Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Кафедра прикладной математики и компьютерной безопасности

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. А. Кытманов
подпись
« ____» _____ 20 __ г.

БАКАЛАВРСКАЯ РАБОТА

01.03.04 – Прикладная математика

Повышение точности определения траектории спутника системы GPS
по его навигационным сигналам

Руководитель _____ профессор, д.ф.-м.н. С. П. Царев
подпись, дата

Выпускник _____ А. С. Тетерятников
подпись, дата

Красноярск 2017

Содержание

Реферат

Выпускная квалификационная работа по теме «Повышение точности определения траектории спутника системы GPS по его навигационным сигналам» содержит 47 страниц текстового документа, 17 иллюстраций, 1 таблицу, 14 использованных источников, 3 приложения.

МЕТОД СОПРЯЖЕННЫХ ГРАДИЕНТОВ, ЯЗЫК ПРОГРАММИРОВАНИЯ JULIA, ИНТЕРПОЛЯЦИЯ, СИСТЕМА ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ, GPS, ГЛОНАСС.

Цели и задачи:

- Решение больших систем линейных уравнений в числах большой точности;
- Изучение языка программирования Julia. Поиск наиболее быстрого метода чтения-записи координат спутников с жесткого диска.

В задачах космической навигации встречаются большие системы линейных алгебраических уравнений. Их решение занимает продолжительное время на современных ЭВМ. Требуется найти быстрый и точный способ решения таких систем в числах большой точности. В языке программирования Julia есть специальный тип данных для работы с числами произвольной точности, причем, по заявлению разработчиков языка, Julia почти не уступает в быстродействии языку C++. По этим причинам целесообразно использовать язык Julia для написания программы.

В результате была реализована программа на языке Julia для решения системы линейных алгебраических уравнений Годунова методом сопряженных градиентов в числах произвольной точности, проведено сравнение быстродействия языков Julia и C++, проведено сравнение метода сопряженных градиентов со встроенным в Julia оператором для решения СЛАУ, на Julia разработан метод хранения в бинарном файле координат спутников.

Введение

При решении задач космической навигации появляются большие системы линейных алгебраических уравнений (СЛАУ). К примеру, задача определения координат приемника GPS/ГЛОНАСС. По сигналам навигационных спутников получаем разряженную систему из 4 миллионов ненулевых элементов. Требуется найти быстрый и точный способ решения таких систем в числах высокой точности. Один из методов решения СЛАУ – «Метод сопряженных градиентов» (МСГ).

Метод сопряженных градиентов [10] - предназначен для решения систем линейных алгебраических уравнений вида: $A \cdot x = b$, где A – матрица размерности M*N, b – вектор размерности M.

Метод сопряженных градиентов при точных вычислениях приводит к ответу за конечное число шагов, но по сути является итерационным процессом, слабым местом которого является критерий остановки – определение номера шага процесса, после которого точность приближения к решению системы линейных уравнений на данном компьютере не может быть существенно улучшена.

Для тестирования алгоритма МСГ была выбрана СЛАУ Годунова. Эта система является плохо обусловленной. Для размерности равной 200, число обусловленности для СЛАУ Годунова составляет 7.76e16. Эту систему удобно использовать для проведения тестов корректности работы программ для решения СЛАУ, так как её решение для любой размерности легко можно вычислить по формулам. СЛАУ Годунова приведена в приложении А.

Выяснилось, что у МСГ есть множество вариантов реализации алгоритма. В работе будут рассмотрены все найденные варианты алгоритма с целью выявления наиболее быстрого и при этом не уступающего другим алгоритмам в точности нахождения решения. МСГ будет применен для решения СЛАУ, полученной из задачи определения положения спутников по

их навигационным сигналам. В процессе исследования МСГ оказалось, что МСГ требователен к точности представления чисел в ЭВМ.

На сегодняшний день набирает популярность язык программирования Julia. Так как в Julia есть специальный тип данных (BigFloat) для работы с числами высокой точности, где мантисса числа указывается пользователем, то в целях ознакомления и изучения языка, алгоритм МСГ будет реализован на Julia. В процессе изучения языка, возникали вопросы, которые не описываются в официальной документации.

