

Prosjekt 1

Live Wang Jensen

September 14, 2016

Abstract

I dette prosjektet skal vi se nærmere på en numerisk løsning av den velkjente Poisson-ligningen, hvor Dirichlets grensebetingelser er brukt. Den andrederiverte er blitt tilnærmet med en tre-punkts-formel, og selve problemet løses ved hjelp av et lineært ligningssett. Vi skal løse disse ligningene på to ulike måter; ved Gauss-eliminasjon og ved LU-faktorisering. De to løsningsmetodene skal så sammenlignes når det kommer til FLOPS (floating point operations per second) og beregningstid.

1 Introduksjon

Vi starter med å se på den én-dimensjonale Poisson ligningen for en sfærisk kule funksjon (?). Denne ligningen løses numerisk ved å dele opp, eller diskretisere, intervallet $x \in [0,1]$, og løsningen f på ligningen er gitt. Teoridelen viser at det å finne en diskret løsning vil være det samme som å løse et lineært ligningssett, $A\mathbf{u} = \tilde{\mathbf{b}}$. Det vil vise seg at matrisen A er en såkalt tridiagonal matrise, noe som forenkler Gauss-eliminasjonen betraktelig. Selve metoden er implementert i et Python-program, hvor vi har variert antall grid-points n . Den numeriske løsningen sammenlignes så med den analytiske løsningen. Deretter beregnes den maksimale relative feilen i den numeriske løsningen. Resultatet plottes som en funksjon av n . Helt til slutt skal vi bruke LU-faktorisering på matrisen A til å finne den numeriske løsningen på Poisson ligningen. Antall FLOPS og beregningstid sammenlignes så mellom de to løsningmetodene.

2 Teori

Mange differensialligninger innenfor fysikk kan skrives på formen

$$\frac{d^2 y}{dx^2} + k^2(x)y = f(x) \tag{1}$$

hvor f er det uhomogene leddet i ligningen, og k^2 er en reell funksjon. Et typisk eksempel på en slik ligning finner vi i elektromagnetismen, nemlig Poisson-ligningen

$$\Delta^2 \Phi = -4\pi\rho(\mathbf{r}) \quad (2)$$

Her er Φ det elektrostatiske potensialet, mens $\rho(\mathbf{r})$ er ladningsfordelingen. Dersom vi antar at Φ og $\rho(\mathbf{r})$ er sfærisk symmetriske, kan vi forenkle ligning (2) til en én-dimensjonal ligning,

$$\frac{1}{r^2} \frac{d}{dr} \left(r^2 \frac{d\Phi}{dr} \right) = -4\pi\rho(r) \quad (3)$$

Hvis vi bruker at $\Phi(r) = \phi(r)/r$, kan ligningen skrives som

$$\frac{d^2\phi}{dr^2} = -4\pi r\rho(r) \quad (4)$$

Det uhomogene leddet f er nå gitt ved produktet $-4\pi\rho$. Dersom vi lar $\phi \rightarrow u$ og $r \rightarrow x$, ender vi opp med en generell, én-dimensjonal Poisson ligning på formen

$$-u''(x) = f(x) \quad (5)$$

Det er denne ligningen vi skal se nærmere på i denne oppgaven. Nærmere bestemt skal vi løse ligningen

$$-u''(x) = f(x), \quad x \in (0, 1), \quad u(0) = u(1) = 0 \quad (6)$$

Vi skal altså holde oss innenfor intervallet $x \in (0, 1)$, med Dirichlet-grensebetingelsene gitt ved $u(0) = u(1) = 0$. Vi definerer den diskrete tilnærmingen til løsningen u med v_i , slik at $x_i = ih$, hvor $h = 1/(n+1)$ er steglengden. Vi får da at $x_0 = 0$ og $x_{n+1} = 1$. Den andrederiverte av u kan da tilnærmes med en tre-punkts formel

$$-\frac{v_{i+1} + v_{i-1} - 2v_i}{h^2} = f_i \quad \text{når} \quad i = 1, \dots, n \quad (7)$$

hvor $f_i = f(x_i)$ og grensebetingelsene er gitt som $v_0 = v_{n+1} = 0$. Feilleddet her går som $O(h^2)$.

