UPPSALA
UNIVERSITET

Workout

Uppgifter till Beräkningsvetenskap I

Sep 2017

WORKOUT

Institutionen för informationsteknologi

Om Workout

Detta häfte innehåller uppgifter som ingår i de s k workout-passen i kursen Beräkningsvetenskap I. Syftet med dessa är att du på egen hand ska arbeta igenom övningar med papper och penna. Arbetet med workout-uppgifterna, med möjlighet till hjälp från lärare, ger tillfälle att bearbeta det stoff som har behandlats under föreläsningarna.

Skriftliga redovisningar av lösningar till workout-uppgifterna är obligatoriska. Du bör delta under workout-passet och redovisar då direkt till läraren. Om du inte kan närvara så lämnar du din redovisning senast dagen efter workout-passet.

Uppgifterna löses lämpligen i grupp. Det är viktigt att alla i gruppen har förstått lösningarna. *Har* man förstått, så har man goda chanser att klara tentamen.

Du kan använda workout-uppgifterna som underlag för en *självdiagnos*. Om du *inte* förstår en lösning som gruppen kommit fram till, bör du läsa på mer om det avsnittet och be lärare eller någon kompis förklara. Lägg inte en workout-uppgift åt sidan förrän du tycker att du verkligen förstår den och kan lösa den (och andra av samma typ) på egen hand.

Observera att workout-övningarna är tillrättalagda så att de går att genomföra med handräkning. De är enbart ett stöd för ditt lärande och ska *inte* uppfattas som exempel på verkliga tillämpningar. I de tillämpningar som metoderna egentligen är tänkta för krävs datorer för att genomföra beräkningarna inom rimlig tid. Beräkningsvetenskap handlar om metoder och algoritmer för stora problem som inte kan lösas för hand. Den typen av tillämpningar ger miniprojekten exempel på.

Kurvanpassning

Uppvärmning – obligatorisk

1. Hitta interpolationspolynomet av grad 3 genom de fyra första punkterna i tabellen nedan:

x	-2	-1	0	1	2
$f(x)$	2	14	4	2	2

   Newtons interpolationsansats ska användas.

2. Använd minstakvadratapproximation för att anpassa en rät linje till följande punkter:

x	9	10	11	12	13
y	5	5	4	3	1

   Använd ansatsen $p_1(x) = a_0 + a_1 \cdot x$, formulera de s.k. normalekvationerna och lös dem. Vilken blev formeln för den resulterande räta linjen? Rita ett diagram, där du dels prickar in de givna punkterna, dels ritar den räta linjen.

Medel – obligatorisk

3. För samma data som i uppgift 2 ovan, formulera normalekvationerna igen men nu med ansatsen $p_1(x) = a_0 + a_1 \cdot (x - \hat{x})$, där $\hat{x}$ är medelvärdet över x. Du behöver inte lösa normalekvationerna, utan det räcker om du ställer upp dem. Vilken kan fördelen vara med denna ansats i jämförelse med den i uppgift 2?

4. I ett experiment studeras förloppet i en kemisk process, där ett visst ämne bildas. Under experimentets gång mäts koncentrationen c av ämnet i fråga vid olika tidpunkter t. I tabellen nedan visas några mätvärden (i lämpliga enheter, som vi här bortser ifrån).

t	0	0.5	1.0	1.5	2.0
c	0	0.19	0.26	0.29	0.31

   Skissa på ett Matlabprogram som minstakvadratanpassar ett andragradspolynom till dessa data samt ritar en graf som visar både mätpunkterna och andragradspolynomet. Använd Matlab-kommandona `polyfit` och `polyval`. Här ges de inledande raderna till de bådas hjälptexter:

```
polyfit
  Fit polynomial to data.
  P = polyfit(X,Y,N) finds the coefficients of a polynomial
  P(X) of degree N that fits the data Y best in a least-squares
  sense.

polyval
  Evaluate polynomial.
  Y = polyval(P,X) returns the value of a polynomial P evaluated
  at X.
```

5. I en databas för termodynamiska beräkningar ingår bland annat värden på Gibbs fria energi för olika kemiska ämnen vid olika temperaturer. Nedan visas databasens ”rad” för svaveldioxid:

S1O2	71748	76281	82564	85795	89080

   De fem sifferkolumnerna visar Gibbs fria energi i kalorier/mol för svaveldioxid för temperaturerna 25, 100, 200, 250 respektive 300 grader Celsius.

