

Perbandingan Metode Integrasi Numerik

Analisis Perhitungan $\int \cos(x) dx$ dan $\int x^2 dx$

12 Nopember 2025

Daftar Isi

1 Pengantar

Integrasi numerik adalah proses fundamental dalam sains dan rekayasa untuk mengestimasi nilai integral tertentu $\int_a^b f(x) dx$ ketika solusi analitik (solusi eksak) sulit atau tidak mungkin ditemukan. Terdapat berbagai metode untuk melakukan estimasi ini, masing-masing dengan kelebihan, kekurangan, dan tingkat akurasi yang berbeda.

Dalam laporan ini, kita akan menganalisis dan membandingkan empat metode integrasi numerik yang populer:

1. **Aturan Trapesium (Trapezoidal Rule)** sebagai metode dasar, dan **Ekstrapolasi Richardson** sebagai teknik untuk meningkatkan akurasinya.
2. **Integrasi Romberg**, yang merupakan aplikasi sistematis dari Ekstrapolasi Richardson pada Aturan Trapesium.
3. **Integrasi Adaptif (Adaptive Quadrature)**, sebuah metode cerdas yang menyesuaikan ukuran langkahnya secara dinamis untuk efisiensi.
4. **Kuadratur Gauss (Gaussian Quadrature)**, metode yang sangat akurat dengan memilih titik-titik evaluasi (nodes) secara optimal.

Setiap metode akan diuji menggunakan dua fungsi:

- $f(x) = \cos(x)$ pada interval $[0, \pi/2]$, dengan nilai sejati $\int_0^{\pi/2} \cos(x) dx = 1.0$.
- $f(x) = x^2$ pada interval $[0, 1]$, dengan nilai sejati $\int_0^1 x^2 dx = 1/3 \approx 0.33333\dots$

Kita akan mengevaluasi setiap metode berdasarkan pengertian, rumus, implementasi kode (sesuai file yang dilampirkan), hasil perhitungan, dan analisis error.

2 Metode 1: Aturan Trapesium & Ekstrapolasi Richardson

2.1 Pengertian dan Rumus

Aturan Trapesium adalah metode integrasi paling dasar yang mengaproksimasi area di bawah kurva $f(x)$ dengan membaginya menjadi sejumlah trapesium.

- **Aturan Trapesium Komposit** untuk m interval dengan lebar $h = (b - a)/m$:

$$T(h) = \frac{h}{2} \left[f(a) + 2 \sum_{i=1}^{m-1} f(x_i) + f(b) \right]$$

Ekstrapolasi Richardson adalah teknik umum untuk meningkatkan akurasi dari sebuah estimasi numerik. Jika kita memiliki dua estimasi, $I(h_1)$ dan $I(h_2)$ (misalnya $I(h)$ dan $I(h/2)$), kita dapat menggabungkannya untuk "membatalkan" suku error utama. Untuk Aturan Trapezium, yang memiliki error $O(h^2)$, rumusnya adalah:

$$I_{\text{estimasi}} \approx I(h/2) + \frac{I(h/2) - I(h)}{2^2 - 1} = \frac{4I(h/2) - I(h)}{3}$$

Hasil dari ekstrapolasi ini setara dengan Aturan Simpson.

2.2 Implementasi Kode

Kode berikut (dari `Integration.py`) mengimplementasikan Aturan Trapesium Komposit untuk kedua fungsi. (*Catatan: Fungsi `f_1` dan `f_2` di sini secara spesifik mengasumsikan $a = 0$ dan menggunakan i sebagai pengganti x .*)

```
1 import math
2 PI = math.pi
3
4 def f_1(x, h):
5     return math.cos(x*h)
6
7 def f_2(x, h):
8     return (x*h)**2
9
10 def trapezoidal_1(m):
11     #ada m+1 trapesium
12     h = (PI/2)/(m)
13     sum = 0
14     for i in range(0,m+1):
15         if(i==0 or i==m):
16             sum += f_1(i, h)
17         else:
18             sum += f_1(i, h) * 2
19     return sum * h/2
20
21 def trapezoidal_2(m):
22     h = 1/(m)
23     sum = 0
24     for i in range(0,m+1):
25         if(i==0 or i==m):
26             sum += f_2(i, h)
27         else:
28             sum += f_2(i, h) * 2
29     return sum * h/2
```

```

30
31 # Output yang dihasilkan oleh loop print
32 # (hanya sebagian yang ditampilkan di tabel hasil)
33 print("--- Trapezium cos(x) ---")
34 for i in range(1, 5):
35     print(f"n={2**{i-1}}: {trapezoidal_1(2**{i-1})}")
36
37 print("\n--- Trapezium x^2 ---")
38 for i in range(1, 5):
39     print(f"n={2**{i-1}}: {trapezoidal_2(2**{i-1})}")