Julia[5] — высокоуровневый высокопроизводительный свободный язык программирования с динамической типизацией, созданный для математических вычислений. Эффективен также и для написания программ общего назначения. Julia написана на Си, C++ и Scheme.

На официальном сайте языка программирования Julia указано, что Julia практически не уступает в скорости языку C++. В работе будет проведено сравнение скорости работы программы на языке Julia с подобной программой, написанной на C++.

Необходимо проверить, как изменяется скорость работы программы при использовании чисел высокой точности. Сравниваться будут 2 подобные программы. В одной будет использоваться тип данных BigFloat, в другой – Float64.

Считывание координат спутников из текстовых файлов формата “txt” и составление матрицы в Julia занимает много времени и оперативной памяти. Поэтому поднимается вопрос о наиболее эффективном способе чтения и записи на жесткий диск.

1. Реализация метода сопряженных градиентов на Julia

На языке программирования Julia была реализована программа для решения СЛАУ Годунова методом сопряженных градиентов. В дальнейшем такую программу будем называть “godunovMatrix”.

Далее в разбираемой программе используется алгоритм МСГ из презентации Киреева И.В. «Метод сопряженных градиентов» [2] (В обозначения этого алгоритма, решается СЛАУ вида A·x=b, где A – произвольная матрица размерности N*M, b – вектор размерности N). Алгоритм приведен на рисунке 1.

Initialize

$$\boldsymbol{p}^{(0)} = \boldsymbol{s}^{(0)} = A^T\left(\boldsymbol{b} - A\boldsymbol{x}^{(0)}\right), \gamma_0 = \|\boldsymbol{s}^{(0)}\|_m^2;$$

for $k = 0, 1, 2, \ldots$ **while** $\gamma_k > tol$ **real compute**

$$\begin{aligned}
\boldsymbol{q}^{(k)} &= A\boldsymbol{p}^{(k)}, \\
\alpha_k &= \gamma_k / \|\boldsymbol{q}^{(k)}\|_m^2, \\
\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{p}^{(k)}, \\
\boldsymbol{s}^{(k+1)} &= \boldsymbol{s}^{(k)} - \alpha_k \left(A^T \boldsymbol{q}^{(k)}\right), \\
\gamma_{k+1} &= \|\boldsymbol{s}^{(k+1)}\|_n^2, \\
\beta_k &= \gamma_{k+1} / \gamma_k, \\
\boldsymbol{p}^{(k+1)} &= \boldsymbol{s}^{(k+1)} + \beta_k \boldsymbol{p}^{(k)}.
\end{aligned}$$

Рисунок 1 - алгоритм метода сопряженных градиентов

В качестве критерия остановки в цикле алгоритма МСГ используется максимально допустимое отклонение нормы градиента, цикл завершается при достижении нормы градиента не превышающей максимально допустимое отклонение нормы градиента (на рисунке 1: *tol* – максимальное отклонение нормы градиента, *s* – градиент в точке *x*). Дополнительный критерий остановки – ограничение на количество итераций.

Во время написания программы использовалась Julia версии 0.4.5.

Для запуска программы нужно запустить файл godunovMatrix.jl при помощи Julia. Например, из консоли запуск программы будет производиться следующей командой:

Julia godunovMatrix.jl

В программе для расчетов используется тип данных BigFloat (для работы с числами произвольной точности).

Число с плавающей запятой состоит из следующих частей:

- знак мантиссы (указывает на отрицательность или положительность числа);
- мантисса (выражает значение числа без учёта порядка);
- знак порядка;
- порядок (выражает степень основания числа, на которое умножается мантисса).

Мантисса типа данных BigFloat указывается пользователем (в конфигурационном файле config.txt параметр mantissa).

Программа состоит из 3 файлов: главная часть программы godunovMatrix.jl, подключаемого в главной части модуля readFile.jl и конфигурационного файла config.txt.

