

Numerička analiza

26. predavanje

Autor: Saša Singer

Predavač: Nela Bosner

nela@math.hr

web.math.hr/~nela/nad.html

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Multigrid (višemrežna) metoda za diskretnu Poissonovu jednadžbu:
 - Uvod u Multigrid.
 - Grubi opis Multigrida u 2D.
 - Multigrid V–ciklus (MGV).
 - Puni Multigrid (FMG).
 - Složenost Multigrida.
 - Detaljni opis Multigrida u 1D.
- Usporedba metoda za diskretnu Poissonovu jednadžbu:
 - Tablica složenosti.

Multigrid metoda za diskretnu Poissonovu jednadžbu

Uvod u multigrid

Multigrid (višemrežne) metode izmišljene su za rješavanje

• parcijalnih diferencijalnih jednažbi, poput **Poissonove**, ali se mogu primijeniti i na **širu** klasu problema.

Osnovna prednost pred ostalim **iterativnim** metodama:

• brzina **konvergencije multigrid** metode **ne ovisi** o veličini N problema, tj. o finoći diskretizacije — koraku h .

Drugim riječima, **nema usporenja** za velike probleme!

Posljedica: **multigrid** metoda rješava problem s n nepoznanica

• u $O(n)$ vremena (sekvencijalno), ili

• u **konstantnom** broju **operacija** po nepoznanici.

To je **optimalno**, do na multiplikativnu konstantu.

Problem kod standardnih iterativnih metoda

Što je **problem** kod svih **ostalih iterativnih** algoritama koje smo dosad spominjali? Ukratko:

- **brzina širenja** “**informacije**” u **aproksimacijama** rješenja pripadnog **linearnog sustava** dobivenog **diskretizacijom**.

Ta brzina je **premala!**

Razlog tome je

- način **generiranja** **sljedeće** aproksimacije rješenja $x^{(m+1)}$, koji se svodi na
 - “**lokalno**” **usrednjavanje** vrijednosti iz **prethodne** aproksimacije $x^{(m)}$ i **desne strane** (vanjskog djelovanja).

Ilustrirajmo to na primjeru **Poissonove** jednadžbe u **2D**.

Problem kod standardnih iterativnih metoda

Za korak diskretizacije $h = 1/(N + 1)$, jednačbe **linearnog sustava** imaju oblik

$$4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = h^2 f_{i,j}, \quad 1 \leq i, j \leq N.$$

Svaka jednačba je prirodno “**lokalna**” i veže

- rješenje i vanjsko djelovanje u točki (i, j) ,
- s rješenjem u **najbližim susjednim** točkama mreže.

To vrijedi **bez obzira** na **poredak** jednačbi u sustavu, tj. neovisno o **numeraciji** čvorova.

Problem kod **standardnih** iterativnih metoda je što

- matrica **iteracije** T odražava istu “**lokalnost**”.

Ilustracija problema kod Jacobijeve metode

Na primjer, u **Jacobijevoj** metodi, **iteracije** imaju sljedeći oblik (pisan “prostorno”, bez eksplicitne numeracije čvorova):

$$x_{i,j}^{(m+1)} = \frac{1}{4} \left(x_{i-1,j}^{(m)} + x_{i+1,j}^{(m)} + x_{i,j-1}^{(m)} + x_{i,j+1}^{(m)} + h^2 f_{i,j} \right),$$

za $1 \leq i, j \leq N$ i $m \geq 0$.

Uz oznaku za **desnu stranu** $b_{i,j} := h^2 f_{i,j}$, očito je da se nova aproksimacija $x_{i,j}^{(m+1)}$ dobiva “**lokalno**”,

- **usrednjavanjem** vrijednosti iz **prethodne** aproksimacije $x^{(m)}$ i **desne strane**,
- preko najbližih **susjednih čvorova** mreže, uključujući i **dani** čvor.

Ilustracija problema kod Jacobijeve metode (n.)

Neka je $N = 31$ (neparan) i zamislimo da desna strana b ima

- jedan jedini element različit od nule (na pr. jednak 1),
- i to u centru kvadrata — točki $(i, j) = (16, 16)$.

To odgovara koncentriranom djelovanju u centru.

Broj nepoznanica (red sustava) je $n := N^2$.

Pravo (egzaktno) rješenje $u_{i,j}$ je

- različito od nule u svim unutarnjim čvorovima mreže,
- i pada prema rubovima kvadrata (homogeni rubni uvjet).

Slike su malo dalje.

Ilustracija problema kod Jacobijeve metode (n.)

Za početnu iteraciju uzmimo $x^{(0)} = 0$. Pogledajmo što se događa u Jacobijevoj metodi nakon k iteracija. Zanima nas

- u kojim čvorovima je $x^{(k)}$ različito od nule,
- i kolika je greška obzirom na pravo rješenje.

U iteraciji $x^{(k)}$, zbog usrednjavanja preko najbližih susjeda,

- samo vrijednosti u čvorovima udaljenim za najviše k od centra mogu biti različite od nula.

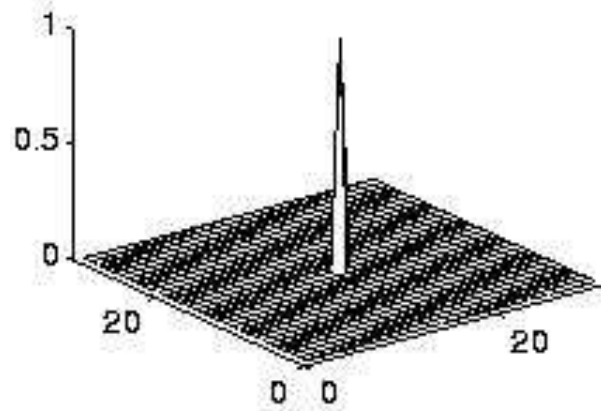
Do ostalih čvorova, “informacija” iz centra još nije stigla,

- jer se “širi” brzinom od jednog čvora po iteraciji.

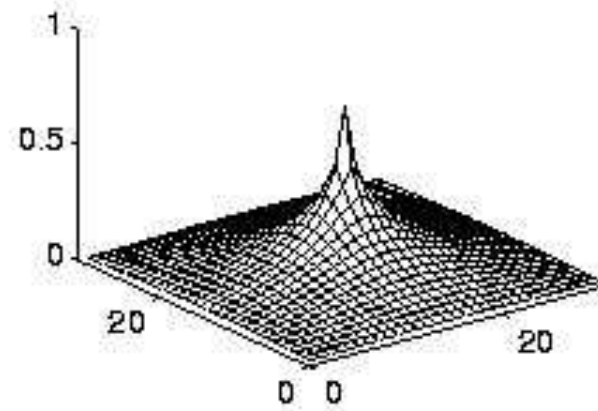
Slike za $k = 5$ su na sljedećoj stranici.

Ilustracija problema na 31×31 mreži

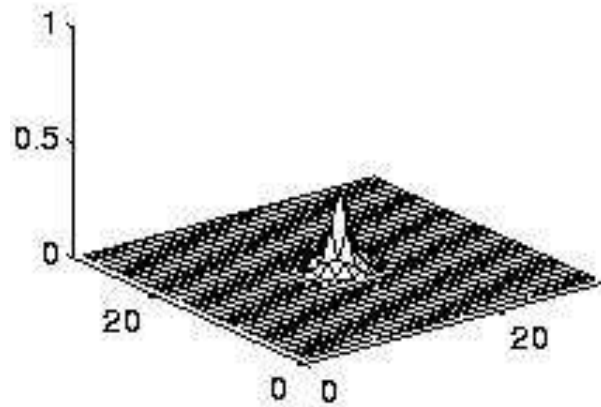
Right Hand Side



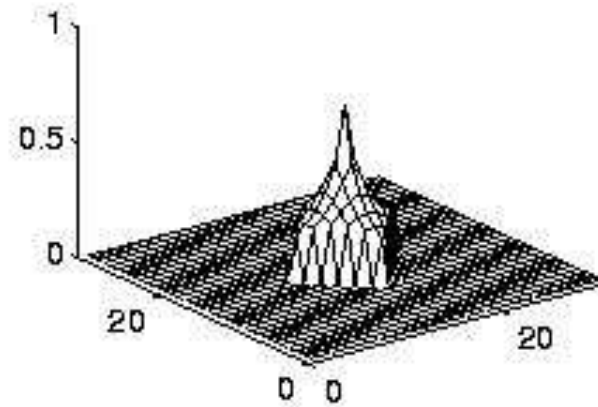
True Solution



5 steps of Jacobi



Best 5 step solution



Lokalnost i brzina konvergencije iteracija

Uz takvo “širenje” informacije, najbolja moguća aproksimacija rješenja nakon k iteracija je

- restrikcija pravog rješenja na odgovarajuću okolinu centra (v. prethodnu sliku),
- jer aproksimacija mora biti nula u svim točkama udaljenim za više od k od centra.

Greška te aproksimacije jednaka je

- pravom rješenju u čvoru udaljenom za $k + 1$ od centra.

Kako pada ta greška s porastom k ? Potrebno je barem

- $O(\log n)$ iteracija, odnosno, $O(n \log n)$ operacija,
- da greška padne za konstantni faktor manji od 1.

Lokalnost i brzina konvergencije iteracija (n.)

Potpuno **isto** ponašanje vrijedi i za sve ostale **standardne** iterativne metode, poput

- **Gauss–Seidelove**, **SOR(ω)** (čak i uz optimalni izbor parametra ω), **Krilovljevih** iteracija (množenje matricom sustava $G_{N \times N}$) i sl.

Dakle, **niti jedna iterativna** metoda bazirana

- **samo** na **lokalnom usrednjavanju**
ne može biti **optimalna**.

Informacije kroz **mrežu** čvorova treba prenesti

- **brže** no što je to **jedna** točka po **iteraciji**.

To je osnovna **motivacija** za **multigrid** metodu.

Osnovne ideje multigrida

U suštini, **multigrid** je **rekurzivni** algoritam, baziran na

- “**divide-and-conquer**” (“podijeli-pa-vladaj”) pristupu,
- koji koristi **više različitih** mreža za generiranje aproksimacije rješenja.

Multigrid spada u **iterativne** metode, ali se **iteracije** odvijaju

- na **raznim** mrežama — od **grubljih**, prema sve **finijima** (a, dijelom, i **obratno!**).

Poanta: Sve **iteracije nisu** na samo **jednoj** mreži, već na **više** njih, pa zato i naziv **multigrid**.

Osnovne ideje multigrida (nastavak)

“Lokalne” iterativne metode koriste se samo za

- (iterativno) poboljšanje rješenja na svakoj pojedinoj mreži.

Usput, to poboljšanje se radi s vrlo preciznim ciljem, a ne bilo kako (detalji malo kasnije).

Međutim, ovdje “lokalnost” nema loših posljedica, jer se

- početno rješenje na svakoj mreži dobiva “globalno”,
- na bazi “divide-and-conquer” — iz dvostruko grublje mreže.

Upravo to osigurava

- brže širenje informacija kroz mrežu (ili mreže) čvorova.

“Podijeli-pa-vladaj” u multigridu

U **Multigridu** se pristup “podijeli-pa-vladaj” koristi na dva povezana načina:

- u prirodnoj **prostornoj** domeni — po **gustoći** mreže,
- u **frekvencijskoj** domeni — **prigušivanjem gornje** polovine, tj. visokih frekvencija **pogreške**.

Za ilustraciju principa, opet gledamo

- **Poissonovu** jednadžbu u **2D** (kao na slikama).

Kako tamo — u **2D**, izgleda

- **prostorna** rekurzija po **gustoći** mreže?

“Podijeli-pa-vladaj” u prostornoj domeni

Početno rješenje na $N \times N$ mreži dobiva se

- korištenjem **grublje** — “polovične” $(N/2) \times (N/2)$ mreže kao aproksimacije,
- uzimanjem svake **druge** mrežne točke iz $N \times N$ mreže, **raspolavljanjem** u **svakoj** dimenziji.

Ova **grublja** $(N/2) \times (N/2)$ mreža opet se aproksimira

- **još** **grubljom** $(N/4) \times (N/4)$ mrežom,
- i tako redom — **rekurzivno**.

Cilj: Očuvanje **globalnosti** u **svim** fazama rješavanja problema,

- tj. — **brzo** širenje informacija.

“Podijeli-pa-vladaj” u frekvencijskoj domeni

Za “podijeli-pa-vladaj” u frekvencijskoj domeni, rješenje problema i, posebno, grešku rješenja treba promatrati

- kao linearnu kombinaciju svojstvenih vektora, ili
- sinusnih funkcija s različitim frekvencijama.

Intuitivno govoreći, posao koji radimo na određenoj mreži

- treba umanjiti (ili eliminirati) grešku u onoj polovini frekvencijskih komponenti,
- čija greška još nije smanjena (ili eliminirana) na grubljim mrežama.

