TF2202 Komputasi Rekayasa

Integral dan Diferensiasi Numerik

Tim Praktikum Komputasi Rekayasa 2021
Teknik Fisika
Institut Teknologi Bandung

Soal 1. Chapra Latihan 21.1-21.7 Untuk integral-integral berikut ini

$$\int_0^{\pi/2} (6 + 3\cos(x))\ \mathrm{d}x \tag{1}$$

$$\int_0^{3} (1 - e^{-2x})\ \mathrm{d}x \tag{2}$$

$$\int_{-2}^{4} (1 - x - 4x^3 + 2x^5)\ \mathrm{d}x \tag{3}$$

$$\int_1^{2} (x - 2/x)^2\ \mathrm{d}x \tag{4}$$

$$\int_{-3}^{5} (4x - 3)^3 \mathrm{d}x \tag{5}$$

$$\int_0^{3} x^2 e^x \mathrm{d}x \tag{6}$$

$$\int_0^{1} 14^{2x}\ \mathrm{d}x \tag{7}$$

dengan menggunakan metode:

(a) analitik

(b) aturan trapesium

(c) aturan 1/3 Simpson

(d) aturan 3/8 Simpson

(e) aturan Boole

Variasikan parameter numerik yang terkait, seperti jumlah titik yang dievaluasi, untuk setiap metode yang digunakan. Bandingkan hasil numerik yang diberikan dengan hasil analitik. Anda dapat menggunakan SymPy untuk menghitung integral secara analitik.

Soal 2. Chapra 21.10 dan 21.11 Hitung integral dari data berikut ini dengan menggunakan aturan trapesium dan aturan Simpson.

Data Chapra 21.10

x	0	0.1	0.2	0.3	0.4	0.5
$f(x)$	1	8	4	3.5	5	1

Data Chapra 21.11

x	-2	0	2	4	6	8	10
$f(x)$	35	5	-10	2	5	3	20

Soal 3. Chapra 21.13 Diketahui data berikut ini dihasilkan dari fungsi $f(x) = 2e^{-1.5x}$

x	0	0.05	0.15	0.25	0.35	0.475	0.6
$f(x)$	2	1.8555	1.5970	1.3746	1.1831	0.9808	0.8131

Hitung integral dari data ini dari $a = 0$ dan $b = 0.6$ dengan menggunakan aturan trapesium dan aturan Simpson. Bandingkan hasil yang Anda peroleh dengan melakukan integrasi $f(x)$ secara analitik.

Soal 4. Chapra Latihan 22.1, 22.2, dan 22.3 Gunakan metode Romberg untuk menghitung integral berikut.

$$\int_0^3 xe^{2x}\,\mathrm{d}x \tag{8}$$

$$\int_1^2 \left(x + \frac{1}{x}\right)^2\,\mathrm{d}x \tag{9}$$

$$\int_0^2 \frac{e^x \sin(x)}{1 + x^2}\,\mathrm{d}x \tag{10}$$

Variasikan parameter numerik yang Anda gunakan, misalnya jumlah maksimal selang yang digunakan. Bandingkan hasil yang Anda dapatkan dengan hasil analitik atau dari SymPy.

Soal 5. Chapra Latihan 22.8 Gunakan formula Gauss-Legendre dengan dua sampai 6 titik untuk menghitung integral

$$\int_{-3}^3 \frac{1}{1 + x^2}\,\mathrm{d}x \tag{11}$$

Bandingkan hasil yang Anda dapatkan dengan hasil analitik atau dari SymPy.

Soal 6. Chapra Latihan 22.9 Gunakan metode integrasi numerik untuk menghitung integral:

$$\int_2^\infty \frac{1}{x(x+2)}\,\mathrm{dx} \tag{12}$$

$$\int_0^\infty e^{-x} \sin^2 x\,\mathrm{dx} \tag{13}$$

Bandingkan hasil yang Anda peroleh dengan hasil analitik atau dari SymPy. Gunakan metode standard (trapesium, Simpson, Romberg, dll) secara langsung dan dengan melakukan penggantian variabel. Anda juga dapat menggunakan kombinasi metode tersebut jika integral yang Anda lakukan dibagi menjadi dua integral (Lihat Contoh 22.6 pada Chapra). Plot juga integran yang terlibat.

Soal 7. Chapra Latihan 23.8 Hitung turunan pertama dari fungsi-fungsi berikut dengan menggunakan metode beda hingga tengah $O(h^4)$:

- $y = x^3 + 4x - 15$ pada $x = 0$ dengan $h = 0.25$
- $y = x^2 \cos(x)$ pada $x = 0.4$ dengan $h = 0.1$
- $y = \tan(x/3)$ pada $x = 3$ dengan $h = 0.5$
- $y = \sin(0.5\sqrt{x})/x$ pada $x = 1$ dengan $h = 0.2$
- $y = e^x + x$ pada $x = 2$ dengan $h = 0.2$

Bandingkan hasil yang Anda peroleh dengan hasil analitik atau SymPy.

