MA2501 Numeriske metoder

Tellende hjemmeøving 2

Praktisk informasjon

- **Utlevering**: Mandag 24. april
- **Innlevering**: Fredag 12. mai, senest klokken 18:00.
- **Veiledning underveis**:
  - Faglærer er tilgjengelig på kontor 1354 i Sentralbygg II i de vanlige forelesningstimene og øvingstimene samt i tidsrommet 14:15 – 16:00 alle dager bortsett fra onsdag 26 april. Spørsmål kan også sendes på epost til `<bardsk@math.ntnu.no>` og vil bli besvart i den grad de er rimelige.
  - Øvingslærers treffetid og -sted offentliggjøres på kursets øvingssider.

  All epost til fagstaben *må* ha «MA2501» i tittelen.
- **Gruppestørrelse**: Maksimalt tre studenter pr. gruppe.

Innledning

En plate laget av et materiale med homogene varmeledningsegenskaper utsettes for en uniformt fordelt ekstern varmekilde mens sidekantene holdes fast på 0° C. Varmekilden har stått på lenge, så temperaturen i platen er nå stasjonær (forandres ikke med tiden). Det kan da vises at temperaturen $u(x, y)$ i platen vil tilfredsstille en partiell differensialligning med randkrav gitt ved

$$\begin{aligned} -\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) &= 1, \quad \text{i platen} \\ u(x, y) &= 0, \quad \text{på randen.} \end{aligned} \tag{1}$$

Vi skal senere i kurset se hvordan vi kan omformulere problemet (1) til et problem som er mer egnet til løsning på en datamaskin. I denne sammenhengen er det imidlertid tilstrekkelig å vite at det omformulerte problemet er gitt ved et lineært ligningssystem,

$$A\mathbf{u} = \mathbf{f}, \tag{2}$$

hvor løsningen $\mathbf{u}$ er temperaturverdiene i et endelig antall *nodepunkter* i platen. Oppgaven denne gang går ut på å studere og sammenligne ulike iterative løsningsmetoder for ligningssystemet (2).

For å redusere programmeringsarbeidet er det på kursets hjemmesider lagt ut et program som setter opp ligningssystemet (dvs. lager A og $\mathbf{f}$), løser det ved hjelp av Gausseliminasjon og plotter løsningen. Programmet heter `driver` og det benytter funksjonene `setup` og `plot_solution` som begge er tilgjengelige på kursets hjemmesider. Programmet kan for eksempel brukes som følger

```
>> R = 'B'; n = 32;
>> driver(R, n)
```

Parameteren `n`, som må være et heltall større enn 0, angir nøyaktigheten av løsningen. Større verdier av `n` gir en mer nøyaktig løsning, men krever også mer beregningsarbeid for å bestemme løsningen. Spesielt vokser dimensjonen av systemet (2) svært hurtig med økende verdi av `n`. Videre er `R` et *navn* på en gitt plateform. `R = 'B'` betegner en kvadratisk plate med et hull formet som en sommerfugl utskåret i midten. Andre mulige valg er listet i funksjonen `setup`. Det anbefales at dere tar utgangspunkt i `driver` når dere skal skrive deres egen kode.

Vi bemerker at matrisen A, uansett hvilke valg vi gjør for `R` og `n`, er symmetrisk og positiv definitt (SPD). Den er dessuten *glissen*, hvilket betyr at kun svært få matriseelementer er forskjellige fra null – i dette tilfellet maksimalt 5 matriseelementer pr. rad. MATLAB-funksjonen `spy` lar deg visuelt inspisere hvilke matriseelementer som er null og hvilke som ikke er det.

Rapporten

Hver gruppe skal levere en selvstendig rapport fra arbeidet. Rapportens lengde bør ikke overstige $\mathbf{3 + 2}\boldsymbol{n}$ sider hvor $\boldsymbol{n}$ er antall personer i gruppen. Denne begrensningen inkluderer *ikke* utskrift av MATLAB-kode. Diskuter gjerne ulike problemstillinger med andre grupper, men hver gruppes rapport skal som sagt være et selvstendig arbeid. Det forutsettes at dere ikke kopierer andres MATLAB-kode og at hvert gruppemedlem står inne for rapportens innhold.

