

SØLVE SELSTØ

Praktisk introduksjon til numeriske metodar

Øvingsoppgåver for MATLAB

UNIVERSITETSFORLAGET

Praktisk introduksjon til numeriske metodar

Sølve Selstø

Praktisk introduksjon til numeriske metodar

Øvingsoppgåver for MATLAB®

Universitetsforlaget

© Universitetsforlaget, Oslo, 2019
ISBN: 9788215039657
1. utgave, 1. versjon, Universitetsforlaget, 2019
E-utgaven baserer seg på 1. trykte utgave, Universitetsforlaget 2017

Omslagsdesign: Cecilie Mohr
Sats: Sølve Selstø

Materialet i denne publikasjonen er omfatta av reglane i åndsverklova. Utan særskild avtale med rettshavarar er det berre tillate å framstilla eksemplar og gjera innhaldet i boka tilgjengeleg dersom det er heimla i lova eller tillate gjennom avtale med Kopinor, interesseorgan for rettshavarar til åndsverk. Utnytting i strid med lov eller avtale kan føra til erstatningsansvar og inndraging og kan bli straffa med bøter eller fengsel.

www.universitetsforlaget.no
post@universitetsforlaget.no

Til Anne Birte, Halvard og Ingeborg

Denne boka har blitt til i samspel med flotte ingeniørstudentar ved Høgskolen i Oslo og Akershus og med dyktige og hyggelege kollegaer ved Allmenn-seksjonen ved Institutt for informasjonsteknologi. Tusen takk!

Innhald

Introduksjon **11**

1 Å komme i gang **15**
1.1 MATLAB som kalkulator 15
1.2 Tilordning 17
1.3 Litt om variabelnamn 18
1.4 Løysingsforslag 19

2 Om indeksering og om komplekse tal **27**
2.1 Vektorar 28
2.2 Matriser 29
2.3 Komplekse tal 30
2.4 Løysingsforslag 31

3 Funksjonar og plotting **41**
3.1 Å lage eit plott 41
3.2 Å lagre eit plott 42
3.3 Fleire grafar samtidig 44
3.4 Flo og fjære 45
3.5 Løysingsforslag 47

4 Matriser **53**
4.1 Redusert trappeform og løysing av lineære likningssystem 54
4.2 Med og utan punktum 54
4.3 Nokre spesielle matriser 54

4.4 Invertering 56
4.5 Testing av "rekneregla" 56
4.6 Eigenverdiar og -vektorar 58
4.7 Ekstraoppgåve: Rekkeoperasjonar 59
4.8 Løysingsforslag 59

5 Skript og funksjonar 77
5.1 Kva gjer desse skripta? 77
5.2 Logiske variablar, `if`-satsar 79
5.3 Litt meir om `if`-satsar 80
5.4 Funksjonar i MATLAB 81
5.5 Funksjonsfiler 83
5.6 Litt om forskjellen på funksjonsfiler og skript 84
5.7 Løysingsforslag 84

6 `for`-løkker 95
6.1 Summar og `for`-løkker 95
6.2 Konvergens? 97
6.3 Halveringsmetoden 97
6.4 Ekstraoppgåve: Fakultetsfunksjonen 100
6.5 Ekstraoppgåve: Maksimering 100
6.6 Løysingsforslag 100

7 Å løyse likningar 115
7.1 Halveringsmetoden – igjen 115
7.2 Newtons metode 116
7.3 Fikspunktiterasjon 117
7.4 Ekstraoppgåve: Bursdag same dag 118
7.5 Ekstraoppgåve: Tredjegradslikninga 118
7.6 Løysingsforslag 120

8 Numerisk derivasjon 137
8.1 Endeleg differanse 137
8.2 Den deriverte frå ein tabell 139
8.3 Derivasjon som lineær transformasjon 140

8.4 Ekstraoppgåve: Gammafunksjonen vs. eksponentialfunksjonen . 141
8.5 Ekstraoppgåve: Å bestemme feilen 142
8.6 Løysingsforslag 142

