Studio 4: Dubbel- och trippelintegral.

Analys och Linjär Algebra, del C, K1/Kf1/Kb1, vt03

6 februari 2003

Samtliga integrander nedan antas vara Lipschitzkontinuerliga funktioner på sitt integrationsområde.

1 Repetition av enkelintegral och `my_int`

Betrakta differentialekvationen

$$\begin{aligned} u'(x) &= f(x), \qquad a < x < b, \\ u(a) &= u_a. \end{aligned} \tag{1}$$

Enligt *Analysens Fundamentalsats* ges lösningen till (1) av

$$u(x) = u_a + \int_a^x f(y)\,dy, \quad a < x < b. \tag{2}$$

I ALA-B skrev du en MATLAB-funktion som heter `my_int` vilken löser (1) *numeriskt* genom att beräkna integralen i (2) med *kvadratur* (numerisk integration). Funktionsfilen `my_int.m` kan t.ex. se ut som följer:

```
function [x, u] = my_int(f, I, ua, h)
a = I(1);
b = I(2);
N = ceil((b-a)/h);
x(1) = a;
u(1) = ua;
for i = 1:N   % loop över samtliga delintervall
   xi = a + i*h;   % delintervallets högra ändpunkt
   x(i+1) = xi;
   u(i+1) = u(i) + feval(f, xi)*h;   % “right-hand rectangle rule”
end
x = x';
u = u';
```

Funktionen ovan använder kvadratur baserad på integrandens värde i höger ändpunkt av delintervallen $I_i = [a + (i-1)h, a + ih],\ i = 1, 2, \ldots, N$. Den returnerar två (kolonn)vektorer:

$\texttt{x}$, vars element innehåller nodpunkternas koordinater:

$$\texttt{x(1)} = a,\ \texttt{x(2)} = a + h,\ \texttt{x(3)} = a + 2h,\ \ldots,\ \texttt{x(N + 1)} = a + Nh = b,$$

där vi för enkelhets skull antagit att $b - a$ är delbart med h (d.v.s. $b - a = Nh$).

$\texttt{u}$, vars element innehåller motsvarande (approximativa) funktionsvärden:

$$\texttt{u(1)} = u_a = u(a),$$

$$\texttt{u(2)} = u_a + f(a+h)\,h \approx u_a + \int_a^{a+h} f(x)\,dx = u(a+h),$$

$$\texttt{u(3)} = u_a + f(a+h)\,h + f(a+2h)\,h \approx u_a + \int_a^{a+2h} f(x)\,dx = u(a+2h),$$

$$\ldots$$

$$\texttt{u(N + 1)} = u_a + \sum_{i=1}^{N} f(a+ih)\,h \approx u_a + \int_a^b f(x)\,dx = u(b).$$

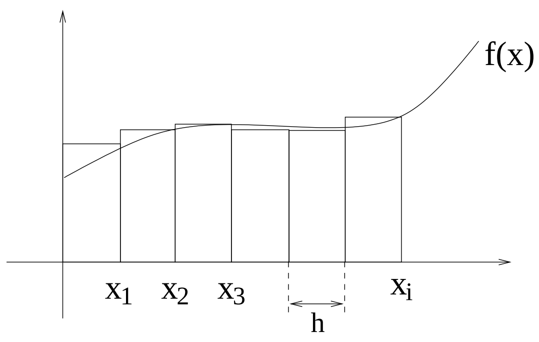
Ifall vi endast är intresserade av värdet på den bestämda integralen

$$\int_a^b f(x)\,dx, \tag{3}$$

väljer vi alltså $u_a = 0$, varpå (3) approximeras av element $\texttt{u(N+1)}$:

$$\int_a^b f(x)\,dx \approx f(a+h)\,h + f(a+2h)\,h + \ldots + f(b)\,h = \sum_{i=1}^{N} f(a+ih)\,h. \tag{4}$$

Notera att $f(a+ih)\,h$ är *arean* av en rektangel med basen h och höjden $f(a+ih)$, så att summan i (4) är en approximation till arean under kurvan $y = f(x)$ mellan $x = a$ och $x = b$.



Figur 1: Approximation av integralen genom kvadratur

För att direkt approximera (3) genom att beräkna summan i (4) kan `my_int.m` förenklas till `my_quad.m`:

```
function q = my_quad(f, a, b, h)
N = ceil((b-a)/h);
q = 0;
for i = 1:N   % loop över samtliga delintervall
   xi = a + i*h;   % delintervallets högra ändpunkt
   q = q + feval(f, xi)*h;   % “right-hand rectangle rule”
end
```

Exempel 1: `>> q = my_quad('cos', 0, 1, 1/100)` beräknar en approximation till $\int_0^1 \cos(x)\,dx$ baserad på en indelning av $[0, 1]$ i 100 delintervall.