Hvis dere henter materiale på internett eller i andre bøker enn Kincaid og Cheney må dere oppgi referanser.

Rapporten skal merkes med et ***gruppenummer*** som utdeles av faglærer på faglærers kontor når gruppene er dannet samt ***alle*** gruppemedlemmenes ***navn***.

Oppgave 1 – Klassiske iterative metoder

Bakgrunnsmateriale for denne oppgaven er teksten om iterative metoder for lineære ligningssystemer i kapittel **8.2** i Cheney & Kincaid.

1) Gi en kort beskrivelse av metoden kjent som *Suksessiv Overrelaksasjon* (SOR). Begrunn spesielt at SOR kan benyttes på ligningssystemet (2).

2) Implementer SOR og test implementasjonen på ligningssystemet (2).

3) Hvordan varierer antall SOR-iterasjoner med relaksasjonsparameteren, ω? Hvordan varierer "optimal" relaksasjonsparameter med dimensjonen av (2) og hvor sensitiv er antall SOR-iterasjoner overfor den optimale verdi av ω?

4) Foreslå en strategi for å utnytte glissenhetsstrukturen i matrisen. Det er ikke nødvendig å implementere denne strategien.

5) Dekomponer matrisen A i (2) som

$$A = D - C_L - C_U$$

hvor $D = \mathrm{diag}(A)$, $C_L = -\,\mathrm{tril}(A,-1)$ og $C_U = -\,\mathrm{triu}(A,1)$. *Symmetrisk* suksessiv overrelaksasjon (SSOR) er på matriseform gitt ved relasjonene

$$\begin{aligned}(D-\omega C_L)\mathbf{x}^{(k+1/2)} &= \omega(C_U\mathbf{x}^{(k)}+\mathbf{b}) + (1-\omega)D\mathbf{x}^{(k)}\\ (D-\omega C_U)\mathbf{x}^{(k+1)} &= \omega(C_L\mathbf{x}^{(k+1/2)}+\mathbf{b}) + (1-\omega)D\mathbf{x}^{(k+1/2)},\end{aligned}$$

for $k = 0, 1, 2, \ldots$. Gi en intuitiv beskrivelse av denne algoritmen.

Formuler algoritmen på komponentform og implementer denne. Undersøk hvordan ytelsen, målt i antall iterasjoner *og* beregningstid, for SSOR forholder seg til tilsvarende mål for SOR. Studer til slutt SSORs avhengighet av relaksasjonsparameteren.

Oppgave 2 – konjugerte gradienter

La A være en fiksert symmetrisk, positiv definitt matrise av dimensjon $n \times n$ og $\mathbf{b}$ en fiksert vektor av dimensjon n. Definer funksjonen $f : \mathbb{R}^n \to \mathbb{R}$ ved

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^\mathsf{T} A\mathbf{x} - \mathbf{x}^\mathsf{T}\mathbf{b}. \tag{3}$$

1) Vis at f har et entydig kritisk punkt $\mathbf{x}_*$ gitt som løsningen av ligningsystemet $A\mathbf{x}_* = \mathbf{b}$ og at $f(\mathbf{x}_*) < f(\mathbf{x})$ for alle $\mathbf{x} \in \mathbb{R}^n$ med $\mathbf{x} \neq \mathbf{x}_*$.

Hint: Skriv $\mathbf{x} = \mathbf{x}_* - \mathbf{e}$ med $\mathbf{e} \neq \mathbf{0}$.

2) La $\mathbf{x}$ være en vilkårlig vektor i $\mathbb{R}^n$. Begrunn at "residualet" $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ er den retningen hvor funksjonen f avtar hurtigst fra $\mathbf{x}$.

