TMR4162 - Ramme Analyse

Simen Haugerud Granlund

November 27, 2012

Forord

Dette Notatet er laget som en ledd av en innlevering i faget TMR4162 - Prosedyreorientert Programering i 2012. Det inneholder teori og informasjon om et program, kalt FEMA, som utfører en enkel styrkeberegning på en rammekonstruksjon. Programmet kan lastes ned via https://github.com/hgranlund/FEMA.

Simen Haugerud Granlund, Trondheim 2011.

Contents

1 Innledning

Jeg valgte å ta oppgaven "ramme analyse". Grunnen til dette er at jeg synes teorien rundt finite element analyses (FEA) er mye mer interessent og passer bedre til mitt studie enn Poisson-likningen. Jeg valgte også å legge til flere elementer i oppgaven. Oppgaven presiserte at vi bare trengte 2 frihetsgrader per bjelke, samt at vi kun trengte å presentere resultater for bøye-momentet. Jeg valgte å bruke 6 frihetsgrader per bjelke, slik at jeg fikk meg både vertikal og horisontal forskyvninger og rotasjon. De resultatene jeg ville skulle vises var; Moment diagram, skjærkraft diagram, Normalkraft diagram og starttilstanden. Til dette trenger programmet å regne ut Momenter og normale og aksielle krefter til alle bjelker.

Det jeg ville legge mest vekt må under oppgaven var god struktur av koden slik at jeg enkelt kan utvide med nye funksjonaliteter. Med en god struktur er det enkelt å sette seg inn i koden og en slipper å strukturere mye på nytt om en vil legge til nye funksjonaliteter. Jeg liker generelt ikke mye kommentarer i koden. Koden skal være leselig og forståelig uten en hel haug av kommentarer. Selvfølgelig er det steder hvor kommentarer er passende, men generelt holder det med god struktur og navngiving.

2 Teori

Her har jeg tenkt å kort beskrive de teoriene som er brukt i programmet. Jeg vil kun ta med de prinsippene, likningene og matrisene jeg brukte i koden og ikke beskrive i detalj hvordan jeg kan frem til dem basert på lærebøker.

2.1 Elementmetoden

Teorien bak elementmetoden ble utarbeidet på 1940-tallet, men man så ikke den store nytteverdien av den før datamaskinen kom til verden. Det første programmet som implementerte elementmetoden var NASTRAN skrevet i Fortran i 1965 [1]. Elementmetoden er i prinsippet en numerisk metode for å finne approksimasjoner til differensiallikninger. Den er ofte brukt innenfor en rekke felter: strømning, varmeleding, svinginger, elektriske felt og liknende. I denne oppgaven brukte jeg den for å løse styrkebergnegrer på en rammekonstruksjon.

Elementmetoden går ut på å dele konstruksjonen inn i elementer, jo flere elementer jo mer nøyaktige resultater får man. For hvert element blir stivheten til elementet kalkulert. Stivheten er basert på e-modulen, arealet og det andre arealmomentet til elementet. Disse stivhetene blir så addert sammen for å finne systemets stivhet. Vi kan deretter bruke sammenhangen mellom stivhet(k), forskyvninger(v) og krefter(S) (1) til å kalkulere knutepunktenes forskyvninger.

$$S = k * v \tag{1}$$

2.1.1 Lokal stivhetsmatrise

For å regne ut den lokale stivheten til et element kan en bruke stivhetsmatrisen (2).

$$k = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} & 0 & \frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} & 0 & \frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \tag{2}$$

Denne stivhetsmatrisen vil nå gjelde for lokalt for det enkelte elementet. Vi må derfor transformere matrisen slik at den gjelder globalt. Dette kan vi gjøre med en enkel rotasjonsmatrise (3).