2 Dubbelintegral

Vi skall nu utvidga ovanstående till beräkning av dubbelintegral över en rektangel $Q = [a, b] \times [c, d] = \{x = (x_1, x_2) : a \leq x_1 \leq b,\ c \leq x_2 \leq d\}$, och utgår från approximationen

$$
\begin{gathered}
\int_a^b \int_c^d f(x_1,\, x_2)\, dx_2 dx_1 \approx \\
f(a+h,\, c+h)\, h^2 + f(a+2h,\, c+h)\, h^2 + \ldots + f(b,\, c+h)\, h^2 + \\
f(a+h,\, c+2h)\, h^2 + f(a+2h,\, c+2h)\, h^2 + \ldots + f(b,\, c+2h)\, h^2 + \\
\ldots + \\
f(a+h,\, d)\, h^2 + f(a+2h,\, d)\, h^2 + \ldots + f(b,\, d)\, h^2 = \\
\sum_{i=1}^{N} \sum_{j=1}^{M} f(a+ih,\, c+jh)\, h^2, \qquad (5)
\end{gathered}
$$

där vi nu använder kvadratur baserad på integrandens värde i det övre, högra hörnet av delkvadraterna $Q_{ij} = [a+(i-1)h,\ a+ih] \times [c+(j-1)h,\ c+jh]$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, M$. Notera att $f(a+ih, c+jh)\, h^2$ är ***volymen*** av ett rätblock med basarean h^2 och höjden $f(a+ih, c+jh)$, så att (dubbel)summan i (5) är en approximation till volymen under ytan $x_3 = f(x_1, x_2)$, $(x_1, x_2) \in Q$.

Uppgift 1: Skriv en funktionsfil `my_quad2D.m`

```
function q = my_quad2D(f, a, b, c, d, h)
% ...
```

som approximerar dubbelintegralen av $f = f(x_1, x_2)$ över rektangeln $Q = [a, b] \times [c, d]$ genom att beräkna (dubbel)summan i (5).

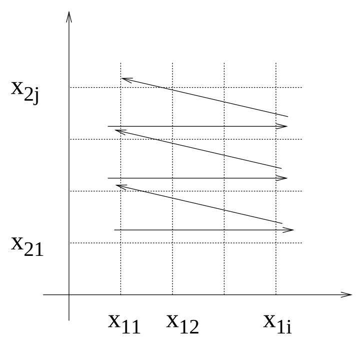
Tips:

i) Utgå från funktionen `my_quad`.

ii) Du kan införa ett nytt index `j`, och istället för variabeln `xi` t.ex. använda variablerna `x1i` och `x2j`.

iii) Du behöver nu två stycken for-loopar. Idén är att i den yttre loopen gå igenom alla värden på x_1 "`for i = 1:N`" och i den inre (för *varje* fixt x_1) gå igenom alla värden på x_2 "`for j = 1:M`". (Du kan förstås också göra tvärtom!) Detta kallas att "nästla" loopar. Resultatet blir en *dubbel-loop*[1].

iv) Tänk på att f nu är en funktion av *två* variabler, så du får ändra rad 6 i `my_quad.m` till exempelvis `q = q + feval(f, x1i, x2j)*h*h;`



Figur 2: Pilarna visar hur vi löper igenom nodpunkterna i den beskrivna nästlade loopen.

Exempel 2: `>> q = my_quad2D('funk64_1a', 0, 1, 0, 1, 1/100)` beräknar en approximation till dubbelintegralen i Uppgift 64.1 (a) i *Applied Mathematics: Body & Soul* baserad på en indelning av $[0,1] \times [0,1]$ i $100 \times 100 = 10000$ delkvadrater. Du måste förstås i detta fall först ha skapat funktionsfilen `funk64_1a.m`:

```
function x3 = funk64_1a(x1, x2)
x3 = x1 + x2;
```

Uppgift 2: Beräkna med `my_quad2D` dubbelintegralerna i Uppgift 64.1 i *Applied Mathematics: Body & Soul.* Räkna också ut integralerna $(a) - (c)$ analytiskt och jämför med dina beräknade värden. *Det är viktigt att på detta sätt alltid testköra dina program på exempel där du känner den exakta lösningen.*

[1] Skriver du `>> help for` så får du ett exempel på hur en dubbel-loop kan se ut.

Uppgift 3: Välj en av de tre integraler du räknat ut analytiskt i Uppgift 2 och beräkna den med olika värden på h, t.ex. $h = 0.1,\ 0.01,\ 0.001$. Hur beror felet (skillnaden mellan det exakta och det beräknade värdet) på h? Jämför gärna med Studioövning 2.1 i ALA-B: *How large (small) is the error?*
(*Kommentar*: Om $x = (x_1, x_2)$ och $y = (y_1, y_2)$ tillhör samma delkvadrat Q_{ij} ger Lipschitzkontinuiteten hos f att $|f(x) - f(y)| \leq L_f \, \|x - y\| \leq L_f \sqrt{2}\, h$, eftersom $\sqrt{2}\, h$, längden på diagonalen, är det maximala avståndet mellan två punkter i Q_{ij}. Felbidraget från Q_{ij} blir därmed maximalt $L_f \sqrt{2}\, h\, A(Q_{ij})$, där $A(Q_{ij}) = h^2$ är arean av Q_{ij}. Adderar vi nu felbidragen från samtliga delkvadrater inser vi att det totala felet måste vara mindre än $L_f \sqrt{2}\, h\, A(Q)$, där $A(Q) = (b-a)(d-c)$ är arean av Q. Det förefaller alltså som om felet borde vara proportionellt mot h.)

