimacija λ^j, $j = 1, 2 \ldots$ dok ne pronađe optimalnu poziciju λ^* kroz slijedeću shemu:

- izračuna inicijalni smjer traženja

$$d^0 = -\nabla\xi(\lambda^0)$$

- za $j = 0, \ldots, g-1$

  * minimizira $\quad \theta(\alpha) = \xi(\lambda^j + ad^j)$
    da bi dobili α^* i $\lambda^{j+1} = \lambda^j + \alpha^* d^j$

  (a) promjeniti smjer traženja, tj.

$$d^{j+1} = -\nabla\xi(\lambda^{j+1}) + \frac{\|\nabla\xi(\lambda^{j+1})\|^2}{\|\nabla\xi(\lambda^j)\|^2} d^j$$

- stavi $\lambda^0 = \lambda^g$ i ponovno pokreni računanje

Proces završava i λ^{j+1} se prihvaća kao λ^* čim je

$$\frac{|\xi(\lambda^{j+1} - \xi(\lambda^j)|}{\xi(\lambda^j} < \varepsilon_1 \quad \text{i} \quad \frac{\|\lambda^{j+1} - \lambda^j\|}{\|\lambda^j\|} < \varepsilon_2.$$

Zbog gore navedenih razloga ε_1 i ε_2 ne trebamo uzimati premale (dovoljno je $\varepsilon_1 = \varepsilon_2 = 0.0005$).

Što se tiče računanja gradijenta $\nabla\xi(\lambda)$, lako se pronađe njegov analitički izraz

$$\nabla\xi(\lambda) = \nabla\delta(\lambda) + p\nabla P(\lambda).$$

Komponente $\nabla P(\lambda)$ se izračunaju iz definicije i dobije se

$$\frac{\partial P(\lambda)}{\partial\lambda_l} = (\lambda_{l+1} - \lambda_l)^{-2} - (\lambda_l - \lambda_{l-1})^{-2} \quad l = 1, \ldots, g$$

dok se one od $\delta(\lambda)$ izračunaju i iznose

$$\frac{\partial\delta(\lambda)}{\partial\lambda_l} = -2\sum_{j=1}^{m} w_j^2 (y_j - s(x_j)) \frac{\partial s(x_j)}{\partial\lambda_l}.$$

Koristeći pravila derivacije B- splajn baze dobivamo

$$\frac{\partial B_{i,k+1}(x)}{\partial\lambda_l} = d_{i+1,l}(x) - d_{i,l}(x)$$

gdje je

$$\begin{aligned} d_{j,l}(x) &= 0, \text{ za } j > l \text{ ili } j < l-k \\ &= \Delta_k^{k+1}(\lambda_j, \ldots, \lambda_l, \lambda_l, \ldots, \lambda_{j+k})(t-x)_+^k, \text{ ako je } l-k \le j \le l \end{aligned}$$

iz čega daljnim računom dobivamo

$$\frac{\partial s(x)}{\partial\lambda_l} = \sum_{i=-k}^{g+1} e_i \tilde{B}_{i,k+1}(x)$$

uz

$$\begin{aligned} e_i &= \frac{c_{i-1}-c_i}{\tilde{\lambda}_{i+k+1}-\tilde{\lambda}_i}, \quad i=l-k,\ldots,l \\ &= 0 \\ \tilde{\lambda}_j &= \lambda_j \quad j=-k,\ldots,l \\ &= \lambda_{j-1} \quad j=l+1,\ldots,g+k+2 \end{aligned} \tag{36}$$

i $\tilde{B}_{i,k+1}(x)$ je normalizirani B-splajn definiran s čvorovima $\tilde{\lambda}_j$, $j=-k,\ldots,i+k+1$. Lako se vidi da je

$$\frac{\partial s(x_i)}{\partial \lambda_l}=0, \quad \text{za } x_i \le \lambda_{l-k} \ \text{ ili } x_i \ge \lambda_{l+k},$$

što će nam uvelike skratiti računanje.

Također želimo ograničiti broj iteracija pri minimizaciji $\theta(\alpha)$ pošto svako njegovo izračunavanje zahtjeva pronalaženje splajna najmanjih kvadrata s fiksnim čvorovima (potrebno za računavanje $\delta(\lambda^j+\alpha d^j)$). Zbog toga ćemo biti zadovoljni i s grubom aproksimacijom α^*. Ne želimo trošiti vrijeme na pronalaženje derivacije $\theta'(\alpha)$, osim za $\theta'_0 = \theta'(0)=\nabla\xi(\lambda^j)d^j$ koju već znamo. Teoretski θ'_0 mora biti negativna (ako računanje pokaže da to nije točno ponovno startamo Fletcher-Reevesov algoritam u točki λ^j), pa tražimo pozitivnu vrijednost za α^*. Također imamo gornju granicu $\alpha_{\max}$: minimalna vrijednost α za koju $\lambda^j+\alpha d^j$ ima dvije ili više jednakih komponenti, tj. uz $d^j_0=d^j_{g+1}=0$

$$\alpha_{\max}=\min\left\{\frac{\lambda^j_{l+1}-\lambda^j_l}{d^j_l-d^j_{l+1}} \;\middle|\; l=0,\ldots,g; d^j_l>d^j_{l+1}\right\}.$$