   För beräkning av Gibbs fria energi för temperaturer mellan dem som ingår i databasen kan man använda interpolation baserad på närmast kringliggande databasvärden. Du ska nu använda den metoden för att beräkna ett approximativt värde på Gibbs fria energi för svaveldioxid vid 260 grader Celsius. Använd linjär interpolation med Newtons interpolationsansats.

6. I kursen ingår Newtons interpolationsmetod, minsta kvadratanpassning och kubiska splines. Ge exempel när respektive metod är lämplig att använda.

Intensiv

7. För följande kväveoxider känner man molekylvikterna med tre decimalers noggrannhet:

NO	30.006	N_2O	44.013	NO_2	46.006
N_2O_3	76.012	N_2O_5	108.010	N_2O_4	92.001

   Använd minstakvadratanpassning för att beräkna atomvikten för kväve respektive syre utifrån dessa data. Samtliga data skall utnyttjas.

   Ledning: Låt x och y beteckna kvävets respektive syrets atomvikt. Enligt mätvärdena ovan är då $x + y = 30.006$, $2x + y = 44.013$, etc. På så vis ger tabellen sex ekvationer för de två obekanta atomvikterna. Genom att lösa detta överbestämda ekvationssystem med minstakvadratapproximation får man fram fram värden på atomvikterna.

8. En tidserie har följande mätvärden

t	0	0.50	1.00	2.00	4.00
r	2.01	1.79	1.42	0.99	0.42

   Man vet att sambandet mellan r och t är på formen

$$r(t) = \alpha e^{\beta t}.$$

   Hitta den bästa approximationen av α och β i Minsta kvadratmening. Tips: Logaritmera uttrycket och använda det som ansats.

9. Antag att du efter examen får arbete på ett institut som sysslar med samhällsekonomiska utredningar. En forskare vid institutet har ställt upp hypotesen att priset p på en bostadsrättslägenhet i tätort *ungefär* kan uttryckas med formeln

$$p = \frac{c}{r^2}$$

   där r är avståndet från lägenheten till stadskärnan. För varje tätort antas c vara en konstant, men c kan vara olika för olika tätorter.

   Som nyanställd får du i uppgift att genomföra en studie för att testa hypotesen och bestämma värdet på c för Uppsala tätort. Beskriv hur du skulle lägga upp studien och skissa ett Matlab-program för beräkning av c.

10. Om man vill åstadkomma en visuellt tilltalande kurva genom givna punkter, exempelvis i datorgrafik, så är det lämpligt att lägga till kontinuitetsvillkor på derivatorna hos det styckvisa interpolationspolynomet. Det vanligaste är kubiska splines, där både första och andra derivatan är kontinuerlig. För att inte komplicera uppgiften i onödan tar vi här emellertid upp fallet med *kvadratiska* splines. Det innebär att man i varje delintervall interpolerar med ett andragradspolynom samt att man på gränsen mellan två delintervall kräver att förstaderivatan skall vara kontinuerlig. Nu följer en mera precis beskrivning av idén:

    Låt värdena $(x, f(x))$ vara givna i $n+1$ stycken punkter i intervallet $[a, b]$. Punkterna betecknas med $x_0, \ldots, x_n$ och det gäller att:

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b.$$

    Det kvadratiska spline-polynomet till dessa punkter är en funktion q som uppfyller följandes villkor:

(a) $q(x)$ är kontinuerlig på intervallet $[a, b]$;

(b) i varje delintervall $[x_i, x_{i+1}]$, $i = 0, 1, \ldots, n-1$, sammanfaller q med ett kvadratiskt polynom

$$q(x) = q_i(x) = a_i + b_i\, x + c_i\, x^2;$$

(c) $q(x_i) = f(x_i)$ för $i = 0, 1, \ldots, n$.

