(MATLAB) Zašto se koeficijenti formiranog splajna ne podudaraju sa koeficijentima koje vrati csape()?

Pod podacima u okviru p.coefs (gde je p=csape(x,y,'uslov')) nalaze se koeficijenti koji množe $(x-x_i)^k$, tj. naši $[\delta, \gamma, \beta, \alpha]$ sa časa.
Vaš zadatak za domaći je bio da formirate splajn čije koeficijente ćete smestiti u neku matricu dimenzije nx4. To će biti sređen polinom, tako da i treba da se razlikuje, ali vrednosti polyval(vrsta_vase_matrice,neko_x) i ppval(p,neko_x) treba da budu iste.

(Furijeov red) Zašto je u zapisu Furijeovog reda nulti član $\frac{a_o}{2}$ a ne a_0?

Koeficijenti a_k množe $\cos(kx)$ dok koeficijenti b_k množe $\sin(kx)$, tj.
$Q_o(x) = a_0\cos(0\cdot x) + b_0\sin(0\cdot x) + a_1\cos(1\cdot x) + b_1\sin(1\cdot x) + ...$
Pošto je $\sin(0) = 0$ i $\cos(0) = 1$ ovo bi mogli zapisati kao
$Q_o(x) = a_0 + a_1\cos(1\cdot x) + b_1\sin(1\cdot x) + ... = a_0 + \sum_{k=1}^{\infty}(a_k\cos(kx) + b_k\sin(kx))$ pri čemu se svi ovi koeficijenti računaju preko formule $\frac{(f,g_k)}{(g_k,g_k)}$
Videli ste da se svi a_k i b_k, za $k = 1, 2, ...$ računaju preko formula (19) u knjizi (strana 74). U slučaju kada je $k = 0$ imamo: $a_0 = \frac{(f,g_0)}{(g_0,g_0)} = \frac{(f,\cos(0))}{(\cos(0),\cos(0))} = \frac{(f,1)}{(1,1)} = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)dx$.
Da nam se ne bi razlikovala formula po kojoj se dobija a_0 i svi ostali a_k, $k = 1, 2, ...$, kao prvi koeficijent se uzima $\frac{a_0}{2}$ i onda formule (19) važe za sve $k = 0, 1, 2....$ Tj. $a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(0x)dx$, ali koristimo ono što nam treba da bi formula gore bila tačna tj. $\frac{a_0}{2} = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)\cos(0x)dx$ (dvojka iz $\frac{1}{2\pi}$ "pređe" ovde).

(Splajn) Zašto je kod prirodnog splajna $\mu_0 = \nu_0 = \lambda_0 = 0$? (analogno za $\mu_n = \nu_n = \lambda_n = 0$)

Kako je $M_0 = 0$ (to je prva jednačina sistema koji treba da formiramo/rešimo), kada bismo formirali jednačinu $\mu_i M_{i-1} + 2M_i + \nu_i M_{i+1} = \lambda_i$ za $i = 0$ dobili bi:
$\mu_0 M_{-1} + 2M_0 + \nu_0 M_1 = \lambda_0$ (Ova jednačina treba da bude ekvivalentna jednačini $M_0 = 0$).

Za početak, λ_i se računaju preko podeljenih razlika reda 2. Na osnovu veze podeljenih razlika i izvoda, pošto je $M_0 = S''(f, x_0)$, treba i λ_0 da bude 0. Dalje, M_{-1} ne možemo da odredimo, dok M_1 (najverovatnije) će biti različito od 0. Da bi dobili da važi jednakost (šta god bili M_{-1} i M_1) to je moguće samo ako su $\mu_0 = \nu_0 = 0$.

(MATLAB) Zašto u kodovima na netu je uprotreba ugrađene funkcije quad() sa pomoćnim funkcijama?

Na primer:
A(i) = quad(@Af,-pi,pi,[],[],fun,i)/quad(@(x)cos(i*x).*cos(i*x),-pi,pi)
function y = Af(x,f,i)
y=feval(f,x).*cos(i*x);

To je moja stara navika da tako radim. Kao prvi argument quad()-u se moze proslediti funkcija kao string, inline objekat, anonimna funkcija (ono kako ste vi učili na UNM), a takođe i pokazivač na novi function gde ćete sami definisati funkciju (ovako kako ja radim). Za ovako jednostavne podintegralne funkcije nema mnogo smisla ta pomoćna funkcija, tek kod složenijih ima smisla zbog preglednosti. Vi ne morate tako da radite. Gornji deo koda je ekvivalentan:
A(i) = quad(@(x) f(x).*cos(i*x),-pi,pi)/quad(@(x)cos(i*x).*cos(i*x),-pi,pi)

(MATLAB) Čemu služe promenljive `diffs` i `diffs_1` u kodu sa sajta za Hermiteov interpolacioni poliom?