Dersom vi multipliserer leddet h^2 i ligning (7) på hver side, kan vi definere leddet på høyre side av ligningen som $\tilde{b}_i = h^2 f_i$. Vi ender altså opp med

$$-v_{i+1} - v_{i-1} + 2v_i = \tilde{b}_i \quad (8)$$

Dersom vi setter inn ulike verdier av i i ligningen ovenfor, ender vi opp med en tilhørende ligning på samme form:

$$i = 1 : \quad v_2 + v_0 - 2v_1 = \tilde{b}_1$$

$$i = 2 : \quad v_3 + v_1 - 2v_2 = \tilde{b}_2$$

osv. Vi ender altså opp med et lineært ligningssett som kan settes opp som en matriseligning:

$$\begin{pmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ \dots \\ \dots \\ v_n \end{pmatrix} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \dots \\ \dots \\ \dots \\ \tilde{b}_n \end{pmatrix} \quad (9)$$

hvor A er en tridiagonal $n \times n$ -matrise. Hvis vi kaller vektorene

$$\begin{pmatrix} v_1 \\ v_2 \\ \dots \\ \dots \\ \dots \\ v_n \end{pmatrix} = \mathbf{v} \quad \text{og} \quad \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \dots \\ \dots \\ \dots \\ \tilde{b}_n \end{pmatrix} = \tilde{\mathbf{b}} \quad (10)$$

kan vi på forkortet form skrive ligningen som

$$\mathbf{A}\mathbf{v} = \tilde{\mathbf{b}} \quad (11)$$

Her er A og $\tilde{\mathbf{b}}$ kjent, mens vektoren \mathbf{v} er den ukjente.

I vårt tilfelle er funksjonen f er gitt som $f(x) = 100e^{-10x}$. Dersom vi bruker dette i ligning (5) og integrerer denne ligningen analytisk, ender vi opp med en løsning på formen

$$u(x) = 1 - (1 - e^{-10})x - e^{-10x} \quad (12)$$

Det er denne analytiske løsningen vi skal sammenligne den numeriske løsningen med.

3 Løsningsmetoder

3.1 Gauss-eliminasjon

Generell tridiagonal matrise

Vi kan tenke oss at elementene langs diagonalen i matrisen vår, A , utgjør en vektor b , samtidig som elementene på hver side av diagonalen utgjør vektorene a og c . Alle andre elementer i matrisen er null. Vi antar nå at elementene i hver vektor a , b og c *ikke* er identiske. Matriseligningen kan da skrives

$$\begin{pmatrix} b_1 & c_1 & 0 & \dots & \dots & \dots \\ a_2 & b_2 & c_2 & \dots & \dots & \dots \\ & a_3 & b_3 & c_3 & \dots & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ & & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ \dots \\ \dots \\ v_n \end{pmatrix} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \dots \\ \dots \\ \dots \\ \tilde{b}_n \end{pmatrix}. \quad (13)$$

hvor vektorene a , b og c har lengden n . Hvis vi nå tenker oss at $n=4$, forenkler denne matriseligningen seg noe:

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \tilde{b}_3 \\ \tilde{b}_4 \end{pmatrix}. \quad (14)$$

Vi vil da få ligninger på formen

$$\begin{aligned} i = 1: & \quad b_1 v_1 + c_1 v_2 = \tilde{b}_1 \\ i = 2: & \quad a_2 v_1 + b_2 v_2 + c_2 v_3 = \tilde{b}_2 \\ i = 3: & \quad a_3 v_2 + b_3 v_3 + c_3 v_4 = \tilde{b}_3 \\ i = 4: & \quad a_4 v_3 + b_4 v_4 = \tilde{b}_4 \\ & \quad \vdots \\ & \quad a_i v_{i-1} + b_i v_i + c_i v_{i+1} = \tilde{b}_i \quad \text{når } i = 1, 2, \dots, n \end{aligned} \quad (15)$$

For å kunne løse dette ligningssettet, bruker vi metoden *forlengs substitusjon*. Det vi ønsker er å få element a_1 til å bli null. Vi starter med å multiplisere første rad i matrisen med a_2/b_1 . Deretter trekker vi første rad fra andre rad. Skriver vi matrisen på utvidet form vil vi få:

$$\left[\begin{array}{cccc|c} 1 & c_1/b_1 & 0 & 0 & \tilde{b}_1/b_1 \\ 0 & b_2 - \frac{c_1}{b_1}a_2 & c_2 & 0 & \tilde{b}_2 - \frac{\tilde{b}_1}{b_1}a_2 \\ 0 & a_3 & b_3 & c_3 & \tilde{b}_3 \\ 0 & 0 & a_4 & b_4 & \tilde{b}_4 \end{array} \right]$$

