Numerické integrovanie

Budeme sa zaoberať funkciou x^x a jej integrálom na intervale [1, 9]. Ako si pamätáme z analýzy, x^x nemá explicitne vyjadriteľnú primitívnu funkciu. Matlab nám to pripomenie:

```
>> f=inline('x.^x');
>> s=sym('s');
>> int(f(s))
                Warning: Explicit integral could not be found.
                > In C:\MATLAB6p5p1\toolbox\symbolic\@sym\int.m at line 58
  ans =int(s^s,s)
```

Odpoveďou je zopakovanie otázky. Analyticky to teda riešiť nejde.[1]

I. Obdĺžniková metóda

Rozdelíme interval na 8 rovnakých častí.

```
>> h=1; x=1:h:9;
```

Vyjadrím si vektor stredov týchto dielov a funkčnú hodnotu v nich:

```
>> n=length(x); xs=(x(2:n)+x(1:n-1))/2; y=f(xs);
```

Numerická aproximácia integrálu obdĺžnikovou metódou bude

```
>> In=h*sum(y)
                        In =    8.330461002341646e+007
```

Odhadnime chybu. Potrebujeme na to ohraničiť druhú deriváciu f na intervale.

```
>> diff(diff(f(s)))
                        ans =s^s*(log(s)+1)^2+s^s/s

>> f2=inline('x.^x.*(log(x)+1).^2+x.^x./x')

>> xh=1:0.0001:9; M2=max(f2(xh))
                                M2 =    4.003353876773043e+009
```

Maximum sme našli numericky. Ak by sme chceli byť dôslednejší,museli by sme si všimnúť, že f2 je kladná a rastúca. Maximum absolútnej hodnoty f2 je preto v pravom bode intervalu. Odhad chyby podľa vzorca je

```
    >> en=M2*8^3/((n-1)^2)/24
                                en =    1.334451292257681e+009
```

[1] Nepriamo je možné vyjadriť riešenie úlohy aj analyticky ako súčet nekonečného radu.

Chyba je fatálna, väčšia než sám výsledok. To sa tak trochu dalo čakať, delenie intervalu je veľmi hrubé a M2 je príliš veľké. Skôr než pôjdeme ďalej, ukážeme si na obrázku, čo sme sa to vlastne pokúsili urobiť:

```
>> plot(xh, f(xh), 'r'), hold on
>> bar(xs, f(xs),1)
```

Obrázok ukazuje, že chyba môže byť ozaj veľká.

Počítajme integrál postupne pre stále hustejšie delenia intervalu. Pripravme sa na to, že matlabu to môže chvíľu trvať.

```
>> for k=2:20, h=h/2; x=1:h:9; n=length(x); xs=(x(2:n)+x(1:n-1))/2; y=f(xs); In=h*sum(y); en=M2*8^3/((n-1)^2)/24; [n-1,In, en], end
```

Počet podintervalov	Integrál	Odhad chyby
1.600000000000000e+001	1.106156932609747e+008	3.336128230644202e+008
3.200000000000000e+001	1.194278743778552e+008	8.340320576610506e+007
6.400000000000000e+001	1.217900671372854e+008	2.085080144152626e+007
1.280000000000000e+002	1.223912682645421e+008	5.212700360381566e+006
2.560000000000000e+002	1.225422461855185e+008	1.303175090095392e+006
5.120000000000000e+002	1.225800332064019e+008	3.257937725238479e+005
1.024000000000000e+003	1.225894826233665e+008	8.144844313096197e+004
2.048000000000000e+003	1.225918451440128e+008	2.036211078274049e+004
4.096000000000000e+003	1.225924357845755e+008	5.090527695685123e+003
8.192000000000000e+003	1.225925834453661e+008	1.272631923921281e+003
1.638400000000000e+004	1.225926203606045e+008	3.181579809803202e+002
3.276800000000000e+004	1.225926295894166e+008	7.953949524508005e+001
6.553600000000000e+004	1.225926318966196e+008	1.988487381127001e+001
1.310720000000000e+005	1.225926324734204e+008	4.971218452817503e+000
2.621440000000000e+005	1.225926326176207e+008	1.242804613204376e+000
5.242880000000000e+005	1.225926326536699e+008	3.107011533010939e-001
1.048576000000000e+006	1.225926326626832e+008	7.767528832527348e-002
2.097152000000000e+006	1.225926326649356e+008	1.941882208131837e-002
4.194304000000000e+006	1.225926326654990e+008	4.854705520329593e-003