9 Numerisk integrasjon **157**
9.1 Mange rektangel (og nokre trapes) 157
9.2 Med fasit 160
9.3 Utan fasit 161
9.4 Enda ein lineær transformasjon 162
9.5 Ekstraoppgåve: Plott feilen 162
9.6 Ekstraoppgåve: Monte Carlo-integrasjon 163
9.7 Løysingsforslag 164

10 Eulers metode **185**
10.1 Med fasit 185
10.2 Utan fasit 187
10.3 Ekstraoppgåve: Litt optimering 188
10.4 Ekstraoppgåve: Eit konvergensbevis 190
10.5 Løysingsforslag 191

11 Intermesso **199**
11.1 Kort oppsummering 199
11.2 Introduksjon til oppgåvekapitla 200

12 Svingelikninga **203**
12.1 Frå andre til første orden 203
12.2 Den rette svingelikninga for ein pendel 204
12.3 Løysingsforslag 206

13 Normalfordeling **213**
13.1 Løysingsforslag 214

14 Svampen **221**
14.1 Løysingsforslag 223

15 Temperaturfordeling **229**

15.1 Løysingsforslag 231

16 Endeleg differanse og differensiallikning 237
16.1 1. orden 237
16.2 2. orden med randkrav 239
16.3 2. orden med startkrav 241
16.4 Løysingsforslag 241

17 Redusert trappeform 255
17.1 Løysingsforslag 256

18 Å krysse ei elv 265
18.1 Løysingsforslag 267

Tillegg 275

A Numerikk-ordliste 275

B Liste over MATLAB-funksjonar 281

Introduksjon

Sommaren 2006 skulle eg, som meir eller mindre ung forskar, begynne å jobbe med ein dyktig og anerkjent professor innan atomfysikk. Vi diskuterte kva problemstilling som kunne vere interessant for oss å ta fatt på. Då vi kom fram til eit prosjekt som vi begge syntest høyrtest spennande ut, såg eg for meg fleire veker med lesing av teori, løysing av øvingsoppgåver og generell grubling. Eg måtte jo forstå problemet fullt ut før eg skulle sette meg ned med datamaskina for å skrive eit program som til sjuande og sist skulle løyse det, tenkte eg. Eg vart derfor ganske paff då professoren sa "kan du ikkje berre begynne å implementere det, då?". Korleis kunne eg skrive eit dataprogram for å rekne ut noko eg ikkje fullt ut hadde forstått? Men eg tok henne på ordet og sette meg ned ved datamaskina. Og etter kvart forstod eg kva ho meinte. Då eg ikkje heilt såg vegen vidare, gjekk eg til litteraturen og fann ut av akkurat det eg trengde for å programmere vidare. Og denne kodeskrivinga var definitivt med på kaste lys over teorien – det var ikkje berre teoriforståinga som dikterte kodinga. Dette gjekk begge vegar. Og hadde eg ikkje erfart tidlegare at innsikter eg trudde eg hadde komme fram til, måtte justerast då eg skulle implementere dei?

Dei fleste lærebøker i matematikk er nok lagt opp på den måten at ein først går gjennom ein del teori – med eksempel – og så skal studenten sjølv bruke teorien på konkrete oppgåver etterpå. Her har vi prøvd å gjere ting på ein litt annan måte. Her blir du sett til å gjere oppgåver med ein gong. Innimellom er der drypp av teori. Men framfor alt håpar vi at teorien skal bli til ei *erfaring* frå første stund. Litt "flåsete" kan vi seie at det som står i denne boka, skal *gjerast*, ikkje *lesast*.

Håpet er at alle skal klare å gjere desse øvingane – også dei som aldri har koda ei linje før eller aldri har brukt datamaskinar til noko særleg anna enn nettsurfing og tekstbehandling. Vi forsøker å forklare ting så grundig vi klarar. Derfor vil nok enkelte oppleve at det heile kan bli litt "teksttungt" og omstendeleg. Dette gjeld ikkje minst i løysingsforslaga. Men det er viktig å vere klar over at sjølv om boka er ganskje tjukk, er det ikkje nødvendigvis veldig mykje å gjere.