3) La nå $\{\mathbf{p}^{(0)}, \mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(n-1)}\}$ være en mengde av *konjugerte* (også kalt *A-ortogonale*) vektorer, dvs.

$$\mathbf{p}^{(i)\mathsf{T}} A\mathbf{p}^{(j)} = 0$$

når $i \neq j$. La $\mathbf{x}^{(0)}$ være vilkårlig og definer en følge vektorer $\mathbf{x}^{(k+1)}$ ved

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}.$$

Bestem α_k slik at $\mathbf{e}^{(k+1)} = \mathbf{x}_* - \mathbf{x}^{(k+1)}$ blir A-ortogonal til $\mathbf{p}^{(k)}$ og begrunn hvorfor dette i eksakt aritmetikk (dvs. uten avrundingsfeil) betyr at $\mathbf{x}^{(n)} = \mathbf{x}_*$.

Metoden av konjugerte gradienter er en smart måte for iterativ konstruksjon av de konjugerte *søkeretningene* $\mathbf{p}^{(i)}$.

Ved å utnytte ulike polynomidentiteter og egenskaper ved noen spesielle vektorrom kalt *Krylovrom* kan metoden til slutt formuleres som

$$\begin{aligned}\alpha_k &= \frac{\mathbf{r}^{(k)\mathsf{T}}\mathbf{r}^{(k)}}{\mathbf{p}^{(k)\mathsf{T}} A\mathbf{p}^{(k)}}\\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}\\ \mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} - \alpha_k A\mathbf{p}^{(k)}\\ \beta_{k+1} &= \frac{\mathbf{r}^{(k+1)\mathsf{T}}\mathbf{r}^{(k+1)}}{\mathbf{r}^{(k)\mathsf{T}}\mathbf{r}^{(k)}}\\ \mathbf{p}^{(k+1)} &= \mathbf{r}^{(k+1)} + \beta_{k+1}\mathbf{p}^{(k)}\end{aligned}$$

for alle $k \geq 0$ med $\mathbf{p}^{(0)} = \mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$. Hvor mye lagerplass (minne) trengs for å implementere algoritmen når vi bare er interessert sluttresultatet $\mathbf{x}_*$?

Historisk ble denne metoden introdusert rundt 1950 som et alternativ til Gausseliminasjon for å finne den eksakte løsningen til et ligningsystem, dvs. man konstruerte $\mathbf{x}^{(n)}$. Imidlertid ble metoden ikke brukt i praksis fordi den hadde en høyere totalkostnad enn Gausseliminasjon. Metoden fikk derimot fornyet interesse som en *iterativ* metode for $A\mathbf{x}_* = \mathbf{b}$ ca. 20 år senere da det ble oppdaget at $\mathbf{x}^{(k)} \approx \mathbf{x}_*$ selv for $k \ll n$. Dermed fikk man et nyttig verktøy for løsning av den type store ligningssystemer som oppstår i løsning av partielle differensialligninger som f.eks. (1).

MATLAB har en innebygget funksjon, `pcg`, som implementerer denne metoden. `pcg` benytter et stoppkriterium på formen

$$\|\mathbf{r}^{(k)}\|_2 \leq \varepsilon \|\mathbf{r}^{(0)}\|_2 \tag{4}$$

der *toleransen* ε kan spesifiseres av brukeren, men har en standardverdi på 10^{-6}. Bruk `pcg` til å løse ligningssystemet (2). Hvordan varierer antall iterasjoner i "CG" med systemdimensjonen?

Det kan vises at feilen $\mathbf{e}^{(k)} = \mathbf{x}_* - \mathbf{x}^{(k)}$ oppfyller

$$\|\mathbf{e}^{(k)}\|_A \leq 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^k \cdot \|\mathbf{e}^{(0)}\|_A$$

hvor κ er matrisen As kondisjonstall og

$$\|\mathbf{v}\|_A^2 = \mathbf{v}^\mathsf{T} A\mathbf{v}$$

er kjent som A-normen til en positiv definitt matrise. Hvordan stemmer dette med observasjonene for antall CG-iterasjoner?

Hint: Hvordan avhenger antall CG-iterasjoner nødvendig for å tilfredsstille kravet $\|\mathbf{e}^{(k)}\|_A \leq \varepsilon \|\mathbf{e}^{(0)}\|_A$ av κ?