Soal 8. Chapra Latihan 23.9 Diketahui data jarak dan waktu yang ditempuh pada suatu roket.

t (s)	0	25	50	75	100	125
y (km)	0	32	58	78	92	100

Gunakan metode numerik untuk untuk mengestimasi kecepatan dan percepatan roket pada masing-masing waktu pada tabel (jika memungkinkan dengan metode yang Anda gunakan).

Kode Python

Contoh penggunaan SymPy

Contoh penggunaan SymPy untuk perhitungan integral tentu.

```
import sympy
x = sympy.symbols("x")
func_symb = 6 + 3*sympy.cos(x)
resExact = sympy.N(sympy.integrate(func_symb, (x, 0, sympy.pi/2)))
# fungsi sympy.N digunakan untuk memaksa hasil dalam bentuk numerik.
```

Untuk batas integral yang melibatkan tak-hingga, Anda dapat menggunakan `sympy.oo`. Contoh:

```
import sympy
x = sympy.symbols("x")
func_symb = sympy.exp(-x)*sympy.sin(x)**2
resExact = sympy.N(sympy.integrate(func_symb, (x, 0, sympy.oo)))
```

Beberapa metode integrasi dengan interval tertutup

Metode trapesium:

```
# Satu interval
def integ_trapz( f, a, b ):
    I = (b - a) * (f(a) + f(b)) / 2
    return I

# Jumlah interval N bisa lebih dari satu
def integ_trapz_multiple( f, a, b, N ):
```

```
    x0 = a
    xN = b
    h = (b - a)/N
    ss = 0.0
    for i in range(1,N):
        xi = x0 + i*h
        ss = ss + f(xi)
    I = h/2 * ( f(x0) + 2*ss + f(xN) )
    return I
```

Metode Boole (lihat Tabel 21.2 pada Chapra):

```
# hanya untuk satu interval, f diberikan dalam bentuk analitik (bukan tabel)
def integ_boole(f, a, b):
    h = (b-a)/4
    x0 = a
    x1 = a + h
    x2 = a + 2*h
    x3 = a + 3*h
    x4 = b

    s = 7*f(x0) + 32*f(x1) + 12*f(x2) + 32*f(x3) + 7*f(x4)
    return (b-a)*s/90
```

Metode Newton-Cotes untuk interval terbuka

Metode Newton-Cotes, interval terbuka (lihat Tabel 21.4 pada Chapra)

```
# hanya untuk satu interval, f diberikan dalam bentuk analitik (bukan tabel)
def integ_newtoncotes_open6seg(f, a, b):
    # menggunakan 6 segmen atau 5 titik
    # silakan gunakan formula lain jika diperlukan
    h = (b-a)/6
    x1 = a + h
    x2 = a + 2*h
    x3 = a + 3*h
    x4 = a + 4*h
    x5 = a + 5*h
    #
    s = 11*f(x1) - 14*f(x2) + 26*f(x3) - 14*f(x4) + 11*f(x5)
    return (b-a)*s/20.0
```

Fungsi Python untuk integral multi-interval yang seragam

Mengaplikasikan aturan integrasi pada beberapa interval homogen dengan menggunakan fungsi aturan integrasi satu interval, diberi nama `quad_func` pada argumen fungsi berikut.

```
def apply_quadrature_multi_interval(quad_func, f, a, b, Ninterval):
    s = 0.0
    Δ = (b-a)/Ninterval
    for i in range(Ninterval):
        aa = a + i*Δ
        bb = a + (i+1)*Δ
```

```
        s = s + quad_func(f, aa, bb)
    return s
```

Contoh penggunaan:

```
from math import cos, pi

import sympy
x = sympy.symbols("x")
func_symb = 6 + 3*sympy.cos(x)
resExact = sympy.N(sympy.integrate(func_symb, (x, 0, sympy.pi/2)))

def my_func(x):
    return 6 + 3*cos(x)

# Pastikan fungsi-fungsi yang terkait sudah di-import atau didefinisikan

a = 0.0
b = pi/2
resN = apply_quadrature_multi_interval(
    integ_newtoncotes_open6seg, my_func, a, b, 10
)
print("resN = %18.10f" % resN)
print("res  = %18.10f" % resExact)
print("err  = %18.10e" % abs(resExact-resN))

resN = apply_quadrature_multi_interval(
    integ_simpson38, my_func, a, b, 10
)
print("\nUsing integ_simpson38 10 intervals")
print("resN = %18.10f" % resN)
print("res  = %18.10f" % resExact)
print("err  = %18.10e" % abs(resExact-resN))

resN = apply_quadrature_multi_interval(
    integ_boole, my_func, a, b, 10
)
print("\nUsing integ_boole 10 intervals")
print("resN = %18.10f" % resN)
print("res  = %18.10f" % resExact)
print("err  = %18.10e" % abs(resExact-resN))
```

