Tentamen i MA069G matematisk Modellering 2018-10-31 – exempel på bra studentsvar

Uppgift 1 (studentsvar som gav 5 av 5 poäng)

A, C_a och D_a

$f(x) = \arctan(1+x^2)$

Approximera derivatan av f(x) I x = ½ genom framåtdifferenser med steglängder h=0.02 och h=0.01

```
clear
clc

format long
f = @(x,h) (atan(1+x^2));                %funktion

x0 = 1/2;                                %Derivatan nära värdet x = 1/2

h1=0.02;                                 %Steglängd
h2=0.01;

df1 = (f(x0+h1)-f(x0))/(h1)               %Framåtdifferens med steglängd h1=0.02
utan extrapolation
df2 = (f(x0+h2)-f(x0))/(h2)               %Framåtdifferens med steglängd h2= 0.01
utan extrapolation

REC = ((2*df2)-df1)                                 %Richardsson extrapolation

f_prime = ((2*x0)/((x0^2 + 1)^2 + 1));              %exakta värdet vid x0


edc1 = abs(f_prime-df1);                  %Error hur långt ifrån man var riktiga
värdet med steglängd h1
edc2 = abs(f_prime-df2);                  %Error hur långt ifrån man var riktiga
värdet med steglängd h2
eREC = abs(f_prime-REC);                  %Error hur långt ifrån man var riktiga
värdet med Richardsson extrapolation

Resultat = [df1 df2 REC]
Error = [edc1 edc2 eREC]
```

Svar på A och C_a:

H1 (0.02) = 0.394118576161906 med fel = 0.003874673722882

H2 (0.01) = 0.392211959048172 med fel = 0.001968056609148

Extrapolerat = 0.390305341934438 med fel = 0.000061439495414

B, C_b och D_b

```
clear
```

```
clc

format long

f = @(x,h)(atan(1+x^2));          %funktion

x0 = (1/2);                          %bestämma derivatan nära värdet x =
1/2

h1=0.02;                           %steglängd
h2=0.01;

dc1 = (f(x0+h1)-f(x0-h1))/(2*h1);    %Central differens med steglängd
h1=0.02 utan extrapolation
dc2 = (f(x0+h2)-f(x0-h2))/(2*h2);    %Central differens med steglängd
h2=0.01 utan extrapolation


REC = (dc2 + (dc2-dc1)/3);              %Richardsson extrapolation med
central differens

f_prime = ((2*x0)/((x0^2 + 1)^2 + 1)); %exakta värdet vid x0


edc1 = abs(f_prime-dc1);           %Error hur långt ifrån man var riktiga
värdet med steglängd h1
edc2 = abs(f_prime-dc2);           %Error hur långt ifrån man var riktiga
värdet med steglängd h2
eREC = abs(f_prime-REC);           %Error hur långt ifrån man var riktiga
värdet med Richardsson extrapolation

Resultat = [dc1 dc2 REC]
Error = [edc1 edc2 eREC]
```

Svar:

H1 (0.02) = 0.390120855986512 med fel = 1.0e-03 * 0.123046452512821
H2 (0.01) = 0.390213136340584 med fel = 1.0e-03 * 0.030766098440771
Extrapolerat = 0.390243896458608 med fel = 1.0e-03 * 0.000005980416773

D

format rat

f_prime

Svar:

