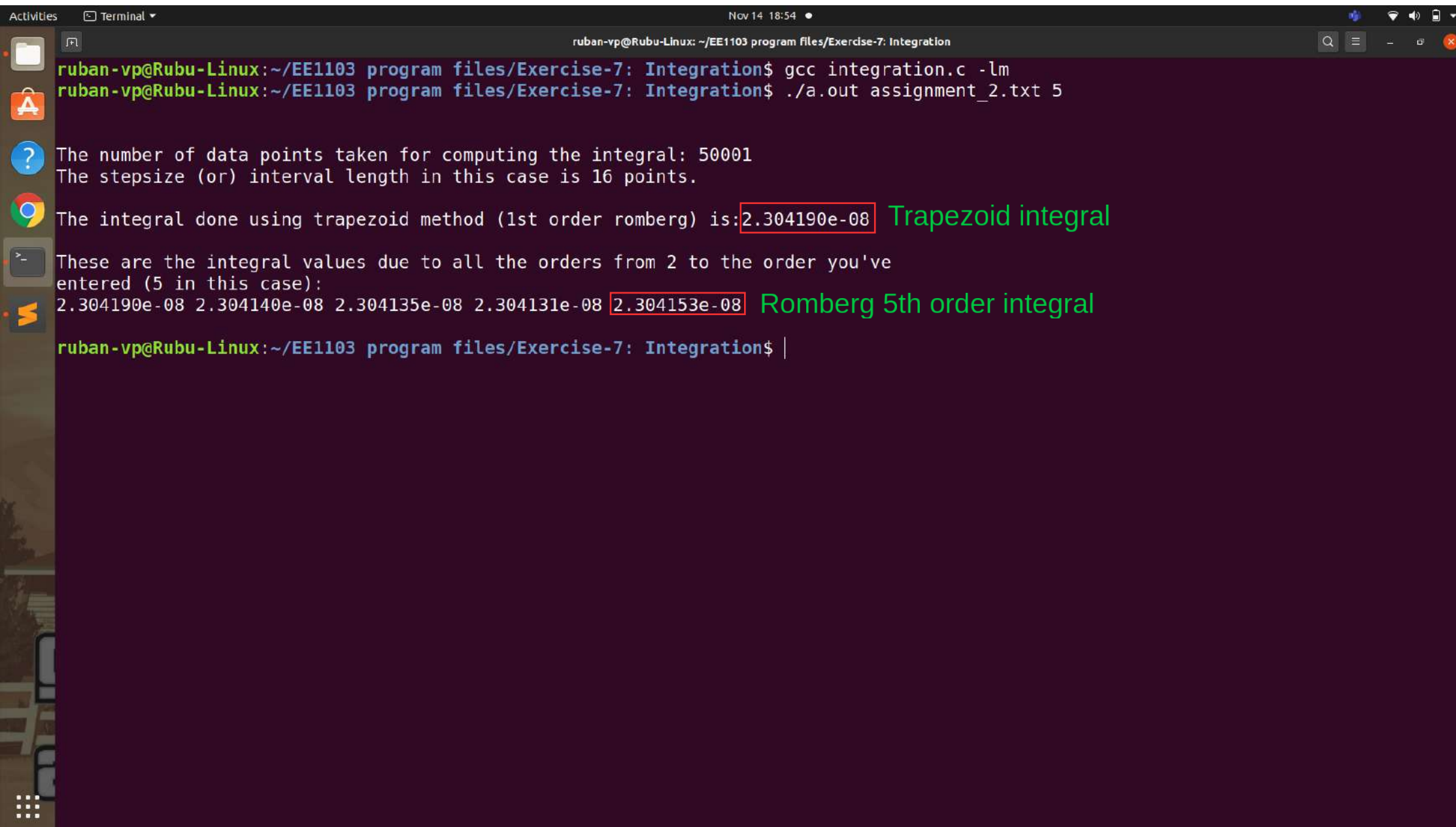


The Trapezoid and Romberg integrals are calculated for order 5 and displayed



The image shows a terminal window titled "ruban-vp@Rubu-Linux: ~/EE1103 program files/Exercise-7: Integration". The terminal displays the following commands and output:

```
ruban-vp@Rubu-Linux:~/EE1103 program files/Exercise-7: Integration$ gcc integration.c -lm
ruban-vp@Rubu-Linux:~/EE1103 program files/Exercise-7: Integration$ ./a.out assignment_2.txt 5
```