Zapravo, algoritam će tražiti tri vrijednosti $0\le\ \alpha_0<\alpha_1<\alpha_2<\alpha_{\max}$ takve da je $\theta(\alpha_0)>\theta(\alpha_1)$ i $\theta(\alpha_2)\ge\theta(\alpha_1)$ i pronaći aproksimaciju $\bar{\alpha}$ za α^* zahtjevajući da je $I'(\bar{\alpha})=0$, gdje je $I(\alpha)$ interpolacijska funkcija ($I(\alpha_i)=\theta(\alpha_i),\ i=0,1,2$). Prvo ćemo dati shematski opis kompletnog algoritma, a zatim komentirati korake. Zapravo, objasnit ćemo razloge raznih izbora i objasniti kako se posebne osobine $\theta(\alpha)$ mogu uvesti u račun. Treba napomenuti da je većina detalja rezultat testiranja i eksperimentiranja. Algoritam je slijedeći ($\theta(\alpha_i)$ označavamo s θ_i):

- inicijalizacija: stavi

$$\begin{aligned} \text{iter} &= 0; \quad \alpha_0=0 \\ \alpha_1 &= \frac{\alpha_{\max}}{2}\left(1-\frac{\theta_0}{\alpha_{\max}\theta'_0}\right)^{-1} \\ \alpha_2 &= 2\alpha_1 \end{aligned} \tag{37}$$

- dok je $\theta_1 \geq \theta_0$

  - izračunaj
$$\tilde{\alpha}_1 = -\frac{\theta_0' \alpha_1^2}{2(\theta_1 - \theta_0 - \theta_0' \alpha_1)} \tag{38}$$
  - stavi iter $\longleftarrow$ iter + 1; $\alpha_1 \longleftarrow \max\left(\frac{\alpha_1}{10}, \tilde{\alpha_1}\right)$

- ako je iter > 0 tada prihvati $\alpha^* = \alpha_1$, inače

  - dok je $\theta_2 < \theta_1$ stavi
$$\begin{aligned} \alpha_0 &\longleftarrow \alpha_1;\ \alpha_1 \longleftarrow \alpha_2 \\ \alpha_2 &\longleftarrow \min\left(2\alpha_1, \alpha_1 + \frac{\alpha_{\max} - \alpha_1}{2}\right) \end{aligned} \tag{39}$$
  - prihvati $\alpha^* = \tilde{\alpha}$ za koji je $I'(\tilde{\alpha}) = 0$, gdje je $I(\alpha)$ funkcija oblika
$$I(\alpha) = Q(\alpha) + pR(\alpha) \tag{40}$$
    uz
$$Q(\alpha) = a_0 + a_1(\alpha - \alpha_0) + a_2(\alpha - \alpha_0)^2 \tag{41}$$
$$R(\alpha) = b_0 + b_1(\alpha - \alpha_0) + \frac{b_2}{\alpha_{\max} - \alpha} \tag{42}$$
    pri čemu koeficijente a_i i b_i trebamo odrediti tako da vrijedi
    $Q(\alpha_i) = \delta(\lambda^j + \alpha_i d^j),\ R(\alpha_i) = P(\lambda^j + \alpha_i d^j),\ i = 0, 1, 2.$

Početna vrijednost (37) za α_1 dobivena je sa zahtjevom da je $\tilde{R}'(\alpha_1) = 0$, gdje je $\tilde{R}(\alpha)$ racionalna funkcija oblika
$$\tilde{R}(\alpha) = \frac{c_0}{(\alpha + c_1)(\alpha_{\max} - \alpha)} \tag{43}$$
koja zadovoljava $\tilde{R}(0) = \theta_0$ i $\tilde{R}'(0) = \theta_0'$. Nadalje $\lim_{\alpha \to \alpha_{\max}} \tilde{R}(\alpha) = \infty$ kao što je slučaj za $\theta(\alpha)$. Dakle koristimo sve dostupne informacije o $\theta(\alpha)$ u točki $\alpha_0 = 0$. Također treba primjetiti da je $\alpha_1 < \frac{\alpha_{\max}}{2}$ pošto je $\theta_0' < 0$. Ako korištenje ove početne vrijednosti za α_1 na rezultira manjom vrijednosti za θ ($\theta_1 \geq \theta_0$) tražimo bolji $\tilde{\alpha}_1$ unutar $\langle 0, \alpha_1 \rangle$ tražeći da je $Q'(\tilde{\alpha}_1) = 0$ gdje je $Q(\alpha)$ kvadratna funkcija oblika (41) koja zadovoljava $Q(\alpha_0 = 0) = \theta_0,\ Q'(0) = \theta_0'$ i $Q(\alpha_1) = \theta_1$. Ako ova vrijednost nije premala ($\tilde{\alpha}_1 \geq \frac{\alpha_1}{10}$) i ako daje bolju vrijednost za θ ($\theta(\tilde{\alpha}_1) < \theta(0)$) tada je prihvaćamo kao dobru aproksimaciju za α^*. Inače nastavljamo tražiti u smanjenom intervalu $\langle 0, \alpha_1 \rangle$. Potrebno je ograničiti