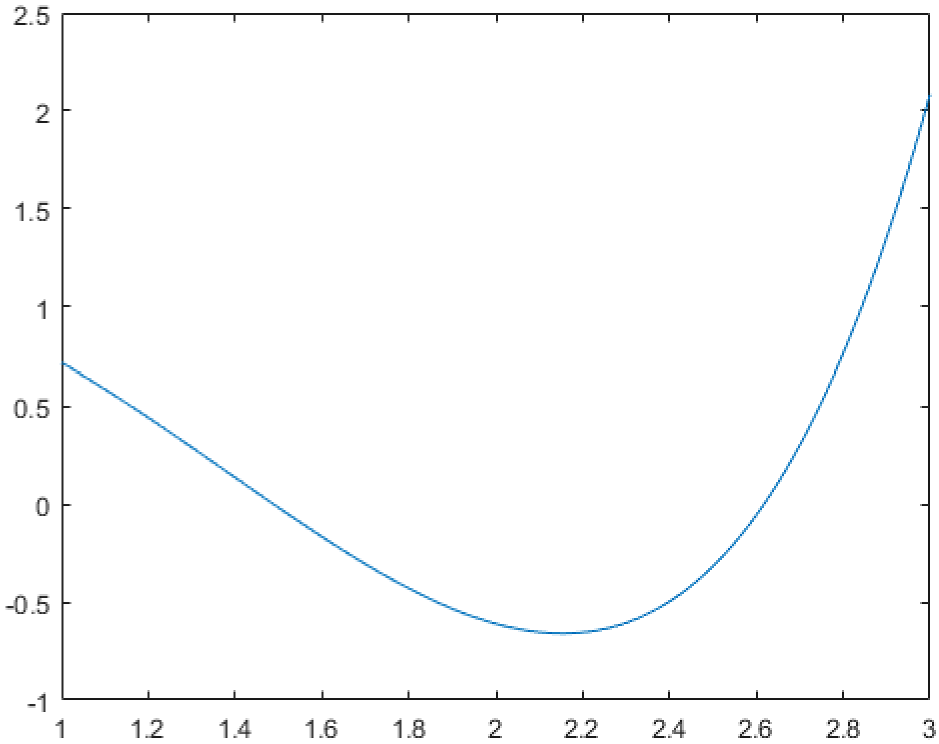
f'(1/2) = 16/41

felvärden står i svar för a) och b)

Uppgift 2 (studentsvar som gav 4,5 av 5 poäng)

$$f(x) = e^x - 2x^2$$

a)



b) Halveringsmetod för att hitta lösningen till $f(x) = 0$ i intervallet $[2,3]$, tills intervall längden är mindre än 0.01.

I MatLab:

```
Funktion för att kolla funktionsvärdet

function y = upggift_2_f( x )
%Returns the y-value of the inputted x-value

y = exp(x) - 2*x^2;

end
```

```
Script för att lösa uppgiften

format long;

a = 2; %Start of intervall
b = 3; %End of intervall

if uppgift_2_f(a)*uppgift_2_f(b) < 0 %Guarantee there actually is a 0 
between
    while b - a > 0.01 %loop while the intervall length is greater than 
0.01
        c = (b - a)/2; %Half of the intervall
        if uppgift_2_f(b-c)*uppgift_2_f(b) < 0 %Checks if the 0 is in the 
upper half of the limit
            a = a + c; %New lower limit
        elseif uppgift_2_f(a)*uppgift_2_f(a+c) <= 0 %Checks if the 0 is in 
the lower half of the limit
            b = b - c; %New higher limit
        end
    end
end

x = a + c;
uppgift_2_f(x)
```

Utdata:

a	2.6172
b	2.6250
c	0.0078

Utifrån tabellen kan vi läsa att $f(x) = 0$ ligger någonstans på intervallet $[2.6172, 2.6250]$

c)

Funktion för Newton Rhapson

```
function [ x ] = NewRap( xstart )

x = xstart - uppgift_2_f(xstart)/uppgift_2_f_prim(xstart); %Creates the 
first x value

while abs(xstart - x) >= 10^-7 %Stop when the difference is less than 10^-7
    xstart = x; %Assign the old start value a new value from previous 
iteration
    x = xstart - uppgift_2_f(xstart)/uppgift_2_f_prim(xstart); %Assign the 
new x value using Newton Rhapsons method
end
```

ger tillbaka värdena 1.487962065498177 för lägre punkt med 1.5 som startvärde och 2.617866613066812 som högre punkt med 2.6 som startvärde.

d) Ifall vi tänker igenom algoritmen för halveringsmetoden så håller den på så länge $f(a) \cdot f(b) < 0$, vilket betyder att vi vill kolla $f(1) \cdot f(3)$, vilket är större än noll som innebär att det finns antingen ingen eller ett jämt antal lösningar på problemet. Ifall vi skulle kringgå detta så skulle man aldrig få ett exakt svar med halveringsmetoden, då det naturliga talet har ett oändligt antal decimaler och svaret då antagligen också skulle behöva det. I MatLab skulle man kanske kunna få 14 decimalers precision, men det är inget jämfört med totalen. Däremot skulle det ju gå att lösa ekvationen på ett annat sätt, men intervallhalveringsmetoden är inte den rekommenderade metoden för ett exakt svar.