Nå som første element i andre rad har blitt redusert til 0, kan vi gå i gang med å redusere andre element i tredje rad. La oss først, for ordens skyld, definere $\tilde{d}_1 = b_1$, $\tilde{d}_2 = b_2 - \frac{c_1}{b_1}a_2$, $\tilde{e}_1 = \tilde{b}_1/\tilde{d}_1$ og $\tilde{e}_2 = (\tilde{b}_2 - \frac{\tilde{b}_1}{b_1}a_2)/\tilde{d}_2$. Vi kan nå multiplisere rad nummer to med $1/\tilde{d}_2$ og døpe om variablene til noe mer oversiktelig:

$$\left[\begin{array}{cccc|c} 1 & c_1/\tilde{d}_1 & 0 & 0 & \tilde{b}_1/\tilde{d}_1 \\ 0 & 1 & c_2/\tilde{d}_2 & 0 & (\tilde{b}_2 - \frac{\tilde{b}_1}{b_1}a_2)/\tilde{d}_2 \\ 0 & a_3 & b_3 & c_3 & \tilde{b}_3 \\ 0 & 0 & a_4 & b_4 & \tilde{b}_4 \end{array} \right] \sim \left[\begin{array}{cccc|c} 1 & c_1/b_1 & 0 & 0 & \tilde{e}_1 \\ 0 & 1 & c_2/\tilde{d}_2 & 0 & \tilde{e}_2 \\ 0 & a_3 & b_3 & c_3 & \tilde{b}_3 \\ 0 & 0 & a_4 & b_4 & \tilde{b}_4 \end{array} \right]$$

Trekker nå $a_3 \cdot$ (rad to) fra rad tre:

$$\left[\begin{array}{cccc|c} 1 & c_1/\tilde{d}_1 & 0 & 0 & \tilde{e}_1 \\ 0 & 1 & c_2/\tilde{d}_2 & 0 & \tilde{e}_2 \\ 0 & 0 & b_3 - \frac{c_2}{\tilde{d}_2}a_3 & c_3 & \tilde{b}_3 - \tilde{e}_2 a_3 \\ 0 & 0 & a_4 & b_4 & \tilde{b}_4 \end{array} \right] \sim \left[\begin{array}{cccc|c} 1 & c_1/\tilde{d}_1 & 0 & 0 & \tilde{e}_1 \\ 0 & 1 & c_2/\tilde{d}_2 & 0 & \tilde{e}_2 \\ 0 & 0 & 1 & c_3/\tilde{d}_3 & (\tilde{b}_3 - \tilde{e}_2 a_3)/\tilde{d}_3 \\ 0 & 0 & a_4 & b_4 & \tilde{b}_4 \end{array} \right]$$

I den siste overgangen har vi multiplisert tredje rad med $1/\tilde{d}_3$. Vi ser et gjentakende mønster hvor

$$\tilde{d}_i = b_i - \frac{c_{i-1}}{\tilde{d}_{i-1}} a_i \quad \text{og} \quad \tilde{e}_i = \frac{\tilde{b}_i - c_{i-1} \tilde{a}_i}{\tilde{d}_i} \quad \text{hvor} \quad i = 2, 3, \dots, n \quad (16)$$

med initialbetingelser $\tilde{d}_1 = b_1$ og $\tilde{e}_1 = \tilde{b}_1/\tilde{d}_1$. Vi har nå utført en forlengs substitusjon. For å finne den endelige (diskrete) løsningen \mathbf{v} , bruker vi det vi kaller *baklengs substitusjon*. Vi starter med å se på det endelige resultatet etter at alle pivotelementene har blitt radredusert til ledende enere:

$$\begin{pmatrix} 1 & c_1/\tilde{d}_1 & 0 & 0 \\ 0 & 1 & c_2/\tilde{d}_2 & 0 \\ 0 & 0 & 1 & c_3/\tilde{d}_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} \tilde{e}_1 \\ \tilde{e}_2 \\ \tilde{e}_3 \\ \tilde{e}_4 \end{pmatrix} \quad (17)$$

