**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Курсовая работа**

по курсу «Численные методы»

На тему «Вычисление несобственных интегралов численными методами»

Студент: Гаврилов М.С.

Группа: 80-306б

Преподаватель: Ревизников Д.Л.

Оценка:

Москва, 2022

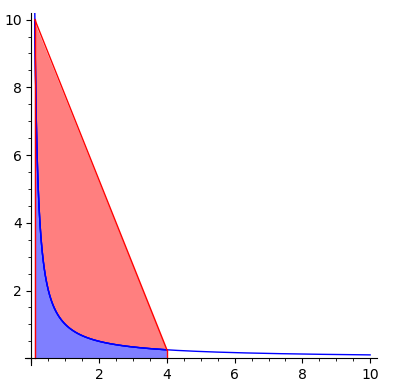
1. **Исследование проблемы / Введение**

При вычислении несобственных интегралов численными методами могут возникнуть следующие проблемы:

1. Катастрофическое падение точности при попадании узла сетки в значение, близкое к бесконечности.

Данный эффект возникает, когда мелкости сетки не хватает, чтобы адекватно оценивать значение несобственного интеграла второго рода, когда функция резко стремится к бесконечности. Особенно сильно он проявляется при использовании метода трапеций.

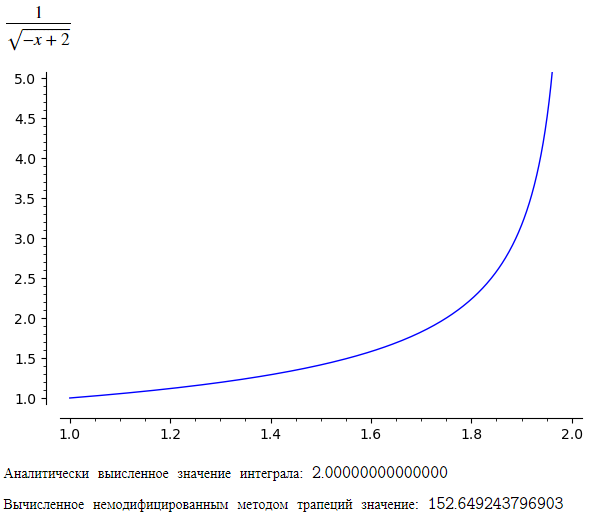
Эту проблему хорошо иллюстрирует данное изображение:



Синим выделена истинная площадь под графиком, красным – дополнительная площадь, которая будет посчитана методом трапеций на таком шаге.

Эта проблема существенна, так как возникает даже на крайне малых шагах ввиду близости значения функции к бесконечности, и, в отличие от обычной неточности методов очень плохо устраняется уменьшением шага.

Пример неверно вычисленного из-за этой ошибки интеграла:



1. Попадание сетки в особую точку

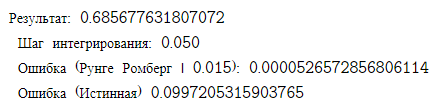
Случается деление на ноль и ничего нельзя посчитать.

1. Непонятно, как интегрировать до бесконечности.

Если графически выбирать точку, в которой значение функции «близко нулю», погрешность вычислений будет непредсказуемой и не будет существенно уменьшаться с уменьшением шага, так как вызвано не неточностями приближения, а неучетом бесконечного «хвоста» функции.

Пример проблемы интегрирования до бесконечности:

|  |
| --- |
|  |



Видно, что метод Рунге-Ромберга не дает сколь либо точной оценки погрешности.

1. **Предлагаемые решения / Выполнение работы**

**Метод выделения особенности.**

В данном методе подынтегральная функция разделяется на так, что . , при этом, должна содержать ту же особенность, что и и быть легко интегрируемой, должна быть достаточно гладкой, т.е. интегрируемой с помощью обычных квадратурных методов. Тогда интеграл вычисляется аналитически, а интегрируется численно.

Данный метод применим к функциям, которые можно представить в виде:

Где – особая точка.