På den andre sida: Om du synest at oppgåvene er i overkant detaljstyrande, kjenn

deg fri til å løyse oppgåvene på din eigen måte. Og dersom du skulle bli nyfiken på ting som kanskje ligg litt på sida av det oppgåva legg opp til, så ikkje ver redd for å gå ein liten omveg!

Som tittelen tyder på, er oppgåvene knytt til programvara MATLAB. Ein kan kanskje meine at dette er uheldig då denne programvara kostar pengar. Til det er det å seie at det finst ein heil del "kloningar"[1] som er gratis. Så godt som alt vi gjer her kan gjerast på same måte i for eksempel Octave (UPM), SciLab eller Freemat. Overgongen til andre gratis verktøy som for eksempel Python eller Julia burde heller ikkje vere så stor. Og då er vi framme ved eit anna poeng: Ein kan meine at boka ville hatt større nytteverdi om kode og implementering hadde vore gjort meir generisk – alstå formulert på litt meir generelle måtar. Men no er det slik at MATLAB-kode alt er ganske intuitiv. Om vi skulle ha brukt ein slags pseudokode i staden, ville nok denne ha likna ganske mykje på MATLAB-kode uansett. Dermed er det vel like greit å berre skrive MATLAB-kode; "oversettinga" til andre språk, som nemnde Python, eller Java, C(++) eller Fortran blir uansett høgst overkommeleg for dei som alt er komfortable med desse språka.

Så godt som alle velkjente numeriske metodar har ein eller annan ferdig funksjon i MATLAB som kan brukast ganske direkte. MATLAB kan for eksempel finne integral numerisk og løysingar av likningar ved berre å gi éin kommando. Vi skal i liten grad bruke slike ferdiglaga løysingar. Dette kan kanskje sjå ut som ein effektiv måte for å gjere ting meir kompliserte enn dei treng å vere. Men poenget med dette kurset er jo at vi skal *lære oss numeriske metodar*, ikkje lære oss korleis vi skal få MATLAB til å gjere dei for oss på enklast mogleg måte. Eit slikt opplegg ville dessutan ha ført med seg ein del kjedeleg pugging. Når vi vel har lært desse metodane – og kva dei er sterke og kva dei er svake på – vil det vere meir naturleg å bruke slike ferdige pakkar. Men her, i utgangspunktet, vil vi bygge opp metodane ved hjelp av dei enklaste bitane. På den måten vil også det vi treng å lære av kodetekniske ting bli halde på eit minimum.

Når vi skal implementere dei aktuelle numeriske metodane, vil vi forsøke å skrive kodane så enkelt og gjennomsiktig som vi klarar. Dette kan nok til tider gå ut over effektiviteten; om vi opplever at det skulle vere konflikt mellom å gjere ein kode pedagogisk og å gjere han optimal, vil vi typisk velge å gjere det første. Derfor er kanskje ikkje denne boka heilt *optimal* for dei som har tenkt å bli veldig gode programmerarar.

Boka er lagt opp på den måten at dei første kapitla fokuserer på eitt avgrensa tema om gongen. Dersom du blir ståande fast på nokre av oppgåvene, skuldast

[1] Med "kloningar" meiner vi alternative programvarer som etterliknar MATLAB sin måte å fungere på.

dette mest sannsynleg at oppgåva ikkje er tydeleg nok formulert – eller at du har tatt litt for lett på eitt av dei tidlegare kapitla. Om du følger progresjonen, skal det ikkje vere nødvendig å gruble lenge for å få ting til. Rettnok har visse av kapitla nokre ekstraoppgåver der spesielt interesserte får utfordra seg litt. Men dei verkelege utfordringane kjem i den siste delen av boka. Her blir det gitt ganske krevjande oppgåver der du vil få bruk for alt det du har lært i dei første kapitla. Det er ikkje utan vidare opplagd kva metodar ein skal ta i bruk i kvar av oppgåvene, og ofte må ein kombinere fleire av dei – i tillegg til å ta i bruk analytiske ("papir og blyant"-)metodar. Og det er akkurat slik vi som jobbar med kvantitative metodar til dagleg, held på; vi vekslar mellom notatblokk og datamaskin heile tida. (I alle fall mange av oss.)