broj iteracija, te u slučaju ako iter postane prevelik prihvatiti λ^j kao dobru aproksimaciju λ^*.

Ako je naš prvi izbor α_1 uspješan (iter=0) nastavljamo potragu za α_2 u intervalu $\langle \alpha_1, \alpha_{\max} \rangle$ startajući s $\alpha_2 = 2\alpha_1$. Sve dok je $\theta(\alpha_2) < \theta(\alpha_1)$ povečavamo ovu vrijednost i prema tome u (39) mijenjamo druge parametre α_i. Pošto je $\lim_{\alpha \to \alpha_{\max}} \theta(\alpha) = \infty$ na kraju sigurno postižemo vrijednost za koju je $\theta(\alpha_2) \geq \theta(\alpha_1)$.

Što se tiče oblika interpolacijske funkcije $I(\alpha)$ najčešće se koristi kvadratna funkcija ako ne znamo vrijednosti derivacija. Naš izbor (40) uzima u obzir činjenicu da je $\theta(\alpha)$ sastavljema od dvije funkcije $\delta(\lambda^j + \alpha d^j)$ i $P(\lambda^j + \alpha d^j)$ s različitim ponašanjem. Dok se prva može zadovoljavajuće aproksimirati s polinomom, druga ima singularitet za $\alpha = \alpha_{\max}$ što racionalna funkcija (42) opisuje. Eksperimentalno smo vidjeli da je u većini slučajeva predložena shema efikasnija (daje manje vrijednosti za $\theta(\tilde{\alpha})$), nego jednostavna upotreba kvadratne interpolacije ($I(\alpha) = Q(\alpha)$, uz $Q(\alpha_i) = \theta(\alpha_i),\ i = 0, 1, 2$) posebno ako je α_2 blizu $\alpha_{\max}$ (tj. ako optimalni skup čvorova λ^* ima gotovo podudarajuće čvorove). To zapravo nije iznenađujuće pošto na taj način koristimo više podataka o funkciji koju interpoliramo (u tom slučaju šest, dok kod kvadratnog polinoma samo tri).

Početna pozicija čvorova

Uspjeh algoritma najmanjih kvadrata za slobodne čvorove čvrsto će ovisiti o odabiru inicijalne pozicije čvorova λ^0. Zaista, vezano uz nekonveksnost $\delta(\lambda)$ jedino možemo očekivati da pronađemo lokalni minimum u blizini λ^0. Određivanje dobrog inicijalnog skupa čvorova je zato bitan, ali nažalost i netrivijalan zadatak. Ekvidistantna distribucija čvorova, na primjer, rijetko postiže globalni minimum $\delta(\lambda)$. Nadalje, unaprijed ne znamo potreban broj čvorova koji će dati prihvatljivu aproksimaciju najmanjih kvadrata $s_g(x)$. Zato predlažemo shemu za koju se nadamo da će vratiti uspješan splajn najmanjih kvadrata $s_g(x)$, $g = 1, 2, \ldots$ i koja koristi skup čvorova λ^* od $s_g(x)$ da bi pronašla odgovarajuću početnu poziciju λ^0 za određivanje $s_{g+1}(x)$.

- inicijalizacija:
$$\lambda_1^0 = \frac{a+b}{2},\ \lambda^0 = (\lambda_1^0)$$
- za $g = 1, 2 \ldots$

- počevši s $\lambda^0 = (\lambda_1^0, \ldots, \lambda_g^0)$ odredi optimalni položaj $\lambda^* = (\lambda_1^*, \ldots, \lambda_g^*)$ i pripadajući splaj najmanjih kvadrata $s_g(x)$
- izračunaj broj $\tilde{\delta}_i$, $i = 0, 1, \ldots, g$ uz

$$\tilde{\delta}_i = \frac{\sum_{j=q_i+1}^{q_i+m_i} (w_j(y_j - s_g(x_j)))^2}{m - m_i}$$