Разделение функции выполняется при помощи разложения функции в ряд Тейлора в окрестности точки . Тогда при умножении на первые членов ряда получится функция, имеющая особенность в точке , а при умножении на вторую часть разложения получится сумма некоторого числа членов вида , где , не имеющих особенности в точке .

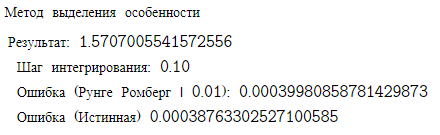
Функция, реализующая этот метод:

|  |
| --- |
| **def** pairTaylor\_series**(**f**,**a**,**n**,**sep **=** 0.5**):**  df **=** f**.**derivative**()**  scur **=** f**(**x **=** a**)**  ssec **=** 0  **for** i **in** **range(**1**,**n **+** 1**):**  **if(** i **<** n**\***sep**):**  scur **+=** df**(**x **=** a**)/(**factorial**(**i**))** **\*** **(**x **-** a**)\*\*(**i**)**  **else:**  ssec **+=** df**(**x **=** a**)/(**factorial**(**i**))** **\*** **(**x **-** a**)\*\*(**i**)**  df **=** df**.**derivative**()**  **return** **(**scur**,**ssec**)**  **def** Improper\_integral\_SP\_isolation**(**h **=** 0.01**,** #решение несобственного интеграла второго рода методом выделения особенности  trace **=** false**,** #применимо лишь для функций определенного вида  sep **=** 0.5**,**  method **=** "rect"**):**  Tf**,**Ts **=** pairTaylor\_series**(**csi**,**0**,**10**,**sep **=** sep**)**  T **=** Tf **+** Ts  **if(**trace**):**  show**(**T**)**  show**(**plot**(**T**,**xmin **=** X0**,**xmax **=** X1**,**legend\_label **=** "разложение Тейлора"**)** **+**  plot**(**csi**,**xmin **=** X0**,**xmax **=** X1**,**  color **=** "red"**,**legend\_label **=** "истинная функция"**,**linestyle **=** "-."**),**  ymin **=** **-**5**,**ymax **=** 5**)**    g**(**x**)** **=** **((**x **-** a**)\*\***al**)\***Tf  phi**(**x**)** **=** **((**x **-** a**)\*\***al**)\***Ts  g**(**x**)** **=** g**(**x**).**expand**().**simplify**()**  phi**(**x**)** **=** phi**(**x**).**expand**().**simplify**()**    **if(**trace**):**  show**(**"g(x) = "**)**  show**(**g**(**x**).**expand**().**simplify**())**  show**(**"phi(x) = "**)**  show**(**phi**(**x**).**expand**().**simplify**())**  show**(**plot**(**g**(**x**),**xmin **=** X0**,**xmax **=** X1**,**legend\_label **=** "g(x)"**)** **+**  plot**(**phi**(**x**),**xmin **=** X0**,**xmax **=** X1**,**  color **=** "green"**,**legend\_label **=** "phi(x)"**),**  ymin **=** 0**,**ymax **=**5**)**    result **=** integral**(**g**(**x**),**x**,**X0**,**X1**)** **+** integrate**(**phi**(**x**),**X0**,**X1**,**h**,**method**)**  **return** result |

Работа этого метода на примере функции , интегрируемой от 0 до 0.5

|  |
| --- |
|  |
| Разложение в ряд Тейлора (используется 10 первых членов) |
|  |
| Разделение на с |
|  |
| Разложение на графике |
|  |
| Ответ |
|  |

Вычисление точности метода методом Рунге-Ромберга путем сравнения с аналитически вычисленным решением. Шаг численного интегрирования равен 0.1



Стоит заметить, что, чем большая часть ряда Тейлора участвует в формировании функции , тем точнее будет результат, ибо тем большая часть интеграла будет вычисляться аналитически.

Вообще можно видеть, что помимо высокой точности вообще, ошибка метода достаточно точно считается методом Ругне-Ромберга, а значит, точность расчетов может быть дополнительно повышена методом Рунге-Ромберга-Ричардсона.

Основной недостаток этого метода заключается в том, что он применим только к функциям особого вида.