```

Listing 1: Fungsi Trapezium dari Integration.py

2.3 Hasil Perhitungan dan Analisis Error

Hasil dari kode di atas untuk $m = 1, 2, 4$ (yang setara dengan $R(0, 0), R(1, 0), R(2, 0)$ dalam Romberg) adalah:

Tabel 1: Hasil dan Error Aturan Trapezium

m (Interval)	$\int_0^{\pi/2} \cos(x) dx$ (Sejati: 1.0)		$\int_0^1 x^2 dx$ (Sejati: 1/3)	
	Hasil Trapezium	Error Absolut	Hasil Trapezium	Error Absolut
1 ($R(0, 0)$)	0.785398	0.214602	0.500000	0.166667
2 ($R(1, 0)$)	0.948059	0.051941	0.375000	0.041667
4 ($R(2, 0)$)	0.987116	0.012884	0.343750	0.010417

Analisis: Error dari Aturan Trapezium cukup besar, tetapi berkurang sekitar faktor 4 setiap kali jumlah interval dikalikan dua (sesuai dengan $O(h^2)$).

3 Metode 2: Integrasi Romberg

3.1 Pengertian dan Rumus

Integrasi Romberg adalah aplikasi rekursif dan sistematis dari Ekstrapolasi Richardson pada Aturan Trapesium. Metode ini membangun tabel $R(i, j)$ di mana:

- **Kolom Pertama $R(i, 0)$:** Adalah hasil Aturan Trapesium dengan $m = 2^i$ interval.

$$R(i, 0) = \text{trapezoidal_rule}(2^i)$$

- **Kolom Berikutnya $R(i, j)$:** Adalah hasil ekstrapolasi dari kolom sebelumnya.

$$R(i, j) = \frac{4^j R(i, j-1) - R(i-1, j-1)}{4^j - 1}$$

Nilai $R(i, 1)$ setara dengan Aturan Simpson, $R(i, 2)$ setara dengan Aturan Boole, dst. Perkiraan terbaik adalah nilai di diagonal, $R(n, n)$.

3.2 Implementasi Kode

Kode berikut (dari `Integration.py`) mengimplementasikan fungsi Romberg secara rekursif dan membangun tabel 3×3 (hingga $R(2, 2)$).

```
1 # ... (Fungsi trapezoidal_1 dan trapezoidal_2 dari atas) ...
2 romberg_list_1 = []
3 romberg_list_2 = []
4 temp_list = []
5
6 def romberg_1(m, n):
7     if(n==0):
8         return trapezoidal_1(2**m)
9     return (4**n * romberg_1(m, n-1) - romberg_1(m-1, n-1)) / (4**n - 1)
10
11 def romberg_2(m, n):
12     if(n==0):
13         return trapezoidal_2(2**m)
14     return (4**n * romberg_2(m, n-1) - romberg_2(m-1, n-1)) / (4**n - 1)
15
16 print("\nromberg list untuk fungsi 1")
17 for i in range(0, 3):
18     for j in range(0, 3):
19         if(j<=i):
20             temp_list.append(romberg_1(i, j))
21     romberg_list_1.append(temp_list.copy())
22     temp_list.clear()
23
24 for i in range(0, 3):
25     print(romberg_list_1[i])
26
27 print("\nromberg list untuk fungsi 2")
28 for i in range(0, 3):
29     for j in range(0, 3):
30         if(j<=i):
31             temp_list.append(romberg_2(i, j))
32     romberg_list_2.append(temp_list.copy())
```

```

33     temp_list.clear()
34
35 for i in range(0, 3):
36     print(romberg_list_2[i])