$\lambda_i^* \leq x_{q_i+1} < x_{q_i+2} < \ldots < x_{q_i+m_i} \leq \lambda_{i+1}^*$ i $\lambda_0^* = a,\ \lambda_{g+1}^* = b.$

- pronađi početnu poziciju čvorova λ^0 za određivanje $s_{g+1}(x)$, tj. neka je $\tilde{\delta}_l = \max\{\tilde{\delta}_i, i = 0, \ldots, g\}$, tada

$$\lambda_i^0 = \begin{cases} \lambda_i^* \ , \ i = 1, \ldots, l, \\ (\lambda_l^* + \lambda_{l+1}^*)/2 \ , \ i = l+1, \\ \lambda_{i-1}^* \ , \ i = l+2, \ldots, g+1. \end{cases}$$

S ovom shemom dodatni čvor za λ^0 postavljen je u sredinu intertvala $[\lambda_l, \lambda_{l+1}]$, tj. u dijelu intervala $[a, b]$ gdje s jedne strane $s_g(x)$ ima veliku težinsku sumu kvadrata reziduala, a s druge strane koncentracija čvorova nije prevelika ($m - m_l$ nije prevelik). Drugi uvijet bi trebao osigurati da imamo početni skup čvorova λ^0 koji nije blizu granice prostora $S_{g+1}[a, b]$.

Optimalni broj čvorova

Ako pretpostavimo da gore navedena shema zaista daje zadovoljavajući niz splajnova najmanjih kvadrata $s_g(x)$, jedino trebamo odlučiti koji je optimalni broj čvorova. Ovaj problem je analogan izboru optimalnog stupnja polinoma najmanjih kvadrata $P_g(x)$. Ako je g jako mali, pripadajuća aproksimacija nije dovoljno točna, a ako je prevelik, tada je rezultirajuća aproksimacija pod prevelikim utjecajem statističkih grešaka u vrijednostima podataka. Uobičajeni način procjene optimalnog stupnja polinoma najmanjih kvadrata je ispitivanje root-mean-square reziduala σ_g koji računamo tako da težinsku sumu kvadratnog rezidula δ_g podijelimo s $m - g - 1$, tj. brojem stupnjeva slobode $m - \dim(\mathcal{P}_g)$. U zadovoljavajućem slučaju σ_g će opadati kako povećavamo g i na kraju se "smiriti" na gotovo konstantnoj vrijednosti. Slično, za naš problem trebamo ispitati brojeve

$$\sigma_g = \frac{\delta_g}{m - \dim(\eta_k)} = \frac{\sum_{j=1}^{m} (w_j(y_j - s_g(x_j)))^2}{m - g - k - 1}$$

i prihvatiti broj čvorova koji približno daje konstantnu vrijednost za σ_g. možemo također primjeniti druge statističke testove, kao na primjer test trenda. U tom slučaju računamo

brojeve

$$T_g = \frac{\sqrt{m-1}\sum_{j=2}^{m} r_j r_{j-1}}{\sum_{j=1}^{m} r_j^2}$$

gdje je r_j $j-$ti netežinski rezidual, te prihvatiti za prvi g za koji je $T_g < 1$. Konačno, pažljivo ispitivanje promjene u zaglađujućoj normi $\tilde{\eta}_g$ (vidi [6]) za s_g kako g raste obično daje jasnu indikaciju o optimalnom broju čvorova.

3 Numerički primjeri

Literatura

[1] C. de Boor, *Practical Guide to Splines*, Springer-Verlag, New York, Heidelberg, Berlin, 1978.

[2] C. de Boor, J. R. Rice, *Least squares cubic spline approximation I : Fixed Knots* , CSD TR 20, Purdue Univ., Lafayette, IN, 1968.

[3] C. de Boor, J. R. Rice, *Least squares cubic spline approximation II : variable Knots* , CSD TR 21, Purdue Univ., IN, 1968.

[4] M. G. Cox, P. M. Harris, H. M. Jones, *Strategies for knot placement in least squares data fitting by splines*, TR DITC 101/87, National Physical Laboratory, 1987.

[5] M. G. Cox, P. M. Harris, H. M. Jones, *A knot placement strategiy for least squares spline fitting based on use of local polynomial approximation*, TR DITC , National Physical Laboratory, 1988.

[6] P. Dierckx, *Curve and Surface Fitting with Splines*, Oxford University Press, Oxford, 1993.

[7] D. L. B. Jupp, *Approximation to data by splines with free knots*, SIAM Journal of Numerical Analysis, 15(1978),328 – 343.

[8] C. L. Lawson, R. J. Hanson, *Solving Least Squares Problems*, SIAM, Philadelphia, 1995.

[9] R. Scitovski, *Numerička matematika*, Elektrotehnički fakultet Osijek, Osijek, 1999.

[10] H. Schwetlick, T. Schütze, *Least squares approximation by splines with free knots*, BIT, 35(1995), 361–384.