To znači da treba prigušiti grešku

- u gornjoj ili višoj polovini frekvencija,
- jer se one pojavljuju tek na finijoj mreži.

“Podijeli-pa-vladaj” u frekvencijskoj domeni (n.)

Realizacija ovog “poboljšanja” rješenja na pojedinoj mreži je

- usrednjavanje rješenja u svakoj točki mreže, obzirom na njezine susjede,
- varijacijom Jacobijeve iterativne metode.

Taj postupak “izgladuje” rješenje, što je ekvivalentno

- prigušivanju visokih frekvencija (brzih oscilacija) u pogrešci.

Detaljna ilustracija ovog postupka (u 1D) slijedi malo kasnije.

Matlab implementacija Multigrida

Jednostavnu Matlab implementaciju Multigrida u 1D i 2D napravio je Jim Demmel i nalazi se na Web adresi

[http://www.cs.berkeley.edu/~demmel/
ma221_Fall09/Matlab/MG_README.html](http://www.cs.berkeley.edu/~demmel/ma221_Fall09/Matlab/MG_README.html)

Napomena: link iz Demmelove knjige ne radi.

Grubi opis Multigrida u 2D

Mreže za Multigrid u 2D

Počinjemo s opisom algoritma na globalnom nivou, a onda ćemo ga dopuniti potrebnim detaljima.

Krenimo od specifikacije mreža za Multigrid u 2D.

Princip “raspolavljanja” u domeni

- “uzmi svaku drugu točku” (u svakoj dimenziji), zgodnije je interpretirati kao raspolavljanje intervala, tako da
- broj podintervala, a ne točaka, treba biti potencija od 2.

Zbog toga se Multigrid (u 2D) koristiti na mreži koja ima

- $(2^k - 1) \times (2^k - 1)$ nepoznanica — to su nepoznate vrijednosti rješenja u unutrašnjim čvorovima mreže.

Mreže za Multigrid u 2D (nastavak)

Još treba dodati **rubne** čvorove, u kojima su **zadane** vrijednosti rješenja (rubni uvjeti).

U našem **modelnom** problemu, radi jednostavnosti, uzimamo **homogeni** rubni uvjet, tj.

- rubne vrijednosti su 0.

Tako dobivamo **cijelu** $(2^k + 1) \times (2^k + 1)$ mrežu na kojoj radi algoritam. Ova mreža odgovara

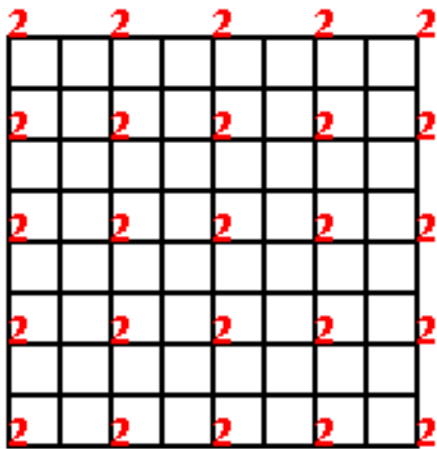
- **podjeli** svake stranice **kvadrata** na 2^k **jednakih** podintervala.

Označimo još $N = N_k := 2^k - 1 =$ broj **nepoznanica** po **stranici** kvadrata, odnosno, **dimenziji** problema.

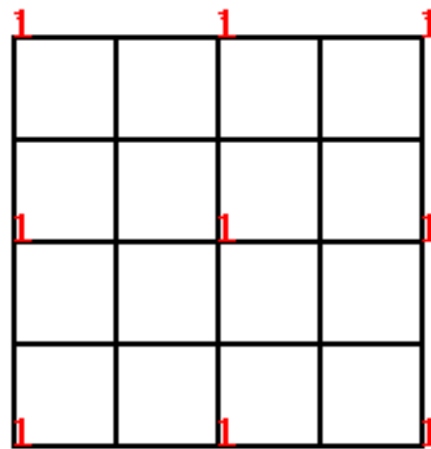
Slika mreža za Multigrid u 2D

Cijele mreže za Multigrid, za $k = 3, 2, 1$, prikazane su na sljedećoj slici.

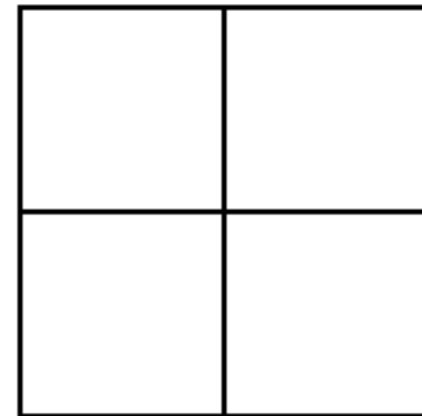
Sequence of Grids Used by Multigrid



P3: 9 by 9 grid of points
7 by 7 grid of unknowns
Points labeled **2** are
part of next coarser grid



P2: 5 by 5 grid of points
3 by 3 grid of unknowns
Points labeled **1** are
part of next coarser grid



P1: 3 by 3 grid of points
1 by 1 grid of unknowns

Oznake za potprobleme u Multigridu

Neka je $P^{(i)}$ problem rješavanja **diskretne Poissonove** jednačbe s (homogenim) rubnim uvjetima na mreži

- od $(2^i + 1) \times (2^i + 1)$ čvorova, s $(2^i - 1)^2$ nepoznanica.

Ekvivalentno, veličina **cijele** mreže je

- $(N_i + 2) \times (N_i + 2)$, a broj nepoznanica je N_i^2 .

Taj problem $P^{(i)}$ je određen, ili **zadan**, sljedećim podacima:

- **veličinom** mreže $N_i = 2^i - 1$ (dovoljno je znati i),
- **matricom** koeficijenata sustava $G^{(i)} := G_{N_i \times N_i}$ i
- **desnom** stranom sustava $b^{(i)}$.

Približno rješenje problema $P^{(i)}$, tj. pripadnog linearnog sustava, označavamo s $x^{(i)}$.

Oznake za potprobleme u Multigridu (nastavak)

Tada su $b^{(i)}$ i $x^{(i)}$ polja (ili vektori)

- veličine, odnosno, duljine $(2^i - 1) \times (2^i - 1)$,
- s vrijednostima u svakoj unutarnjoj točki pripadne mreže.

Rubni uvjeti (ako nisu homogeni) ulaze implicitno u ovaj opis problema, na desnoj strani sustava — ne pamte se posebno!

Ovo su osnovni podaci s kojima radi Multigrid.

Međutim, stvar (osim na trivijalnoj mreži) ne radi tako

- da je $b^{(i)}$ ulaz,
- a izlaz je egzaktno (ili približno) rješenje $x^{(i)}$.

Baš to je ključna ideja Multigrida.

Ključna ideja Multigrida

U trenutku kad “rješavamo” problem $P^{(i)}$,

- ulaz su desna strana $b^{(i)}$ i neko približno rješenje $x^{(i)}$,
- a izlaz je “poboljšano” rješenje $x^{(i)}$!

Dakle, poznato približno rješenje $x^{(i)}$ koristi se

- za dobivanje još boljeg rješenja problema $P^{(i)}$,

koje i opet ne mora biti egzaktno.

Kako to radi? Generira se niz problema $P^{(i-1)}, P^{(i-2)}, \dots, P^{(1)}$ na sve grubljim i grubljim mrežama, i to tako da je

- rješenje problema $P^{(i-1)}$
- dobra aproksimacija za grešku rješenja problema $P^{(i)}$.

Tri operatora u Multigridu

Da bismo objasnili kako algoritam zaista **radi**, potrebno je uvesti **tri** vrste **operatora**

- operator “**poboljšanja**” rješenja S (operator rješenja),
- operator **restrikcije** R , i
- operator **interpolacije** I_n .

Ti operatori

- uzimaju problem $P^{(i)}$ s rješenjem $x^{(i)}$, na **jednoj** mreži,
- i onda ga “**poboljšavaju**” ili **transformiraju** u odgovarajući **problem** na nekoj **drugoj** mreži — **grubljoj** ili **finijoj**.

Sasvim općenito, ovi operatori rade na parovima oblika $(P^{(i)}, x^{(i)})$, sastavljenim od **problema** i njegovog **približnog** rješenja.

Zapis argumenata operatora

Radi jednostavnosti, možemo zamisliti da je **problem** $P^{(i)}$ s **približnim** rješenjem $x^{(i)}$ na **nekoj** mreži

- **potpuno opisan** (zadan) samo vektorima $b^{(i)}$ i $x^{(i)}$,
- jer se “**indeks**” i i **veličina** mreže N_i mogu “**očitati**” iz **duljine** ovih vektora.

Matrica $G^{(i)}$ je, također, potpuno određena s i

- i **konstantna** je za fiksni problem $P^{(i)}$.

Zato uzimamo da operatori rade na

- **parovima** vektora oblika $(b^{(i)}, x^{(i)})$,
- ili samo **jednom** od ta dva vektora, ako je **drugi konstantan** — nema utjecaja na rezultat operatora.

Operator rješenja

Dodatno, radi preglednosti, **izostavljamo indekse** operatora, jer su **očiti** iz indeksa ili **duljine** argumenata.

Operator **rješenja** S

- uzima **problem** $P^{(i)}$ s poznatim **približnim** rješenjem $x^{(i)}$,
- i **izračunava** novo “**poboljšano**” rješenje, u oznaci $x_{\text{imp}}^{(i)}$, za taj **isti** problem $P^{(i)}$.

Zapis djelovanja je

$$x_{\text{imp}}^{(i)} = S(b^{(i)}, x^{(i)}).$$

Poboljšanje se dobiva

- prigušivanjem** komponenti “**visokih frekvencija**” u vektoru **pogreške** rješenja.

Operator rješenja (nastavak)

Operator rješenja S se implementira

- težinskim usrednjavanjem svakog pojedinog čvora s njegovim najbližim susjedima.

Može se interpretirati i kao

- varijacija Jacobijeve iterativne metode za rješavanje linearnog sustava problema $P^{(i)}$ s matricom $G^{(i)}$.

Detaljni opis implementacije u 1D — ide malo kasnije.

To znači da je složenost ovog operatora

- konstantan broj aritmetičkih operacija po nepoznanici,
- ili, $O(n)$, za n nepoznanica.

Operator restrikcije

Operator **restrikcije** R

- uzima **problem** $P^{(i)}$, zadan **desnom** stranom $b^{(i)}$,
- i **preslikava** ga u novi problem $P^{(i-1)}$ na **grubljoj** mreži, s **desnom** stranom $b^{(i-1)}$.

Približno rješenje $x^{(i)}$ **nije** potrebno za R .

Zapis djelovanja je

$$b^{(i-1)} = R(b^{(i)}).$$

Trivijalna **implementacija** ovog operatora

- “**restrikcijom**” — tj. **kopiranjem** odgovarajućih komponenti vektora $b^{(i)}$,

se **ne koristi** u praksi (s dobrim razlozima).

Operator restrikcije (nastavak)

Operator **restrikcije** se **implementira**

- težinskim **usrednjavanjem** svakog pojedinog **čvora** **grublje** mreže
- s njegovim **najbližim** susjedima (na **finijoj** mreži).

Razlog za to je **simetrija** sa standardnim operatorom interpolacije In (ilustracija malo kasnije).

Operator interpolacije

Operator interpolacije In

- uzima približno rješenje $x^{(i-1)}$ problema $P^{(i-1)}$ na grubljoj mreži
- i pretvara ga u približno rješenje $x^{(i)}$ problema $P^{(i)}$ na sljedećoj finijoj mreži.

Desne strane $b^{(i-1)}$ i $b^{(i)}$ ovdje nisu bitne.

Zapis djelovanja je

$$x^{(i)} = In(x^{(i-1)}).$$

Standardna implementacija ovog operatora u praksi je

- obična linearna interpolacija.

Operator interpolacije (nastavak)

Općenito, operator **interpolacije** se, također, **implementira**

- težinskim **usrednjavanjem** svakog pojedinog **čvora** **finije** mreže
- s njegovim **najbližim** susjedima (na **grubljoj** mreži).

Vidimo da se **sva tri** operatora S , R i In **implementiraju**

- nekim oblikom težinskog **usrednjavanja** preko **najbližih susjednih** čvorova odgovarajuće mreže.

Zato **svaki** od ovih operatora zahtijeva

- **konstantan** broj aritmetičkih operacija po **nepoznanici**,
- ili, složenost operatora je $O(n)$, za n nepoznanica.

To je **ključ** za **optimalnu** složenost cijelog **Multigrid** algoritma.

Osnovni algoritam — Multigrid V-ciklus

Ovaj **funkcionalni** opis operatora dovoljan je za formulaciju **osnovnog** algoritma koji se naziva **Multigrid V-ciklus**, ili, skraćeno, **MGV**.