**Метод исключения особой точки**

Идея данного метода – перед началом численного интегрирования пройти по сетке и проверить каждый ее узел на наличие особенности. Если в узле обнаружена неопределенность, численная бесконечность значения функции, или комплексное значение функции, данный узел исключается из сетки.

Это позволяет исключить возможность катастрофической потери точности при попадании узла в окрестность особой точки.

Проверка может выполняться непосредственно при интегрировании, это увеличит скорость работы метода, но сделает его менее универсальным – придется переписывать уже реализованные методы численного интегрирования.

Идея метода графически изображена на графиках ниже.

|  |
| --- |
| Первый узел сетки попадает в малую окрестность точки 0. |
|  |

|  |
| --- |
| После исключения этого узла: |
|  |

Функция, реализующая метод исключения особой точки:

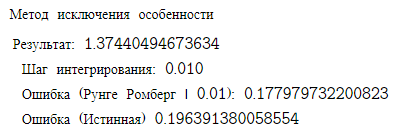
|  |
| --- |
| **def** Improper\_integral\_SP\_exclusion**(**#функция обнаруживает точечнык особенности, и при интегрировании пропускает их  f**,**X0**,**X1**,**h**,**  trace **=** false**,**  method **=** "rect"**,**  eps **=** 1e-4**):**  #без сдвига сетки  hc **=** h #с шагом h будем вычислять производную  h **/=** 2 #проверять надо точки, из которых будем брать занчения f  infpos **=** 1**/**h  infneg **=** **-**1**/**h  xcur **=** X0  **if(**trace**):**  **print(**"{} -- {} | checking..."**.format(**X0**,**X1**))**    i **=** 0  repmark **=** **(**X1**-**X0**)/(**hc**\***10**)**  result **=** 0  **while** **(**xcur **<** X1 **+** h**\***eps**):**    **try:**  f**(**x **=** xcur**)**  **if(**trace **and** **((** i **%** **(**repmark**)** **==** 0**)** **or** **abs(**f**(**x **=** xcur**))** **>** infpos**/**10 **or** **complex(**f**(**x **=** xcur**)).**imag **!=** 0**)):**  **print(**"{} -- {} ({}) -- {} \t|{}|"**.format(**X0**,**xcur**,**f**(**x **=** xcur**),**X1**,**h**))**  **except(BaseException):** #проверка на деление на 0  **if(**trace**):**  **print(**" |0|"**)**  **print(**" !{} -- {} (NAN) -- {} \t|{}|"**.format(**X0**,**xcur**,**X1**,**h**))**  **if(**xcur **-** eps **<** X0**):**  #это начальная точка  #сдвигаем начало на шаг  X0 **+=** hc  xcur **+=** h  #проверяем следующую точку  **continue**  **else:**  **if(**xcur **+** eps **>** X1**):**  #это последняя точка  #после инетгрировать не нужно  **if(**trace**):**  **print(**"integrating ps..."**)**  **return** integrate**(**f**,**X0**,**X1 **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)**  **else:**  #это точка в середине  **if(**trace**):**  **print(**"integrating fu ps..."**)**  #интегрироуем до последнего проверенного узла  result **+=** integrate**(**f**,**X0**,**xcur **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)**  #продолжаем проверку со следующего узла  X0 **=** xcur **+** hc  **continue**    **if** **(**f**(**x **=** xcur**)** **>** infpos **or** f**(**x **=** xcur**)** **<** infneg**):** #проверка на сверхбольшие значения  **if(**trace**):**  **print(**" |inf|"**)**  **print(**" !{} -- {} ({}) -- {} \t|{}|"**.format(**X0**,**xcur**,**f**(**x **=** xcur**),**X1**,**h**))**  **if(**xcur **-** eps **<** X0**):**  #это начальная точка  #сдвигаем начало на шаг  X0 **+=** hc  xcur **+=** h  #проверяем следующую точку  **continue**  **else:**  **if(**xcur **+** eps **>** X1**):**  #это последняя точка  #после инетгрировать не нужно  **if(**trace**):**  **print(**"integrating ps..."**)**  **return** integrate**(**f**,**X0**,**X1 **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)**  **else:**  #это точка в середине  **if(**trace**):**  **print(**"integrating fu ps..."**)**  #интегрироуем до последнего проверенного узла  result **+=** integrate**(**f**,**X0**,**xcur **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)**  #продолжаем проверку со следующего узла  X0 **=** xcur **+** hc  **continue**    **if** **(complex(**f**(**x **=** xcur**)).**imag **!=** 0**):**  **if(**trace**):**  **print(**" |complex|"**)**  **print(**" !{} -- {} ({}) -- {} \t|{}|"**.format(**X0**,**xcur**,**f**(**x **=** xcur**),**X1**,**h**))**  **if(**xcur **-** eps **<** X0**):**  #это начальная точка  #сдвигаем начало на шаг  X0 **+=** hc  xcur **+=** h  #проверяем следующую точку  **continue**  **else:**  **if(**xcur **+** eps **>** X1**):**  #это последняя точка  #после инетгрировать не нужно  **if(**trace**):**  **print(**"integrating ps..."**)**  **return** integrate**(**f**,**X0**,**X1 **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)**  **else:**  #это точка в середине  **if(**trace**):**  **print(**"integrating fu ps..."**)**  #интегрироуем до последнего проверенного узла  result **+=** integrate**(**f**,**X0**,**xcur **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)**  #продолжаем проверку со следующего узла  X0 **=** xcur **+** hc  **continue**    xcur **+=** h  i **+=** 1    **if(**trace**):**  **print(**"integrating..."**)**    **return** result **+** integrate**(**f**,**X0**,**X1**,**hc**,**method **=** method**)** |

