TANA17 Matematiska beräkningar med MATLAB för M, DPU

Fredrik Berntsson, Linköpings Universitet

Föreläsning 8

- God programmeringsstil.
- Sammansatta datatyper: Poster. Cell-matriser.
- Tillämpning: En stack.
- Tentamensinformation.

Programmeringstips

Exempel Vi vill beräkna en approximation av derivatan $f'(2)$ då $f(x) = \sqrt{1+x}$ med formeln

$$f'(2) \approx \frac{f(2+h) - f(x-h)}{2h}, \quad h > 0.$$

Beräkna felet som funktion av h för $h = 1/n$, $n = 10, 20, \ldots, 100$. Plotta sedan resultatet med `loglog`.

Lösning Gör följande steg

1. Skapa en vektor `h` med alla h-värden. Skapa en vektor `Df` med nollor.
2. En `for`-loop där `Df(i)` beräknas som derivata approximationen för steglängd `h(i)`.
3. Beräkna en vektor med fel. Plotta i log-skala.

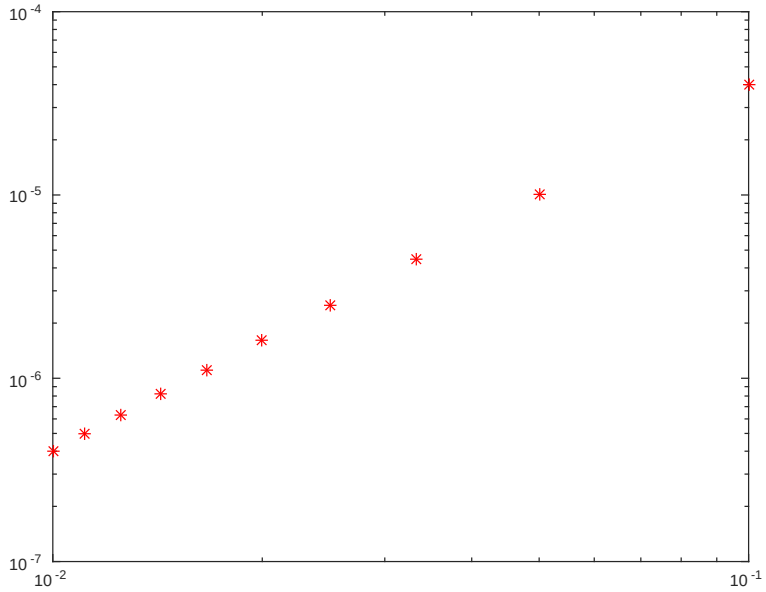
I Matlab skriver vi

```
n=10:10:100;  % Får n=( 10 20 30 .... 100 )
h=1./n;       % Får h=(1/10 1/20 .... 1/100 )
f = @(x) sqrt(1+x);
Df = zeros(size(h));

for i=1:length(h)
    % Beräkna Df(i) med steg h(i)
    Df(i)=( f(2+h(i)) - f(2-h(i)) )/2/h(i);
end
e=abs(Df - 1/2/sqrt(3) );
loglog( h , e ,'r*')
```

Kommentarer Vi väljer först `h` vektorn. I `for`-loopen använder vi ett värde h_i i taget. Undvik `for`-loopar där loop-variabeln inte kan ses som ett index.

Använd variabelnamn som är naturliga för uppgiften `h`, `Df`, etc.



Uttryck av typen $e(h) = Ch^p$ blir räta linjer då de plottas med `loglog`.

Tips För att få lättlästa program bör man vara konsekvent med hur variabler namnges

- Index variabler för loopar: i, j, k.
- Vektorlängd, Matrisstorlek, Antal: n, m, N, M
- Reella tal: t, x, y, z, u, v
- Matriser: A, B, C,...
- Vektorer: x, y, b,...
- Funktionsvärden: f, g, q, w,...

Undvik att använda l, o, O, 0 ty svåra att se skillnad på.

Långa förklarande namn blir tydliga men jobbiga att skriva.

Sammansatta Datatyper

Exempel En *bok* beskrivs av dess *Titel*, *Författaren*, och *Utgivningsåret*.

Vi vill skapa en funktion som presenterar denna information på ett trevligt sätt. Vi skriver en Matlab funktion

```
function []=DisplayBook( Title , Author, Year )
  fprintf(1,'The book %s by %s',Title,Author);
  fprintf(1,' was published %i.\n',Year);
end
```

Problem Då boken är ett objekt vore det mera praktiskt att kunna skriva en funktion

```
function []=DisplayBook( TheBook )
```

Där all information om boken samlats i en enda variabel.