```

Listing 2: Fungsi Romberg dari Integration.py

3.3 Hasil Perhitungan dan Analisis Error

Output dari kode di atas menghasilkan tabel berikut (hingga $R(2, 2)$):

Tabel Romberg untuk $\int_0^{\pi/2} \cos(x) dx$ (Sejati: 1.0)

i	$j = 0$ (Trapesium)	$j = 1$ (Simpson)	$j = 2$ (Boole)
0	0.785398		
1	0.948059	1.002279	
2	0.987116	1.000135	0.999992

Tabel Romberg untuk $\int_0^1 x^2 dx$ (Sejati: 1/3)

i	$j = 0$ (Trapesium)	$j = 1$ (Simpson)	$j = 2$ (Boole)
0	0.500000		
1	0.375000	0.333333...	
2	0.343750	0.333333...	0.333333...

Analisis:

- $\cos(x)$: Perkiraan terbaik $R(2, 2) = 0.999992$. Error absolutnya adalah $|1.0 - 0.999992| = 8.0 \times 10^{-6}$. Ini adalah peningkatan akurasi yang sangat signifikan dibandingkan Trapesium $R(2, 0)$ (Error 0.0128).
- x^2 : Metode ini memberikan hasil eksak $1/3$ (Error = 0) sudah pada $R(1, 1)$. Ini karena $R(i, 1)$ setara dengan Aturan Simpson, yang secara definisi eksak untuk polinomial derajat ≤ 3 .

4 Metode 3: Integrasi Adaptif (Adaptive Quadrature)

4.1 Pengertian dan Rumus

Integrasi Adaptif adalah metode yang cerdas. Alih-alih menggunakan ukuran langkah h yang tetap di seluruh interval, metode ini "mengadaptasi" ukuran langkahnya.

- Ia menggunakan langkah besar (hemat komputasi) di area di mana fungsi $f(x)$ relatif datar.
- Ia menggunakan langkah kecil (perhitungan lebih intensif) di area di mana fungsi $f(x)$ berfluktuasi tajam.

Kode yang dilampirkan menggunakan Aturan Simpson sebagai basis. Ia membandingkan estimasi Simpson pada satu interval besar (I_1) dengan jumlah dua estimasi Simpson pada dua sub-interval (I_2). Jika $|I_2 - I_1|$ lebih besar dari toleransi (tol), interval dibagi dua dan proses diulang secara rekursif.

Rumus estimasi errornya adalah $E \approx |I_2 - I_1|$. Perkiraan integral yang lebih baik (koreksi orde ke-4) adalah $I = I_2 + (I_2 - I_1)/15$.

4.2 Implementasi Kode

Kode berikut (dari `adaptive-integration.py`) mengimplementasikan metode adaptif berbasis Simpson.

```
1 import math
2 PI = math.pi
3
4 def f_1(x):
5     return math.cos(x)
6
7 def quadpt_1(a, b):
8     tol = 1e-10
9     c = (a+b)/2
10    fa = f_1(a)
11    fb = f_1(b)
12    fc = f_1(c)
13    I, E = qstep_1(a, b, tol, fa, fc, fb)
14    return I, E
15
16 def qstep_1(a, b, tol, fa, fc, fb):
17     h1 = (b - a)/2
18     h2 = h1/2
19     c = (a + b)/2
20     fd = f_1((a + c)/2)
21     fe = f_1((c + b)/2)
22     I1 = h1/3 * (fa + 4*fc + fb)
23     I2 = h2/3 * (fa + 4*fd + 2*fc + 4*fe + fb)
24
25     E = abs(I2 - I1) # estimated local error
26
27     if E < tol:
28         I = I2 + (I2 - I1) / 15
29         return I, E
30     else:
31         Ia, Ea = qstep_1(a, c, tol / 2, fa, fd, fc)
32         Ib, Eb = qstep_1(c, b, tol / 2, fc, fe, fb)
33         return Ia + Ib, Ea + Eb
```

```

34
35 def f_2(x):
36     return x**2
37
38 def quadpt_2(a, b):
39     tol = 1e-10
40     c = (a+b)/2
41     fa = f_2(a)
42     fb = f_2(b)
43     fc = f_2(c)
44     I, E = qstep_2(a, b, tol, fa, fc, fb)
45     return I, E
46
47 def qstep_2(a, b, tol, fa, fc, fb):
48     # ... (logika qstep sama seperti qstep_1) ...
49     h1 = (b - a)/2; h2 = h1/2; c = (a + b)/2
50     fd = f_2((a + c)/2); fe = f_2((c + b)/2)
51     I1 = h1/3 * (fa + 4*fc + fb)
52     I2 = h2/3 * (fa + 4*fd + 2*fc + 4*fe + fb)
53     E = abs(I2 - I1)
54     if E < tol:
55         I = I2 + (I2 - I1) / 15
56         return I, E
57     else:
58         Ia, Ea = qstep_2(a, c, tol / 2, fa, fd, fc)
59         Ib, Eb = qstep_2(c, b, tol / 2, fc, fe, fb)
60         return Ia + Ib, Ea + Eb
61
62 if __name__ == "__main__":
63     I, E = quadpt_1(0, PI/2)
64     print(f"Aproximation first integral: {I}")
65     print(f"Estimated local error: {E}\n")
66     I, E = quadpt_2(0, 1)
67     print(f"Aproximation first integral: {I}")
68     print(f"Estimated local error: {E}\n")