Вычисление интеграла функции в пределах от 0 до 0.5 методом исключения особой точки с трассировкой. Шаг равен 0.00001:

|  |
| --- |
|  |

Видно, что в точке 0 была обнаружена неопределенность, и данная точка была исключена из сетки.

Определение точности метода, шаг равен 0.01:



Видно, что точность этого метода крайне низка, однако он позволяет интегрировать функции любого вида, не опасаясь ошибок, связанных с крайне большими значениями функции в окрестности особой точки.

**Модифицированный метод исключения особой точки**

Потеря точности численного интегрирования в окрестности особой точки возникает из-за того, что перепад значения функции там существенно больше, чем допускает шаг интегрирования. Мне вообще кажется, что численное интегрирование не особо применимо, если функция не равномерно непрерывна.

На практике это означает, что при уменьшении шага интегрирования предел перепада значения функции, до которого можно интегрировать, не испытывая катастрофического падения точности возрастает. Однако, если интегрировать всю функцию с таким малым шагом, время, требуемое для выполнения вычислений неприемлемо велико.

Модификация метода исключения особой точки заключается в том, что, помимо исключения узла сетки в окрестности особой точки, выполняется дополнительное интегрирование между исключенным и следующим узлом интегрирования. Графически это выглядит так:

|  |
| --- |
| Сетка интегрирования после исключения особой точки |
|  |

|  |
| --- |
| Сетка интегрирования после применения модифицированного метода исключения особой точки |
|  |

Стоит отметить, что сетку дополнительного интегрирования также необходимо проверить на предмет особенностей. Если дополнительное интегрирование осуществлять также модифицированным методом исключения особой точки, то получится итеративный процесс, на который желательно наложить ограничение глубины, после которой будет уже применяться обычный метод исключения особой точки.

