Prosjekt 2: Jacobirotasjon

Henrik Modahl Breitenstein and Carl Petter Duedahl
(Dated: September 26, 2021)

PROBLEM 1

Vi har

$$\gamma\frac{d^2u(x)}{(dx)^2} = -Fu(x)$$

og skal vise at ved skalering blir dette

$$\frac{d^2u(\hat{x})}{(d\hat{x})^2} = -\lambda u(\hat{x})$$

hvor $\hat{x} = \frac{1}{L}$ og $\lambda = \frac{FL^2}{\gamma}$.
Vi starter med å se at

$$\frac{1}{dx} = \frac{d\hat{x}}{dx}\frac{d}{d\hat{x}} = \frac{d(\frac{x}{L})}{dx}\frac{d}{d\hat{x}} = \frac{1}{L}\frac{d}{d\hat{x}}$$

Så da får vi at

$$\frac{d^2u(x)}{dx^2} = \frac{1}{L^2}\frac{d^2u(\hat{x})}{d\hat{x}^2}$$

som gir oss

$$\frac{\gamma}{L^2}\frac{d^2u(\hat{x})}{d\hat{x}^2} = -Fu(\hat{x})$$

så flytter vi over og får

$$\frac{d^2u(\hat{x})}{d\hat{x}^2} = -\frac{L^2F}{\gamma}u(\hat{x})$$

så setter vi inn λ og får:

$$\frac{d^2u(\hat{x})}{d\hat{x}^2} = -\lambda\gamma u(\hat{x})$$

som vi skulle vise. □

PROBLEM 2

Vi vet at $UU^T = UU^{-1} = I$ og at $v_jv_i = \delta_{ji}$. Vi skal så vise at for

$$w_j^T w_i = \delta_{ji}$$

for å vise at U tar var på ortonormaliteten til v_i under multiplikasjon.
Vi starter først med

$$w_j = Uv_j$$

og transponerer denne:

$$w_j^T = (Uv_j)^T = v_j^TU^T = v_j^TU^{-1}$$

så tar vi

$$w_j^Tw_i = v_j^TU^{-1}Uv_i = v_j^TIv_i = v_j^Tv_i = \delta_{ji}$$

som vi skulle vise. □

PROBLEM 3

Koden kan finnes i som prob3.cpp.
Vi konstruerer de analytiske egenverdiene som

$$\lambda_i = d + 2 * a\cos\left(\frac{i\pi}{N+1}\right)$$

og egenvektorene i en matrise som

$$v_i = \begin{bmatrix}\sin\left(\frac{i\pi}{N+1}\right), \sin\left(\frac{i\pi}{N+1}\right), (\cdots), \\ \sin\left(\frac{ji\pi}{N+1}\right), (\cdots)\sin\left(\frac{Ni\pi}{N+1}\right)\end{bmatrix}^T$$

og konstruerer A som den tridiagonale matrisen og bruker arma::eig_sym til for å finne egenverdiene for å sammenligne med de analytiske verdiene. normalise funksjonen normaliserer egenvektorene også så vi må også normalisere de analytiske egenvektorene og sammenlikner vi nå ser vi at de armadillos egenvektorer og de analytiske egenvektorene stemmer.

PROBLEM 4

Problem a

Skriver funkjsonen inn i "Project2func.cpp"

Problem b

Skriver progremmet "LargestOffDiagTest.cpp" som kan kjøres ved kommandoen
$make LargestOD

PROBLEM 5

Vi lagde funskjonene i Project2func.cpp og brukte dette sammen med Porject2.cpp til å finne egenverdiene og de tilhørende egenvektorene for A som en 6×6 matrise. Vi sammmenliknet også med de analytiske verdiene vi fikk fra oppgave 3 og de stemte.

PROBLEM 6

Problem a

Vi kjører programet for $n = 7$ til $n = 100$. Vi får da plottet i 1.