Algoritam MGV. Zapis u notaciji nalik na **Matlab** je

```
function MGV( $b^{(i)}$ ,  $x^{(i)}$ )
```

```
    /* Zamjenjuje približno rješenje  $x^{(i)}$  problema  $P^{(i)}$   
       poboljšanim rješenjem. */
```

```
if  $i = 1$     /* samo jedna nepoznanica */
```

```
    izračunaj egzaktno rješenje  $x^{(1)}$  problema  $P^{(1)}$ 
```

```
    return  $x^{(1)}$ 
```

```
else
```

```
    ...
```

Multigrid V-ciklus (nastavak)

- /* poboljšaj početno rješenje $x^{(i)}$ */
- (1) $x^{(i)} = S(b^{(i)}, x^{(i)})$
- /* izračunaj rezidual tog rješenja */
- (2) $r^{(i)} = G^{(i)} \cdot x^{(i)} - b^{(i)}$
- /* riješi rekurzivno na grubljoj mreži */
- (3) $d^{(i)} = In(MGV(4 \cdot R(r^{(i)}), 0))$
- /* korigiraj rješenje na finoj mreži */
- (4) $x^{(i)} = x^{(i)} - d^{(i)}$
- /* još jednom poboljšaj rješenje */
- (5) $x^{(i)} = S(b^{(i)}, x^{(i)})$
- return** $x^{(i)}$
- endif**

Kratki opis pojedinih koraka MGV-a

Algoritam **počinje** na **finijoj** mreži, s problemom $P^{(i)}$

• i njegovim **približnim** rješenjem $x^{(i)}$.

Odakle se **dobiva** to **početno** rješenje — o tome malo kasnije!

U **netrivijalnom** slučaju $i > 1$, algoritam radi sljedeće korake.

- (1) Prvo **poboljšava** **polazno** rješenje, prigušivanjem visokih frekvencija greške

$$x^{(i)} = S(b^{(i)}, x^{(i)}).$$

- (2) Računa **rezidual** $r^{(i)}$ novog aproksimativnog rješenja $x^{(i)}$.

Ideja: dobiti **približno** rješenje $d^{(i)}$ problema $P^{(i)}$ s **desnom** stranom $r^{(i)}$, tj. jednačbe $G^{(i)} \cdot d^{(i)} = r^{(i)}$.

Onda je $d^{(i)}$ **korekcija** rješenja $x^{(i)}$.

Zašto rezidual približnog rješenja?

Opravdanje. Neka je $d^{(i)}$ **egzaktno** rješenje te jednačbe. Onda je

$$G^{(i)} \cdot d^{(i)} = r^{(i)} = G^{(i)} \cdot x^{(i)} - b^{(i)}.$$

Preuređivanjem dobivamo

$$G^{(i)}(x^{(i)} - d^{(i)}) = b^{(i)},$$

pa je $d^{(i)}$ **korekcija** rješenja, a $x^{(i)} - d^{(i)}$ je **egzaktno** rješenje.

Ako je $d^{(i)}$ **približno** rješenje jednačbe $G^{(i)} \cdot d^{(i)} = r^{(i)}$, onda je $x^{(i)} - d^{(i)}$ **ново** **približno** rješenje **polaznog** problema $P^{(i)}$ s **desnom** stranom $b^{(i)}$.

Dodatno, ako je $x^{(i)}$ **egzaktno** rješenje, onda su **rezidual** $r^{(i)}$ i **korekcija** $d^{(i)}$ jednaki **nula**.

Kratki opis pojedinih koraka MGV-a (nastavak)

Korištenje **reziduala** **izbjegava** restrikciju rješenja $x^{(i)}$.

- (3) Ovaj korak ima nekoliko **podkoraka** — u **istoj** “naredbi”.
- Prvo se aproksimira **rezidual** na sljedećoj **grubljoj** mreži kao **restrikcija** $R(r^{(i)})$ ovog reziduala $r^{(i)}$ na **finijoj** mreži. Taj **grublji** rezidual će biti desna strana za problem na **grubljoj** mreži.
 - Rješava se **grublji** problem (**rekurzivno**), s **nulom** kao **početnom** aproksimacijom rješenja

$$MGV(4 \cdot R(r^{(i)}), 0).$$

Faktor 4 dolazi zbog faktora h^2 na **desnoj** strani **Poissonove** jednačbe. Taj se promijeni za faktor 4, kad s **finije** mreže prelazimo na 2 puta **grublju** mrežu ($h \mapsto 2h$).

Kratki opis pojedinih koraka MGV-a (nastavak)

(3) (kraj)

- Dobiveno rješenje za **korekciju** na **grubljoj** mreži preslikava se **interpolacijom** na **finiju** mrežu

$$d^{(i)} = \text{In}(MGV(4 \cdot R(r^{(i)}), 0)).$$

To je **približna korekcija** rješenja.

(4) Ta **korekcija** $d^{(i)}$ se oduzima od prošlog rješenja $x^{(i)}$, što daje **ново приближно** rješenje (na finijoj mreži)

$$x^{(i)} = x^{(i)} - d^{(i)}.$$

(5) Još jednom se **poboljšava** to rješenje

$$x^{(i)} = S(b^{(i)}, x^{(i)}).$$

Kratki opis pojedinih koraka MGV-a (kraj)

Na kraju, uočimo da se **trivijalni** problem $P^{(1)}$ za $i = 1$

- rješava **egzaktno** (jedna jedina operacija),
- a **početno** rješenje $x^{(1)}$ se **ne koristi!**

Zato se ovo rješenje $x^{(1)}$ može iskoristiti kao

- **početno** rješenje u cijelom algoritmu (tzv. **Full Multigrid**).

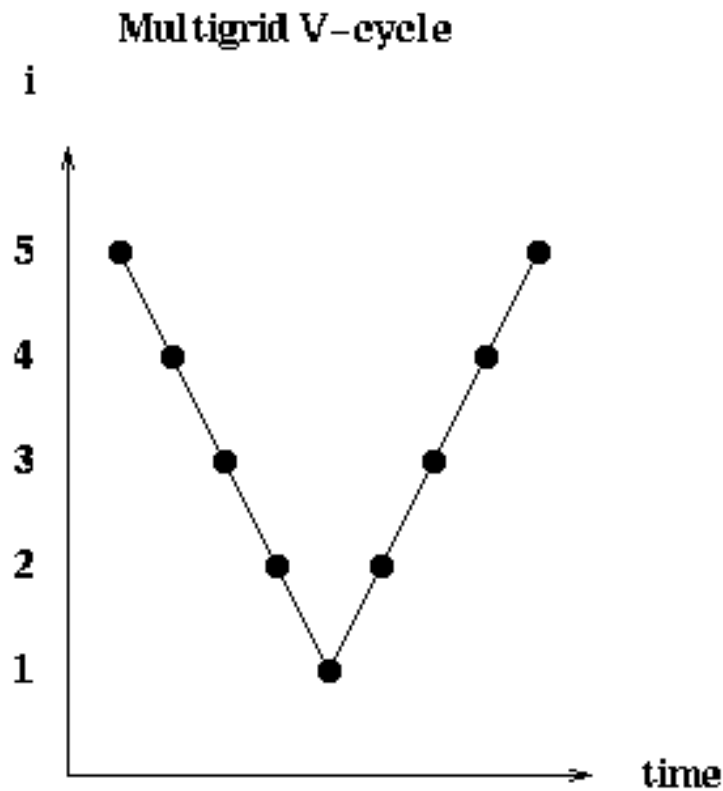
Drugim riječima, **nije** potrebno

- **posebno** tražiti **početno** rješenje $x^{(i)}$.

Detalji slijede nakon analize složenosti **MGV**-a.

Slika V-ciklusa u Multigridu

Zašto se algoritam zove **V-ciklus**? Ako nacrtamo **točke** za **rekurzivne** pozive **MGV**-a u koordinatnom sustavu s osima (**vrijeme**, broj čvora i), dobivamo sljedeću sliku:



Početak za $MGV(b^{(5)}, x^{(5)})$ je u gornjem lijevom kutu.

Algoritam zove **MGV**, redom, na mrežama za $i = 4, 3, 2$ i 1 , a zatim se vraća na $k = 5$.

Analiza složenosti MGV-a

Za grubu analizu složenosti MGV-a dovoljno je znati da svaki od operatora S , R , In

- mijenja vrijednost u pojedinom čvoru mreže
- nekom težinskom sredinom vrijednosti u samom tom čvoru i konstantnom broju njegovih susjeda.

Dakle, za svaki čvor mreže trebamo (najviše)

- konstantan broj računskih operacija,

pa je složenost svakog operatora za n nepoznanica jednaka $O(n)$, tj. linearna u n .

Potpuno isto vrijedi za sve korake (1)–(5) u algoritmu MGV, do na rekurzivni poziv.

Analiza složenosti MGV-a (nastavak)

Rekurzivni pozivi MGV-a opisani su V-ciklusom.

- U svakoj “točki” ● na razini i u V-ciklusu,
- prije i poslije rekurzivnog poziva,

algoritam, također, treba

- $O(N_i^2) = O((2^i - 1)^2) = O(4^i)$ računskih operacija.

Kvadrat dolazi iz dimenzije problema — 2D.

Ako se najfinija mreža nalazi na razini k , s $n = (2^k - 1)^2 \approx 4^k$ nepoznanica, onda ukupna količina posla u MGV algoritmu ima red veličine opisan geometrijskom sumom

$$\sum_{i=1}^k O(4^i) = O(4^k) = O(\text{broj nepoznanica}).$$

Složenost MGV-a — zaključak

Dakle, broj operacija u MGV algoritmu je

• linearan u broju nepoznanica.

U sekvencijalnoj implementaciji algoritma, to vrijedi i za vremensku složenost.

Isti zaključak izlazi i direktno — rekurzijom za složenost.

Neka je $T(i)$ = broj računskih operacija u algoritmu MGV

• na “ulaznoj” mreži s indeksom i ,

tj. na mreži s $N_i^2 = (2^i - 1)^2 = O(4^i)$ nepoznanica. Onda je

$$T(i) = T(i - 1) + O(4^i), \quad i > 1,$$

uz $T(1) = \text{const}$, pa je $T(k) = O(4^k)$.

Cijeli algoritam — ideja

Problem. Osnovni algoritam **MGV** na **netrivijalnoj** mreži, u startu treba

- neko **početno** rješenje $x^{(k)}$, kojeg onda **poboljšava**.

Zato je **ideja**:

- **startati** s trivijalnom **najgrubljom** mrežom za $i = 1$,
- na kojoj **MGV** računa **egzaktno** rješenje (i **ništa** mu ne treba za start),

a onda se polako “**penjati**”,

- “**V-ciklus**, po **V-ciklus**” — za po **jednu** mrežu **finije**,
- do one **najfinije** koja nam treba.

Cijeli algoritam — ideja (nastavak)

Prijelaz iz jednog V -ciklusa u “za jedan viši” V -ciklus

- ide jednostavno — interpolacijom,
- koja daje početnu aproksimaciju na finijoj mreži, tj. baš ono što nedostaje u startu za sljedeći MGV .

Cijeli algoritam se zove Puni Multigrid (engl. Full Multigrid), ili, skraćeno, FMG .

- On koristi MGV kao građevni blok, kojeg iterira.

Cijeli algoritam — Puni Multigrid

Algoritam FMG. Zapis u notaciji nalik na Matlab je

```
function  $FMG(b^{(k)}, x^{(k)})$ 
```

```
    /* Vraća približno, ali vrlo točno rješenje  $x^{(k)}$   
       problema  $P^{(k)}$ . */
```

```
    riješi problem  $P^{(1)}$  egzaktno da dobiješ prvo rješenje  $x^{(1)}$ 
```

```
    for  $i = 2$  to  $k$ 
```

```
         $x^{(i)} = MGV(b^{(i)}, In(x^{(i-1)}))$ 
```

```
    end for
```

```
    return  $x^{(k)}$ 
```


Kratki opis pojedinih koraka FMG-a

Drugim riječima, algoritam FMG radi sljedeće.