1.1 Конфигурационный файл

В конфигурационном файле "config.txt" программы godunovMatrix указываются параметры, с которыми будет производиться расчет в программе godunovMatrix. Программа имеет следующие параметры:

- limit – ограничение на количество итераций в цикле МСГ;
- Godunov_matrix_dim – размерность СЛАУ Годунова;
- norm_residual – максимально допустимое отклонение нормы градиента в точке *x*;
- mantissa – значение мантиссы для типа данных BigFloat, указывается в битах (мантисса, с которой производится расчет в алгоритме МСГ);
- mantissa_out – значение мантиссы для типа данных BigFloat, предназначено для уменьшения количества знаков после запятой при записи результатов в файл;

- pathResult – путь до каталога, где будут храниться файлы с результатами работы программы.

Содержание файла config.txt на рисунке 2.

```
# ограничение на количество итераций
limit = 100

# размерность матрицы Годунова
Godunov_matrix_dim = 200

# максимально допустимое отклонение нормы градиента в точке
norm_residual = 1e-500

# мантисса
mantissa = 2000

# мантисса при записи вектора X в файл
mantissa_out = 2000

# путь к каталогу для вывода результатов
pathResult = /home/8ulia8y/8ulia_conj
```

Рисунок 2 – содержимое файла config.txt

1.2 Главная часть программы

Файл godunovMatrix.jl является главной частью программы. В нем происходит создание и заполнение матрицы СЛАУ Годунова, расчет решения системы методом сопряжены градиентов, расчет решения СЛАУ Годунова специальными формулами и производится вывод результатов в файлы.

В начале кода программы (фрагмент кода на рисунке 3) происходит запуск таймера, после чего программа определяет расположение каталога, в котором программа была запущена и подключается файл readFile.jl, в нем происходит считывание параметров программы из конфигурационного файла config.txt.

Содержимое модуля readFile.jl представлено в приложении Б.

```
1 tic()  #запуск таймера
2
3 path = @__FILE__            # @__FILE__  - возвращает путь до исполняемого
4                             # файла godunovMatrix.jl.
5 path = dirname(path)        # dirname() возвращает путь до дириктории в
6                             # которой находится файл, указываемый в path.
7 Include(string(path,"/readFile.jl")) # Выполнение модуля readfile.jl.
8 set_bigfloat_precision(mantissa) # Установка величины мантиссы BigFloat.
```

Рисунок 3 – Фрагмент кода программы godunovMatrix

На этом этапе имеем следующие переменные:

- path — путь до каталога с программой (тип String);
- norm_residual — максимально допустимое отклонение нормы градиента (тип Float64);
- limit — ограничение на количество итераций (Int64);
- Godunov_matrix_dim — размерность матрицы Годунова (тип Int64);
- mantissa — величина мантиссы BigFloat, с которой происходят расчеты в программе (тип Int64);
- mantissa_out — величина мантиссы BigFloat, с которой происходит запись результатов в файлы (тип Int64);
- pathResult — путь до каталога, в который будут записаны результаты (тип String).

Вводится функция для расчета квадрата нормы вектора norma(). Функция norma() принимает вектор и возвращает число типа BigFloat. Фрагмент на рисунке 4.

```
11 function norma(x)
12   result = big(0.0)                    # Переменной result присваивается
13                                        # тип BigFloat.
14   X = x.^2          # Все элементы вектора возводятся в квадрат.
15   Result = sum(x)   # Все элементы вектора суммируются.
16   Return result     # функция возвращает переменную result.
17 End
```

Рисунок 4 – Фрагмент кода программы godunovMatrix

На следующем участке кода создается матрица A размерностью n*n и вектор B размерностью n. Матрица A и вектор B заполняются числами типа BigFloat так, что A является левой и B правой частью системы линейных уравнений Годунова. Создается вектор real_result размерностью n, и заполняется числами типа BigFloat. По формулам для известного решения системы Годунова рассчитываются компоненты вектора real_result.