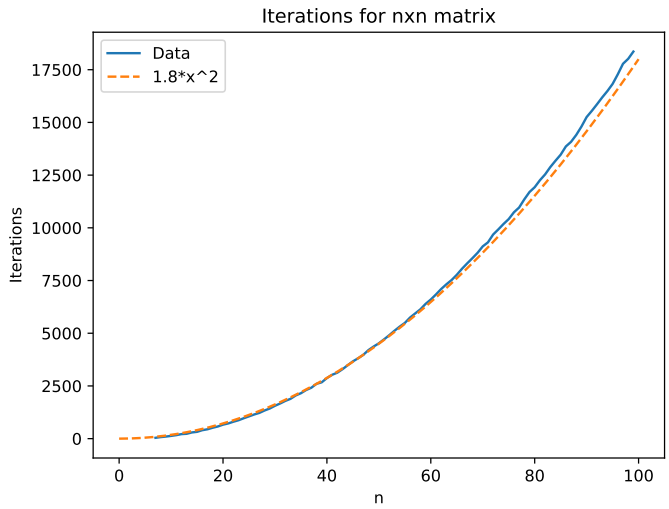

Figure 1. "Antallet iterasjoner for forskjellige n. Har testet oss frem til en analytsik funksjon som passer bra til i området."

Tenker at algoritmen er så treg fordi når man roterer ut et annet element så kan det hende at et annet element øker. Med mange elementer, selv om de er null, så vil det skje ganske ofte.

Problem b

Vi ser at for en tridiagonal matrise så øker antall rotasjoner med ca. $1.8N^2$. Da er utgangspunktet bare $N-4$ elementer som må roteres ut, som vil si at antall rotasjoner er proporsjonal med r^2 hvor r er antall elementer som må roteres ut. I en tett matrise så har vi N^2-N elementer å rotere ut, så om det følger samme system så vil antall rotasjoner være proporsjonal med N^4.

PROBLEM 7

a

Vi brukte igjen problem7.cpp og Project2func.cpp til å finne egenverdier og egenvektorene slik som i oppgave 5, men denne gangen med $N = 10$ og $n = 11$. Så brukte skrev vi inn disse egenverdiene i eigenvecs7.txt og leste dem av i Python for å plotte dem i . Vi ser at grafene ikke er så jevne som kommer av at vi bare brukte 12 punkter. Plotter vi før høyere n og N får vi mer nøyaktige grafer.

b

Vi trengte bare å endre fra $n = 11$ til $n = 101$ i problem7.cpp for å finne løse denne. Vi skrev også datane over i eignevecs7n100.txt, og plottet disse sammen med grafene i a-oppgaven og fikk Figur . Vi ser her at grafene er blitt mye jevnere siden vi har flere punkter og mer nøyaktige verdier, men vi kan fortsatt se de har samme form. Mode 2 for $n = 101$ er negativ av $n = 11$, men dette er fordi den negative versjonen av en egenvektor er fortsatt en egenvektor så dette gjør ikke en av grafene mer eller mindre korrekt.

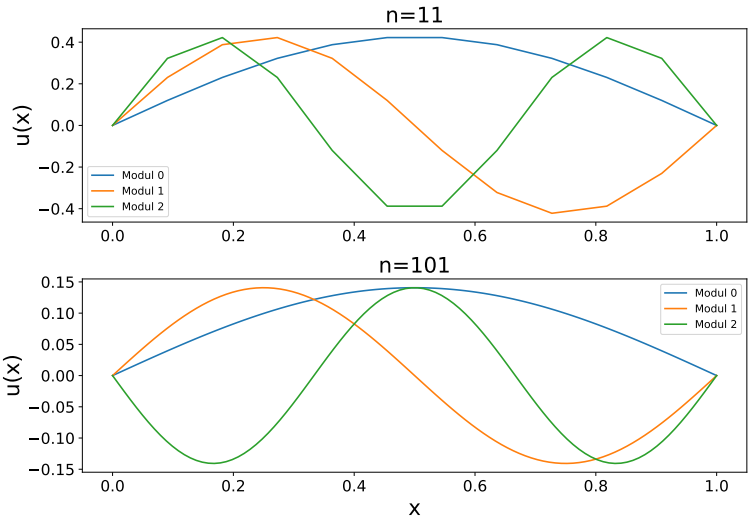

Figure 2. Problem 7 sine grafer av de tre laveste egenvektorene og ytterpunktene for $N = 10$ og $N = 100$. Vi ser at grafene blir jevnere for $N = 100$ enn for $N = 10$.