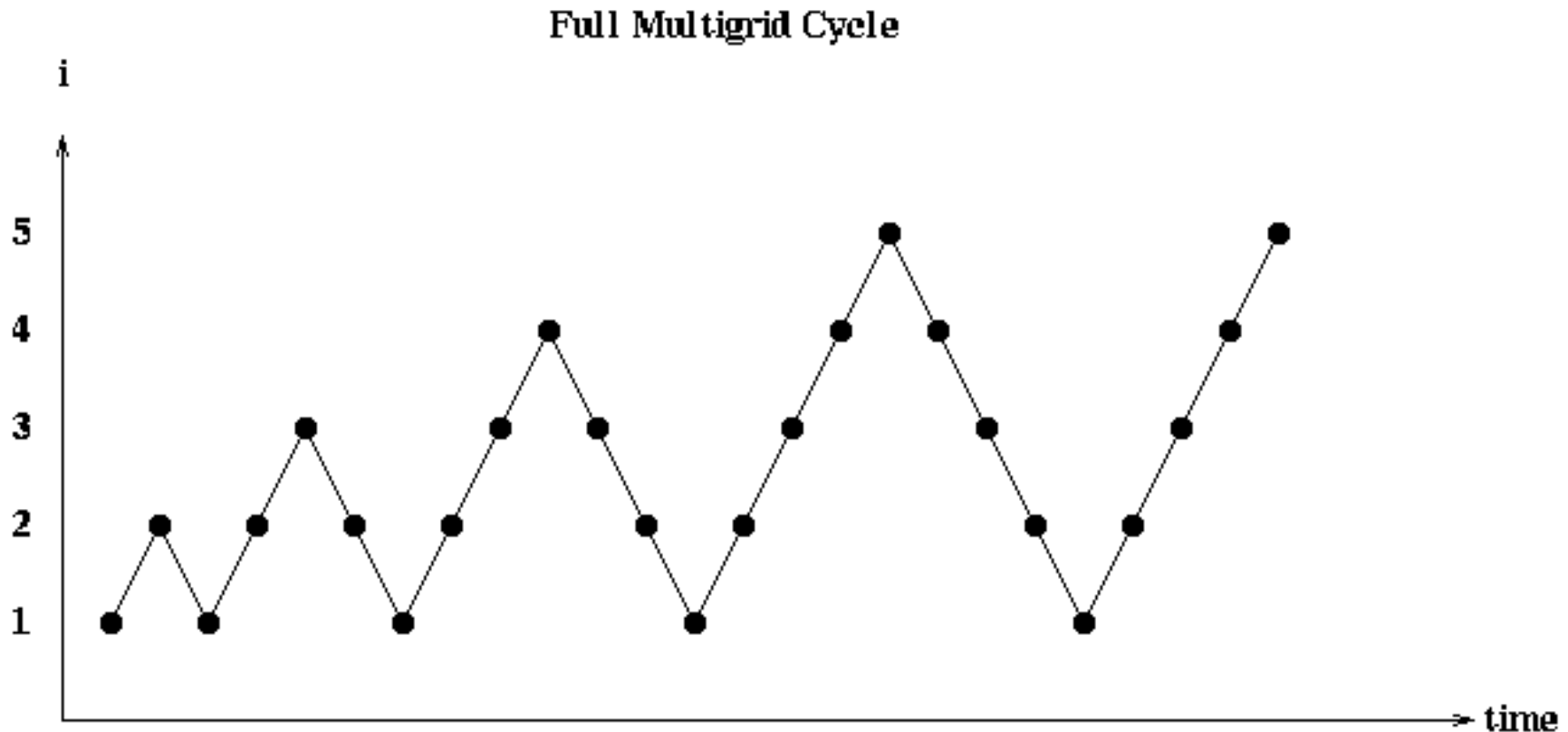
- Rješava problem $P^{(1)}$ egzaktno.
- Dobiveno (približno) rješenje $x^{(i-1)}$ grubljeg problema $P^{(i-1)}$ preslikava interpolacijom u početnu aproksimaciju $x^{(i)}$ sljedećeg finijeg problema $P^{(i)}$

$$In(x^{(i-1)}).$$

- Rješava finiji problem $P^{(i)}$ korištenjem MGV-a s tom početnom aproksimacijom

$$MGV(b^{(i)}, In(x^{(i-1)})).$$

Slika cijelog Multigrida



Napomena. Postoje i drugačije (malo složenije) realizacije cijelog algoritma. Ovo je najjednostavnija.

Analiza složenosti FMG-a

Složenost cijelog algoritma FMG izlazi trivijalno.

Svaki V-ciklus na prethodnoj slici predstavlja

- jedan poziv MGV-a unutar petlje FMG-a, s tim da
- V-ciklus koji počinje (i završava) na nivou i treba $T(i) = O(4^i)$ operacija.

Složenost FMG-a je zbroj složenosti svih poziva MGV-a.

Ukupan broj operacija je

$$\sum_{i=1}^k T(i) = \sum_{i=1}^k O(4^i) = O(4^k) = O(\text{broj nepoznanica}).$$

Konstanta proporcionalnosti, “skrivena” u oznaci O , je malo veća (ali ne puno) od one za osnovni algoritam.

Složenost FMG-a — zaključak

Broj operacija u cijelom FMG algoritmu je

- linearan u broju nepoznanica n ,

što je optimalno, do na multiplikativnu konstantu, jer

- zahtijeva konstantan broj operacija po svakoj nepoznanici.

Isti zaključak vrijedi i za vremensku složenost sekvencijalne implementacije algoritma.

Detaljni opis Multigrida u 1D

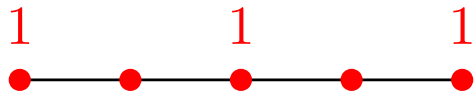
Detaljni opis Multigrid metode u 1D

Pojednostavljenje: objašnjenje multigrid algoritma na 1D problemu.

U 1D, problem $P^{(i)}$ ima $2^i + 1$ čvorova s $2^i - 1$ nepoznanica.



$P^{(3)}$: 9 čvorova, 7 nepoznanica
grublja mreža – indeksi 2



$P^{(2)}$: 5 čvorova, 3 nepoznanice
grublja mreža – indeksi 1



$P^{(1)}$: 3 čvora, 1 nepoznanica

Matrica problema

Neka je:

- $G^{(i)}$ tridijagonalna matrica koeficijenata problema $P^{(i)}$, reda $2^i - 1$;

$$G^{(i)} = \begin{bmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -1 & 2 & -1 & & \\ & & & -1 & 2 & & \\ & & & & & -1 & 2 \end{bmatrix}.$$

Svojstvene vrijednosti vektori matrice problema

Promotrimo vektor rješenja i greške u rješenju kao linearne kombinacije svojstvenih vektora $z^{(j)}$ matrice $G^{(i)}$.

Sjetimo se da je za $G^{(i)}$

- red matrice $N = 2^i - 1$,
- njezine svojstvene vrijednosti su

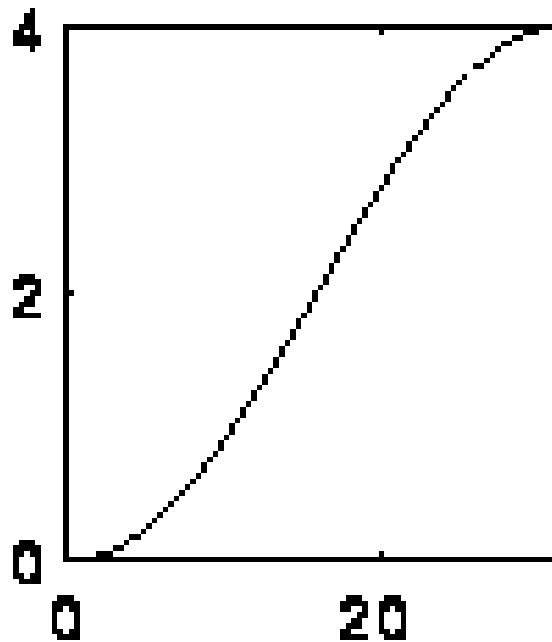
$$\lambda_j = 2 \left(1 - \cos \frac{\pi j}{N + 1} \right),$$

- a komponente svojstvenog vektora $z^{(j)}$ su

$$z_k^{(j)} = \frac{2}{N + 1} \sin \frac{jk\pi}{N + 1}.$$

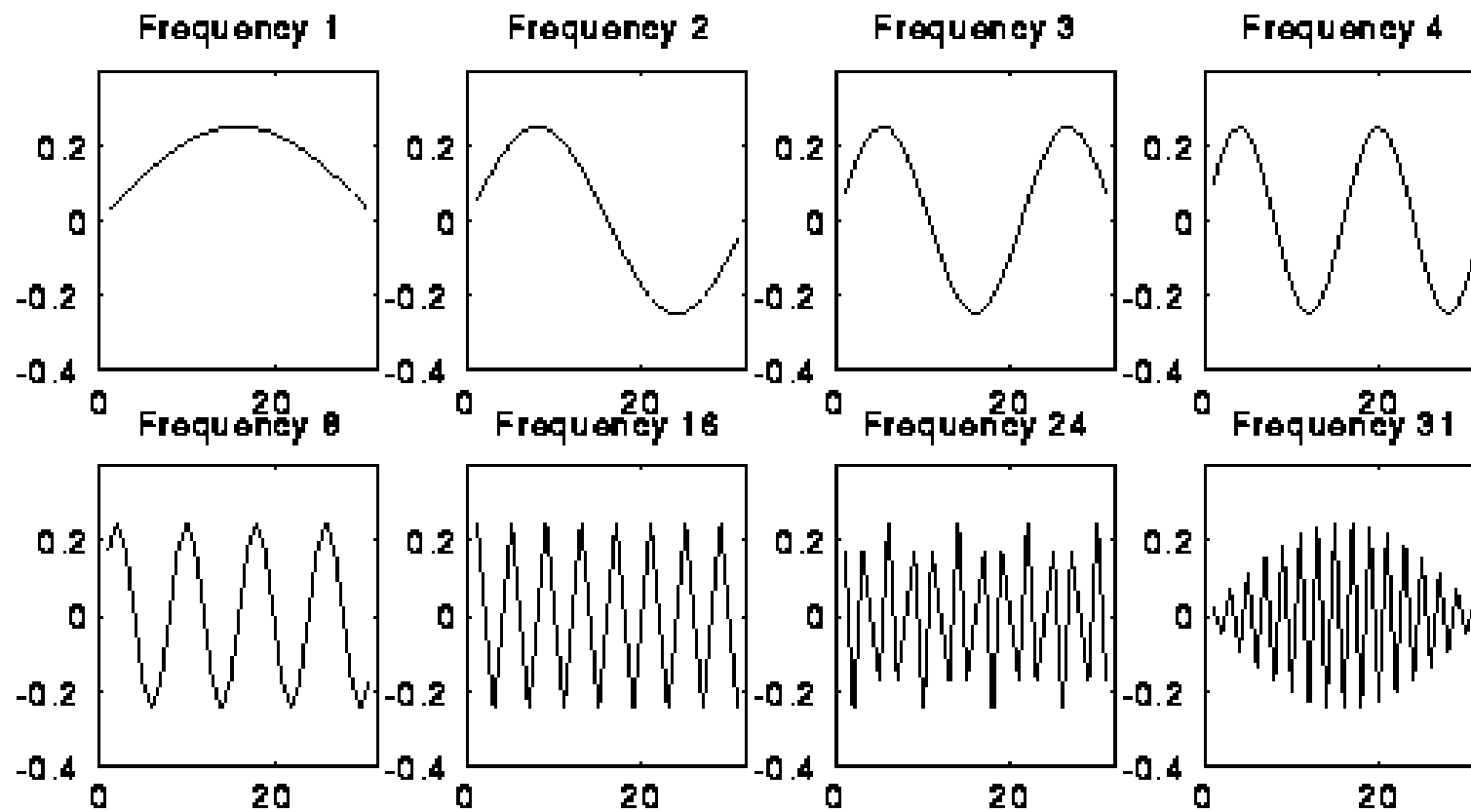
Primjer svojstvenih vrijednosti i vektora

Za $i = 5$, tj. za matricu $G^{(5)}$ je red matrice $N = 2^5 - 1 = 31$.
Sljedeći graf predstavlja **svojstvene vrijednosti** (frekvencije) λ_j
matrice $G^{(5)}$ u rastućem poretku, $j = 1, \dots, N$.



Primjer svojstvenih vrijednosti i vektora (nast.)

Komponente $z_k^{(j)}$ nekih svojstvenih vektora (sinusne krivulje) za pripadne svojstvene vrijednosti λ_j (frekvencije).



Svojstva operatora rješenja

Neka je Z matrica svojstvenih vektora

$$Z = [z^{(1)}, z^{(2)}, \dots, z^{(N)}].$$

Pokazat ćemo kako operator rješenja “ruši” ili “prigušuje” najgornjih pola frekvencija greške. Multigrid metodu možemo opisati i ovako:

- Na najfinijoj mreži $P^{(m)}$ prigušuje se gornja polovina frekvencijskih komponenti pogreške.
- To se obavlja korištenjem operatora rješenja $S^{(i)}$.
- Na sljedećoj grubljoj mreži multigrid prigušuje polovinu od preostale donje polovine frekvencijskih komponentata u grešci, sve dok ne dođemo do egzaktnog rješenja za problem $P^{(1)}$.

Operator rješenja S

Operator rješenja $S(i)$ je $\text{JOR}(\omega)$ iterativna metoda.