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\theta & -\sin\theta & 0 \\ 0 & 0 & 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Vi kan enkelt transformere kreftene S og forskyvningene v ved sammenhengene:

$$S_G = RS_L \text{ og } v_G = Rv_L \tag{4}$$

Vi kan nå ved hjelp av 1 og 4 finne en sammenheng mellom den globale $_G$ og den lokale $_L$ stivhetsmatrisen.

$$S_G = RS_L = Rk_Lv_L = Rk_LR^{-1}v_G = k_Gv_G \tag{5}$$

$$k_G = Rk_LR^{-1} \tag{6}$$

2.1.2 Global stivhetsmatrise

For å addere alle elementstivhetsmatrisene til en global stivhetsmatrise er det normalt å bruke en IEG-matrise. IEG-matrisen er en relasjon mellom elementene og det tilhørende nodeparret. Den globale stivhetsmatrisen holder på stivhetene til alle nodene i systemet, mens de enkelte elementstivhetsmatrise inneholder kun stivheten til sine to tilhørende noder. Når de skal adderes skal stivhetene til elementstivhetsmatrisene adderes inn i de tilhørende nodene i den globale stivhetsmatrisen.

2.1.3 Grensebetingelser

Nå står vi bare igjen med et lineært ligningssystem av typen $kv = S$, hvor forskyvningene (v) er ukjente. Dette likningssystemet vil nå være uløselig, fordi stivhetsmatrisen k er singulær. Denne singulariteten skyldes at vi enda ikke har sakt hvor konstruksjonen er fastbundet. For et system med 3 frihetsgrader per node trenger vi 3 fastbuende frihetsgrader. Når vi har innført disse grensebetingelsene vil systemet være løselig.

2.1.4 Elementkrefter

Når vi vil regne ut krefter på de forskjellige elementene kan vi igjen bruke ligning 1. Vi må nå huske at vi må transformere matrisene til det koordinatsystemet vi ønsker.

2.2 Ligningsløser

' Når man skal modellere mange fysiske og interiørmessigere problemer kan en nesten ikke unngå å løse store linear ligningssystem. Det finnes derfor mange forskjellige metoder å løse disse på og alle har visse fordeler og ulemper. Det finnes to kategorier av ligningsløsere: dirkete og iterative. De direkte løser settet eksakt, men bruker ofte lenger tid en en iterativ løser. Gauss metoden er et eksempel på en direkteløser. En iterativ løser finner en approksimasjon av løsningen, og kan ha problemer med konvergens i visse tilfeller.

2.2.1 Gauss

Gauss metoden består av to deler. Den ene er eliminasjon, hvor du utfører elementære rad operasjoner til du for en triangulær form. Deretter gjøres en tilbakesubstitusjon for å finne x-verdiene. De finnes i hovedsak 3 forskjellige rad operasjoner:

1. Multiplisere en rad med et tall som ikke er null.
2. Addere to rader
3. Bytte om to rader

Kjøretiden til gauss kan kalkuleres ut i fra antall iterasjoner som gjøres. Om vi antar det er n likninger som skal løses kan kjøretiden kalkuleres som vist i (7).

$$E(n) = \sum_{k=1}^{n-1}(n-k) + 2\sum_{k=1}^{n-1}(n-k)(n-k-1) = O(n^3) \tag{7}$$

På samme måte kan vi kalkulere kjøretiden til tilbakesubstitusjonen:

$$T(n) = 2\sum_{i=1}^{n}(n-i) + n = O(n^2) \tag{8}$$

2.2.2 Gauss-Seidel

Gauss-seidel er en iterativ ligningsløser, som fungerer best på spinkle matriser. Den har en iterativ del, beskrevet i likning **??**. I hver iterasjon bruker Gauss-Seidel den forrige utregnede verdien, som gjør at den ikke kan kjøre parallelt.

$$x_j^{m+1} = \frac{1}{a_{jj}}(b_j - \sum_{k=1}^{j-1} a_{jk}x_k^{m+1} - \sum_{k=j+1}^{n} a_{jk}x^m) \tag{9}$$

Her kan vi ikke finne kjøretiden på samme måte som ved en direkteløser, siden vi hele tiden kan forbedre resultatet ved å kjøre en iterasjon til. Vi må derfor se på konvergens, parallellitet og hvilken nøyaktighet man skal ha. Det finnes noen enkle konvergeringsregler for Gauss-Seidel, som sier at metoden garantert konvergerer om:

1. Matrisen (A) er strengt eller uavvendelige diagonalt dominerende.
2. Matrisen (A) er positive bestemt (Positive-definite)

file:///home/simenhg/invece

2.3 OpenGL

Opengl er et API for å rendre 2d og 3d grafikk. Mange av kallene går rett på GPU'en, slik at vi får økt ytelsen. Strukturen er bygget opp som en tilstandsmaskin, hvor vi hele tiden forandrer tilstanden til programmet. Vi kan for eksempel sette hvilken farge eller linjetykkelse som skal brukes videre, ved kommandoene glColor3f() og glLineWith().