(d) $q(x)$ har kontinuerlig förstaderivata i $x_i, i = 1, \ldots, n-1$, det vill säga

$$q'_{i-1}(x_i) = q'_i(x_i), i = 1, \ldots, n-1.$$

Din uppgift är att använda villkoren ovan för att ställa upp ett ekvationssystem för bestämning av koefficienterna i de olika polynom $q_i(x)$ som tillsammans utgör $q(x)$. Du kommer att märka att det behövs ytterligare ett villkor för att ekvationssystemet skall få entydig lösning. Ett sätt att åstadkomma ett sådant villkor är att föreskriva att interpolationspolynomets derivata skall vara noll i ändpunkten x_0.

Det räcker med att du ställer upp ekvationssystemet för fallet $n = 3$. Observera att du inte behöver lösa systemet.

Integrering

Uppvärmning – obligatorisk

1. Beräkna integralen
$$I = \int_0^1 x^2 e^x dx$$
numeriskt med trapetsformeln. Använd steglängderna $h = 1/2$ och $h = 1/4$.

2. Beräkna samma integral som ovan men nu med Simpsons formel och steglängden $h = 1/4$.

Medel – obligatorisk

3. Gör följande undersökningar utgående från resultaten i uppgift 1.

   (a) Just den här integralen kan beräknas analytiskt, med partialintegration, och har värdet $e-2$. Jämför trapetsformelns resultat i uppgift 1 med det exakta värdet på integralen. Hur stort blev absoulutbeloppet av det absoluta felet för steglängden $h = 1/2$ respektive $h = 1/4$? Med vilken faktor förbättrades noggrannheten i trapetsformeln när steglängden halverades? Stämmer denna förbättringsfaktor med vad du hade förväntat dig på basis av teorin?

   (b) De numeriska integreringsformlerna är tänkta att användas när den exakta integralen är svår eller omöjlig att beräkna. Då har man alltså ingen exakt lösning att jämföra med. Man vill ändå kunna avgöra hur noggrann det beräknade, approximativa integralvärdet är. För trapetsformeln kan man använda tredjedelsregeln för att åstadkomma en sådan uppskattning av felet. Använd nu tredjedelsregeln för att uppskatta felet i det värde du fick med steglängden $h = 1/4$ i uppgift 1.

   I just det här fallet vet vi också hur stort felet verkligen är. Jämför det verkliga felet med den uppskattning tredjedelsregeln gav. Verkar tredjedelsregeln tillförlitlig?

   (c) Tredjedelsregeln uppkommer som ett delresultat när man gör richardsonextrapolation utgående från trapetsformeln. Själva extrapolationen består i att det beräknade trapetsvärdet korrigeras med det uppskattade felet enligt tredjedelsregeln, vilket förmodas leda till en förbättrad approximation av integralen.

   Använd nu Richardsonextrapolation på de värden du fick i uppgift 1. Blev resultatet en förbättrad approximation?

   Jämför det approximativa integralvärde du nu fått fram med det svar du fick i uppgift 2. Hur förhåller sig dessa värden till varandra?

4. För en funktion $f(t)$ har man mätt upp följande mätvärden

t	0	0.125	0.25	0.375	0.5
$f(t)$	1.00	1.13	1.28	1.45	1.65

   (a) Beräkna integralen, dvs $\int_0^{0.5} f(t)\, dt$ med trapetsmetoden, steglängd $h = 0.125$.

   (b) Lös samma problem med Simpsons metod.

   (c) Antag att noggrannheten ska förbättras med en faktor 16. Hur många fler mätvärden behövs i (a) respektive (b)?

5. Sist i denna workout finns en algoritm i form av en Matlab-funktion, `trap`. För att öva dig i att följa programmeringskod, torrexekvera koden för integralen i uppgift 1, med $n = 2$.

   Du kan du hoppa över de inledande `if`-satserna och börja med raden som börjar med `x = a`.

   Kommentar: Med torrexekvering menas att du kör igenom koden för hand, rad för rad, och beräknar och skriver ner de siffervärden som programmet beräknar. Det här är ett sätt att dels förstå hur algoritmerna fungerar, dels lära sig flödet i en programmeringskod.