Функция, реализующая модифицированный метод исключения особой точки:

|  |
| --- |
| **def** Improper\_integral\_SP\_exclusion\_mod**(**#функция обнаруживает точечнык особенности, и при интегрировании пропускает их  f**,**X0**,**X1**,**h**,**  trace **=** false**,**  method **=** "rect"**,**  eps **=** 1e-4**):**  #без сдвига сетки  hc **=** h #с шагом h будем вычислять производную  h **/=** 2 #проверять надо точки, из которых будем брать занчения f  hsub **=** hc**\***eps #шаг для вспомогательного интегрирования  infpos **=** 1**/(**hc**\***eps**)**  infneg **=** **-**1**/(**hc**\***eps**)**  xcur **=** X0  repmark **=** **(**X1**-**X0**)/(**hc**\***10**)**  **if(**trace**):**  **print(**"{} -- {} | checking..."**.format(**X0**,**X1**))**    i **=** 0  **while** **(**xcur **<** X1 **+** h**\***eps**):**    **try:**  f**(**x **=** xcur**)**  **if(**trace **and** i **%** **(**repmark**)** **==** 0**):**  **print(**"{} -- {} ({}) -- {} \t|{}|"**.format(**X0**,**xcur**,**f**(**x **=** xcur**),**X1**,**h**))**  **except(BaseException):** #проверка на деление на 0  **if(**trace**):**  **print(**" |0|"**)**  **print(**" !{} -- {} (NAN) -- {} \t|{}|"**.format(**X0**,**xcur**,**X1**,**h**))**  **if(**xcur **-** h**\***eps **<** X0**):**  **return** **(**Improper\_integral\_SP\_exclusion**(**f**,**X0 **+** hc**,**X1**,** h **=** hc**,**method **=** method**,**eps **=** eps**,** trace **=** trace**)** **+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**X0 **+** hsub**,**xcur **+** hc**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**))**  **if(**xcur **+** h**\***eps **>** X1**):**  **return** **(**integrate**(**f**,**X0**,**X1 **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)** **+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**X1 **-** hc**,**xcur **-** hsub**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**))**    **return** **(**integrate**(**f**,**X0**,**xcur **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)** **+** #эта часть сетки уже проверена  Improper\_integral\_SP\_exclusion**(**f**,**xcur **+** hc**,**X1**,** h **=** hc**,**method **=** method**,**eps **=** eps**,** trace **=** trace**)** **+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**xcur **-** hc**,**xcur **-** hsub**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**)+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**xcur **+** hsub**,**xcur **+** hc**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**))**    **if** **(**f**(**x **=** xcur**)** **>** infpos **or** f**(**x **=** xcur**)** **<** infneg**):** #проверка на сверхбольшие значения  **if(**trace**):**  **print(**" |inf|"**)**  **print(**" !{} -- {} ({}) -- {} \t|{}|"**.format(**X0**,**xcur**,**f**(**x **=** xcur**),**X1**,**h**))**  **if(**xcur **-** h**\***eps **<** X0**):**  **return** **(**Improper\_integral\_SP\_exclusion**(**f**,**X0 **+** hc**,**X1**,** h **=** hc**,**method **=** method**,**eps **=** eps**,** trace **=** trace**)+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**X0 **+** hsub**,**xcur **+** hc**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**))**  **if(**xcur **+** h**\***eps **>** X1**):**  **return** **(**integrate**(**f**,**X0**,**X1 **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)** **+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**X1 **-** hc**,**xcur **-** hsub**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**))**    **return** **(**integrate**(**f**,**X0**,**xcur **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)** **+** #эта часть сетки уже проверена  Improper\_integral\_SP\_exclusion**(**f**,**xcur **+** hc**,**X1**,** h **=** hc**,**method **=** method**,**eps **=** eps**,** trace **=** trace**)+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**xcur **+** hsub**,**xcur **+** hc**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**)** **+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**xcur **-** hc**,**xcur **-** hsub**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**))**  **if** **(complex(**f**(**x **=** xcur**)).**imag **!=** 0**):**  **if(**trace**):**  **print(**" |complex|"**)**  **print(**" !{} -- {} ({}) -- {} \t|{}|"**.format(**X0**,**xcur**,**f**(**x **=** xcur**),**X1**,**h**))**    **if(**xcur **-** h**\***eps **<** X0**):**  **return** **(**Improper\_integral\_SP\_exclusion**(**f**,**X0 **+** hc**,**X1**,** h **=** hc**,**method **=** method**,**eps **=** eps**,** trace **=** trace**)+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**X0 **+** hsub**,**xcur **+** hc**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**))**  **if(**xcur **+** h**\***eps **>** X1**):**  **return** **(**integrate**(**f**,**X0**,**X1 **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)** **+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**X1 **-** hc**,**xcur **-** hsub**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**))**    **return** **(**integrate**(**f**,**X0**,**xcur **-** hc**,** h **=** hc**,**method **=** method**,**eps **=** hc**\***eps**,** trace **=** trace**)** **+** #эта часть сетки уже проверена  Improper\_integral\_SP\_exclusion**(**f**,**xcur **+** hc**,**X1**,** h **=** hc**,**method **=** method**,**eps **=** eps**,** trace **=** trace**)+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**xcur **+** hsub**,**xcur **+** hc**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**)** **+**  Improper\_integral\_SP\_exclusion\_mod**(**f**,**xcur **-** hc**,**xcur **-** hsub**,**hsub**,**method **=** method**,**eps **=** eps**,** trace **=** trace**))**    xcur **+=** h  i **+=** 1    **if(**trace**):**  **print(**"integrating..."**)**    **return** integrate**(**f**,**X0**,**X1**,**hc**,**method **=** method**,**eps **=** eps**\***hc**)** |