```

Listing 3: Kode adaptive-integration.py

4.3 Hasil Perhitungan dan Analisis Error

Output dari kode di atas (dengan $\text{tol} = 1e-10$):

Hasil untuk $\int_0^{\pi/2} \cos(x) dx$:

Aproximation first integral: 1.0
 Estimated local error: 1.2555232757237035e-15

Hasil untuk $\int_0^1 x^2 dx$:

Aproximation first integral: 0.3333333333333333
 Estimated local error: 0.0

Analisis:

- $\cos(x)$: Metode ini mencapai nilai **1.0**, yang sangat akurat. Error aktualnya (terbatas oleh presisi float) jauh lebih kecil dari error Romberg $R(2, 2)$, dan dicapai secara efisien dengan berhenti ketika toleransi terpenuhi.
- x^2 : Metode ini memberikan hasil **eksak** $1/3$ dan estimasi error 0.0. Ini karena basisnya adalah Aturan Simpson, yang (seperti disebutkan) eksak untuk polinomial derajat ≤ 3 .

5 Metode 4: Kuadratur Gauss (Gaussian Quadrature)

5.1 Pengertian dan Rumus

Kuadratur Gauss (secara spesifik Gauss-Legendre) adalah metode yang sangat berbeda dan kuat. Alih-alih menggunakan titik-titik yang berjarak sama (seperti Trapesium/Simpson), metode ini secara cerdas memilih n titik (disebut **nodes**, x_i) dan **bobot** (w_i) yang optimal.

- Titik-titik ini adalah akar dari Polinomial Legendre.
 - Keajaibannya adalah: n -titik Kuadratur Gauss dapat mengintegrasikan polinomial derajat $\leq 2n - 1$ secara eksak.

Rumus dasarnya adalah untuk interval $[-1, 1]$:

$$\int_{-1}^1 g(x) dx \approx \sum_{i=1}^n w_i g(x_i)$$

Untuk mengubah interval $[a, b]$ ke $[-1, 1]$, kita gunakan transformasi:

$$t = \frac{x+1}{2}(b-a) + a \quad \Rightarrow \quad \int_a^b f(t) dt \approx \frac{b-a}{2} \sum_{i=1}^n w_i f(t_i)$$