   `nargin` står för "number of arguments in" och innhåller ett heltalsvärde som är antalet inparameterar som skickats in i funktionen vid anropet.

Intensiv

6. Hur liten steglängd skulle vi behöva med trapetsmetoden för att få sex korrekta decimaler i beräkningen av integralen I ovan? Utgå från feluppskattningen i uppgift 3c ovan.
   Vilken steglängd skulle behövas för den noggrannheten om vi istället använde Simpsons formel? Antag att integranden kan evalueras så noggrant att vi kan bortse från felet i funktionsberäkningarna och bara behöver ta hänsyn till diskretiseringsfelet.

7. I laborationen till detta kursblock ingår bland annat ett exempel med vattenströmning i en cylindrisk vattenledning. Flödesvolymen per tidsenhet, Q, beräknas ur följande formel:

$$Q = \int_0^1 2\pi r v \mathrm{d}r.$$

   Längdskalan är här satt så att vattenledningens radie är 1. Med r avses koordinaten i radiell riktning, så att $r = 0$ motsvarar rörets mitt och $r = 1$ dess vägg. Strömningshastigheten betecknas med v och beror på r.

   I laborationen var strömningshastigheten given som en formel. Nu tänker vi oss istället att strömningshastigheten v *mäts* genom att sensorer placeras radiellt i ett tvärsnitt av röret.

   De uppmätta värdena (i lämpliga enheter) framgår nedan:

r	0.0	0.25	0.5	0.75	1.0
v	1.00	0.93	0.74	0.43	0.00

   Hastighetsvärdena v är avrundade till det antal decimaler som sensorernas mätnoggrannhet medger.

   (a) Använd samtliga de givna mätvärdena och numerisk kvadratur för att beräkna ett approximativt värde på Q. Välj själv en av de kvadraturformler som har ingått i kursen.

   (b) Resultatet av din beräkning ovan blir att Q approximeras med ett värde $\tilde{Q}$. Gör en välgrundad uppskattning av antalet korrekta decimaler i $\tilde{Q}$. Som motivering räcker hänvisning till kända formler, utan att dessa behöver härledas.

8. Härled de formler som du använde i deluppgift b) ovan.

9. Härled trapetsformeln genom att ansätta

$$\int_a^b f(x)dx \approx a_0 f(x_0) + a_1 f(x_1)$$

och bestämma koefficienterna a_0 och a_1 så att formeln blir exakt för ett förstagradspolynom.

10. Om du är van programmerare kan du nu ta ytterligare en titt på algoritmen för trapetsformeln nedan. Ett alternativt sätt att implementera trapetsformeln i Matlab skulle kunna vara följande: Skapa en vektor av x-värden genom att använda `linspace`. Skapa en vektor av funktionsvärden genom att tillämpa `func` på x-vektorn. Skapa en vektor som innehåller koefficienterna i trapetsformeln. Beräkna trapetsvärdet som skalärprodukten mellan koefficientvektorn och funktionsvärdesvektorn.

    Skriv en alternativ version av `trap` enligt ovanstående idé.

Algoritm för Trapetsformeln:

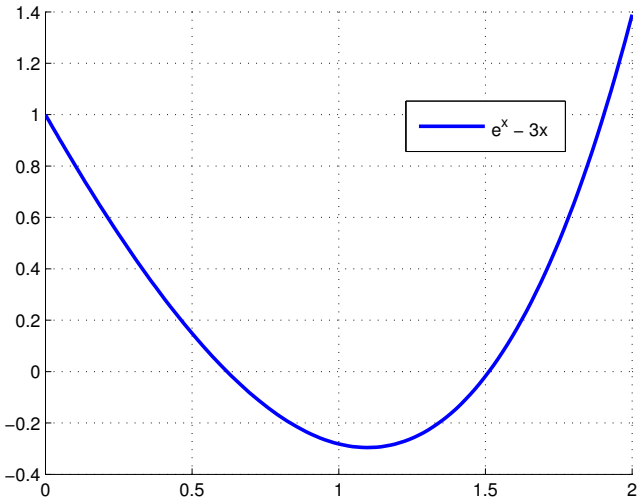
```
function I = trap(func, a, b, n)
% trap: composite trapezoidal rule quadrature
%    I = trap(@func, a, b, n);
%    I = trap(@(x) func(x,p1,p2,...), a, b)
% Input:
%    func       = name of function to be integrated
%    a, b       = integration limits
%    n          = number of segments (default = 100)
%    p1, p2,... = additional parameters used by func
% Output:
%    I = integral estimate

if nargin<3
   error('At least 3 input arguments required');
end
if ~(b>a)
   error('Upper bound must be greater than lower');
end
if (nargin<4) || isempty(n)
   n = 100;
end
x = a; h = (b-a)/n;
s = func(a);
for i = 1:n-1
   x = x + h;
   s = s + 2*func(x);
end
s = s + func(b);
I = (b-a)*s/(2*n);
```