Работа метода на примере интегрирования функции в пределах от 0 до 0.5 с шагом 0.0001.

|  |
| --- |
|  |

Значение данного интеграла, вычисленное аналитически:



Определение ошибки метода, h = 0.01:

|  |
| --- |
|  |

Видно, что данный метод существенно точнее обычного метода исключения особой точки.

**Вычисление несобственных интегралов первого рода**

В основе методов численного интегрирования собственных функций лежит поиск точки , в которой значение функции столь близко к нулю, что при заданной точности интеграл от несущественно мал.

В качестве критерия для поиска точки можно использовать следующее выражение:

Где – задаваемый критерий точности.

Остающаяся после определения границ интегрирования проблема – получившийся отрезок интегрирования может быть любой длины, так что при задавании постоянного шага вычисление интеграла может занять крайне много времени.

Самым простым решением будет задавать шаг в зависимости от длины получившегося отрезка, а в качестве параметра принимать значение – какую часть от отрезка занимает шаг.

Функция, реализующая такую версию метода вычисления несобственного интеграла первого рода:

|  |
| --- |
| **def** Improper\_integral\_1**(**f**,**X0**,**X1**,**h**,**  method **=** "rect"**,**  trace **=** false**,**  eps **=** 1e-5**):**    **if(** **(**X0 **>** 1**/**eps **and** X1 **>** 1**/**eps**)** **or** **(**X0 **<** 1**/**eps **and** X1 **<** 1**/**eps**)** **):** #при данной погрешности выражение не имеет смысла  **return** false    a **=** var**(**"a"**)**  **if(**X0 **<** **-**1**/**eps**):**#первый \ предел в -бесконечности  **if(**trace**):**  **print(**"searching for a"**)**    criteria **=** f**(**x **=** a**)/**a  step **=** h**\***10    **if(**X1 **>** 1**/**eps**):** #если и второй предел в бесконечности, то за базу берем ноль  a **=** 0  **else:**  a **=** X1  **while(**a **>** **-**1**/**eps**):**  **if(**trace**):**  **print(**" a: {} | {}"**.format(**a**,**step**))**  **try:**  criteria**(**a **=** a**)**  **if(**trace**):**  **print(**" a: {} ({}) | {}"**.format(**a**,** criteria**(**a **=** a**),**step**))**  **break**  **except(BaseException):**  a **-=** step  step**\*=**2  **if(**a **<=** **-**1**/**eps**):** #на данной сетке интеграл не существует  **return** false  **while(abs(**criteria**(**a **=** a**))** **>** eps**):**  a **-=** step  step**\*=**2    **if(**trace**):**  **print(**" a: {} ({}) | {}"**.format(**a**,** criteria**(**a **=** a**),**step**))**    **if(**a **<=** **-**1**/**eps**):** #при данной погрешности интеграл не сходится  **return** false  **else:**  a **=** X0    b **=** var**(**"b"**)**  **if(**X1 **>** 1**/**eps**):**#второй предел в бесконечности  **if(**trace**):**  **print(**"searching for b"**)**    criteria **=** f**(**x **=** b**)/**b  step **=** h**\***10  b **=** a  **while(**b **<** 1**/**eps**):**  **if(**trace**):**  **print(**" b: {} | {}"**.format(**b**,**step**))**    **try:**  criteria**(**b **=** b**)**  **if(**trace**):**  **print(**" b: {} ({}) | {}"**.format(**b**,**criteria**(**b **=** b**),**step**))**  **break**  **except(BaseException):**  b **+=** step  step**\*=**2  **if(**b **>=** 1**/**eps**):** #на данной сетке интеграл не существует  **return** false  **while(abs(**criteria**(**b **=** b**))** **>** eps**):**  b **+=** step  step**\*=**2    **if(**trace**):**  **print(**" b: {} ({}) | {}"**.format(**b**,**criteria**(**b **=** b**),**step**))**    **if(**b **>=** 1**/**eps**):** #при данной погрешности интеграл не сходится  **return** false  **else:**  b **=** X1    **if(**trace**):**  **print(**"integrating from {} to {}"**.format(**a**,**b**))**  **print(**"w step {}"**.format(**h**\*(**b**-**a**)/**200**))**    **return** integrate**(**f**,**a**,**b**,**h**\*(**b**-**a**)/**1000**,**method **=** method**)** |