I Matlab används kommandot `struct` för att skapa sammansatta datatyper. De olika delarna kallas `fält`.

Exempel I Matlab skriver vi

```
>> Book = struct('Title',[],'Author',[],'Year',[])
```

så skapas en tom *post* med tre fält. Vi kan lägga in värden med

```
>> Book.Title='Linjär algebra';
>> Book.Year=1990;
>> disp( Book )
  Title: 'Linjär algebra'
 Author: []
   Year: 1990
```

Detta är mycket användbart om flera variabler logiskt hör ihop.

Exempel Vi kan skapa en post med givna fält genom tilldelningar

```
>> Frukt.Sort = 'Äpple';
>> Frukt.Vikt = '49';
>> Frukt.Farg = 'Röd';    % ä får ej användas
>> disp( Frukt )
  Sort: 'Äpple'
  Vikt: '49'
  Farg: 'Röd'
```

Efter att en post skapats använder vi den genom att skrivas

```
>> x = Frukt.Vikt;
>> fprintf(1,'Frukten är %s.\n',Frukt.Farg)
Frukten är Röd.
```

Exempel Vi vill skapa ett Matlab program som hanterar *bankkonton*.

Ett bankkonto karakteriseras av ett `Kontonummer`, kontoinnehavarens *Namn*, och ett *Saldo*.

Vi vill kunna skapa nya konton, sätta in och ta ut pengar, samt dela ut ränta.

På filen `SkapaKonto.m` skriver vi

```
function [Konto]=SkapaKonto( Namn , Nummer );
      Konto.Saldo=0;
      Konto.Namn=Namn;
      Konto.Nummer=Nummer;
end
```

På filen `Uttag.m` skriver vi

```
function [Konto]=Uttag( Konto , Belopp );

      if Konto.Saldo < Belopp
         error('Ogiltigt uttag');
      else
         Konto.Saldo=Konto.Saldo-Belopp;
      end
end
```

Cell matriser

En `cell` matris har elemelent som kan vara av olika typ. En tom cell matris skapas med kommandot `cell`.

Exempel Vi skapar en vektor av `cell` typ och lägger in information i den.

```
>> C = cell( 1 , 3 );
>> C{1} = 'Fredrik';
>> C{2} = rand(2,2);
>> C{3} = 5.3;
>> C
C =
 'Fredrik'    [2x2 double]    [5.3000]
```

För indexering ersätts oftast `( )` med `{ }`. Skriver vi `C{1}` fås elementet på plats 1. Alltså strängen `'Fredrik'`.

Datastruktur - Stack

Ofta opraktiskt att tänka i termer av vektorer och matriser då man programmerar.

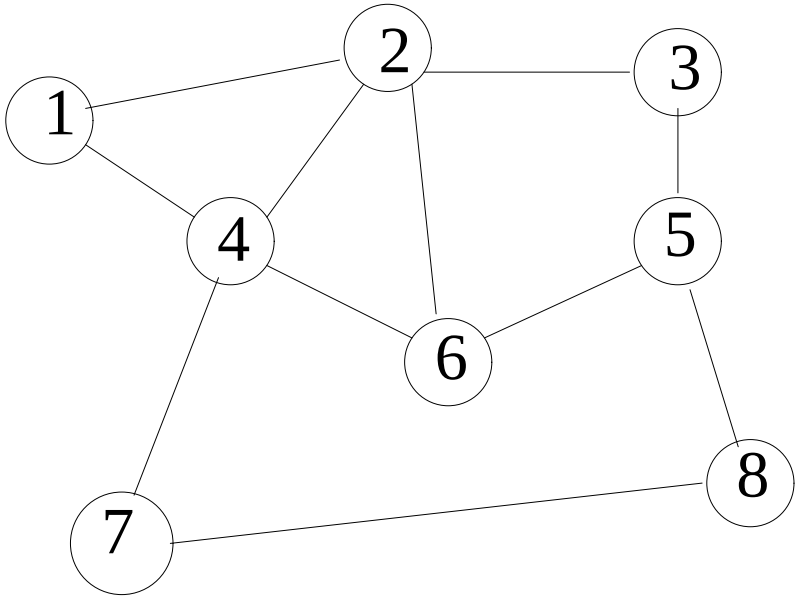
> **Definition** En *Stack* är en abstrakt datatyp som motsvarar en hög. Man kan lägga till nya element överst på högen och man kan ta bort det element som ligger överst.