Standardna Jacobijeva metoda za rješavanje sustava $Gx = b$ (indekse (i) izostavljamo zbog jednostavnosti) je

$$x^{(m+1)} = T_J x^{(m)} + c_J, \quad c_J = D^{-1}b = \frac{1}{2}b$$

i

$$T_J = -D^{-1}(L + U) = \frac{1}{2} \begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 1 & 0 \end{bmatrix} = I - \frac{1}{2}G.$$

Operator rješenja S (nastavak)

Standardna Jacobijeva metoda j -tu komponentu približnog rješenja $x^{(i)}$ zamjenjuje sa:

$$\begin{aligned}(x_{\text{imp}}^{(i)})_j &= 0.5(x_{j-1}^{(i)} + x_{j+1}^{(i)} + 4^i \cdot b_j) \\ &= x_j^{(i)} + 0.5(x_{j-1}^{(i)} - 2x_j^{(i)} + x_{j+1}^{(i)} + 4^i \cdot b_j).\end{aligned}$$

JOR(ω) metoda, umjesto matrice T_J , uzima težinsku matricu $T_{JOR(\omega)}$, a umjesto c_J , uzima $c_{JOR(\omega)}$,

$$T_{JOR(\omega)} = (1 - \omega)I - \omega T_J = I - \frac{\omega}{2}G, \quad c_{JOR(\omega)} = \frac{\omega}{2}b.$$

Time smo dobili JOR(ω) metodu koja glasi

$$(x_{\text{imp}}^{(i)})_j = x_j^{(i)} + 0.5\omega(x_{j-1}^{(i)} - 2x_j^{(i)} + x_{j+1}^{(i)} + 4^i \cdot b_j).$$

Operator rješenja S (nastavak)

Za težinu $\omega = 1$ dobivamo **običnu** Jacobijevu metodu.

Neka je svojstvena dekompozicija matrice G

$$G = Z\Lambda Z^T.$$

Zato što su i T_J i $T_{JOR(\omega)}$ **matrični polinomi** u G , njihova je svojstvena dekompozicija

$$T_J = Z \left(I - \frac{1}{2}\Lambda \right) Z^T, \quad T_{JOR(\omega)} = Z \left(I - \frac{\omega}{2}\Lambda \right) Z^T.$$

Prisjetimo se, da bi iterativna metoda **konvergirala** mora biti

$$\text{spr}(T) < 1.$$

Greške u rješenju

Neka je $e^{(m)}$ greška m -te iteracije $JOR(\omega)$ metode,

$$e^{(m)} = x^{(m)} - x.$$

Onda imamo

$$\begin{aligned} e^{(m)} &= (T_{JOR(\omega)}x^{(m-1)} + c_{JOR(\omega)}) - (T_{JOR(\omega)}x + c_{JOR(\omega)}) \\ &= T_{JOR(\omega)}(x^{(m-1)} - x) = T_{JOR(\omega)}e^{(m-1)} = T_{JOR(\omega)}^m e^{(0)} \\ &= \left(Z \left(I - \frac{\omega}{2} \Lambda \right) Z^T \right)^m e^{(0)} = Z \left(I - \frac{\omega}{2} \Lambda \right)^m Z^T e^{(0)}, \end{aligned}$$

pa zbog ortogonalnosti matrice Z vrijedi

$$Z^T e^{(m)} = \left(I - \frac{\omega}{2} \Lambda \right)^m Z^T e^{(0)}.$$

Greške u rješenju (nastavak)

U prethodnom vektoru, j -ta komponenta je

$$(Z^T e^{(m)})_j = \left(I - \frac{\omega}{2} \Lambda \right)_{jj}^m (Z^T e^{(0)})_j.$$

Ovu komponentu $(Z^T e^{(m)})_j$ zovemo j -ta **frekvencijska komponenta greške** $e^{(m)}$, budući da je

$$e^{(m)} = Z(Z^T e^{(m)}).$$

To je **težinska suma** stupaca matrice Z s težinom $(Z^T e^{(m)})_j$

$$e^{(m)} = \sum_{j=1}^N (Z^T e^{(m)})_j z^{(j)},$$

što je zapis greške $e^{(m)}$ u **bazi** svojstvenih vektora matrice Z .

Greške u rješenju (nastavak)

Znamo da su stupci od Z **sinusoide** raznih frekvencija. Onda svojstvene vrijednosti matrice $T_{JOR(\omega)}$, a to su

$$\lambda_j(T_{JOR(\omega)}) = 1 - \frac{\omega}{2} \lambda_j,$$

određuju kojom će se **brzinom** svaka od frekvencijskih komponenti $(Z^T e^{(m)})_j$ **prigušivati**, kako m raste.

Ako uzmemo $\omega = 2/3$, onda vrijedi

$$|\lambda_j(T_{JOR(\omega)})| = \left| 1 - \frac{1}{3} \lambda_j \right|,$$

a iz $\lambda_j = 2 \left(1 - \cos \frac{\pi j}{N+1} \right)$, za $j > N/2$ izlazi da je $2 < \lambda_j < 4$.

Greške u rješenju (nastavak)

Zaključak: za $j > N/2$ vrijedi

$$|\lambda_j(T_{JOR(\omega)})| < \frac{1}{3}.$$

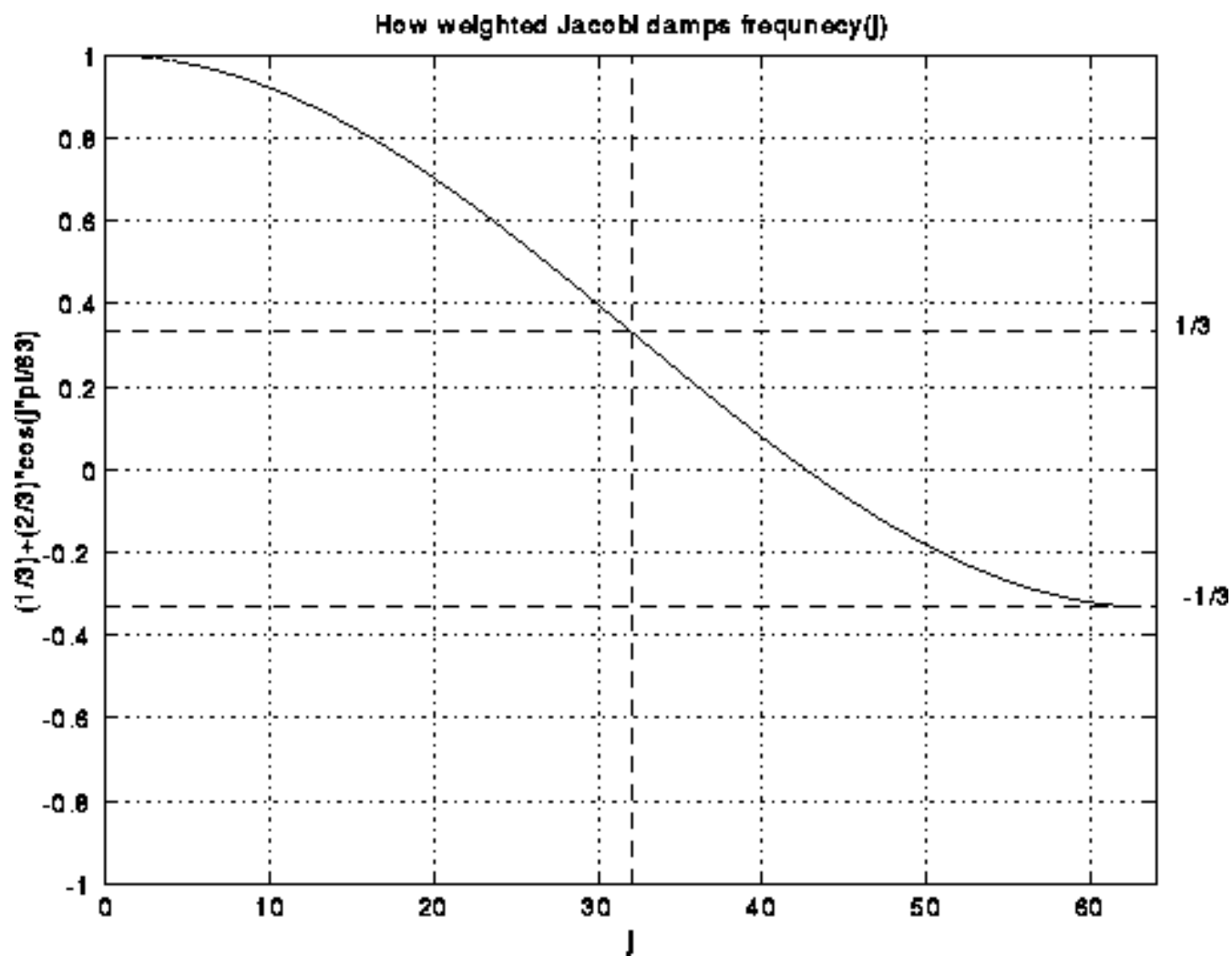
To znači da se u JOR(ω) metodi

- gornja polovina frekvencijskih komponentata $(Z^T e^{(m)})_j$ greške $e^{(m)}$
- množi s $1/3$ ili manje, u **svakoj** iteraciji, bez obzira na N .

Taj ω je **dobar izbor**, a pripadna JOR(ω) metoda za $S^{(i)}$ glasi

$$(x_{\text{imp}}^{(i)})_j = \frac{1}{3}(x_{j-1}^{(i)} + x_j^{(i)} + x_{j+1}^{(i)} + 4^i b_j).$$