Описываемый участок кода представлен на рисунке 5.

```
18 n = Godunov_matrix_dim
19 A = fill(big(0e0),n,n)  # Матрица A размерности n*n типа BigFloat создается
20                         # функцией fill() и заполняется нулевыми элементами.
21 B = Array(BigFloat, n) # Вектор B – массив размерности n типа BigFloat.
22 for i in 1:1:n
23   A[i,i] = big"7.0"/5  # Заполнение диагоналей матрицы A, как левая часть
24                         # СЛАУ Годунова.
25 end
26 for i in 1:1:(n-1)
27  A[i,i+1] = big"11.0"/3
28 end
29 for i in 1:1:(n-2)    #Заполнение вектора B, как правая часть СЛАУ Годунова
30  B[i] = (152*i+118)
31  B[i] = B[i]/(15*(2*i+1)*(2*i+3))
32 end
33 B[n] = 7
34 B[n] = B[n]/(5*(2*n+1))
35 B[n-1] = (152*n-34)
36 B[n-1] = B[n-1]/(15*(2*n-1)*(2*n+1))
37 real_result = Array(BigFloat, n) # real_result - вектор, решение матрицы
38                                  # Годунова.
39 for i in 1:1:(n-2)  # Заполнение вектора real_result по формуле для
40                     # известного решения СЛАУ Годунова.
41   real_result[i] = big(1e0)/(2*i+1)
42 end
43 real_result[n-1] = big(1e0)/(2*n-1)
44 real_result[n] = big(1e0)/(2*n+1)
```

Рисунок 5 – Фрагмент кода программы godunovMatrix

Примечание: при преобразовании дробных чисел в BigFloat, следует сначала преобразовать числитель в BigFloat и только потом делить на знаменатель. Пример: big(1)/3. Если преобразовать по-другому big(1/3), то

дробь 1/3 будет посчитана в формате Float64 с мантиссой 53 бита, и при конвертировании в BigFloat, с установленной мантиссой большей, чем 53 бита, число будет представлено некорректно.

Далее выполняется алгоритм МСГ (рисунок 6).

```
x = fill(big(0.0e),n)
p = A'*(B-A*x)
s=p
y=norma(s)
k = 0
while(y> norm_residual && k<limit)
  q = A*p
  a = y/norma(q)
  x = x + a*p
  s = s - a*A'*q
  y1 = norma(s)
  b = y1/y
  p = s + b*p
  k=k+1
  y=y1
end
display(convert(Array{Float64},x))
```

Рисунок 6 – Фрагмент кода программы godunovMatrix

На участке кода, представленного на рисунке 6, счетчик k (тип Int64) отсчитывает количество итераций, пока не достигнет значения limit (тип Int64) и, по достижению, прерывает выполнение цикла while.

Программа, после выхода из цикла, отображает в консоли вектор *x* — полученное решение методом сопряженных градиентов.

Команда display() отображает значений переменных в консоли. Функция display() не поддерживает тип BigFloat. Поэтому требуется конвертировать массив из BigFloat в Float64.

На следующем участке происходит проверка, существует ли каталог, указанный в pathResult. Если каталог не существует, то функция mkpath() создает все каталоги, указанные в pathResult. В переменную difference

записывается разность векторов x и real_result. Участок программы представлен на рисунке 7.

```
64 if !(isdir(pathResult))
65 mkpath(pathResult)          # Создание директории result в случае
66                             # её отсутствия.
67 end
68 difference = x-real_result
```

Рисунок 7 – Фрагмент кода программы godunovMatrix

Устанавливается значение для мантиссы BigFloat равное mantissa_out. Значение мантиссы mantissa_out (меньше, чем величина мантиссы, хранящейся в переменной mantissa) нужно, чтобы при записи результатов в файлы, числа выглядели более наглядно для пользователя. Таким образом, программа производит вычисления с мантиссой mantissa, а записывает в файлы с мантиссой mantissa_out.

Изменение мантиссы не затрагивает уже существующие переменные, поэтому к векторам типа BigFloat прибавляется и вычитается единица. Фрагмент программы на рисунке 8.

```
71 set_bigfloat_precision(mantissa_out)
72 x=x+1-1
73 real_result = real_result+1-1
74 difference = difference+1-1
```

Рисунок 8 – Фрагмент кода программы godunovMatrix

Далее происходит запись результатов в файлы (участок программы на рисунке 9).