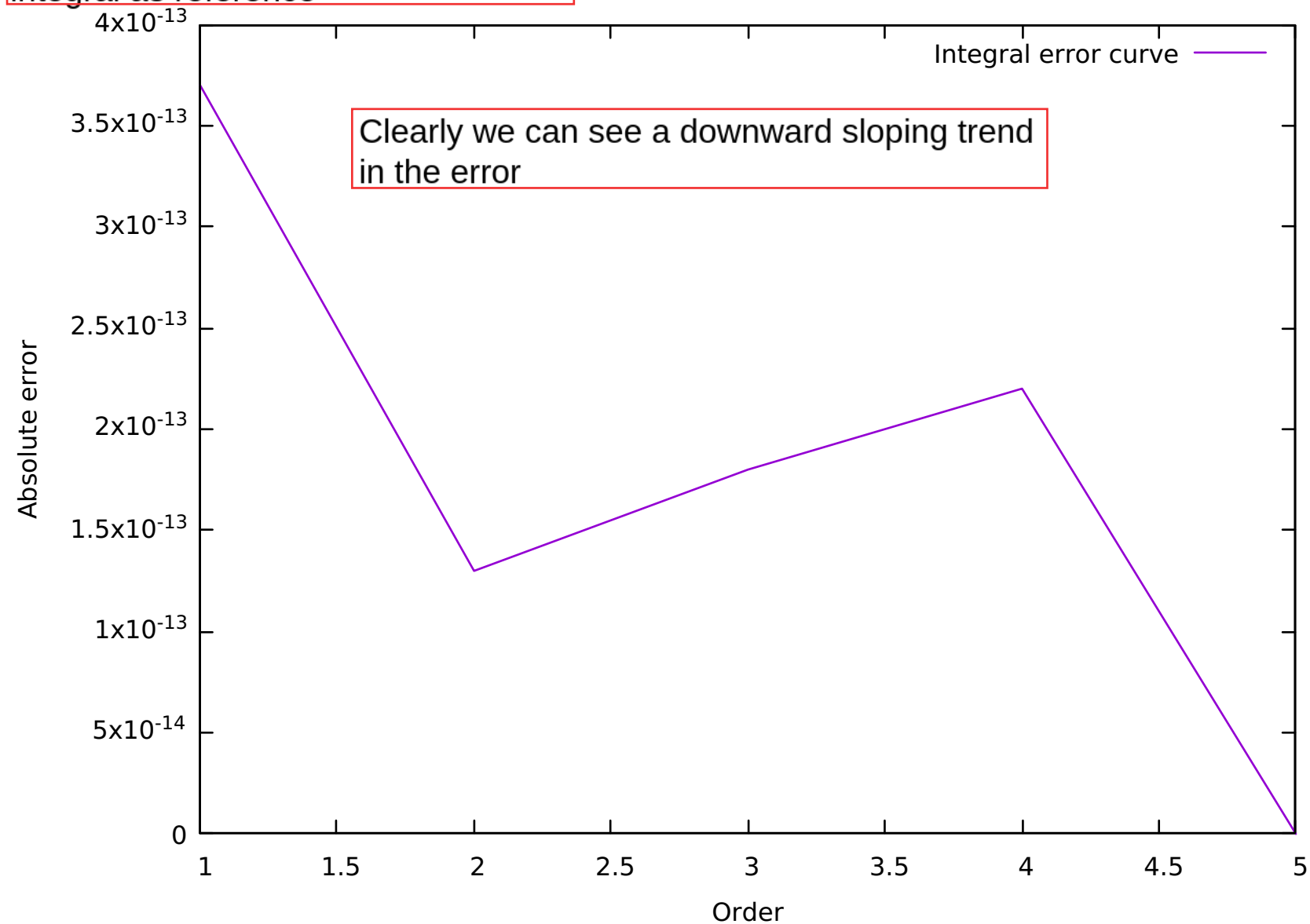
The program outputs the following information:

- The number of data points taken for computing the integral: 50001
- The stepsize (or) interval length in this case is 16 points.
- The integral done using trapezoid method (1st order romberg) is: **2.304190e-08** Trapezoid integral
- These are the integral values due to all the orders from 2 to the order you've entered (5 in this case):
2.304190e-08 2.304140e-08 2.304135e-08 2.304131e-08 **2.304153e-08** Romberg 5th order integral

The terminal prompt is now `ruban-vp@Rubu-Linux:~/EE1103 program files/Exercise-7: Integration$`.