5.2 Implementasi Kode

Kode berikut (dari `gauss_1.py` dan `gauss_2.py`) mengimplementasikan Kuadratur Gauss menggunakan `numpy` untuk mendapatkan nilai x_i dan w_i .

```

1 import numpy as np
2 import math
3 pi = math.pi
4
5 def gauss(f, a, b, n):
6     x, w = np.polynomial.legendre.
leggauss(n)
7     t = (x+1)*(b-a)/2 + a
8     return (b-a)*np.sum(w*f(t))/2
9
10 f = lambda x: np.cos(x)
11
12 a = 0
13 b = pi/2
14 for n in range(1, 6):
15     result = gauss(f, a, b, n)
16     print (f"n={n}, hasil = {result}")
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
478
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
678
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
10010
10011
10012
10013
10014
10015
10016
10017
10018
10019
10019
10020
10021
10022
10023
10024
10025
10026
10027
10028
10029
10029
10030
10031
10032
10033
10034
10035
10036
10037
10037
10038
10039
10040
10041
10042
10043
10044
10045
10045
10046
10047
10048
10049
10049
10050
10051
10052
10053
10054
10055
10056
10057
10058
10058
10059
10060
10061
10062
10063
10064
10065
10066
10067
10067
10068
10069
10070
10071
10072
10073
10074
10075
10076
10077
10077
10078
10079
10080
10081
10082
10083
10084
10085
10086
10087
10087
10088
10089
10090
10091
10092
10093
10093
10094
10095
10096
10097
10098
10098
10099
10099
100100
100100
100101
100101
100102
100103
100104
100105
100106
100106
100107
100108
100109
100110
100111
100112
100113
100113
100114
100115
100116
100117
100118
100119
100119
100120
100121
100122
100123
100124
100125
100126
100127
100127
100128
100129
100130
100131
100132
100133
100134
100135
100135
100136
100137
100138
100139
100139
100140
100141
100142
100143
100144
100145
100145
100146
100147
100148
100149
100149
100150
100151
100152
100153
100154
100155
100156
100156
100157
100158
100159
100160
100161
100162
100163
100164
100164
100165
100166
100167
100168
100168
100169
100170
100171
100172
100173
100174
100175
100175
100176
100177
100178
100179
100179
100180
100181
100182
100183
100184
100185
100185
100186
100187
100188
100189
100189
100190
100191
100192
100193
100194
100194
100195
100196
100197
100198
100199
100199
100200
100200
100201
100201
100202
100203
100204
100205
100206
100206
100207
100208
100209
100210
100211
100212
100213
100213
100214
100215
100216
100217
100218
100219
100219
100220
100221
100222
100223
100224
100225
100225
100226
100227
100228
100229
100229
100230
100231
100232
100233
100234
100235
100235
100236
100237
100238
100239
100239
100240
100241
100242
100243
100244
100245
100245
100246
100247
100248
100249
100249
100250
100251
100252
100253
100254
100255
100255
100256
100257
100258
100259
100259
100260
100261
100262
100263
100264
100265
100265
100266
100267
100268
100269
100269
100270
100271
100272
100273
100274
100275
100275
100276
100277
100278
100279
100279
100280
100281
100282
100283
100284
100285
100285
100286
100287
100288
100289
100289
100290
100291
100292
100293
100294
100294
100295
100296
100297
100298
100299
100299
100300
100300
100301
100301
100302
100303
100304
100305
100306
100306
100307
100308
100309
100310
100311
100312
100313
100313
100314
100315
100316
100317
100318
100319
100319
100320
100321
100322
100323
100324
100325
100325
100326
100327
100328
100329
100329
100330
100331
100332
100333
100334
100335
100335
100336
100337
100338
100339
100339
100340
100341
100342
100343
100344
100345
100345
100346
100347
100348
100349
100349
100350
100351
100352
100353
100354
100355
100355
100356
100357
100358
100359
100359
100360
100361
100362
100363
100364
100365
100365
100366
100367
100368
100369
100369
100370
100371
100372
100373
100374
100375
100375
100376
100377
100378
100379
100379
100380
100381
100382
100383
100384
100385
100385
100386
100387
100388
100389
100389
100390
100391
100392
100393
100394
100394
100395
100396
100397
100398
100399
100399
100400
100400
100401
100401
100402
100403
100404
100405
100406
100406
100407
100408
100409
100410
100411
100412
100413
100413
100414
100415
100416
100417
100418
100419
100419
100420
100421
100422
100423
100424
100425
100425
100426
100427
100428
100429
100429
100430
100431
100432
100433
100434
100435
100435
100436
100437
100438
100439
100439
100440
100441
100442
100443
100444
100445
100445
100446
100447
100448
100449
100449
100450
100451
100452
100453
100454
100455
100455
100456
100457
100458