Ickelinjära ekvationer

Uppvärmning – obligatorisk

1. Den icke-linjära funktionen $f(x) = e^x - 3x = 0$ har följande graf



Du ska nu hitta den rot som ligger mellan 0 och 1.

(a) Lös problemet med bisektionsmetoden. Välj startintervallet $[0.5, 0.8]$. Den beräknade lösningen ska approximera den exakta med minst en korrekt decimal. Följ algoritmen i detalj, kontrollera stoppvillkoret efter varje iteration och avbryt när stoppvillkoret uppfylls.

(b) Lös samma problem med Newton-Raphsons metod så att den beräknade lösningen approximerar den exakta med minst två korrekta decimaler. Välj lämplig startapproximation utgående från plotten. Följ algoritmen i detalj och avbryt när stoppvillkoret uppfylls.

MATLABs `fzero` ger roten 0.619061. Du kan *i efterhand* jämföra dina resultat med detta (så att du ser att gjort rätt). När du använder metoderna ovan är roten givetvis ej känd.

Medel – obligatorisk

2. En sfärisk tank ska fyllas med vatten. Vattenvolymen som tanken innehåller ges av

$$V = \pi h^2 \frac{3R - h}{3},$$

där V är volymen (m^3), R är sfärens radie (m) och h är vattnets höjd (djupet) i tanken (m).

Till vilket djup h måste man fylla tanken om den ska innehålla 30 m^3, givet att $R = 3$?

(a) Formulera Newton-Raphsons metod för att lösa problemet.

(b) Starta med startgissningen $h_0 = 1.5$ och utför några iterationer. Resultatet ska ha så pass många korrekta siffror att höjden har korrekt antal centimeter.

3. Vi fortsätter med samma ekvation som i föregående uppgift. Om du ska lösa den i Matlab är det enklast att använda Matlabs inbyggda kommando `fzero`.

   (a) Första steget blir i så fall att skriva en egen Matlab-funktion som beskriver ekvationen ovan. Hur skulle den funktionen se ut?

   (b) När du har skapat den funktion som beskriver ekvationen, så kan du lösa problemet med `fzero`. Skriv den instruktion i Matlab, som gör att ekvationen löses och resultatet lagras i variabeln `h`.

   Som hjälp ges här de första raderna i hjälptexten till `fzero`:

```
fzero  Single-variable nonlinear zero finding.
  X = fzero(FUN,X0) tries to find a zero of the function FUN
  near X0, if X0 is a scalar. It first finds an interval
  containing X0 where the function values of the interval
  endpoints differ in sign, then searches that interval for
  a zero. FUN is a function handle.
```

Intensiv

4. Vilka svårigheter skulle kunna uppstå när man löser problemet i uppgift 1 med Newton-Raphsons metod?

5. Antag att vi i uppgift 1 istället skulle vilja ha lösningen med minst tolv korrekta decimaler. Hur många ytterligare iterationer skulle då behövas med bisektionsmetoden respektive Newton-Raphsons metod? Du ska besvara frågan utan att *genomföra* några ytterligare iterationer. Det gäller att i stället utnyttja teoretiska samband.