Boka er rik på kodesnuttar. Om du har lyst til å bruke nokon av desse, skal du sleppe å skrive dei av. Alle dei aktuelle filene kan du finne på boka si nettside på Universitetsforlaget.no.

Vi håpar at du etter å ha jobba deg gjennom denne boka sit att med ei offensiv halding til kvantitativ problemløysing: "Dersom eg klarar å bruke matematikk til å *formulere* problemet, skal eg også klare å *løyse* det." Og kanskje opplever også du, som eg, at kodinga kan hjelpe på både forståinga og formuleringa av problemet – ikkje berre motsett.

Kapittel 1

Å komme i gang

Vi skal ikkje gjere nokon avanserte ting i dette kapitlet – berre bli kjent med korleis MATLAB ser ut og korleis vi kan bruke det som ein kalkulator. Sjølvsagt er MATLAB mykje meir enn ein avansert kalkulator. Men vi må jo begynne ein stad.

Vi skal også introdusere *variablar* og korleis ein *tilordnar* desse.

Aller først skal vi kikke litt på grensesnittet. Når du startar opp MATLAB, vil eit panel med fire vidaugo dukke opp:

- Mappevindauga, "*Current Folder*", som viser kva for ei mappe du jobber i.
- Kommandovindauga, "*Command Window*", kor ein utfører dei ulike kommandoane. Dette er det viktigste vindauga, og det er fullt mogleg å bruke MATLAB med dette vindauga aleine.
- Variabelvindauga, "*Workspace*", som viser kva variablar som ligg i minnet til datamaskina.
- Kommandohistorikken, "*Command History*" – ei liste som viser alle kommandoane du har utført i arbeidsøkta.

I dette kapitlet, og for så vidt i heile boka, skal vi stort sett berre prate om kommandovindauga.

1.1 MATLAB som kalkulator

Her skal vi kort forklare korleis ein utfører enkle aritmetiske operasjonar i MATLAB. Dette er definitivt å "skyte spurv med kanon" – MATLAB kan brukast til så mykje meir, men det må jo kunnast dette også.

a) Forsøk å skrive inn enkle aritmetiske reknestykke i kommandovindauga. Du kan for eksempel prøve å rekne ut

```
2+2
3-2
3*2
6^2
6/2
```

Kva betyr '`*`', '`^`' og '`/`' i denne samanhengen?

b) Rekn ut

$$\frac{3^{2.12} \cdot (2+4 \cdot (-3.1))}{2^2 - 1} \quad .$$

Gjer gjerne fleire – og mer avanserte – reknestykke. Merk at MATLAB, akkurat som dei fleste andre språk i verda, brukar punktum – ikkje komma – for desimalar.

Som du sikkert ser, er '`*`' symbolet som blir brukt for multiplikasjon, '`/`' blir brukt for divisjon og '`^`' blir brukt for å skrive potensar.[1]

c) Kva blir rekna ut her?:

```
sin(0.7)
log(5)
sqrt(9)
exp(1)
atan(2)
```

d) Finn ut korleis MATLAB tolkar desse reknestykka:

```
1/0
0^0
5/Inf
0*Inf
Inf/Inf
10^999
1.2e4
NaN+1
```

[1] Hatten, '`^`', kan vere litt "krøkkete" å få fram på visse tastatur. Det hender at ein også må bruke mellomromstasten.