Uppgift 3 (studentsvar som gav 5 av 5 poäng)

a)

```
f=@(x)log(x);
Resultat = mcint(f, [1 exp(1)], 10, 1000)

function [N, mu, err] = mcconv(D, N)
% MCCONV. Function solving a multidimensional integral using %...%

close;
% Number of replications to base the error estimation on
M = 50;

% Default number of pseudorandom numbers
if (nargin < 2)
   N = [100; 500; 1000; 5000; 10000; 1e5];
end

% Default 6D and range [-5 5] in each dimension
if (nargin == 0) | (isempty(D) )
    D = [-5 5;-5 5;-5 5;-5 5;-5 5]; % 6D
end

N = N(:);
% The dimension and range
[rows, columns] = size(D);
if columns ~= 2
    error('There must be 2 numbers (lower and upper limit) in each row in D');
end



dim = numel(D)/2;
disp(['A ',num2str(dim),'-dimensional integral is solved'])
disp(['for random numbers: ' num2str(N')]);

if dim == 1
    fplot(@fnorm, [D(1) D(2)]);
end
if dim == 2
    NN = 100;
    x = linspace(D(1,1), D(1,2), NN);
    y = linspace(D(2,1), D(2,2), NN);
    [X,Y] = meshgrid(x,y);
    Z = fnorm([X(:)';Y(:)']);;
    Z = reshape(Z,NN,NN);
```

```
    surf(X,Y,Z);
    shading interp;
end
drawnow;
j=1;
mu = zeros(length(N),1);
err = mu;
for i=1:length(N)
    [mu(j),err(j)]=mcint(@fnorm, D, N(i), M);
    j=j+1;
end
end


% ------------------------------------------
% Interna funktioner

function y = fnorm(x)
% FNORM. function defining the density of the d-dim standard normal %...%


   [d,N]=size(x);
   C = eye(d);
   y = exp(-sum(x.*(C*x),1));
end

function [val,err] = mcint(f,D,N,M)
% MCINT.  Function that solves an integral using Monte Carlo.%...%

    V   = cumprod(D(:,2)-D(:,1));
    dim = numel(V);
    V   = V(end);

    r=zeros(dim,N);

    for j=1:M
        r(:,:) = repmat(D(:,1),1,N)+rand(dim,N).*repmat(D(:,2)-D(:,1),1,N);
        I(j)=V*mean(f(r));
    end

    val = mean(I);
    err = std(I);

end
```

```
Resultat =

   0.999446021781160
```

Monte Carlo metoden ger ett resultat som går mot 1, med antalet punkter och iterationer på 1000 blir precisionen alltid närmare än 0.01.

b)

```
x = 1:(exp(1)-1)/10:exp(1);
F = log(x);
Resultat = trapz(x, F)
```

Command Window

```
>> Uppgift3b

Resultat =

   0.998447005059135

>> |
```

Beräkningen sker i 10 x värden.

c)

$$\int_1^e \log(x)\,dx = \ [x(\log(x-1))]$$

$$= e(\log(e)-1) - 1(\log(1)-1)$$

$$= e * (1-1) - 1 * (0-1)$$

$$0 + 1 = 1$$

Fel i a) med 100 punkter och 100 upprepningar = 0.003279372035343

Fel i b) med 10 steg = 0.001552994940865

d)

Om integraler i högre dimensioner ska beräknas är det alltid bättre med Monte Carlo då den består oförändrad genom alla olika dimensioner. Men i detta fall skulle jag vilja påstå att det är bättre att använda trapetsmetoden då dimensionen är låg och det ger ett bättre och snabbare resultat.