OpenGL opererer med en stakk av 3 matriser, hvor do på hver matrise kan utføre rotasjoner, forskyvninger og skaleringer. Hver gang en matrise operasjon blir kalt blir den utført på den gjeldene matrisen. De forskjellige operasjonene er standard matrise operasjoner, hvor matrisen blir multiplisert med en operasjons matrise.

$$glRotate(\theta, x, y, z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$glScale(x, y, z) = \begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

$$glTranslate(x, y, z) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

For å lage og dytte ut matriser på stakken brukes glPushMatrix() og glPopMatrix(). Når du har hentet ut en matrise kan du utføre rutiner på denne. Modellmatrisen er den mest essensielle matrisen, her kan du utføre rutiner som å tegne streker, polygoner og lignende.

3 Programmet

Dette programmet er bygget opp av to deler. Det første er en beregnende del, mens den andre er en visualiserende del. Beregningsdelen er skrevet i Fortran 90 som ut i fra en inputfil utfører en elementanalyse. Resultatene av elementanalysen blir så sendt videre til den visualiserende delen. Denne delen er skrevet i c og viser frem resultatene.

3.1 Bruk

3.1.1 Installasjon

3.1.2 Input filen

Input filen beskriver systemet som skal beregnes. Her skal det spesifiseres hvilke noder, elementer og laster som skal være med i kalkulasjonen. Hvilken benevning som brukes er valgfritt så lenge den er konsekvent. Input filen blir lest igjennom standard input enheten. Figur 3.1.2 viser et eksempel på en input fil.

```
6,6,1                   ! AntallNoder, AntallElementer, AntallKrefter
0,0,0,0,0               ! Liste med noder, på format:
0,9000,1,1,1            ! x-verdi, y-verdi, frihet i x-retning, frihet i y-retning, rotasjonsfrihet
6000,9000,1,1,1
9000,9000,1,1,1
9000,6000,1,1,1
9000,0,0,0,0,0
200E3,6500,80E6,2,3     ! Liste med elementer, på format:
200E3,6500,80E6,3,4     ! E-modul,Areal, I, node1, node2
200E3,6500,80E6,3,5
200E3,6500,80E6,4,5
200E3,6500,80E6,5,6
3,2,-40000              ! Liste med krefter,på format:
                        ! NodeNummer, Hvilken frihetsgrad? (x=1,y=2,r=3), Value
```

Figure 1: Eksempel på en input fil

3.2 Fortran

Delen som er skrevet i fortran implementerer en enkel styrkeberegning basert på elementmetoden.

3.2.1 Struktur

3.2.2 Program-analyse

3.2.3 Optimaliseringer

3.2.4 Tidsbruk

3.2.5 Resultater

3.3 C

3.3.1 Struktur

3.3.2 Optimaliseringer

3.3.3 Tidsbruk

3.3.4 Visualisering

3.4 Fargefunksjon

En fargefunksjon er en funksjon genererer en farge basert på en verdi. Den kan for eksempel returene en rgb verdi. Fargemodellen rgb baserer seg på og blande, blå, grønn og rød, til ønsket farge.

4 Diskusjon

hvorfor jeg valte ting: gauss, innpt stdin

5 Konklusjon

lærte nytt tankemønster ved å programere funksjonelt

References

[1] Wikipedia, http://en.wikipedia.org/wiki/Finite_element_method

[2] Åge Ø. Waløen, 1995, Dimensjonering ved hjelp av elementmetoden, NTNU

[3] Dyryl L. Logen, 2011, A First Course in the Finite Element Method

[4] Erwin Kreyszig, 2006, Advanced Engineering Mathematics, 9th Edition

[5] Kolbein Bell, 2007, Programutvikling (Teknisk, prosedyreorientert programmering), NTNU

[6] Kolbein Bell, 2006, Fortran 90, NTNU