Datorlaboration för kursen ML1302, Datorbaserade designverktyg

Obligatoriska uppgifter

1. Om x ligger nära noll så ligger följande summa nära funktionen cosh(x):

$$S = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \frac{x^8}{8!}$$

Denna summa (med fem termer) skulle också kunna skrivas som

$$S = \sum_{n=0}^{4} \frac{x^{2n}}{(2n)!}$$

Man kommer närmare cosh(x) om man använder fler termer. Allmänt kan man skriva:

$$S = \sum_{n=0}^{N} \frac{x^{2n}}{(2n)!}$$

För ovanstående summa blir antalet termer $N+1$

UPPGIFTEN
Skriv en funktion **`fun(x, N)`** som beräknar summan med $N+1$ termer.

Funktionen skall för varje term skriva ut en rad som anger värdet på n, aktuell term, och delsummans storlek.

Funktionen skall också, som jämförelse, skriva ut värdet av cosh(x) för det aktuella x-värdet.

Dialog med funktionen skall till exempel kunna se ut så här:

```
>> fun(2, 4)
    ----------------------------------
     n      term_n          summa_n
   ----------------------------------
     0     1.000000000     1.000000000
     1     2.000000000     3.000000000
     2     0.666666667     3.666666667
     3     0.088888889     3.755555556
     4     0.006349206     3.761904762
    ----------------------------------
cosh(2.000000000) = 3.762195691

ans =

    3.7619
```

I fallet ovan ser vi att summan, med tre siffrors noggrannhet, överensstämde med värdet på cosh(x).

Funktionen får inte använda sig av **`factorial`** eller någon annan funktion för att beräkna fakultet.

2. Pythagoras sats anger som bekant sambandet $a^2 = b^2 + c^2$ för sidorna i en rätvinklig triangel. Det finns oändligt många tripler av heltal som uppfyller detta samband, exempelvis (5, 4, 3) och (13, 12, 5). Sådana uppsättningar av tre heltal kallas pytagoreiska taltripler.

   Vi kan systematiskt hitta pytagoreiska taltripler med följande samband:

   $$a = \frac{1}{2}\left(n^2 + m^2\right) \quad ; \quad b = \frac{1}{2}\left(n^2 - m^2\right) \quad ; \quad c = n \cdot m$$

   formlerna fungerar om n och m är udda tal, och n är större än m.

   UPPGIFTEN:
   Skriv en funktion med indata n_{max} som skriver ut alla pytagoreiska taltripler där n är n_{max} eller mindre.

3. På Canvas hittar ni textfilerna **`TempNacka.txt`** och **`TempSala.txt`**. Filerna skulle kunna vara temperaturdata uppsamlade under ett dygn med hjälp av en temperaturgivare.

   Filerna innehåller bland annat en tabell med två kolumner, där den vänstra kolumnen anger tidpunkt (antal minuter från mätningarnas start) och den högra kolumnen innehåller ett tal mellan 0 och 255, som motsvarar temperaturen.

   Relationen mellan ett tal i den högra kolumnen (låt oss kalla det *u*) och temperaturen är:
   $temperatur = 0.19 \cdot u - 19.3$. Så om $u = 139$ så gäller att temperaturen är 7.1 grader Celsius.

   Precis före den nämnda tabellen står raden: **`& Mätserie start`**
   och precis efter står raden **`$ Mätserie stop`**
   Filens format är sådant att det är endast dessa rader som får börja med **`&`** respektive **`$`**.

   UPPGIFTEN:
   Skriv en funktion vars indata är namn på en fil med temperaturdata (i praktiken blir det **`TempNacka.txt`** eller **`TempSala.txt`**.)

   Funktionen skall läsa in siffrorna i tabellen, och beräkna temperaturerna med formeln som angivits ovan. Om funktionsfilen heter LaesTemp.m skall ett typiskt anrop av filen kunna vara:

   ```
   >> LaesTemp('TempSala.txt')
   ```

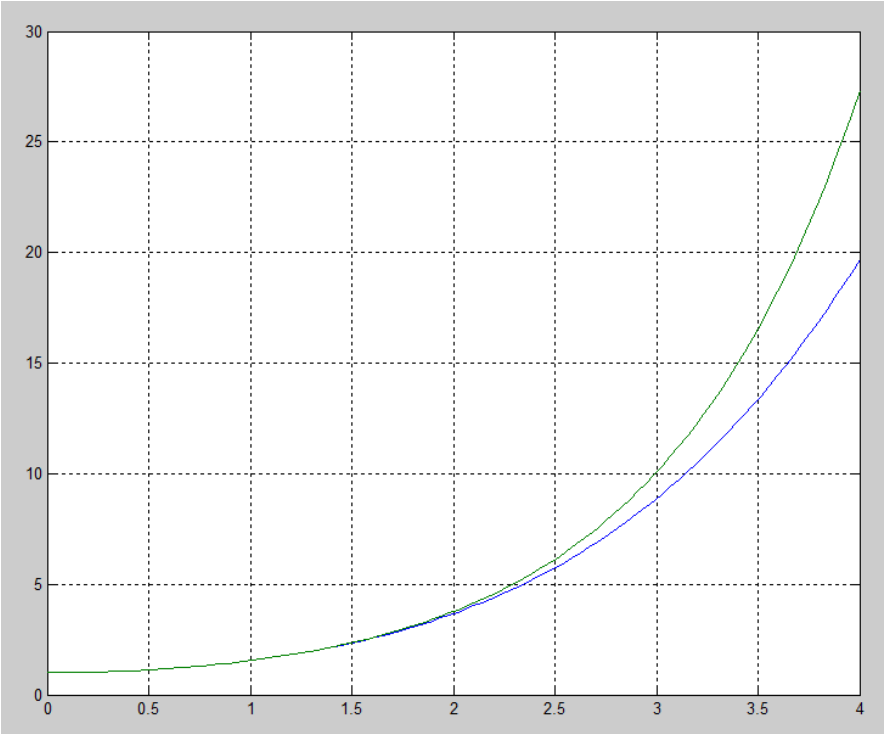
   Funktionen skall skapa en textfil med namnet **`TidTemp.txt`**. I denna fil skall en tabell med temperaturerna i klartext skrivas in (beräknade med formeln ovan). Vänstra kolumnen skall bestå av tidsangivelser i minuter precis som i originaltabellen, medan den högra skall utgöras av temperaturangivelser med en noggrannhet på en tiondels grad.

   Slutligen skall ett diagram presenteras, där temperaturen i grader Celsius är plottad mot tiden i minuter.

Uppgifter för erhållande av högre betyg

4. Detta är en fortsättning på uppgift 1. Vi skall åter beräkna summan $S = \sum_{n=0}^{N} \frac{x^{2n}}{(2n)!}$.

   UPPGIFTEN
   Skapa en funktion med ett *X*-värde och ett *N* -värde som indata. Funktionen skall beräkna värdet på summan S med N+1 termer i intervallet $[\,0, X\,]$. Dessutom skall de beräknade värdena i intervallet plottas tillsammans med funktionen $\cosh(x)$. Figuren nedan visar hur resultatet av plottningen ser ut om man vill ha jämförelse mellan $\cosh(x)$ och summan S, beräknad med N=2 i intervallet [0, 4].



5. Detta är en fortsättning på uppgift 2. Den som löste uppgift 2 lade sannolikt märke till att vissa tal kan vara ”hypotenusa” i en pytagoreisk taltrippel, medan andra tal omöjligt kan vara det.

   UPPGIFTEN
   Skriv en funktion som tar emot ett tal och undersöker om det kan vara det största talet i en pytagoreisk trippel.

   Om funktionen heter **`isPyt3`** skulle en dialog med funktionen kunna se ut så här:

```
>> isPyt3(511)
   Talet 511 ingår i den pythagoreiska taltrippeln [511  336  385]

>> isPyt3(127)
   Talet 127 är inte största tal i någon pythagoreisk taltrippel

>> isPyt3(221)
   Talet 221 ingår i den pythagoreiska taltrippeln [221  104  195]
   Talet 221 ingår i den pythagoreiska taltrippeln [221  204   85]
   Talet 221 ingår i den pythagoreiska taltrippeln [221  140  171]
   Talet 221 ingår i den pythagoreiska taltrippeln [221  220   21]
```

6. Detta är en fortsättning på uppgift 3. Filerna **`TempNacka.txt`** och **`TempSala.txt`** innehåller också information om det klockslag då temperaturmätningarna startade. Omedelbart ovanför raden med detta klockslag finns en rad som börjar med en stjärna. Det kan utnyttjas för att läsa in tidpunkten.

   Tidpunkten är angiven så att före punkten anges timmen med två siffror och efter punkten anges minuten med två siffror. Så 15.37 betyder ”trettiosju minuter över tre på eftermiddagen”.

   UPPGIFTEN
   Skriv en funktion vars indata är namnet på den fil som skall läsas in. Funktionen skall ***läsa in*** en fil formaterad som **`TempNacka.txt`** och **`TempSala.txt`**, och ***skapa*** en textfil som innehåller originalfilernas metadata (vars rader inleds med **#**).

   Tabellen med tidpunkter och temperatur skall naturligtvis vara med, men nu skall tidpunkterna anges som de klockslag då mätningen ägde rum. Filen som är baserad på uppgifter från Sala skall således inledas så här:

```
#  Behandlade data    #
# --------------------#
#    TEMPERATURDATA
# --------------------#
#    Sala väderstation    #
#    Datum: 2018-04-14
#
*   Tidpunkt vid mätseriens början:
15.37
& Mätserie start
  15.37    11.9
  16.06    12.6
  16.35    12.6
```

   …. och så vidare….