6. Miniräknare använder ibland Newton-Raphsons metod för att beräkna kvadratroten av positivt tal y, dvs $x = \sqrt{y}$. Härled en formel för detta, dvs för beräkning av $f(x) = x^{1/2}$. Beräkna sedan $\sqrt{3}$ med denna metod. Använd t ex startvärde $x_0 = 3$ och iterera tills 3 signifikanta siffror är korrekta.

7. Sist i det här häftet visas ett exempel på hur bisektionsmetoden kan implementeras i Matlab. Sätt dig in i programmet i detalj, genom att torrexekvera det för problemet i uppgift 1.
   Kommentar: Med torrexekvering menas att du kör igenom koden för hand, rad för rad, och beräknar och skriver ner de siffervärden som programmet beräknar. Det här är ett sätt att dels förstå hur algoritmerna fungerar, dels lära sig flödet i en programmeringskod.

8. I koden för bisektionsmetoden (enligt föregående uppgift) används ett stoppvillkor som baseras på uppskattning av det absoluta felet. Visa att den formel som används för beräkning av `fel` i detta program verkligen är en uppskattning av det relativa felet.

9. Man skulle kunna kombinera Newton-Raphsons metod och bisektionsmetoden till en ny algoritm enligt följande idé.

   Låt $x^{(k)}$ beteckna den approximativa lösningen efter k iterationer och låt $[a\ b]$ vara ett intervall i vilket såväl den rätta lösningen som $x^{(k)}$ finns. (Iterationsprocessen inleds med att användaren ger ett intervall där lösningen finns och $x^{(0)}$ sätts till mittpunkten i det intervallet.)

   Iteration $k+1$ börjar med att man med Newton-Raphsons metod beräknar en tentativ ny approximation $\tilde{x}^{(k+1)}$.

Därefter kommer en delalgoritm som undersöker om det tentativa värdet blev "tillräckligt bra". I så fall tilldelas $x^{(k+1)}$ värdet $\tilde{x}^{(k+1)}$ och intervallet $[a\ b]$ lämnas oförändrat.

Om å andra sidan det tentativa värdet bedöms vara dåligt, så förkastas det. Då genomförs i stället en iteration med bisektionsmetoden utgående från $[a\ b]$ vilket leder till en halvering av sökintervallet, varefter $x^{(k+1)}$ sätts till mittpunkten i det nya intervallet.

Bedömningen av om det tentativa värdet är "tillräckligt bra" ska gå ut på att undersöka om det nya värdet är en förbättring i förhållande till föregående värde, $x^{(k)}$. Din uppgift blir att föreslå hur det skulle kunna kontrolleras, under förutsättning att den exakta lösningen är okänd. Uppgiften består av två delar: **(i)** föreslå en utformning av den delalgoritm som ska undersöka om $\tilde{x}^{(k+1)}$ är tillräckligt bra och **(ii)** ge en teoretiskt grundad motivering till ditt förslag (härledning eller annan övertygande argumentation byggd på teoretiska överväganden).

Matlabfunktion för Bisektionsmetoden:

```
function [x, fx, fel] = bisektion(f, x0, tol)
% [x, fx, fel] = bisektion(@f, [a b] tol);
% [x, fx, fel] = bisektion(@f, [a b]);
% Hittar nollställe till funktionen f(x) med Bisektionsmetoden.
% Funktionen f(x) definieras i f, [a b] är startintervall, och tol är
% toleransen. Om tol inte ges används 1e-4 som tolerans.
% x är nollstället, fx funktionsvärdet i nollstället, fel en vektor med det
% uppskattade felet i varje iteration.

if nargin < 4
   tol = 10^(-4);
end
xl = x0(1);
xu = x0(2);
if sign(f(xl))==sign(f(xu))
   disp('Funktionsvärdet i ändpunkterna måste ha olika tecken');
   fx = []; x =[]; fel = [];
   return;
end
niter = 1;
fel = 1;
while fel > tol
   xr = xl + (xu - xl)/2;
   x = xr;       % Lagra som ny approximation
   if sign(f(xl)) == sign(f(xr))
      xl = xr;
   else
      xu = xr;
   end
   fel = abs(xu - xl);
   niter = niter + 1;
end
fx = f(x);  % beräkna funktionsvärdet i nollstället
```