Integral error curve with the 5th order
integral as reference

Integral absolute error



Profiling data: For order=5

ActivitiesSublime Text

Nov 14 18:48

~/EE1103 program files/Exercise-7: Integration/profile.txt - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

integration.c x t.txt x EE198138_4.sh x auto.sh x profile.txt x

1 Flat profile:
2
3 Each sample counts as 0.01 seconds.
4 no time accumulated
5
6 % cumulative self self total
7 time seconds seconds calls Ts/call Ts/call name
8 0.00 0.00 0.00 3125 0.00 0.00 romberg
9 0.00 0.00 0.00 3125 0.00 0.00 trapezoid
10
11 % the percentage of the total running time of the
12 time program used by this function.
13
14 cumulative a running sum of the number of seconds accounted
15 seconds for by this function and those listed above it.
16
17 self the number of seconds accounted for by this
18 seconds function alone. This is the major sort for this
19 listing.
20
21 calls the number of times this function was invoked, if
22 this function is profiled, else blank.
23
24 self the average number of milliseconds spent in this
25 ms/call function per call, if this function is profiled,
26 else blank.
27
28 total the average number of milliseconds spent in this
29 ms/call function and its descendents per call, if this
30 function is profiled, else blank.
31
32 name the name of the function. This is the minor sort
33 for this listing. The index shows the location of
34 the function in the gprof listing. If the index is
35 in parenthesis it shows where it would appear in
36 the gprof listing if it were to be printed.
37 <0x0c>
38 Copyright (C) 2012-2020 Free Software Foundation, Inc.
39
40 Copying and distribution of this file, with or without modification,
41 are permitted in any medium without royalty provided the copyright
42 notice and this notice are preserved.
43 <0x0c>
44 Call graph (explanation follows)
45
46
47 granularity: each sample hit covers 2 byte(s) no time propagated
48
49 index % time self children called name

We can't see any time taken as the code is extremely quick such that the time taken is less than the least count

Line 1, Column 1Tab Size: 4Plain Text

File Edit Selection Find View Goto Tools Project Preferences Help

integration.c x t.txt x EE198138_4.sh x auto.sh x profile.txt x

```
49 index % time self children called name
50 0.00 0.00 0.00 3125/3125 main [8]
51 [1] 0.0 0.00 0.00 3125 romberg [1]
52 -----
53 0.00 0.00 0.00 3125/3125 main [8]
54 [2] 0.0 0.00 0.00 3125 trapezoid [2]
55 -----
56
```

57 This table describes the call tree of the program, and was sorted by
58 the total amount of time spent in each function and its children.

60 Each entry in this table consists of several lines. The line with the
61 index number at the left hand margin lists the current function.
62 The lines above it list the functions that called this function,
63 and the lines below it list the functions this one called.

64 This line lists:

65 index A unique number given to each element of the table.

66 Index numbers are sorted numerically.

67 The index number is printed next to every function name so
68 it is easier to look up where the function is in the table.

70 % time This is the percentage of the 'total' time that was spent
71 in this function and its children. Note that due to
72 different viewpoints, functions excluded by options, etc,
73 these numbers will NOT add up to 100%.

75 self This is the total amount of time spent in this function.

77 children This is the total amount of time propagated into this
78 function by its children.

80 called This is the number of times the function was called.

81 If the function called itself recursively, the number
82 only includes non-recursive calls, and is followed by
83 a '+' and the number of recursive calls.

85 name The name of the current function. The index number is
86 printed after it. If the function is a member of a
87 cycle, the cycle number is printed between the
88 function's name and the index number.

91 For the function's parents, the fields have the following meanings:

93 self This is the amount of time that was propagated directly
94 from the function into this parent.

96 children This is the amount of time that was propagated from
97 the function's children into this parent.