7. Den största gemensamma delaren för två tal kan beräknas med en enkel algoritm. Antag att vi vill finna den största gemensamma delaren till talen 42 och 12. Då kan man göra så här:

   Subtrahera det mindre talet från det större: 42 – 12 = 30. Nu har vi talen 12 och 30.
   Subtrahera det mindre talet från det större: 30 – 12 = 18. Nu har vi talen 12 och 18.
   Subtrahera det mindre talet från det större: 18 – 12 = 6. Nu har vi talen 12 och 6.
   Subtrahera det mindre talet från det större: 12 – 6 = 6. Nu har vi talen 6 och 6.
   Subtrahera talen: 6 – 6 = 0.

   Det sista talet vi hade innan det blev noll var 6. Alltså är 6 det största tal som både 42 och 12 är delbara med.

   UPPGIFTEN
   a.
   Skriv en funktion som beräknar den största gemensamma delaren. Om argumenten (indata) exempelvis är 42 och 12 skall funktionen returnera 6. Eller, om argumenten är 889 och 2933 skall funktionen returnera 7.

b.

Skriv en funktion som förkortar bråk. Bråket $\frac{12}{42}$ kan representeras av vektorn [12, 42]. Om funktionen heter **`forkorta`** skulle en dialog med funktionen kunns se ut så här:

```
>> forkorta([12, 42])
ans =

        2
        7
```

eller:

```
>> forkorta([889, 2933])
ans =

      127
      419
```

På så sätt kan man med funktionen konstatera att $\frac{12}{42} = \frac{2}{7}$ eller att $\frac{889}{2933} = \frac{127}{419}$

Naturligtvis skall funktionen använda sig av funktionen för största gemensamma delaren, som programmerades i *a*-uppgiften.

c.

Skriv en funktion som har två tal som indata, och som beräknar vad som skulle vara den minsta gemensamma nämnaren om talen vore nämnare i varsitt bråk. Om funktionens namn var **`mgn`** skulle en dialog med funktionen kunna vara så här:

```
>> mgn(14, 35)
ans =

      70
```

Naturligtvis skall funktionen använda sig av funktionen för största gemensamma delaren, som gjordes i *a*-uppgiften.

d.

Skriv en funktion som adderar bråk. Additionen $\frac{3}{10} + \frac{7}{15}$ kan representeras av vektorn **`[3, 10, 7, 15]`**. Om funktionen heter **`FracAdd`** skulle man kunna tänka sig en dialog med funktionen som är så här:

```
>> FracAdd([3, 10, 7, 15])
ans =

      23
      30

>> FracAdd([1, 14, 1, 35])
ans =

       1
      10
```

Med hjälp av funktionen har man då konstaterat att $\frac{3}{10} + \frac{7}{15} = \frac{23}{30}$ och att $\frac{1}{14} + \frac{1}{35} = \frac{1}{10}$
Det är rekommendabelt att använda sig av minsta-gemensamma-nämnaren-funktionen från *c*-uppgiften och förkortningsfunktionen från *b*-uppgiften.

8. En talföljd är definierad enligt följande:

$F_0 = 1,\ F_1 = 1$

Om $n > 1$ gäller $F_n = F_{n-1} + F_{n-2}$

Skall man beräkna nästa tal i följden erhålls således $F_2 = F_1 + F_0 = 1 + 1 = 2$

Enligt påståenden gäller att $\frac{F_{n+1}}{F_n}$ konvergerar mot $G = \frac{1+\sqrt{5}}{2}$

UPPGIFTEN
Skriv en funktion **`fun(N)`** som beräknar och skriver ut alla tal i talföljden fram till F_N och också beräknar och skriver ut värdet på kvoten $\frac{F_{n+1}}{F_n}$.
Undersök hur långt du kan beräkna tal i talföljden och erhålla exakta heltal (dvs. icke avrundade värden).

Undersök också om påståendet stämmer, att kvoten $\frac{F_{n+1}}{F_n}$ konvergerar mot $G = \frac{1+\sqrt{5}}{2}$.

Exemplet nedan visar hur en dialog med funktionen skulle kunna se ut. Nedan beräknas talföljden fram till det att N är lika med 7.

```
>> fun(7)

-------------------------------------------------------------
  n                    F        F_n+1/F_n        G - F_n+1/F_n
-------------------------------------------------------------
  1                    1
  2                    1      1.00000000     -0.6180339887498949
  3                    2      2.00000000      0.3819660112501051
  4                    3      1.50000000     -0.1180339887498949
  5                    5      1.66666667      0.0486326779167718
  6                    8      1.60000000     -0.0180339887498948
  7                   13      1.62500000      0.0069660112501051

 ------------------------------------------
 G = 0.5*( 1 + sqrt(5) ) = 1.618033989
```