TMA4101 Matematikk 1 for MTELSYS

Noen iterative metoder

I denne øvingen skal vi ta et kvantesprang i forhold til hvordan du har løst matematikkoppgaver til nå. Datamaskinen din kan regne ut egenverdier og egenvektorene til en ganske svær matrise på null komma svisj, mye fortere enn du noensinne kommer til å greie det for hånd. Kommandoen np.linalg.eig er for komplisert til at vi kan analysere den, men vi kan se på en enkel numerisk metode for å finne den største egenverdien.

Det første vi er nødt til å gjøre, er å huske hvordan man normaliserer en vektor.

1 La v være en tilfeldig vektor. Beregn $\|\mathrm{v}\|$. Hva er dette for noe?

Forrige oppgave lærte du å løse på videregående skole. Den neste oppgaven husker jeg ikke om vi lærte på skolen.

2 Beregn $\frac{\mathrm{v}}{\|\mathrm{v}\|}$. Hvor lang er denne vektoren?
(Hvis du er usikker, kan du beregne $\left\|\frac{\mathrm{v}}{\|\mathrm{v}\|}\right\|$, altså lengden til $\frac{\mathrm{v}}{\|\mathrm{v}\|}$.

I for å regne ut lengden av en vektor b i numpy, skriver du bare np.linalg.norm(b). Nå skal vi se på en metode som kalles potensmetoden (power method på engelsk).

3 La

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

og v_0 være en tilfeldig vektor. Beregn $\mathrm{v}_1 = A\mathrm{v}_0$ i Python, og normaliser v_1. Fortsett flere ganger, altså kjør på med rekursjonen

$$\mathrm{v}_{n+1} = \frac{A\mathrm{v}_n}{\|A\mathrm{v}_n\|}$$

til du synes det er greit.

4 Forhåpentligvis husker du tall godt nok til at du kjente igjen noe fra tidligere i uken i forrige oppgave. Beskriv til en medstudent hva som fikk deg til å stoppe iterasjonen. Kjør debatt.

Dersom svaret ditt i forrige oppgave var noe sånt som at du fortsatte til du ikke så noen endringer i desimalene, har du begynt å fikle med noe som kalles 'cauchyfølge'. En suksessrik numerisk løsning av et matematisk er som oftest en cauchyfølge. Elementene i følgen er bedre og bedre tilnærminger til den korrekte løsningen.

Nå skal vi prøve en teknikk for å løse et system av likninger. Kommandoen np.linalg.solve gjør jo dette for oss om vi vil, men vi skal se litt på en matematisk metode som av og til funker.

5 La

$$A = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}$$

og

$$\mathsf{b} = \begin{bmatrix} 4 \\ 5 \\ 3 \end{bmatrix}.$$

Vi skal løse likningssystemet $A\mathsf{x} = \mathsf{b}$ med en iterasjonsteknikk som kalles Jacobis iterasjon. La

$$D = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

altså diagonalen til A. La v_0 være en tilfeldig startvektor, og sett igang iterasjonen

$$\mathsf{v}_{n+1} = D^{-1}\left(\mathsf{b} - (A - D)\mathsf{v}_n\right).$$

Konvergerer iterasjonen mot korrekt løsning?

6 Dersom du ikke har gjort det ennå, kan du google 'for loop python', og lage et script som kjører en for-løkke. Da kan du med et tastetrykk kjører ti tusen iterasjoner.

Dagens moro

7 La

$$A = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

og

$$\mathsf{b} = \begin{bmatrix} 4 \\ 5 \\ 3 \end{bmatrix}.$$

Vi skal løse likningssystemet $A\mathsf{x} = \mathsf{b}$ med en iterasjon. La v_0 være en tilfeldig startvektor, og sett igang iterasjonen

$$\mathsf{v}_{n+1} = \mathsf{v}_n + 0.01(\mathsf{b} - A\mathsf{v}_n)$$

Denne iterasjonen kalles Richardsoniterasjon. Vi kommer tilbake til den om et års tid.