Primjer – gušenje frekvencija $JOR(\omega)$

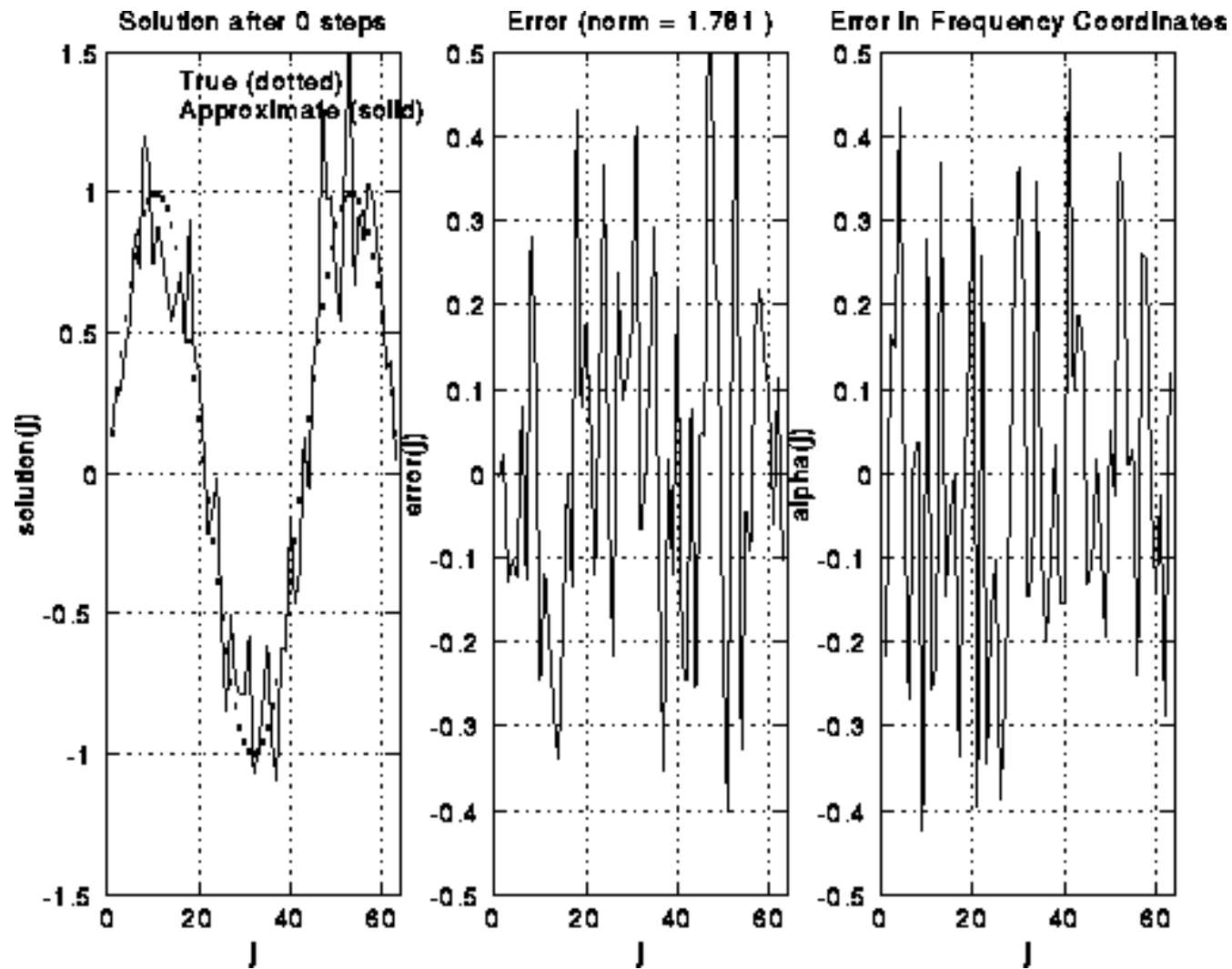


Primjer

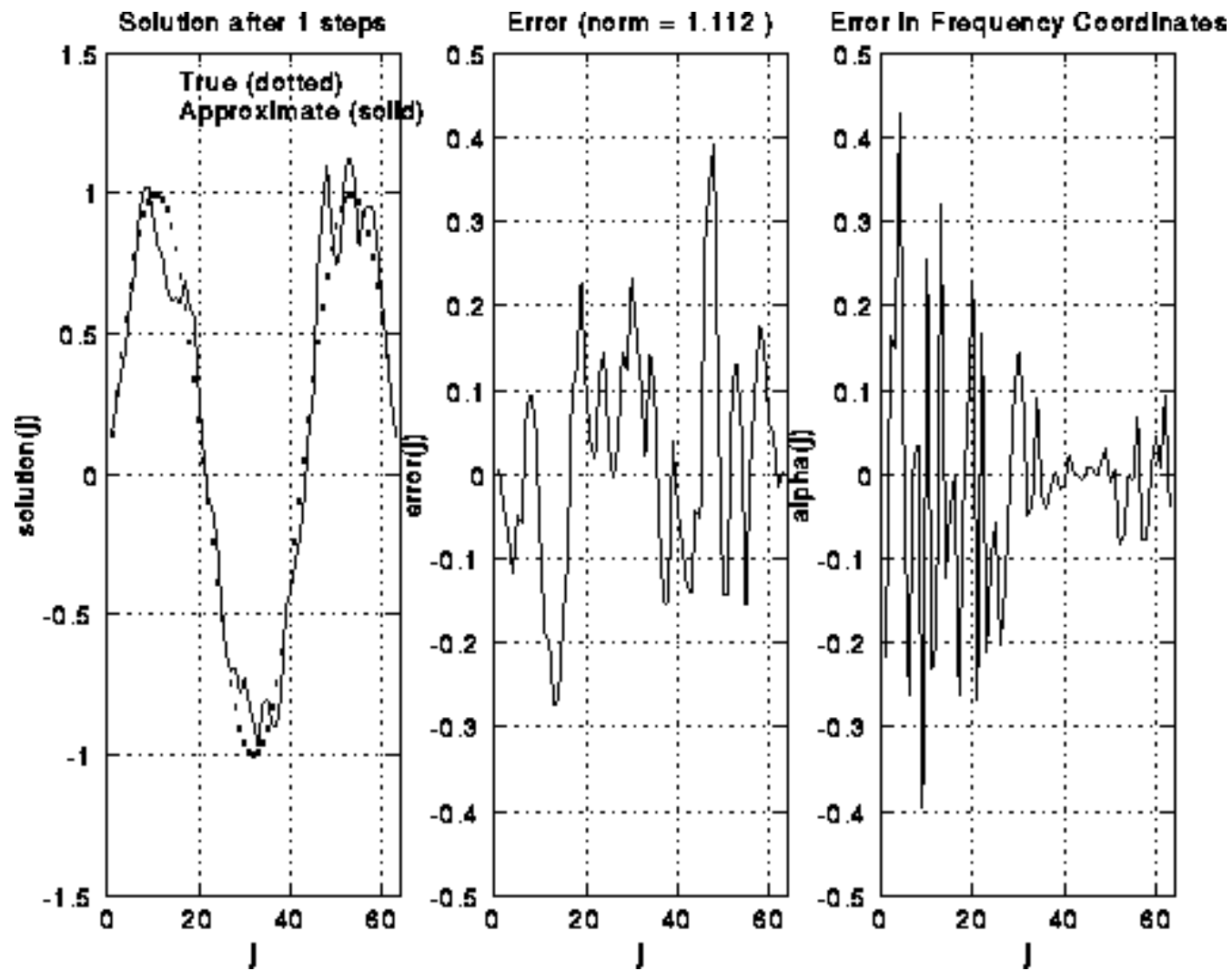
Primjer za $S^{(i)}$, ako je $i = 6$, a $N = 2^6 - 1 = 63$. Opis:

- U prvom redu je početno rješenje i greška tog rješenja.
- Preostalih 5 redova pokazuju rješenja i greške nakon uzastopnih primjena $S^{(i)}$.
- Pravo rješenje je sinusoida nacrtana točkasto na najljevijoj slici u svakom retku.
- Aproksimativno rješenje je prikazano na istoj slici, ali punom linijom.
- Srednja slika u retku prikazuje grešku i njezinu normu (u naslovu slike).
- Najdesnija slika pokazuje frekvencijske komponente greške $e^{(m)}$, tj. komponente vektora $Z^T e^{(m)}$.