Uppgift 4 (studentsvar som gav 5 av 6 poäng)

A Jag tolkar det som att alla maskiner som går att använda för t ex en stol måste användas för varje tillverkad stol. Det vill säga att för att en stol ska tillverkas så måste den gå igenom maskin 1, 2 och 3 medan en bokhylla bara går igenom maskin 1 och 2. Annars påverkar de olika maskinerna inte varandra vilket gör att maskin 1 tillverkar bara bord, maskin 2 bara bokhyllor och maskin 3 bara stolar (detta skulle ge 7400 kr vinst).

```
max = 30 * x1 + 20 * x2 + 40 * x3;
2 * x1 +  x2 + 3 * x3 <= 120;
x1 + x2 + x3 <= 80;
x1 + x2 <= 60;
```

B

Svar:

2000 kr, 60 bord och 20 bokhyllor

C

```
min = 120 * y1 + 80 * y2 + 60 * y3;
2 * y1 + y2 + y3 >= 30;
y1 + y2 + y3 >= 20;
3 * y1 + y2 >=40;
```

Svar:

2000 kr

D

Svar:

100 kr

Y2 har värdet 10 dvs 10 timmar * 10 = 100.

E

Svar: 0 stolar, 50 bord och 23,333 bokhyllor

F

Svar: Antalet tillverkade enheter är inte längre i heltalsform vilket är ett måste vid tillverkning av varor. Det måste tas i beaktande vid vidare beräkningar av sifforna.

Uppgift 5 (studentsvar som gav 3,5 av 5 poäng)

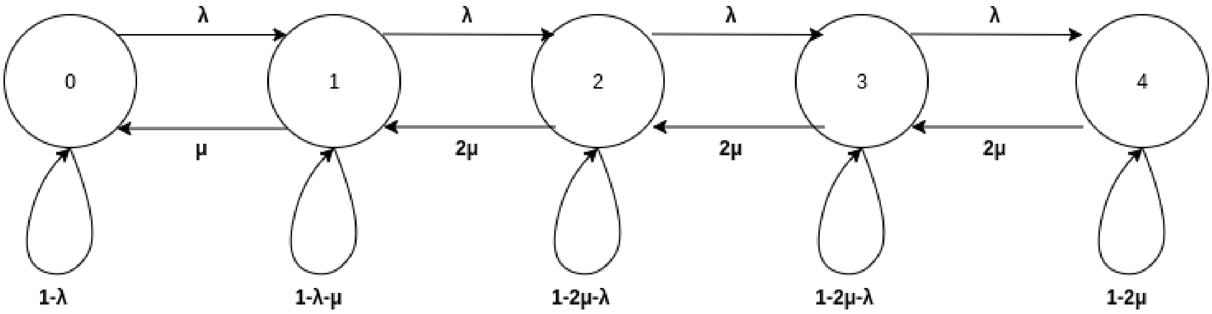
A

Tillstånd = 0,1,2,3,4 kunder i butiken

2 kassörer

Inkomna kunder per min λ = 0.5

Avslutade ärenden per kassör μ = 0.2 per min



B

Matris	0	1	2	3	4
0	0,5	0,5	0	0	0
1	0,2	0,3	0,5	0	0
2	0	0,4	0,1	0,5	0
3	0	0	0,4	0,1	0,5
4	0	0	0	0,4	0,6

C

```
clear
clc

P = [0.5 0.5 0 0 0; 0.2 0.3 0.5 0 0; 0 0.4 0.1 0.5 0 ; 0 0 0.4 0.1 0.5; 0 0
0 0.4 0.6]; %Array med sannolikheter
s0 = 1; %startvärde
T = [0.0000001 0.0000001 0.0000001 0.0000001 0.0000001]; %Tolerans
rest = 80;
i=0;

while abs(rest) > T %Itererar tills att resten är mindre än toleransen
    s1 = s0*P;
    rest = s1 - s0;
    s0 = s1;
    i=i+1;
end

M = sum ( [0 1 2 3 4 5] .* s1) ; % Viktat medelvärde - medelantal kunder


Svar: s1 = [0.1180    0.2220    0.2200    0.2080    0.2320]
```

Viktat medelvärde: M = 3,22 kunder i butiken i genomsnitt.

D