Integral Romberg

Metode Romberg untuk integral numerik.

```
# Lihat Gambar 22.4 pada Chapra
def integ_romberg(f, a, b, es=1e-10, MAXIT=10):
    I = np.zeros( (MAXIT+2,MAXIT+2) )
    n = 1
    # We start from I[1,1], to follow the book's notation
    I[1,1] = integ_trapz_multiple(f, a, b, n)
    iterConv = 0
    for i in range(1,MAXIT+1):
        n = 2**i
        I[i+1,1] = integ_trapz_multiple(f, a, b, n)
```

```
        #
        for k in range(2,i+2):
            j = 2 + i - k
            I[j,k] = .... # LENGKAPI
        #
        ea = abs( (I[1,i+1] - I[2,i])/I[1,i+1] )*100 # in percent
        if ea <= es:
            iterConv = i
            #print("converged")
            break
        iterConv = i
    return I[1,iterConv+1]
```

Integral yang melibatkan batas tak-hingga

Tinjau integral:

$$\int_a^b f(x)\,\mathrm{d}x \tag{14}$$

Jika a dan/atau b bernilai tak-hingga maka, kita dapat menggunakan suatu bilangan yang cukup besar atau cukup kecil, atau *practical infinity*, untuk mengaproksimasi tak-hingga dan metode standard seperti aturan trapesium, Simpson, dan lainnya dapat digunakan.

Cara ini efektif apabila integran $f(x)$ mendekati nol pada tak-hingga dan nilai fungsi sudah sangat kecil pada *practical infinity*. Pada beberapa fungsi atau integran yang terlokalisasi pada suatu titik, seperti Gaussian, hal ini cukup efektif.

Cara ini tidak efektif apabila $f(x)$ mendekati nol dengan sangat lambat sehingga *practical infinity* yang digunakan menjadi sangat besar sehingga diperlukan banyak selang untuk mengevaluasi integral. Metode adaptif dapat digunakan untuk mengatasi hal ini.

Metode lain yang dapat digunakan adalah dengan melakukan penggantian variabel. Kita akan menggunakan penggantian variabel $x \longrightarrow \dfrac{1}{t}$ dan melakukan integrasi pada domain t.

Dengan menggunakan substitusi variabel $x \longrightarrow \dfrac{1}{t}$, sehingga $\mathrm{d}x \longrightarrow -\dfrac{1}{t^2}\mathrm{d}t$. Batas integrasi menjadi $x = a \longrightarrow t = 1/a$ dan $x = b \longrightarrow t = 1/b$. Integral pada Pers. (14) menjadi:

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &= \int_{1/a}^{1/b} \left(-\frac{1}{t^2}\right) f\left(\frac{1}{t}\right)\,\mathrm{dt} \\ &= \int_{1/b}^{1/a} \left(\frac{1}{t^2}\right) f\left(\frac{1}{t}\right)\,\mathrm{dt}\end{aligned}$$

Metode integrasi Newton-Cotes selang terbuka biasanya digunakan untuk menghitung integral ini secara numerik (mengapa?).

Sebagai contoh, kita akan menghitung integral:

$$\int_1^\infty \frac{1}{x^3+1}\,\mathrm{d}x \tag{15}$$

SymPy mengalami kesulitan untuk mengevaluasi integral ini secara analitik jika kita menggunakan `sympy.oo` sebagai batas atas, namun kita dapat melakukan aproksimasi dengan menggunakan suatu bilangan yang besar sebagai pengganti dari tak-hingga. Untuk integral yang ada pada Latihan Chapra 22.9, kita dapat menggunakan `sympy.oo` secara langsung.

Sebagai alternatif lain, Anda juga dapat menggunakan Wolfram Alpha atau Mathematica.

Akses laman `https://www.wolframalpha.com/` dan ketikkan teks berikut pada input teks yang tersedia.

```
N[Integrate[1/(x^3 +1), {x,1,Infinity}], 20]
```

Hasil diberikan dalam 20 digit. Anda dapat membandingkannya dengan hasil SymPy.