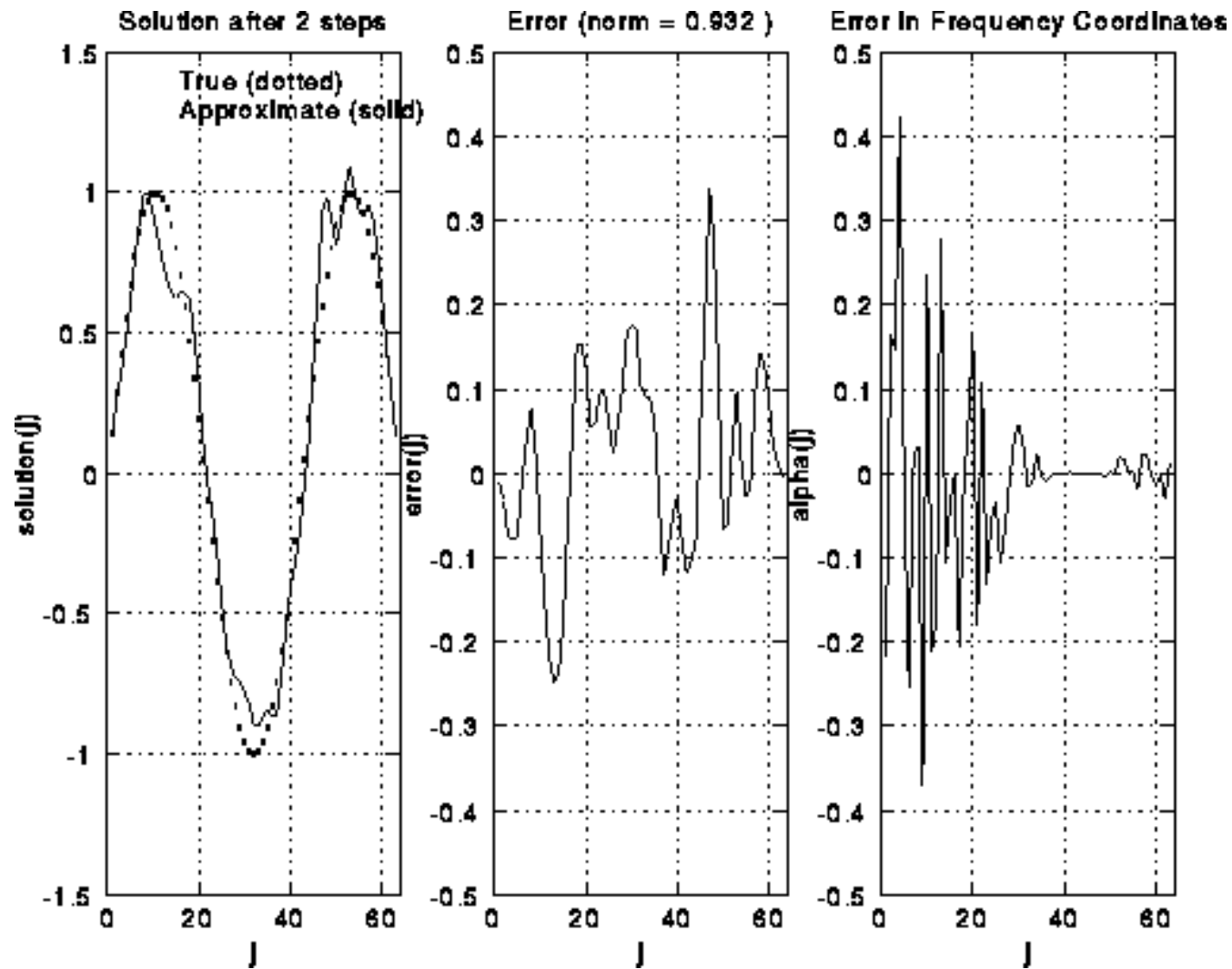
Primjer — 1. red



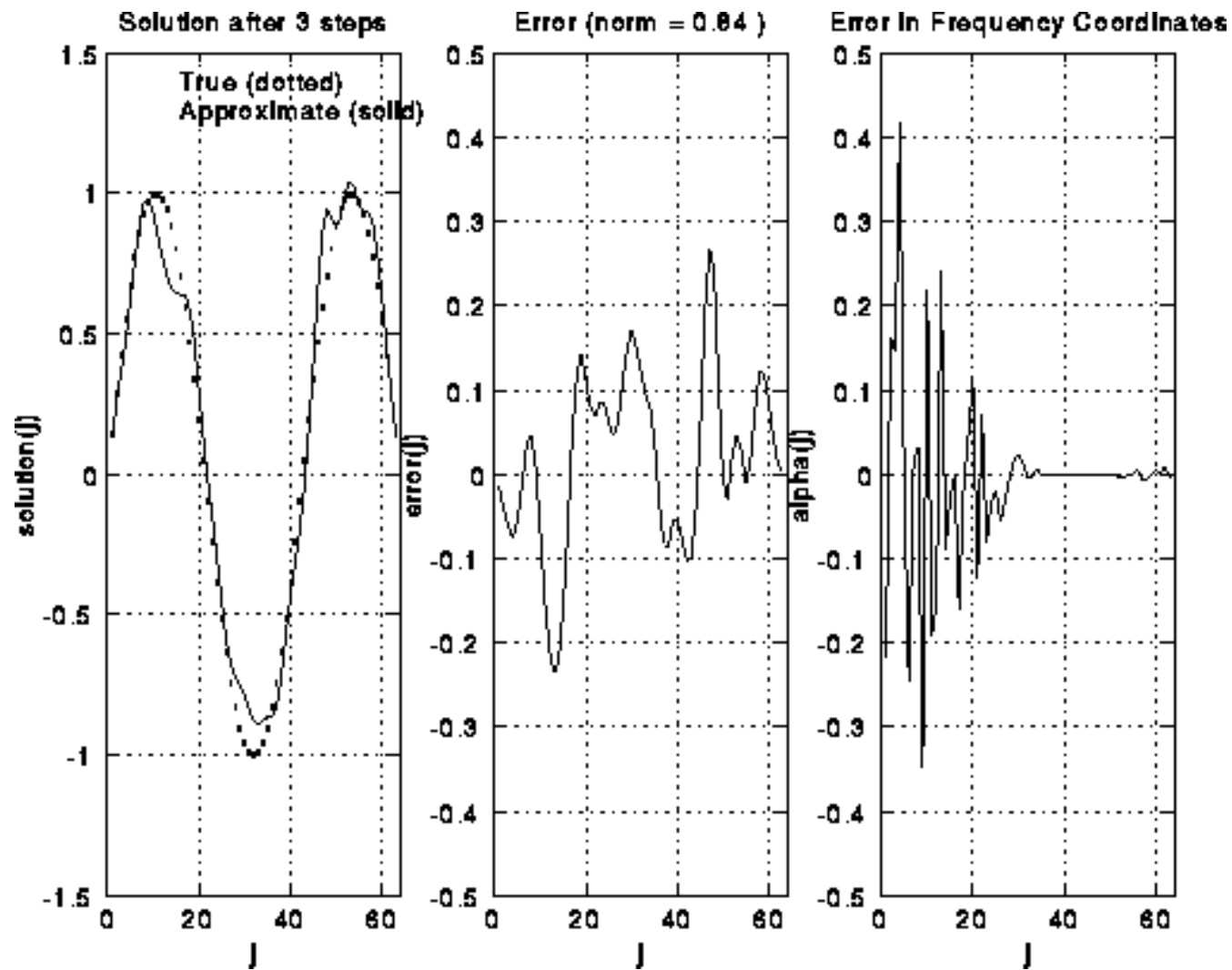
Primjer — 2. red



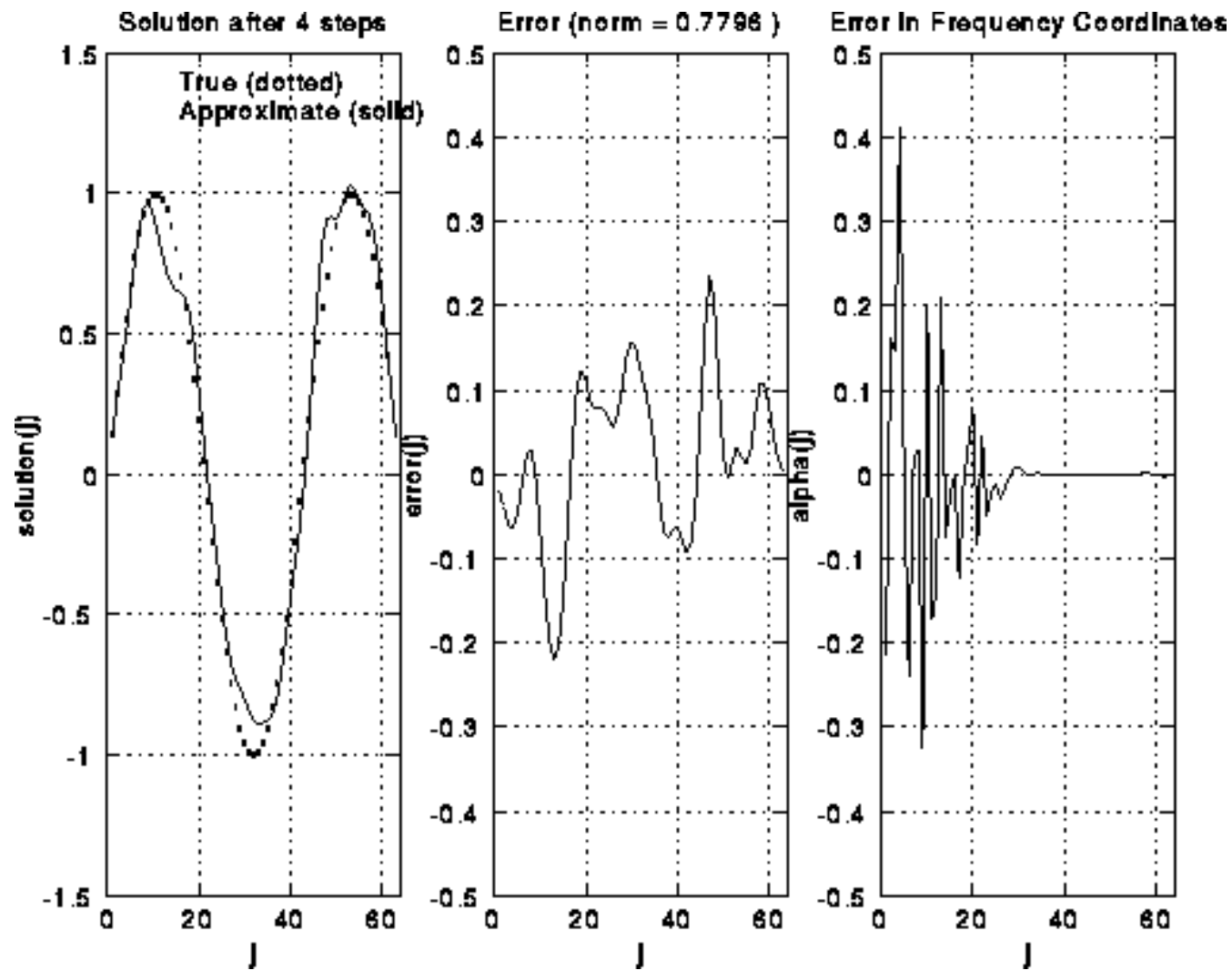
Primjer — 3. red



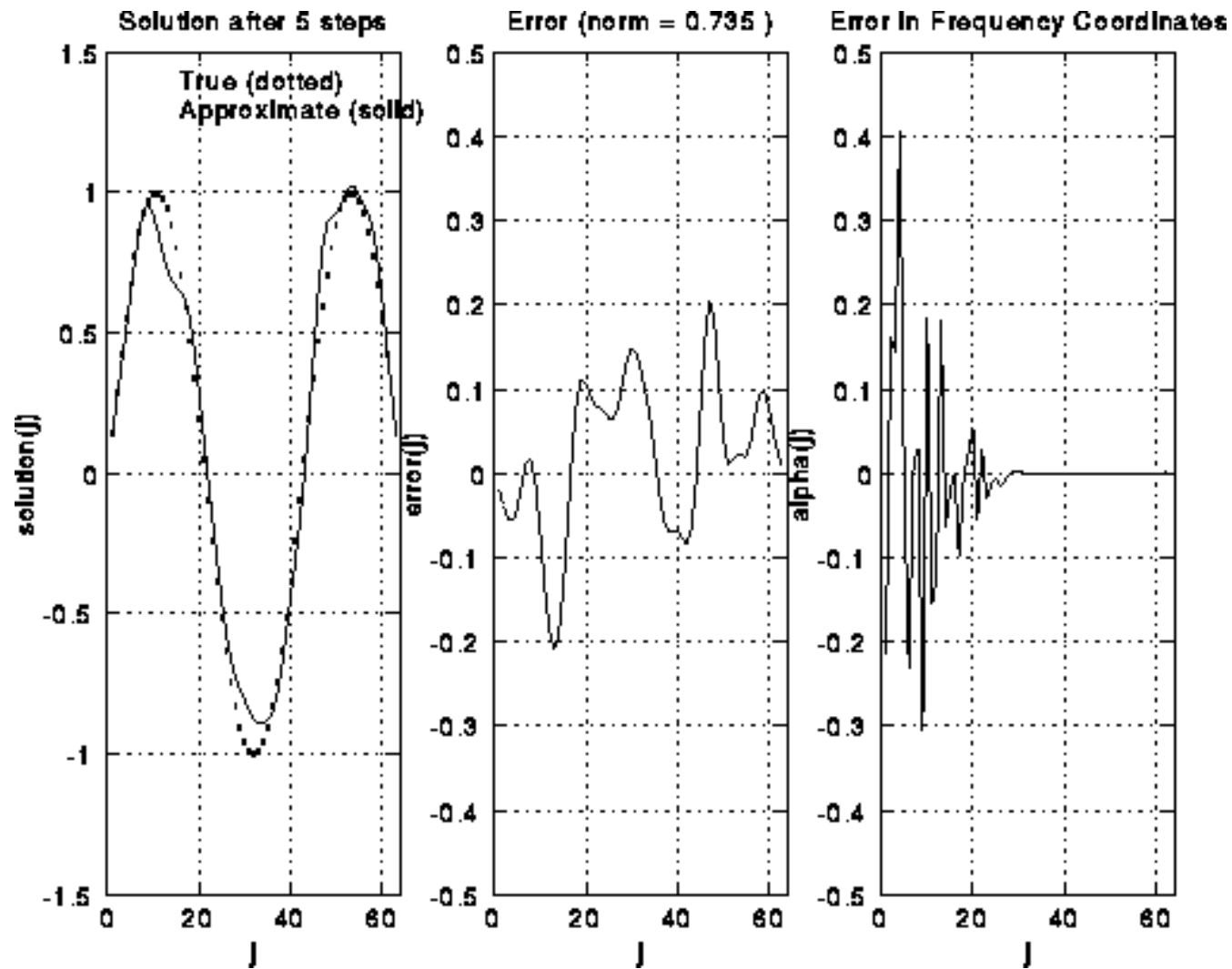
Primjer — 4. red



Primjer — 5. red



Primjer — 6. red



Primjer — zaključak

Ponašanje metode:

- Nakom primjene $S^{(i)}$, **desna polovina** frekvencija se **priguši**.
- To omogućava da aproksimativna rješenja postanu **glada**, jer greške u **nižim** frekvencijama izgledaju **glade** nego u višim.
- Na početku norma vektora **greške rapidno** opada s **1.78** na **1.11**, ali kasnije **sporije** opada, jer treba prigušiti **sve manje** grešaka u visokim frekvencijama.
- Zbog toga ima smisla upotrijebiti **samo 1 do 2 iteracije** operatora $S^{(i)}$ u određenom vremenskom trenutku.

2D slučaj

Ako operator $S^{(i)}$ za JOR(ω) metodu u 1D slučaju napišemo u obliku

$$(x_{\text{imp}}^{(i)})_j = (1 - \omega)x_j^{(i)} + \frac{\omega}{2} \left(x_{j-1}^{(i)} + x_{j+1}^{(i)} + b_j^{(i)} \right),$$

onda operator $S^{(i)}$ u 2D slučaju ima oblik

$$(x_{\text{imp}}^{(i)})_{jk} = (1 - \omega)x_{jk}^{(i)} + \frac{\omega}{4} \left(x_{j-1,k}^{(i)} + x_{j+1,k}^{(i)} + x_{j,k-1}^{(i)} + x_{j,k+1}^{(i)} + b_{j,k}^{(i)} \right).$$

U **oba** slučaja koristi se težina $\omega = 2/3$.

Operator restrikcije R

Promatrajmo **operator restrikcije** $R^{(i)}$, koji uzima desnu stranu $b^{(i)}$ problema $P^{(i)}$ i aproksimativno rješenje $x^{(i)}$, i **preslikava** ga na problem $P^{(i-1)}$ s desnom stranom $b^{(i-1)}$ i aproksimativnim rješenjem $x^{(i-1)}$.

Neka je $r^{(i)}$ **rezidual** aproksimativnog rješenja $x^{(i)}$

$$r^{(i)} = G^{(i)}x^{(i)} - b^{(i)}.$$

Ako je $x^{(i)}$ **egzaktno** rješenje, $r^{(i)} = 0$. Ako riješimo jednadžbu

$$G^{(i)}d^{(i)} = r^{(i)}$$

egzaktno za korekciju $d^{(i)}$, tada je $x^{(i)} - d^{(i)}$ rješenje koje tražimo.

Operator restrikcije R (nastavak)

Naime, očito je

$$G^{(i)}(x^{(i)} - d^{(i)}) = G^{(i)}x^{(i)} - G^{(i)}d^{(i)} = (r^{(i)} + b^{(i)}) - r^{(i)} = b^{(i)}.$$

Ako nađemo **približno** rješenje za **korekciju** $d^{(i)}$, onda je

• $x^{(i)} - d^{(i)}$ novo **približno** rješenje.

Aproksimaciju za $d^{(i)}$ dobivamo iz **grubljeg** problema $P^{(i-1)}$, **interpolacijom** dobivenog rješenja.

Da bismo dobili problem $P^{(i-1)}$, operator restrikcije se **ne primjenjuje** direktno na par $(b^{(i)}, x^{(i)})$,

• jer bismo trebali restringirati **oba** vektora.

Baš zato i idemo na **rezidual**!

Operator restrikcije R (nastavak)

Za dobivanje $P^{(i-1)}$ moramo

- izračunati rezidual $r^{(i)}$,
- **restringirati** ga na sljedeću **grublju** mrežu da dobijemo $b^{(i-1)} = r^{(i-1)}$,
- staviti početnu pretpostavku rješenja $x^{(i-1)} = 0$.

Operator $R^{(i)}$ se primjenjuje **samo** na rezidual $r^{(i)}$.

Problem $P^{(i-1)}$ se onda svodi na rješavanje sustava

$$G^{(i-1)} d^{(i-1)} = r^{(i-1)},$$

za korekciju $d^{(i-1)}$, s **početnom** aproksimacijom $x^{(i-1)} = 0$ (0 je vektor!). Dobiveno (približno) rješenje $d^{(i-1)}$ interpolacijom prevodimo u $d^{(i)}$.

Operator restrikcije R (nastavak)

Najlakši način za računanje restrikcije je

- uzimanje iste vrijednosti u zajedničkim točkama (onima iz grublje mreže).

No, to se ne radi!

Restrikcija se dobiva usrednjavanjem vrijednosti $r^{(i)}$ s najbližim susjedima na finoj mreži, da bi se dobila aproksimacija na grubljoj mreži.

Vrijednost u točki grublje mreže je zbroj:

- 0.5 puta vrijednost u odgovarajućoj točki fine mreže i
- 0.25 puta vrijednost u 2 najbliža susjeda na finoj mreži.

