Norges teknisk–naturvitenskapelige universitet
Institutt for matematiske fag

TMA4320 - Introduksjon til vitenskapelige beregninger Våren 2017

Prosjekt 1

1 Introduksjon

Ved flere anvendelser er man interessert i å bestemme verdier av visse parametre som styrer et gitt system. Selve systemet kan også modeleres ved hjelp av numeriske metoder.

I dette prosjektet skal vi se på en slik situasjon der vårt system beskriver et stasjonært tilfelle av en varmeledningslikning i én dimensjon:

$$\begin{aligned} -u''(x) &= f(x), \qquad a < x < b, \\ u(a) &= u_a, \\ u(b) &= u_b, \end{aligned} \tag{1}$$

hvor varmekilden f er en gitt funksjon, u_a og u_b er gitt som temperaturen på randene av intervallet (a, b), og u er den ukjente temperaturprofilen.

Vi antar videre at varmekilden f avhenger av en parameter $\sigma > 0$; mer konkret, vi skal bruke

$$f(x; \sigma) = \exp\left\{ -\frac{(x-\mu)^2}{\sigma^2} \right\}, \tag{2}$$

hvor $\mu \in [a, b]$ er gitt. La oss definere den gjennomsnittlige temperaturen på intervallet (a, b) som en funksjon av σ:

$$u_{\text{avg}}(\sigma) = \frac{1}{b-a} \int_a^b u(x)\,\mathrm{d}x, \tag{3}$$

hvor $u(x)$ er løsningen til (1) som tilsvarer $f(x) = f(x; \sigma)$.

Vårt mål er:

Bestem $\sigma > 0$ slik at

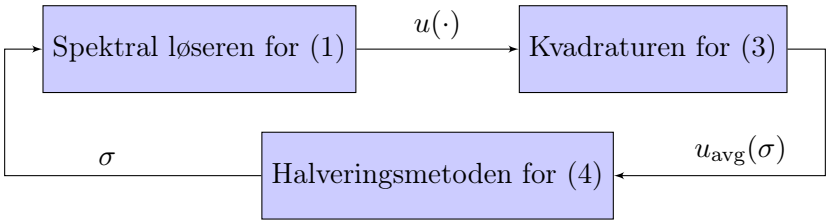
$$u_{\text{avg}}(\sigma) = \hat{u}_{\text{avg}}, \tag{4}$$

hvor $\hat{u}_{\text{avg}}$ er gitt.

2 Numerisk strategi

- Vi må bruke en slags numerisk algoritme til å løse likning (4). Det er lettest å bruke halveringsmetoden her; hvis dere har tid kan dere prøve å bruke sekant eller (mye vanskeligere!) Newtons metode.
- Vi skal bruke en numerisk kvadratur til å beregne integralet i (3). Dere kan selve velge metoden. Kanskje den letteste å bruke her er trapesregelen (Trapezoid Rule).
- Vi skal bruke en spektraldiskretisering av problemet (1) for å finne en tilnærming til løsningen $u(\cdot)$, se seksjon 3.

Sammenhengen mellom de numeriske algoritmene er vist i Fig. 1.



Figur 1: Blokkdiagram som viser samspillet mellom de numeriske algoritmene i dette prosjektet.

3 Numerisk løsning av (1)

3.1 Spektraldiskretisering av (1)

Vi velger N punkter på intervallet $[a, b]$: $a = x_0 < x_1 < \cdots < x_{N-2} < x_{N-1} = b$. La $L_i(x)$, $i = 0, \ldots, N-1$ være Lagrange interpolasjonspolynomet av grad $N-1$ som oppfyller $L_i(x_i) = 1$ og $L_i(x_j) = 0$, $\forall j \neq i$. Vi søker en tilnærming $\tilde{u}(x)$ til løsningen $u(x)$ for (1) i form av

$$\tilde{u}(x) = \sum_{i=0}^{N-1} U_i L_i(x), \tag{5}$$

hvor de ukjente koeffisientene U_i, $i = 0, \ldots N-1$ bestemmes fra likningsystemet:

$$\left\{ \begin{array}{ll} \tilde{u}(x_0) = u_a, & \\ -\tilde{u}''(x_i) = f(x_i), & i = 1, \ldots, N-2, \\ \tilde{u}(x_{N-1}) = u_b, & \end{array} \right\} \iff AU = B, \tag{6}$$

hvor $A \in \mathbb{R}^{N \times N}$, $B \in \mathbb{R}^N$, og $U = (U_0, \ldots, U_{N-1})^{\mathrm{T}}$.