Berikut ini adalah kode Python yang akan kita gunakan.

```
import sympy
x = sympy.symbols("x")
func_symb = 1/(x**3 + 1)
a = 1.0
# SymPy cannot evaluate this integral when we used sympy.oo
# We use a large number instead
resExact = sympy.N(sympy.integrate(func_symb, (x, a, 1e10)))

#
# Don't forget to import the needed functions or add their
# definitions
#

def my_func(x):
    return 1/(x**3 + 1)

b = 1000.0 # practical, approximate infinity

resApproxInf = sympy.N(sympy.integrate(func_symb, (x, a, b)))

print("\nUsing Boole's rule") # naive
for n in [1, 2, 4, 10, 50, 100, 200, 300, 500, 1000, 10000]:
    resN = apply_quadrature_multi_interval(
        integ_boole, my_func, a, b, n
    )
    print("%5d %18.10f %18.10e %18.10e" % (n,
        resN, abs(resN-resExact), abs(resN-resApproxInf)))

print("\nUsing integ_romberg")
resN = integ_romberg(my_func, a, b, MAXIT=14)
print("resN = %18.10f %18.10e %18.10e" % (resN,
    abs(resN-resExact), abs(resN-resApproxInf)))

def my_func2(t):
    x = 1/t
    return (1/t**2)*my_func(x)

# x -> a, t = 1/x -> 1/a
# x -> oo, t = 1/x -> 0
t1 = 0
t2 = 1/a

print("\nUsing Newton-Cotes 6seg")
for n in [1, 2, 4, 10, 50, 100]:
    resN = apply_quadrature_multi_interval(
        integ_newtoncotes_open6seg, my_func2, t1, t2, n
    )
    print("%5d %18.10f %18.10e" % (n, resN, abs(resN-resExact)))
```

```
print()
print("resExact     = %18.12f" % resExact) # "exact" result from SymPy
print("resApproxInf = %18.12f" % resApproxInf) # result with approximate infinity
print("diff         = %18.10e" % abs(resExact-resApproxInf))
```

Hasil:

```
Using Boole's rule
    1      38.8500245106   3.8476473783e+01   3.8476474283e+01
    2      19.4250984308   1.9051547703e+01   1.9051548203e+01
    4       9.7128865195   9.3393357916e+00   9.3393362916e+00
   10       3.8872647405   3.5137140126e+00   3.5137145126e+00
   50       0.8148676096   4.4131688175e-01   4.4131738175e-01
  100       0.4862953453   1.1274461740e-01   1.1274511740e-01
  200       0.3851574116   1.1606683700e-02   1.1607183700e-02
  300       0.3740278872   4.7715934605e-04   4.7765934605e-04
  500       0.3730326569   5.1807101260e-04   5.1757101260e-04
 1000       0.3735156871   3.5040805719e-05   3.4540805720e-05
10000       0.3735502279   5.0002079310e-07   2.0794199695e-11

Using integ_romberg
converged, iterConv =  13
iterConv =  13
resN =       0.3735503753   3.5260186448e-07   1.4739813442e-07

Using Newton-Cotes 6seg
    1       0.3743445981   7.9387016285e-04
    2       0.3735469160   3.8118722618e-06
    4       0.3735507339   5.9745146053e-09
   10       0.3735507279   3.8686998050e-11
   50       0.3735507279   3.5527136788e-15
  100       0.3735507279   7.7715611724e-16

resExact     =     0.373550727891
resApproxInf =     0.373550227891
diff         =   4.9999999890e-07
```

Program di atas menggunakan nilai 1000 sebagai *practical infinity*. Perbandingan antara integral eksak dengan batas atas tak hingga, `resExact`, dengan integral eksak dengan batas atas *practical infinity*, yaitu `resApproxInf`, diberikan pada bagian akhir. Perbedaannya berada pada orde 10^{-7}, sehingga metode yang menggunakan *practical infinity* tidak akan memiliki ketelitian yang lebih tingga dari ini.

Pada bagian awal diberikan perbandingan nilai yang diberikan dengan menggunakan aturan Boole multi interval. Selain nilai aproksimasi integral, juga diberikan perbedaan antara aproksimasi dengan *resExact* dan `resApproxInf`. Dari hasil yang diberikan, kita melihat bahwa aturan Boole multi interval kesulitan untuk mendapatkan hasil yang akurat.

Penggunaan aturan Romberg dapat mencapai orde kesalahan 10^{-7} dengan kurang lebih 13 level rekursi.

Aturan Newton-Cotes 6-segment multi-interval pada integral yang sudah ditransformasi dapat mencapai orde kesalahan 10^{-11} hanya dengan 10 interval (sekitar 50 evaluasi fungsi).

Perhatikan bahwa penggunaan metode transformasi variabel ini tidak selalu memberikan hasil yang lebih baik. Hal ini sangat bergantung pada integran yang terlibat. Anda disarankan untuk membuat plot dari integran maupun integran yang sudah ditransformasi untuk mendapatkan informasi lebih lanjut sebelum mengaplikasikan metode yang sudah ada.