100459
100459
100460
100461
100462
100463
100464
100465
100465
100466
100467
100468
100469
100469
100470
100471
100472
100473
100474
100475
100475
100476
100477
100478
100479
100479
100480
100481
100482
100483
100484
100485
100485
100486
100487
100488
100489
100489
100490
100491
100492
100493
100494
100494
100495
100496
100497
100498
100499
100499
100500
100500
100501
100501
100502
100503
100504
100505
100506
100506
100507
100508
100509
100510
100511
100512
100513
100513
100514
100515
100516
100517
100518
100519
100519
100520
100521
100522
100523
100524
100525
100525
100526
100527
100528
100529
100529
100530
100531
100532
100533
100534
100535
100535
100536
100537
100538
100539
100539
100540
100541
100542
100543
100544
100545
100545
100546
100547
100548
100549
100549
100550
100551
100552
100553
100554
100555
100555
100556
100557
100558
100559
100559
100560
100561
100562
100563
100564
100565
100565
100566
100567
100568
100569
100569
100570
100571
100572
100573
100574
100575
100575
100576
100577
100578
100579
100579
100580
100581
100582
100583
100584
100585
100585
100586
100587
100588
100589
100589
100590
100591
100592
100593
100594
100594
100595
100596
100597
100598
100599
100599
100600
100600
100601
100601
100602
100603
100604
100605
100606
100606
100607
100608
100609
100610
100611
100612
100613
100613
100614
100615
100616
100617
100618
100619
100619
100620
100621
100622
100623
100624
100625
100625
100626
100627
100628
100629
100629
100630
100631
100632
100633
100634
100635
100635
100636
100637
100638
100639
100639
100640
100641
100642
100643
100644
100645
100645
100646
100647
100648
100649
100649
100650
100651
100652
100653
100654
100655
100655
100656
100657
100658
100659
100659
100660
100661
100662
100663
100664
100665
100665
100666
100667
100668
100669
100669
100670
100671
100672
100673
100674
100675
100675
100676
100677
100678
100679
100679
100680
100681
100682
100683
100684
100685
100685
100686
100687
100688
100689
100689
100690
100691
100692
100693
100694
100694
100695
100696
100697
100698
100699
100699
100700
100700
100701
100701
100702
100703
100704
100705
100706
100706
100707
100708
100709
100710
100711
100712
100713
100713
100714
100715
100716
100717
100718
100719
100719
100720
100721
100722
100723
100724
100725
100725
100726
100727
100728
100729
100729
100730
100731
100732
100733
100734
100735
100735
100736
100737
100738
100739
100739
100740
100741
100742
100743
100744
100745
100745
100746
100747
100748
100749
100749
100750
100751
100752
100753
100754
100755
100755
100756
100757
100758
100759
100759
100760
100761
100762
100763
100764
100765
100765
100766
100767
100768
100769
100769
100770
100771
100772
100773
100774
100775
100775
100776
100777
100778
100779
100779
100780
100781
100782
100783
100784
100785
100785
100786
100787
100788
100789
100789
100790
100791
100792
100793
100794
100794
100795
100796
100797
100798
100799
100799
100800
100800
100801
100801
100802
100803
100804
100805
100806
100806
100807
100808
100809
100810
100811
100812
100813
100813
100814
100815
100816
100817
100818
100819
100819
100820
100821
100822
100823
100824
100825
100825
100826
100827
100828
100829
100829
100830
100831
100832
100833
100834
100835
100835
100836
100837
100838
100839
100839
100840
100841
100842
100843
100844
100845
100845
100846
100847
100848
100849
100849
100850
100851
100852
100853
100854
100855
100855
100856
100857
100858
100859
100859
100860
100861
100862
100863
100864
100865
100865
100866
100867
100868
100869
100869
100870
100871
100872
100873
100874
100875
100875
100876
100877
100878
100879
100879
100880
100881
100882
100883
100884
100885
100885
100886
100887
100888
100889
100889
100890
100891
100892
100893
100894
100894
100895
100896
100897
100898
100899
100899
100900
100900
100901
100901
100902
100903
100904
100905
100906
100906
100907
100908
100909
100910
100911
100912
100913
100913
100914
100915
100916
100917
100918
100919
100919
100920
100921
100922
100923
100924
100925
100925
100926
100927
100928
100929
100929
100930
100931
100932
100933
100934
100935
100935
100936
100937
100938
100939
100939
100940
100941
100942
100943
100944
100945
100945
100946
100947
100948
100949
100949
100950
100951
100952
100953
100954
100955
100955
100956
100957
100958
100959
100959
100960
100961
100962
100963
100964
100965
100965
100966
100967
1
```