Bestem koeffisientene A_{ij}, B_i, $i, j = 0, \ldots, N-1$, og implementer funksjonene som beregner disse i Python. Disse funksjonene kan f.eks. se slik ut:

```
import numpy as np

# set up the LHS (left hand side) for the spectral method for Laplace eqn
def spectral_laplace_lhs(x):
    N = len(x)
    a,b = x[0],x[N-1]
    A = np.zeros((N,N))
    # FIXIT: bestem A_ij
    return A

# set up the RHS (right hand side) for the spectral method for Laplace eqn
def spectral_laplace_rhs(x,f,ua,ub):
    N = len(x)
    a,b = x[0],x[N-1]
    B = np.zeros(N)
    # FIXIT: bestem B_i
    return B

# set up the spectral method for Laplace eqn and solve the resulting system
def spectral_laplace(x,f,ua,ub):
    """
    Set up the spectral discretization of Laplace eqn on the interval x
    -u'' = f;  so f is the RHS of the system
    ua, ub - Dirichlet boundary conditions
    """
    #
    A = spectral_laplace_lhs(x)
    B = spectral_laplace_rhs(x,f,ua,ub)
    # solve the system
    return np.linalg.solve(A,B)
```

3.2 Valg av diskretiseringspunkter

Test koden med *både* jevnt fordelte diskretiseringspunkter:

```
import numpy as np
x_uniform = np.linspace(a,b,N)
```

og Chebyshevs intepolasjonspunkter[1]:

```
import numpy as np
x_cheby = (b+a)/2. + (a-b)/2. * np.cos(np.arange(N)*np.pi/(N-1))
```

3.3 Verifikasjon av koden

Vi skal bruke den såkalte "method of manufactured solutions" for å teste om koden fungerer og konvergerer mot løsningen til (1) når antall diskretiseringspunkter øker.

Som eksempel kan vi velge $a = 0$, $b = 1$, $u(x) = \exp(x)\cos(8\pi x)$. Fra denne løsningen kan man beregne u_a, u_b, og $f(x)$ slik at $u(x)$ løser (1). Disse kan da sendes inn i koden, og

[1] Legg merke til at formelen er litt ulik fra den gitt i boka, fordi vi i tillegg inkluderer intervallets sluttpunkter.

feilen

$$e_N = \max_{0 \le i \le N-1} |\tilde{u}(x_i) - u(x_i)|$$

kan beregnes.

Vis på samme figur hvordan feilen e_N endrer seg med antall punkter N for både jevnt fordelte punkter og Chebyshev punkter. Siden vi forventer at feilen er liten, kan den plottes på den logaritmiske skalaen (se `matplotlib.pyplot.semilogy`).

4 Numerisk integrasjon

Bruk en numerisk kvadratur for å evaluere integralet i (3). Man kan gjenbruke diskretiseringspunktene x_i, $i = 0, \ldots, N-1$ fra seksjon 3 til å definere "paneler" (delintervaller), hvis man f.eks. vil bruke Newton-Cotes kvadratur.

5 Løsning av likning (4)

Likningen (4) skal løses numerisk ved hjelp av en passende metode. Det enkleste er å bruke halveringsmetoden, som har veldig robust konvergens uavhengig av likning. Tenk på hvordan vi velger startintervallet for denne metoden.

6 Til slutt:

Ta $a = 0$, $b = 10$, $u_a = -1$, $u_b = 1$, $\mu = 2.0$ (μ inngår i definisjonen (2)), og $\hat{u}_{\text{avg}} = 3$. Bruk $N = 40$ Chebyshevs interpolasjonspunkter og bestem σ som løser (4) med feilen mindre enn 10^{-6}.

- I rapporten skal dere forklare de viktigste detaljene i koden; dere kan gjerne inkludere kortere deler av algoritmene, men ikke inkluder *hele* koden i rapporten.
- Dokumentér og forklar kort om deres valg i forhold til de mulige numeriske metodene som kan brukes i seksjonene 4 og 5.
- Dokumentér verifikasjon av koden i seksjon 3. Diskuter om konvergenskurvene ser ut som dere forventer.
- Forklar kort hvordan dere verifiserer implementasjoner av de numeriske algoritmene som brukes i seksjonene 4 og 5.
- Diskuter den siste oppgaven: hvor mange iterasjoner blir brukt osv. Inkluder grafene av varmekilden, som tilsvarer σ som dere har funnet, samt temperaturprofilen.