Operationer som kan utföras är

- Skapa en tom stack: `>> S = CreateStack();`
- Lägg till ett element överst: `>> S = Push( S , E );`
- Ta bort ett element: `>>[S,E]=Pop( S );`
- Kontrollera stackens storlek: `>> n = Size( S );`

Vi behöver inte bry oss om hur dessa funktioner fungerar.

Exempel Vi vill flytta oss i en graf och komma ihåg färdvägen.



Vi skall stata i Nod 1 och försöka ta oss till Nod 8. Vi har en funktion

```
function [ NextNode ] = Move( Graph,CurrentNode )
```

Hur skall vi göra?

Lösning För varje flytt sparar vi en vektor med nuvarande och nästa nod i en stack i en stack.

På filen `GraphMovement.m` skriver vi

```
S = CreateStack();
CurrentNode = 1;
while CurrentNode ~= 8
     NextNode=Move( Graph , CurrentNode );
     S = Push( S , [CurrentNode,NextNode] );
     fprintf(1,'Move from %i to %i.\n',...
               CurrentNode,NextNode);
     CurrentNode=NextNode;
 end
```

Skriver vi `...` fortsätter ett Matlab kommando på nästa rad.

För att gå tillbaka samma väg skriver vi

```
while StackSize( S ) > 0
   [ Nodes,S]=Pop( S );
   fprintf(1,'Återvänd från %i till %i.\n',...
             Nodes(2),Nodes(1))
end
```

Vi behöver inte veta hur stacken är implementerad för att använda den.

Det gäller att tänka på rätt abstraktionsnivå!

Övning Implementera en stack som kan användas för att larga vilka typer av data som helst.

Följande funktioner skall finnas: `CreateStack`, `StackSize`, `Push`, `Pop`.

Hur skall stacken representeras? Hur ser en tom stack ut?

Stacken används bara genom funktionerna ovan.

Lösning Vi behöver lagringsutrymme för att spara de element som sparats i Stacken. Dessutom måste vi veta hur många element som finns där. Använd en `struct` med fält:

- `Elements` - `cell`-vektor där element sparas.
- `StackSize` - Antalet element sparade i stacken.
- `MaxSize` - Maximala utrymmet i `Elements`.

Frågor Hur ser en tom stack ut? Hur blir funktionen `CreateStack`?

I filen `CreateStack.m` skriver vi

```
function [Stack]=CreateStack( MaxSize )
    Stack.Elements=cell( MaxSize , 1 );
    Stack.StackSize=0;
    Stack.MaxSize=MaxSize;
end
```

Vi kan nu skriva

```
>> S = CreateStack( 100 )
 S =
   Elements: {100x1 cell}
  StackSize: 0
    MaxSize: 100
```

Fråga Hur skall stacken förändras om vi sparar ett element i den?

I filen `Push.m` skriver vi

```
function [ Stack ]=Push( Stack , E )
   Stack.StackSize=Stack.StackSize+1;
   Stack.Elements{ Stack.StackSize }=E;
end
```

I Matlab kan vi nu skriva

```
>> S=Push(S,'a');S=Push(S,'b');S=Push(S,'c')
 S =
 StackSize: 3
   MaxSize: 100
  Elements: {100x1 cell}
>> S.Elements(1:4)'
ans =
  'a'    'b'    'c'    []
```

På filen `Pop.m` skriver vi

```
function [Stack,E]=Pop( Stack )
   E=Stack.Elements{ Stack.StackSize };
   Stack.StackSize=Stack.StackSize-1;
end
```

I Matlab skriver vi

```
>> [S,E]=Pop( S )[
 S =
  StackSize: 2
    MaxSize: 100
   Elements: {100x1 cell}
E =
   c
```

Vi behöver inte radera elementet från `Elements`. Kontroller för tom eller full stack behövs.

Tentamensinformation

- Skriftlig tentamen i datorsal. Lös uppgiften i Matlab. Skriv av ditt program på papper då du är färdig. Inga utskrifter behövs.
- Salar är Boren/Hunn, Roxen/Glan och ISYs salar Asgård, Olympen, Egypten, etc. Anslås på kurshemsidan.
- Antingen 8 – 12 eller 14 – 18. Datorer räcker inte till alla.
- Ni använder särskilda inloggningar för tentan. Dessa delas ut av vakterna.

De två tentor som finns på hemsidan är relevanta att öva på. Förutom

- Funktionen `filter` ingår inte i årets version av kursen.
- Utskrifter med `fprintf` ingick inte förra året.

Uppgifterna består av kortare Matlab program (5-10 rader). Träna på lektionsuppgifterna.

Lycka till!