Farebne sme zvýraznili postupné zvyšovanie presnosti výsledku. Tabuľka naznačuje, že presnosť rastie a po ďalších zjemneniach delenia intervalu by sme mali dospieť k ešte presnejším hodnotám. Kto má dosť silný počítač s veľkou pamäťou, môže skúsiť…

Otázne je, nakoľko pri tejto konvergencii k presnému výsledku rastie podiel chyby spôsobenej zaokrúhľovaním. To posúdime neskôr. Zatiaľ verme matlabu – na základe zistených údajov môžeme za spoľahlivú hodnotu integrálu považovať číslo z posledného riadku

$$122592632.6655 \pm 0.005$$

Pozrime sa na extrapolované hodnoty integrálu získané metódou polovičného kroku. Z posledných dvoch hodnôt integrálu v našej tabuľke vypočítame spresnenú hodnotu:

```
ii=[1.225926326654990e+008, 1.225926326649356e+008]; (4*ii(1)-ii(2))/3

ans =    1.225926326656868e+008
```

Táto hodnota by mala byť presnejšia, ale nevieme s istotou, nakoľko.

II. Lichobežníková metóda

Pri lichobežníkovej metóde neočakávame vyššiu presnosť, len si to chceme skúsiť v matlabe:

```
>> h=1; x=1:h:9;
>> n=length(x); In=h*(0.5*f(x(1))+sum(f(x(2:n-1)))+0.5*f(x(n)))

        In =   211361072
```

Postupujeme podobne ako pri obdĺžnikovej metóde. V každom kroku si navyše necháme vypísať aj spresnený výsledok metódou polovičného kroku. Nebudeme počítať v cykle, ale ručne krok za krokom, aby sme vychutnali, ako bude výpočet trvať stále viac a viac:

```
>> l=1;
>> h=h/2;l=l+1; x=1:h:9; n=length(x); In(l)=h*(0.5*f(x(1))+sum(f(x(2:n-1)))+0.5*f(x(n))); en=M2*8^3/((n-1)^2)/12;
        Inr=(4*In(l)-In(l-1))/3; [n-1,In(l), en, Inr]
```

Tento riadok budeme volať znovu a znovu. Výpisy:

Počet podintervalov	Integrál	Odhad chyby	Spresnenie
1.600000000000000e+001	1.473328410117082e+008	6.672256461288405e+008	1.259900973489443e+008
3.200000000000000e+001	1.289742671363415e+008	1.668064115322101e+008	1.228547425112192e+008
6.400000000000000e+001	1.242010707570983e+008	4.170160288305253e+007	1.226100052973506e+008
1.280000000000000e+002	1.229955689471919e+008	1.042540072076313e+007	1.225937350105564e+008
2.560000000000000e+002	1.226934186058670e+008	2.606350180190783e+006	1.225927018254254e+008
5.120000000000000e+002	1.226178323956928e+008	6.515875450476958e+005	1.225926369923013e+008
1.024000000000000e+003	1.225989328010473e+008	1.628968862619239e+005	1.225926329361655e+008
2.048000000000000e+003	1.225942077122069e+008	4.072422156548098e+004	1.225926326825935e+008
4.096000000000000e+003	1.225930264281098e+008	1.018105539137025e+004	1.225926326667441e+008
8.192000000000000e+003	1.225927311063426e+008	2.545263847842562e+003	1.225926326657536e+008
1.638400000000000e+004	1.225926572758546e+008	6.363159619606404e+002	1.225926326656919e+008
3.276800000000000e+004	1.225926388182295e+008	1.590789904901601e+002	1.225926326656878e+008
6.553600000000000e+004	1.225926342038236e+008	3.976974762254002e+001	1.225926326656883e+008
1.310720000000000e+005	1.225926330502211e+008	9.942436905635006e+000	1.225926326656869e+008
2.621440000000000e+005	1.225926327618209e+008	2.485609226408752e+000	1.225926326656876e+008
5.242880000000000e+005	1.225926326897203e+008	6.214023066021879e-001	1.225926326656868e+008
1.048576000000000e+006	1.225926326716954e+008	1.553505766505470e-001	1.225926326656871e+008
2.097152000000000e+006	1.225926326671888e+008	3.883764416263674e-002	1.225926326656866e+008
4.194304000000000e+006	1.225926326660622e+008	9.709411040659186e-003	1.225926326656867e+008