Uppgift 4: (Ytintegral) Uppgift 65.8 (a) i *Applied Mathematics: Body & Soul* (som består i att beräkna arean av en buktig yta) leder fram till dubbelintegralen

$$\int_0^1 \int_0^1 \sqrt{1 + x_1^2 + x_2^2}\, dx_1 dx_2$$

(visa detta!). Denna integral är *inte* lätt att räkna ut analytiskt. Beräkna den istället med `my_quad2D`.

3 Trippelintegral

Generalisering från enkel- till dubbelintegral går att utvidga till trippelintegral. Nu får vi använda oss av en nästlad loop med tre `for`-satser som löper över delkuber i de tre koordinatriktningarna.

Uppgift 5: Utgå från `my_quad2D.m` och skriv en funktionsfil `my_quad3D.m`

```
function q = my_quad3D(f, a, b, c, d, e, g, h)
% ...
```

som approximerar trippelintegralen av $f = f(x_1,\, x_2,\, x_3)$ över rätblocket $Q = [a,\, b] \times [c,\, d] \times [e,\, g] = \{x = (x_1,\, x_2,\, x_3) : a \leq x_1 \leq b,\ c \leq x_2 \leq d,\ e \leq x_3 \leq g\}$ genom att dela in Q i delkuber med sidlängd h.

Uppgift 6: Beräkna den första av de tre trippelintegralerna i Uppgift 66.3 i *Applied Mathematics: Body & Soul* med `my_quad3D`. Räkna också ut integralen analytiskt och jämför med ditt beräknade värde.

Tips: Notera att $||x||^2 = x_1^2 + x_2^2 + x_3^2$. Välj först t.ex. $h = 1/10$ eller $h = 1/20$ så att beräkningen inte tar alltför lång tid. Prova därefter att successivt minska på h. Hur mycket längre tid tar beräkningen om du minskar h med en faktor 2? Med hur mycket minskar felet?

4 Kodoptimering

Som du märkt kan beräkningarna bli tidskrävande när man beräknar multipelintegraler med kvadratur. Det är lätt att förstå varför: Antag att du vill beräkna en dubbelintegral över *enhetskvadraten* $Q = [0,\,1] \times [0,\,1] = \{x = (x_1,\,x_2) : 0 \le x_1 \le 1,\ 0 \le x_2 \le 1\}$ med en indelning av Q i *delkvadrater* med sidlängd 1/100. Du kommer då att behöva beräkna integrandens värde i $100^2 = 10\,000$ punkter, samt addera dessa värden. För att uppnå motsvarande "finhet" vid beräkning av en enkelintegral över intervallet $I = [0,\,1]$ räcker det med en indelning i 100 delintervall, och vi behöver endast beräkna integrandens värde i 100 punkter, samt addera dessa värden.

Detta är ett problem som förstås blir ännu mera påtagligt vid beräkning av multipelintegraler i högre dimension än 2. Ska du t.ex. beräkna en trippelintegral över *enhetskuben* $Q = [0,\,1]\times[0,\,1]\times[0,\,1] = \{x = (x_1,\,x_2,\,x_3) : 0 \le x_1 \le 1,\ 0 \le x_2 \le 1,\ 0 \le x_3 \le 1\}$ med en indelning av Q i *delkuber* med sidlängd 1/100. Du kommer då att behöva beräkna integrandens värde i $100^3 = 1\,000\,000$ punkter, samt addera dessa värden.

För att få ner beräkningstiden kan man på olika sätt försöka få koden att gå snabbare. När det gäller Matlab så går t.ex. loopar väldigt långsamt medan vektoroperationer går snabbt. I matlab-katalogen på kursens webbsida finns programmet `studio4_1.m`:

```
x = 1:1e7;          % Initierar en vektor
t1=cputime;         % Returnerar starttidpunkt
sum_loop = 0;       % Loopen summerar komponenterna i x
for i=1:length(x)
  sum_loop = sum_loop + x(i);
end
t2=cputime;         % Returnerar ny tidpunkt
sum_vec = sum(x)    % Matlabs eget summeringskommando
t3=cputime;         % Returnerar sluttidpunkt
t_loop = t2-t1      % Tiden för loopen
t_vector = t3-t2    % Tiden för sum-kommandot
t_loop/t_vector
```

Övertyga dig om att du förstår hur loop-summeringen fungerar, det är en vanlig teknik som är mycket användbar.

Kör programmet några gånger. Hur mycket snabbare är Matlabs summeringsrutin jämfört med loopen? Prova att ändra längden på vektorn `x` och se om förhållandet ändras.

Uppgift 7 (Extrauppgift!): Försök att skriva om `my_quad` så att det går betydligt fortare. Jämför med övriga i klassen och se vem som har lyckats bäst.

Uppgift 8 (Extra extrauppgift!!): Försök att göra samma sak med `my_quad2D`! På webben finns en filen `my_quad2DOpt.m` med en variant på hur det kan göras.