Za formiranje tablice podeljenih razlika, možete u MATLAB-u formirati matricu (kao na UNM što ste radili). Ta matrica bi bila (za zadatak sa časa vežbi):

$$PR = \begin{bmatrix} 5 & -3 & -16 & 15 & -10 & 4 & -1 & 1 \\ 5 & -3 & -1 & 5 & -2 & 2 & 1 & 0 \\ 5 & -4 & 4 & 1 & 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 5 & 10 & 0 & 0 & 0 \\ 1 & 6 & 11 & 15 & 0 & 0 & 0 & 0 \\ 7 & 17 & 26 & 0 & 0 & 0 & 0 & 0 \\ 7 & 17 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Ona sadrži 64 broja (elementa). Skoro pola ove matrice čine nule (jer neka vrednost mora biti upisana u donji trougao) - dakle nepotrebno zauzeće memorije. Dodatno, od svih ovih podataka vama je potrebna samo prva vrsta iz ove matrice za formiranje polinoma, kao i samo prethodna kolona podeljenih razlika za računanje tekuće kolone. Nakon što se izračunaju podeljene razlike reda k, ni jedan podatak (osim prvog) iz kolone koja sadrži podeljene razlike reda $k-1$ vam više nije potreban.
Da bi se smanjilo memorijsko zauzeće ideja je sledeća: koristice se 2 vektora umesto matrice: `diffs_1` da čuva samo elemente iz prve vrste matrice (potrebne za polinom) i vektor `diffs` koji će biti tekući vektor u koji upisujemo izračunate elemente tekuće kolone podeljenih razlika preko vrednosti podeljenih razlika nižeg reda (koje nam više nisu potrebni).
Pre ulaska u dvostruku for petlju, `diffs` je prva kolona matrice PR tj. vrednosti funkcije, a `diffs_1` $= 5$ (prvi element potreban za polinom). Nakon prve iteracije unutrašnje petlje, $i = 1$ izračunava se podeljena razlika $PR(1,2) = f[x_0, x_0] = -3$. Pošto nam element na poziciji $PR(1,1) =$ `diffs`(1) više ne treba, ovu novu vrednost (-3) upisujemo na poziciju `diffs(1)` tj. "preko" one petice. U drugoj iteraciji, računa se $PR(2,2) = f[x_0, x_0] = -3$. Opet isto, element na poziciji $PR(2,1) =$ `diffs`(2) nam više ne treba, pa ovo -3 upisujemo na to mesto itd. U narednoj tabeli je prikazan sadržaj vektora `diffs` nakon svake od iteracija unutrašnje petlje.

$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$
$\begin{pmatrix} 5 \\ 5 \\ 5 \\ 1 \\ 1 \\ 7 \\ 7 \\ 7 \end{pmatrix}$	$\begin{pmatrix} -3 \\ 5 \\ 5 \\ 1 \\ 1 \\ 7 \\ 7 \\ 7 \end{pmatrix}$	$\begin{pmatrix} -3 \\ -3 \\ 5 \\ 1 \\ 1 \\ 7 \\ 7 \\ 7 \end{pmatrix}$	$\begin{pmatrix} -3 \\ -3 \\ -4 \\ 1 \\ 1 \\ 7 \\ 7 \\ 7 \end{pmatrix}$	$\begin{pmatrix} -3 \\ -3 \\ -4 \\ 0 \\ 1 \\ 7 \\ 7 \\ 7 \end{pmatrix}$	$\begin{pmatrix} -3 \\ -3 \\ -4 \\ 0 \\ 6 \\ 7 \\ 7 \\ 7 \end{pmatrix}$	$\begin{pmatrix} -3 \\ -3 \\ -4 \\ 0 \\ 6 \\ 17 \\ 7 \\ 7 \end{pmatrix}$	$\begin{pmatrix} -3 \\ -3 \\ -4 \\ 0 \\ 6 \\ 17 \\ 17 \\ 7 \end{pmatrix}$

Nakon zavrsene unutrašnje petlje, promenljivoj `diffs_1` dodajemo prvi element iz `diffs` (potreban za formiranje polinoma). Sada je `diffs` $= [5, -3]$.
U drugoj iteraciji spoljašnje petlje, za $j = 2$, vektor `diffs` menjamo po istom principu.

(Ravnomerna aproksimacija) Kako se došlo do vrednosti za ϵ_1 u okviru dokaza Čebisevljeve teoreme (strana 100 u knjizi)?

Iz $f(x) - Q(x) - \epsilon P(x) < -L$, ako "izvučemo" ϵ dobijamo $\epsilon < \frac{f(x)-Q(x)+L}{P(x)}$. Da bi izraz sa desne strane bio što je moguće manji, izaberemo $\epsilon_1 = \frac{\min\limits_{x\in[z_0,z_1]} |f(x)-Q(x)+L|}{\max\limits_{x\in[z_0,z_1]} |P(x)|}$ i treba da bude $\epsilon < \epsilon_1$.
Ovo je bilo za prvi interval $[z_0, z_1]$. Po istom principu se odredi ϵ_k za svaki od intervala $[z_{k-1}, z_k]$. I konačno, $\epsilon = \min\{\epsilon_1, ..., \epsilon_m\}$.

(Ravnomerna aproksimacija - Remezov algoritam)

Za razumevanje koraka Remezovog algoritma, na linku

http://enastava.matf.bg.ac.rs/~zdrazic/Remez%20primer.pdf

imate uradjen zadatak 4.47 iz Zbirke koristeći Remezov algoritam. To je isti zadatak kao i sa

http://enastava.matf.bg.ac.rs/~zdrazic/RA%20zadatak.pdf

samo urađen na drugi način da bi razumeli korake 0-4.
Jasno je da na baš ovom primeru Remezov algoritam dosta složeniji od rešenja kao u Zbirci. Međutim, ako funkcija nije konveksna/konkavna, ako vam je potrebno više od 3 tačke,... onda biste morali ovako da radite.

(Množenje dva polinoma korišćenjem DFT)

Detaljan postupak zadataka 4.37 koji se odnosi na množenje dva polinoma i čiji postupak takođe stoji okačen na linku

http://poincare.matf.bg.ac.rs/~zdrazic/NA/DFT.pdf

sada detaljnije imate i na

http://enastava.matf.bg.ac.rs/~zdrazic/DFT%20proizvod%20polinoma.pdf

.