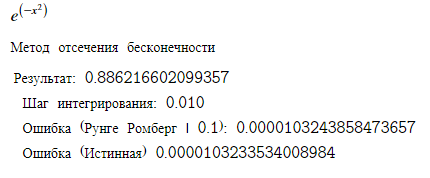
Работа этого метода на примере функции , интегрируемой в пределах от .

|  |
| --- |
|  |

Значение данного интеграла, вычисленное аналитически:



Оценка погрешности решения:



Видим, что получилось крайне точно. При оценке методом Рунге-Ромберга помимо уменьшения шага выполнялось уменьшение требуемой погрешности .

График этой функции:

|  |
| --- |
|  |

Видно, что функция крайне быстро уходит в 0. Попробуем проинтегрировать функцию, которая сходится к 0 медленнее.

|  |
| --- |
|  |

Интегрируем в пределах от .

|  |
| --- |
|  |

Значение этой функции, вычисленное аналитически:



Нахождение погрешности расчетов:

|  |
| --- |
|  |

Видим, что точность по-прежнему высока.

1. **Вывод**

В ходе выполнения этой работы я провел поверхностное исследование довольно интересной и глубокой темы численного интегрирования несобственных интегралов. Все методы, рассмотренные мной в этой работе, являются надстройками над обычными квадратурными методами, которые, проведя те или иные действия над функцией или сеткой интегрирования в итоге сводятся к запускам обычных методов над измененными функциями или сетками. Это означает, что они применимы с любыми квадратурными методами и их реализациями, а не только с теми, что рассматривал я. С другой стороны, существует целый класс методов с переменным шагом интегрирования, которые также применимы для вычисления несобственных интегралов. Эти методы также стоят рассмотрения. Также может иметь смысл более подробное рассмотрение модифицированного метода исключения особой точки, например, поиск более оптимального метода определения применимости или неприменимости квадратурных методов на данной сетке с данными значениями функции в узлах. В целом, рассмотренные методы позволяют с достаточно высокой точностью вычислять несобственные интегралы различных видов.

**Список источников**

1. *Методические указания к решению задач по численному интегрированию*. [Электронный ресурс URL: <http://www.lib.unn.ru/students/src/alkint.pdf>] – НГУ им. Лобачевского 2016.
2. *Математическое обоснование и процедура численного интегрирования несобственных интегралов.* [Электронный ресурс URL: <https://mai.ru/upload/iblock/fe7/chislennoe-integrirovanie-nesobstvennykh-integralov_-vstrechayushchikhsya-v-matematicheskikh-modelyakh-aerodinamiki.pdf>] – МАИ. Смирнов В.Ю.