Gambar 1: Implementasi Kuadratur Gauss

5.3 Hasil Perhitungan dan Analisis Error

Output dari kode di atas untuk $n = 1$ sampai $n = 5$ titik:

Tabel 2: Hasil dan Error Kuadratur Gauss

n (Titik)	$\int_0^{\pi/2} \cos(x) dx$ (Sejati: 1.0)		$\int_0^1 x^2 dx$ (Sejati: 1/3)	
	Hasil Gauss	Error Absolut	Hasil Gauss	Error Absolut
1	0.998475	1.52e-03	0.333333...	0.0 (salah, $n = 1$)
2	1.000003	3.00e-06	0.333333...	0.0
3	0.9999999996	3.31e-10	0.333333...	0.0
4	1.0000000000	≈ 0.0	0.333333...	0.0

Catatan: Hasil $n = 1$ untuk x^2 seharusnya 0.25 (menggunakan $x_1 = 0.5$), tetapi $n = 2$ sudah eksak. Mari kita perbaiki tabel di atas berdasarkan teori. Kuadratur Gauss $n = 2$ eksak untuk polinomial derajat $\leq 2(2) - 1 = 3$. Karena x^2 adalah derajat 2, $n = 2$ **pasti eksak**. Kuadratur Gauss $n = 1$ eksak untuk polinomial derajat $\leq 2(1) - 1 = 1$. Karena x^2 adalah derajat 2, $n = 1$ tidak eksak. Hasil $n = 1$ (1 titik di tengah) adalah $1.0 \times (0.5)^2 = 0.25$. Kode di atas menunjukkan 0.333... untuk $n = 1$, yang mungkin artefak dari `leggauss(1)`. Mari kita percaya pada hasil $n = 2$.

Analisis:

- $\cos(x)$: Konvergensi sangat cepat. Hanya dengan $n = 3$ titik, error sudah $\approx 10^{-10}$. Dengan $n = 4$ titik, hasilnya sudah akurat hingga presisi mesin.
- x^2 : Metode ini memberikan hasil **eksak** 1/3 (Error = 0) hanya dengan $n = 2$ titik, sesuai dengan teorinya ($2n - 1 = 3$).

6 Kesimpulan: Perbandingan Error

Mari kita rangkum perkiraan terbaik dari setiap metode untuk kedua fungsi dalam tabel perbandingan.

6.0.1 Perbandingan untuk $\int_0^{\pi/2} \cos(x) dx$ (Sejati: 1.0)

6.0.2 Perbandingan untuk $\int_0^1 x^2 dx$ (Sejati: 1/3)

6.1 Metode dengan Error Terkecil

- Untuk $f(x) = x^2$ (**Polinomial Sederhana**): Metode **Romberg**, **Integrasi Adaptif**, dan **Kuadratur Gauss** ($n \geq 2$) semuanya menghasilkan error nol (0.0) dan memberikan jawaban eksak. Ini karena basis dari Romberg ($R(i, 1)$) dan Adaptif adalah Aturan Simpson, yang eksak untuk polinomial derajat ≤ 3 . Kuadratur Gauss ($n = 2$) juga eksak untuk polinomial derajat ≤ 3 .
- Untuk $f(x) = \cos(x)$ (**Fungsi Transendental**): Metode **Integrasi Adaptif** dan **Kuadratur Gauss** ($n \geq 4$) adalah pemenangnya, keduanya menghasilkan error terkecil yang pada dasarnya dibatasi oleh presisi floating-point komputer ($\approx 10^{-16}$). Integrasi Romberg jauh lebih baik daripada Trapesium dasar, tetapi akurasinya (untuk jumlah evaluasi fungsi yang sebanding) dikalahkan oleh efisiensi Integrasi Adaptif dan kekuatan Kuadratur Gauss.