Operator restrikcije R (nastavak)

Matrično, to izgleda ovako:

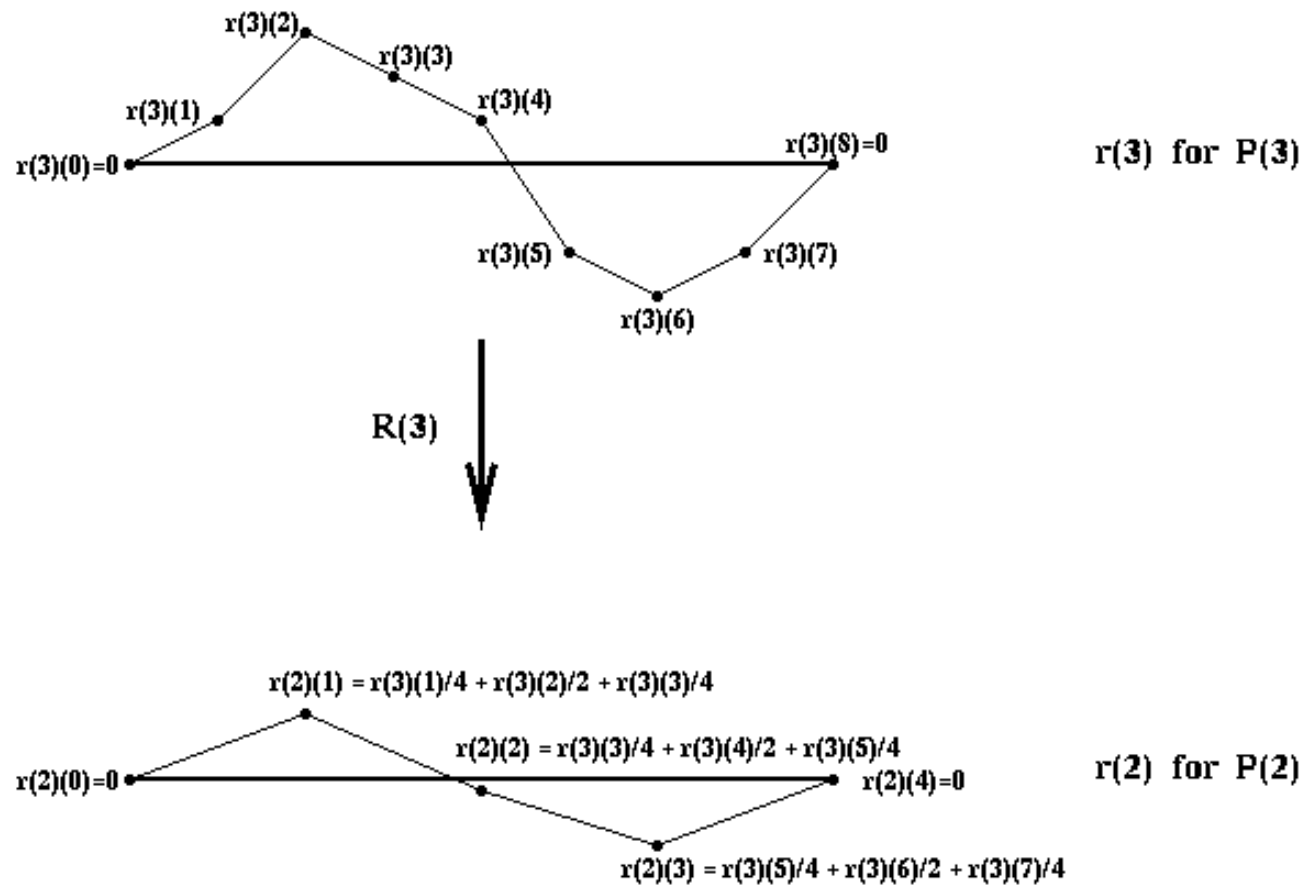
$$b^{(i-1)} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & & & & & & & & \\ & & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & & & & & & \\ & & & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & & & & & \\ & & & & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & & & & \\ & & & & & \ddots & \ddots & \ddots & & & \\ & & & & & & & \ddots & \ddots & \ddots & \\ & & & & & & & & \ddots & \ddots & \ddots \end{bmatrix} b^{(i)}.$$

Ovo usrednjavanje možemo interpretirati “integralno”,

- kao srednju vrijednost integrala u okolini,
- a integral aproksimiramo produljenom trapeznom formulom s dva podintervala (lijevo i desno od točke).

Operator restrikcije R (nastavak)

Restriction Operator in Multigrid



Operator restrikcije u 2D

U 2D slučaju, operator $R^{(i)}$ zahtijeva **usrednjavanje** s (najviše) **8 najbližih** susjeda (u N, S, E, W, NW, SW, SE i NE smjerovima prema kompasu). Princip usrednjavanja je isti, samo se primijenjuje u **oba** smjera.

Vrijednost u (unutrašnjoj) točki **grublje** mreže je zbroj

- $1/4$ vrijednosti u **toj** točki na **finijoj** mreži,
- $1/8$ puta zbroj vrijednosti u susjedima **lijevo, desno, gore** i **dolje** (W, E, N, S)
- $1/16$ puta zbroj vrijednosti u **dijagonalnim** susjedima (NW, NE, SE, SW).

Operator interpolacije I_n

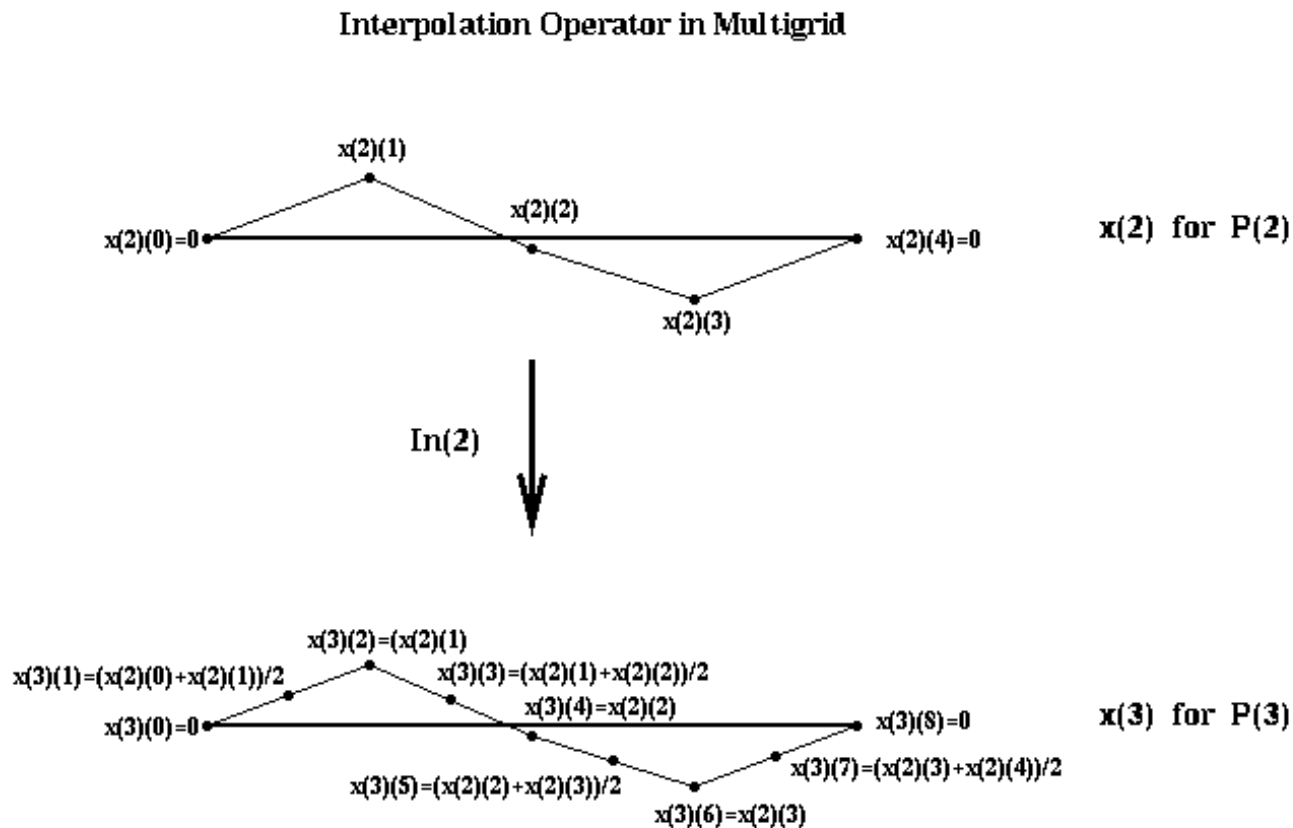
Operator interpolacije $I_n^{(i-1)}$ uzima približno rješenje $d^{(i-1)}$ na **grubljoj** mreži i preslikava ga u aproksimaciju rješenja $d^{(i)}$ na **finijoj** mreži. Standardno se koristi **linearna** interpolacija,

Matematički, interpolaciju rješenja možemo napisati kao

$$d^{(i)} = I_n^{(i-1)} d^{(i-1)} = \begin{bmatrix} \frac{1}{2} & & & & \\ 1 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ & 1 & \cdots & & \\ & \frac{1}{2} & \cdots & \frac{1}{2} & \\ & & \cdots & 1 & \\ & & & \frac{1}{2} & \end{bmatrix} d^{(i-1)}.$$

Operator interpolacije I_n (nastavak)

Rješenje $d^{(i-1)}$ se interpolira na **finojoj** mreži na sljedeći način.



Operator interpolacije u 2D

U 2D slučaju, interpolacija zahtijeva **usrednjavanje** s (najviše) **4 najbliža** susjeda (NW, SW, SE i NE).

- Ako je točka **finije** mreže **ujedno** i točka **grublje** mreže, uzimamo **istu** vrijednost (“usrednjavanje” preko **1** točke).
- Ako točka **finije** mreže ima **dva najbliža** susjeda iz **grublje** mreže (**gore** i **dolje**, ili **lijevo** i **desno**), uzimamo **srednju** vrijednost tih susjeda (usrednjavanje preko **2** točke s faktorom **1/2**, kao u **1D** slučaju).
- Konačno, ako točka **finije** mreže ima **četiri najbliža** susjeda iz **grublje** mreže (**dijagonalno** raspoređenih, tako da je točka u središtu kvadrata), uzimamo **srednju** vrijednost tih susjeda (usrednjavanje preko **4** točke s faktorom **1/4**).

Usporedba metoda za diskretnu Poissonovu jednadžbu

Modelni problem — veličina i metode

Problem. Za zadani $N \in \mathbb{N}$, gledamo **diskretnu** Poissonovu jednadžbu u **2D**,

• na **ekvidistantnoj** mreži od $N \times N$ točaka,
na pr., na jediničnom kvadratu, s korakom $h = 1/(N + 1)$.

Složenost metoda izražavamo u terminu

• broja **nepoznanica** $n := N^2$.

Dakle, “mali” n je **kvadrat** “velikog” N .

Metode za rješavanje diskretne Poissonove jednadžbe grubo dijelimo u **dvije** skupine:

• **D** = **direktne** metode,

• **I** = **iterativne** metode.

“Pravila igre” za usporedbu metoda

Bitna **razlika** između ovih skupina:

- **Direktne** metode daju **točan** rezultat, ako **nema** grešaka zaokruživanja, tj. **složenost ne ovisi** o **točnosti** rezultata.
- Kod **iterativnih** metoda, **broj iteracija** (pa onda i **složenost**) **ovisi** o zadanoj **točnosti**.

Za **korektnu** usporedbu, treba

- **izabrati** traženu **točnost** ε za iterativne metode.

Dogovor: **iteriramo** sve dok greška ne padne **ispod** zadane

- **konstantne** “male” vrijednosti — na primjer, $\varepsilon = 10^{-6}$, tj. greška **ne ovisi** o koraku h , odnosno, o dimenziji n .

Točnost i broj iteracija

Neka je $\rho(T)$ = spektralni radijus matrice iteracija T u iterativnoj metodi. Za broj iteracija n_{iter} onda vrijedi

$$(\rho(T))^{n_{\text{iter}}} \leq \varepsilon, \quad \text{ili} \quad n_{\text{iter}} \approx \frac{\log \varepsilon}{\log \rho(T)}.$$

Znamo da $\rho(T)$ ovisi o n , i ta ovisnost varira, ovisno o metodi. Uz našu pretpostavku, $\log \varepsilon$ je konstantan u gornjoj formuli!

Kad bismo iterirali sve dok greška ne padne na razinu greške odbacivanja, tj.

$$\varepsilon = O(h^2) = O((N + 1)^{-2}) = O(n^{-1}),$$

onda broj iteracija n_{iter} raste za faktor reda veličine $O(\log n)$.

Tablica složenosti metoda

Metoda	sekv. vrijeme	prostor	tip
Puni Gauss/Cholesky	n^3	n^2	D
Eksplicitni inverz	n^2	n^2	D
Vrpčasti Cholesky	n^2	$n^{3/2}$	D
Šuplji Cholesky	$n^{3/2}$	$n \log n$	D
Jacobi	n^2	n	I
Gauss–Seidel	n^2	n	I
Konjugirani gradijenti	$n^{3/2}$	n	I
Optimalni SOR	$n^{3/2}$	n	I
Sim. SOR s Čeb. ubrzanjem	$n^{5/4}$	n	I

Tablica složenosti metoda (nastavak)

Metoda	sekv. vrijeme	prostor	tip
Brza Fourierova transformacija	$n \log n$	n	D
Blok ciklička redukcija	$n \log n$	n	D
Multigrid	n	n	I
Donja ograda	n	n	

Dakle,

- Multigrid metoda je **optimalna**,
- jer “troši” **konstantno** vrijeme i prostor po **nepoznanici**, do na (**malu**) multiplikativnu konstantu, “skrivenu” u $O(n)$.