Skriver vi ut ligningene får vi at

$$v_4 = \tilde{e}_4$$

$$v_3 + (c_3/\tilde{d}_3)v_4 = \tilde{e}_3$$

$$v_2 + (c_2/\tilde{d}_2)v_3 = \tilde{e}_2$$

$$v_1 + (c_1/\tilde{d}_1)v_2 = \tilde{e}_1$$

Den endelige algoritmen for å finne v_i blir derfor på formen

$$v_i = \tilde{e}_i - (c_i/\tilde{d}_i)v_{i+1} \quad \text{når} \quad i = n-1, n-2, \dots, 1 \quad (18)$$

og $v_n = \tilde{e}_n$ når $i = n$. Figur 1 viser hvordan disse algoritmene kan implementeres inn i C++:

```
// Algorithm for finding v:
// a(i)*v(i-1) + b(i)*v(i) + c(i)*v(i+1) = b_twidd(i)
// Row reduction; forward substitution:
double b_temp = b[1];
v[1] = b_twidd[1]/b_temp;
for (int i=2; i<=n; i++) {
    // Temporary value needed also in next loop:
    diag_temp[i] = c[i-1]/b_temp;
    // Temporary diagonal element:
    b_temp = b[i] - a[i]*diag_temp[i];
    // Updating right hand side of matrix equation:
    v[i] = (b_twidd[i] - v[i-1]*a[i])/b_temp;
}

// Row reduction; backward substitution:
for (int i=n-1; i>=1; i--) {
    v[i] -= diag_temp[i+1]*v[i+1];
}
```

Figure 1: Figuren viser hvordan forlengs -og baklengs substitusjon kan implementeres i C++.

Vi ser at den forlengs substitusjons-algoritmen har 6 FLOPS (floating point operations), mens baklengs substitusjon har 2 FLOPS. Vi ender altså opp med at

$$N_{tridiagonal} = O(8n) \quad (19)$$

Spesialtilfelle

I vårt tilfelle har elementene langs diagonalen identisk verdi. Det vil si; alle elementene $a_i = c_i = -1$, og alle elementene langs diagonalen $b_i = 2$. Setter vi disse tallene inn i algoritmen vår, får vi for forlengs substitusjon at

$$\tilde{d}_i = 2 + \frac{1}{\tilde{d}_i} \quad \text{og} \quad \tilde{e}_i = \frac{\tilde{b}_i + \tilde{e}_{i-1}}{\tilde{d}_i} \quad (20)$$

når $i = 2, 3, \dots, n$, mens baklengs substitusjon forenkles til

$$v_i = \tilde{e}_i + \frac{v_{i+1}}{\tilde{d}_i} \quad (21)$$

når $i = n-1, n-2, \dots, 1$. Figur REF viser hvordan dette vil se ut i C++.

FIGUR HER

Vi ser nå at antall FLOPS har blitt redusert til 6, slik at

$$N_{special} = O(6n) \quad (22)$$

3.2 LU-faktorisering

Det finnes en annen måte å løse det lineære ligningssettet vårt på. Matrisen A i ligningen $A\mathbf{v} = \tilde{\mathbf{b}}$ kan dekomponeres til et produkt av to matriser, nemlig L og U , hvor L er en nedre triangulær matrise og U er en øvre triangulær matrise: $A = LU$

$$A = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \dots & 0 \\ \vdots & \vdots & & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & 0 & \dots & u_{nn} \end{pmatrix} \quad (23)$$

Vi kan nå sette inn at $A = LU$ i ligningen $A\mathbf{v} = \tilde{\mathbf{b}}$, slik at $L(U\mathbf{v}) = \tilde{\mathbf{b}}$. Vi ender opp med to ligningssett:

$$L\mathbf{y} = \tilde{\mathbf{b}} \quad \text{og} \quad U\mathbf{v} = \mathbf{y} \quad (24)$$

4 Relativ feil

OPPGAVE d)

Vi kan finne størrelsen på den relative feilen i beregningene vi nå har gjort. Ved å implementere formelen

$$\epsilon_i = \log_{10} \left(\left| \frac{v_i - u_i}{u_i} \right| \right) \quad (25)$$

hvor $i = 1, \dots, n$. ϵ_i er da en funksjon av $\log_{10}(h)$, siden verdiene v_i og u_i avhenger av steglengden h . Vi finner den maksimale relative feilen for ulike verdier av n , helt opp til $n = 10^7$.

5 Vedlegg

Github-adresse: <https://github.com/livewj/Project-1>

References

- [1] *project1.2016.pdf* found at the official Github-page of the course *FYS3150 - Computational Physics* <https://github.com/CompPhysics/ComputationalPhysics>, 03.09.2016