File Edit Selection Find View Goto Tools Project Preferences Help

integration.c x t.txt x EE198138.4.sh x auto.sh x profile.txt x

```
99      called This is the number of times this parent called the
100      function '/' the total number of times the function
101      was called. Recursive calls to the function are not
102      included in the number after the '/'.
103
104      name This is the name of the parent. The parent's index
105      number is printed after it. If the parent is a
106      member of a cycle, the cycle number is printed between
107      the name and the index number.
108
109      If the parents of the function cannot be determined, the word
110      '<spontaneous>' is printed in the 'name' field, and all the other
111      fields are blank.
112
113      For the function's children, the fields have the following meanings:
114
115      self This is the amount of time that was propagated directly
116      from the child into the function.
117
118      children This is the amount of time that was propagated from the
119      child's children to the function.
120
121      called This is the number of times the function called
122      this child '/' the total number of times the child
123      was called. Recursive calls by the child are not
124      listed in the number after the '/'.
125
126      name This is the name of the child. The child's index
127      number is printed after it. If the child is a
128      member of a cycle, the cycle number is printed
129      between the name and the index number.
130
131      If there are any cycles (circles) in the call graph, there is an
132      entry for the cycle-as-a-whole. This entry shows who called the
133      cycle (as parents) and the members of the cycle (as children.)
134      The '+' recursive calls entry shows the number of function calls that
135      were internal to the cycle, and the calls entry for each member shows,
136      for that member, how many times it was called from other members of
137      the cycle.
138      <0x0c>
139      Copyright (C) 2012-2020 Free Software Foundation, Inc.
140
141      Copying and distribution of this file, with or without modification,
142      are permitted in any medium without royalty provided the copyright
143      notice and this notice are preserved.
144      <0x0c>
145      Index by function name
146
147      [1] romberg          [2] trapezoid
```

Line 1, Column 1

Tab Size: 4

Plain Text

Profiling data: For order=10

Activities Sublime Text Nov 14 18:52 ~/EE1103 program files/Exercise-7: Integration/profile.txt - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

integration.c x t.txt x EE198138_4.sh x profile.txt x auto.sh x

```
1 Flat profile:
2
3 Each sample counts as 0.01 seconds.
4 % cumulative self self total
5 time seconds seconds calls Ts/call Ts/call name
6 100.38 0.01 0.01 97 0.00 0.00 main
7 0.00 0.01 0.00 97 0.00 0.00 romberg
8 0.00 0.01 0.00 97 0.00 0.00 trapezoid
9
10 % the percentage of the total running time of the
11 time program used by this function.
12
13 cumulative a running sum of the number of seconds accounted
14 seconds for by this function and those listed above it.
15
16 self the number of seconds accounted for by this
17 seconds function alone. This is the major sort for this
18 listing.
19
20 calls the number of times this function was invoked, if
21 this function is profiled, else blank.
22
23 self the average number of milliseconds spent in this
24 ms/call function per call, if this function is profiled,
25 else blank.
26
27 total the average number of milliseconds spent in this
28 ms/call function and its descendants per call, if this
29 function is profiled, else blank.
30
31 name the name of the function. This is the minor sort
32 for this listing. The index shows the location of
33 the function in the gprof listing. If the index is
34 in parenthesis it shows where it would appear in
35 the gprof listing if it were to be printed.
36 <0x0c>
37 Copyright (C) 2012-2020 Free Software Foundation, Inc.
38
39 Copying and distribution of this file, with or without modification,
40 are permitted in any medium without royalty provided the copyright
41 notice and this notice are preserved.
42 <0x0c>
43 Call graph (explanation follows)
44
45
46 granularity: each sample hit covers 2 byte(s) for 99.62% of 0.01 seconds
47
48 index % time self children called name
49
```

Line 76, Column 66 Tab Size: 4 Plain Text

Here too we aren't able to see any noticeable time taken even for a higher order. The processor is that quick

File Edit Selection Find View Goto Tools Project Preferences Help

integration.c x t.txt x EE198138-4.sh x profile.txt x auto.sh x

```

49 <spontaneous>
50 [1] 100.0 0.01 0.00 main [1]
51      0.00 0.00 97/97 trapezoid [3]
52      0.00 0.00 97/97 romberg [2]
53 -----
54      0.00 0.00 97/97 main [1]
55 [2] 0.0 0.00 0.00 97 romberg [2]
56 -----
57      0.00 0.00 97/97 main [1]
58 [3] 0.0 0.00 0.00 97 trapezoid [3]
59 -----
60
61 This table describes the call tree of the program, and was sorted by
62 the total amount of time spent in each function and its children.
63
64 Each entry in this table consists of several lines. The line with the
65 index number at the left hand margin lists the current function.
66 The lines above it list the functions that called this function,
67 and the lines below it list the functions this one called.
68 This line lists:
69   index A unique number given to each element of the table.
70         Index numbers are sorted numerically.
71         The index number is printed next to every function name so
72         it is easier to look up where the function is in the table.
73
74   % time This is the percentage of the 'total' time that was spent
75         in this function and its children. Note that due to
76         different viewpoints, functions excluded by options, etc,
77         these numbers will NOT add up to 100%.
78
79   self This is the total amount of time spent in this function.
80
81   children This is the total amount of time propagated into this
82         function by its children.
83
84   called This is the number of times the function was called.
85         If the function called itself recursively, the number
86         only includes non-recursive calls, and is followed by
87         a '+' and the number of recursive calls.
88
89   name The name of the current function. The index number is
90         printed after it. If the function is a member of a
91         cycle, the cycle number is printed between the
92         function's name and the index number.
93
94
95 For the function's parents, the fields have the following meanings:
96
97   self This is the amount of time that was propagated directly