Potiaľto sa stroj dostal, na ďalší krok už nestačila pamäť. Je to podobné ako v predošlom prípade. Všimnime si, o koľko krokov skôr dosahuje určitú mieru presnosti výsledok spresnený metódou polovičného kroku.

III. Simpsonova metóda

Simpsonova metóda je pri konkrétnom delení intervalu presnejšia, ale o niečo náročnejšia na výpočet. Okrem samotného výpočtu integrálu nás bude zaujímať, kedy začne mať počítač problémy s objemom operácií a k akej presnosti sa až dopracujeme. Oplatí sa investícia do zložitejších výpočtov? Bude vyvážená tým, čo ušetríme na menej hustom delení intervalu?

```
>> h=1; x=1:h:9; n=length(x); xs=(x(2:n)+x(1:n-1))/2;
        In=(f(x(1))+2*sum(f(x(2:n-1)))+4*sum(f(xs))+f(x(n)))*h/6

                                In =    1.259900973489443e+008
```

Na odhad chyby potrebujeme 4. deriváciu:

```
>> diff(diff(diff(diff(f(s)))))
        ans =
s^s*(log(s)+1)^4+6*s^s*(log(s)+1)^2/s+3*s^s/s^2-4*s^s*(log(s)+1)/s^2+2*s^s/s^3
```

Vidíme (?), že 4. derivácia je kladná a rastúca, maximum dosahuje preto v pravom bode intervalu. Rovno dosadíme:

```
>> s=9;
   M4=s^s*(log(s)+1)^4+6*s^s*(log(s)+1)^2/s+3*s^s/s^2-4*s^s*(log(s)+1)/s^2+2*s^s/s^3

                               M4 =    4.307767766869635e+010
```

Kto tomu neverí, nech nájde s pomocou matlabu maximum f4(xh).

Pokračujme ďalej. Hoci je vzorec na výpočet integrálu zložitejší, počet potrebných operácií nie je rádovo väčší. Mali by sme sa preto dostať až k tomu deleniu intervalu, pri ktorých sme skončili s predošlými metódami. Budeme počítať numericky integrál, odhad chyby a spresnenie integrálu polovičným krokom. Do riadku toho musíme naložiť pomerne veľa. Tu sa začína potreba m-funkcie, ale pokušeniu odoláme.

```
>> l=1;
>> h=h/2; l=l+1; x=1:h:9; n=length(x); xs=(x(2:n)+x(1:n-1))/2;
        In(l)=(f(x(1))+2*sum(f(x(2:n-1)))+4*sum(f(xs))+f(x(n)))*h/6; en=M4*8^5/(n-1)^4/2880;
                Inr=(16*In(l)-In(l-1))/15; [n-1,In(l),en,Inr]
```

Túto sadu príkazov v 1 riadku opakujeme, kým počítač vládze:

Počet podintervalov	Integrál	Odhad chyby	Spresnenie
1.600000000000000e+001	1.228547425112192e+008	7.478763484148672e+006	1.226457188553709e+008
3.200000000000000e+001	1.226100052973506e+008	4.674227177592920e+005	1.225936894830927e+008
6.400000000000000e+001	1.225937350105564e+008	2.921391985995575e+004	1.225926503247701e+008
1.280000000000000e+002	1.225927018254254e+008	1.825869991247234e+003	1.225926329464167e+008
2.560000000000000e+002	1.225926369923013e+008	1.141168744529522e+002	1.225926326700931e+008
5.120000000000000e+002	1.225926329361655e+008	7.132304653309509e+000	1.225926326657565e+008
1.024000000000000e+003	1.225926326825935e+008	4.457690408318443e-001	1.225926326656887e+008
2.048000000000000e+003	1.225926326667442e+008	2.786056505199027e-002	1.225926326656876e+008
4.096000000000000e+003	1.225926326657536e+008	1.741285315749392e-003	1.225926326656876e+008
8.192000000000000e+003	1.225926326656916e+008	1.088303322343370e-004	1.225926326656875e+008
1.638400000000000e+004	1.225926326656878e+008	6.801895764646062e-006	1.225926326656876e+008
3.276800000000000e+004	1.225926326656876e+008	4.251184852903789e-007	1.225926326656875e+008
6.553600000000000e+004	1.225926326656876e+008	2.656990533064868e-008	1.225926326656876e+008
1.310720000000000e+005	1.225926326656873e+008	1.660619083165543e-009	1.225926326656873e+008
2.621440000000000e+005	1.225926326656874e+008	1.037886926978464e-010	1.225926326656874e+008
5.242880000000000e+005	1.225926326656867e+008	6.486793293615400e-012	1.225926326656867e+008
1.048576000000000e+006	1.225926326656873e+008	4.054245808509625e-013	1.225926326656873e+008
2.097152000000000e+006	1.225926326656867e+008	2.533903630318516e-014	1.225926326656866e+008
4.194304000000000e+006	1.225926326656867e+008	1.583689768949072e-015	1.225926326656867e+008

Presnosť rastie oveľa rýchlejšie než pri predošlých metódach. V istom momente sa zdá, akoby sa hodnoty integrálu už ustálili, ale po čase sa to ešte na posledných dvoch pozíciách trochu zmení. Tomu však nesmieme unáhlene veriť. Pri tej číselnej hodnote, ktorú integrál dosahuje, a pri pamäťových možnostiach matlabu, je z čiste technického hľadiska nemožná presnosť vyššia ako zhruba 2e-8 (prečo?). Ak nám vzorec pre en dáva stále menšie hodnoty, presnosť metódy končí tam, kde končí presnosť počítača.

To sme ešte nevzali do úvahy rôzne chyby zaokrúhľovania. Bez toho, aby sme sa nimi podrobne zaoberali, z uvedeného výpisu vidíme, že posledé dve cifry si žiadajú opatrnosť. Predbežne teda určíme hodnotu integrálu orientačne

$$122592632.6656872 \pm 0.0000005$$

Pri všetkej opatrnosti je získaný výsledok oveľa lepší než s použitím metód nižšieho rádu. Skúsme nakoniec použiť ešte metódy samotného matlabu, skryté za príkazom quad. Zvolíme si presnosť primeranú tomu, čo vieme o zobrazovacích možnostiach matlabu, a skúsime mierne zvyšovať nároky:

```
>> quad(f,1,9)
                ans =    1.225926326656876e+008
>> quad(f,1,9,1e-7)
                ans =    1.225926326656876e+008
>> quad(f,1,9,1e-8)
                ans =    1.225926326656876e+008
>> quad(f,1,9,1e-9)
                ans =    1.225926326656876e+008
>> quad(f,1,9,1e-10)
Warning: Maximum function count exceeded; singularity likely. (Type "warning off MATLAB:quad:MaxFcnCount" to
suppress this warning.) In C:\MATLAB6p5p1\toolbox\matlab\funfun\quad.m at line 88
                ans =    1.110973990576650e+008
```

Metódy, ktoré používa matlab, sú o niečo rafinovanejšie než základné Newtonove Cotesove vzorce, takže sa môžeme na ne aj viac spoľahnúť. Výsledok

$$1.225926326656876e+008$$

sa dá považovať za spoľahlivý na toľko miest, koľko je zobrazených.

Ak sa teraz vrátime späť k výpisu hodnôt získaných Simpsonovou metódou, vidíme, že správne sú

```
3.276800000000000e+004   1.225926326656876e+008   4.251184852903789e-007   1.225926326656875e+008
6.553600000000000e+004   1.225926326656876e+008   2.656990533064868e-008   1.225926326656876e+008
```

čiže tie hodnoty, pri ktorých je metodická chyba en zhruba na úrovni chyby zaokrúhľovania a zobrazovania. Akonáhle metodická chyba klesá nižšie, výsledok začína byť menej presný.

Úloha:

1. Počítajte všetkými možnými metódami integrál funkcie x^√x na intervale [1,13].