Er du einig i alle desse tolkingane; verker dei rimelege? '`Inf`' betyr uendeleg (∞) – *infinty*, og '`NaN`' betyr 'udefinert' – *Not a number*. Som du sikkert ser, betyr '1.2e4' $1.2 \cdot 10^4$ – altså eit tal skrive på *normalform*.

e) Nokre tal er så spesielle at dei har fått eigne namn – også i MATLAB. I kommandovindauga, skriv

```
pi
i
eps
```

'`eps`' refererer her til den greske bokstaven epsilon, ε, som ofte blir brukt for å symbolisere små tal. Dette er det minste talet x som er slik at maskina klarar å skilje mellom 1 og $1+x$. Vi kallar dette '*maskinpresisjonen*' og, som namnet tyder på, varierer denne presisjonen noko frå maskin til maskin. Vanlegvis er det eit tal av storleikssorden 10^{-16}. Dette er eit lite tal, men det er ikkje null. Som vi skal sjå seinare, har faktisk det at datamaskinar ikkje klarar å skilje veldig små tal frå null visse praktiske konsekvensar.

Motsett klarar heller ikkje MATLAB å skilje ekstremt store tal frå uendeleg.

i blir kalla 'den imaginære eininga' og kan oppfattast som $\sqrt{-1}$.[2]

1.2 Tilordning

a) Når vi skal rekne ut noko, har vi ofte bruk for å halde styr på fleire ulike storleikar samtidig. Som du sikkert har erfart ved vanleg kalkulatorbruk, blir utrekningar ofte mykje ryddigare om vi legg visse storleikar i *minnet* til maskina. Men då må storleiken ha eit namn – slik at vi kan hente han fram frå minnet når vi treng han. Vi kan *tilordne* talverdier til variabelnamn slik:

```
>> x=7
```

(Symbolet "»" har vi brukt for å indikere at dette blir skrive inn i kommandovindauga i MATLAB.) Vi har nå oppretta variabelen `x` og sagt at denne skal ha verdi 7. Dette "hugsar" MATLAB – det ligg i minnet. Som du sikkert ser, dukkar variabelen `x` no opp i variabelvindauga (*workspace*). Vi kan så bruke denne variabelen i andre utrekningar. Rekn ut følgande:

[2] I fall du stussar: Jo da, visst kan vi tillate oss å ta kvadratrota av negative tal.

```
>> x*2
>> x^2
```

Vi kan fortsette å tilordne fleire variablar:

```
>> y=2
>> x/y
```

Etter kvart som du tilordnar fleire variablar, vil dei også dukke opp i lista i variabelvindauga. Du kan også få opp ei liste over alle variablane dine i kommandovindauga ved å skrive '`who`'.

Når vi gjer ei utrekning utan å gi svaret noko namn, slik vi gjorde i oppgåve 1.1, vil svaret automatisk få namnet `ans` (for "*answer*").

b) Når vi brukar symbolet '`=`' som i den førre deloppgåva, betyr det slett ikkje "er lik", som i matematisk notasjon. Her er "er lik"-teiknet brukt som **tilordning**; vi putter talet 7 inn i variabelen `x`. Dette gir faktisk meining å skrive i MATLAB:

```
>> x=2
>> x=x+1
```

Kva får vi til svar om vi gjer denne operasjonen? Kvifor er den siste linja meiningslaus om vi skal tolke den ut frå vanleg matematikknotasjon?

c) Undersøk kva som skjer om du utfører dette i kommandovindauga:

```
>> pi
>> pi=10
```

Kva for ein verdi vil variabelen `pi` få?

Om du vil at variabelen `pi` igjen skal bli π, berre skriv '`clear pi`'. Berre '`clear`' utan noko meir reinskar heile minnet. Om du også vil reinske sjølve kommandovindauga, får du det til ved å skrive '`clc`'.

1.3 Litt om variabelnamn

I utgangspunktet kan vi gi variablar kva namn som helst; namnet kan godt bestå av fleire bokstavar. Men ein må hugse visse ting:

- Eit variabelnamn får ikkje begynne med eit tal.