```

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <math.h>
5  #include <time.h>
6
7  #define N 1000000
8
9  double f(double x) {
10     return x*x*x*x;
11 }
12
13 double integrate(double a, double b, int n) {
14     double sum = 0.0;
15     for (int i = 0; i < n; i++) {
16         sum += f(a + (b-a)*i/n);
17     }
18     return sum*(b-a)/n;
19 }
20
21 int main() {
22     double a = 0.0, b = 1.0;
23     int n = 1000000;
24     double result = integrate(a, b, n);
25     printf("Integral of x^4 from %f to %f is %f\n", a, b, result);
26     return 0;
27 }

```


File Edit Selection Find View Goto Tools Project Preferences Help

integration.c x t.txt x EE198138_4.sh x profile.txt x auto.sh x

```
97 self This is the amount of time that was propagated directly
98 from the function into this parent.
99
100 children This is the amount of time that was propagated from
101 the function's children into this parent.
102
103 called This is the number of times this parent called the
104 function '/' the total number of times the function
105 was called. Recursive calls to the function are not
106 included in the number after the '/'.
107
108 name This is the name of the parent. The parent's index
109 number is printed after it. If the parent is a
110 member of a cycle, the cycle number is printed between
111 the name and the index number.
112
113 If the parents of the function cannot be determined, the word
114 '<spontaneous>' is printed in the 'name' field, and all the other
115 fields are blank.
116
117 For the function's children, the fields have the following meanings:
118
119 self This is the amount of time that was propagated directly
120 from the child into the function.
121
122 children This is the amount of time that was propagated from the
123 child's children to the function.
124
125 called This is the number of times the function called
126 this child '/' the total number of times the child
127 was called. Recursive calls by the child are not
128 listed in the number after the '/'.
129
130 name This is the name of the child. The child's index
131 number is printed after it. If the child is a
132 member of a cycle, the cycle number is printed
133 between the name and the index number.
134
135 If there are any cycles (circles) in the call graph, there is an
136 entry for the cycle-as-a-whole. This entry shows who called the
137 cycle (as parents) and the members of the cycle (as children.)
138 The '+' recursive calls entry shows the number of function calls that
139 were internal to the cycle, and the calls entry for each member shows,
140 for that member, how many times it was called from other members of
141 the cycle.
142 <0x0c>
143 Copyright (C) 2012-2020 Free Software Foundation, Inc.
144
145 Copying and distribution of this file, with or without modification,
```

```
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
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```


Activities Sublime Text Nov 14 18:53

~/EE1103 program files/Exercise-7: Integration/profile.txt - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

integration.c x t.txt x EE198138_4.sh x profile.txt x auto.sh x

```
145 Copying and distribution of this file, with or without modification,
146 are permitted in any medium without royalty provided the copyright
147 notice and this notice are preserved.
148 <0x0C>
149 Index by function name
150
151 [1] main [2] romberg [3] trapezoid
152
```

Conclusions:

1. In terms of accuracy, the Romberg integrals give more accuracy since they have a lot of extrapolations which take care of errors due to oscillations in the data.
2. In terms of speed, clearly Trapezoid will take less time simply from the complexity of the code. We aren't able to see any time taken for this dataset but we'll be able to see significant time difference between the Romberg and Trapezoid functions if the dataset is quite big.
3. The diagonal elements of the Romberg matrix are nothing but the romberg integrals of the corresponding orders. So for order=1, the romberg matrix will be a single number which is the trapezoid integral. So when we compute the romberg 5th order integral, we are automatically computing romberg integrals of lower orders. Hence, a separate function for trapezoid integral is really not needed. The function is separately coded just for profiling purposes.

Line 13, Column 45 Tab Size: 4 Plain Text