Функция writedlm() предназначена для записи массивов в файл, столбцы отделяются разделителем «\t». Функция не поддерживает тип данных BigFloat, поэтому требуется преобразовать число в строку функцией string(). Полученная строка представляется в следующем виде: текст «BigFloat», квадратные скобки, в квадратных скобках перечисляются

компоненты вектора, записанные через «,». Пример: «BigFloat[1.1e-1,2.5e-1,3.4e-1]». Поэтому от строки берутся только символы с десятого по предпоследний. Функция split() разделяет строку на массив строк, используя в качестве разделителя символ «,». Полученный массив записывается в файл.

Для записи двух и более массивов в один файл в функции writedlm() в квадратных скобках через пробел указываются векторы.

```
76  file = open(string(pathResult, "/resultX.txt"), "w")
77  writedlm(file, split(string(x)[10:end-1],","))
78  close(file)
79
80  file = open(string(pathResult, "/resultDifference.txt"), "w")
81  writedlm(file, split(string(difference)[10:end-1],","))
82  close(file)
83
84  file = open(string(pathResult, "/resultCompare.txt"), "w")
85  writedlm(file, [split(string(x)[10:end-1],",")
86  split(string(real_result)[10:end-1],",")])
87  close(file)
88
89  if (k>=limit)
90  file = open(string(pathResult, "/log.txt"), "w")
91  write(file, string("Программа завершила работу, сработало ограничение на
92  количество итераций\n","Входные параметры:\n", "norm_residual = ",
93  norm_residual, "\n", "mantissa = ", mantissa, "\n", "Godunov_matrix_dim =
94  ", Godunov_matrix_dim, "\n", "limit = ", limit ))
95  close(file)
96  else
97  file = open(string(pathResult, "/log.txt"), "w")
98  write(file, string("Программа завершила работу, отклонение нормы градиента
99  в пределах допустимого значения\n","Входные параметры:\n", "norm_residual
100 = ", norm_residual, "\n", "mantissa = ", mantissa, "\n",
101 "Godunov_matrix_dim = ", Godunov_matrix_dim, "\n", "limit = ", limit ))
102 close(file)
103 end
104 file = open(string(pathResult, "/resultTime.txt"),"w")
105 timer = toc()
106 writedlm(file, timer)
107 close(file)
```

Рисунок 9 – Фрагмент кода программы godunovMatrix

1.3 Результаты работы программы

В процессе работы программы создается каталог result в каталоге, указанном в конфигурационном файле config.txt (параметр pathResult).

По окончанию работы программы godunovMatrix в каталоге result создаются файлы:

- log.txt - содержит информацию о том, как завершилась работа программы (сработало ли ограничение на количество итераций) и с какими входными параметрами программа была запущена;
- resultX.txt - результат работы алгоритма МСГ, вектор *x*;
- resultCompare.txt - выводит в левом столбце вектор, полученный в результате работы алгоритма МСГ примененного к СЛАУ Годунова, в правом столбце решение матрицы Годунова, рассчитанное по формулам (сделано для наглядного сравнения);
- resultDifference.txt - разность вектора, полученного в результате работы алгоритма МСГ, примененного к СЛАУ Годунова, и вектора решения СЛАУ Годунова, рассчитанного по формулам, с увеличением мантиссы на 5 бит;
- resultTime.txt - время работы программы (в секундах);
- maxDiffer.txt - максимальное по модулю число из resultDifference.txt.

2. Сравнение скорости работы программы godunovMatrix с использованием типа BigFloat с программой использующей тип Float64

Для решения больших систем методом сопряженных градиентов будем задействовать числа большой точности. Неизвестно, как их использование влияет на время работы программы. Сравним, во сколько раз изменится время работы программы godunovMatrix, если изменить тип данных с

BigFloat на Float64. Для этого была создана копия программы godunovMatrix, но с изменением типа данных BigFloat на Float64.

Для сравнения, величина мантиссы BigFloat в тесте была равна 53 (величина мантиссы чисел Float64 равна 53).

Результаты (dim – размерность СЛАУ Годунова, norm residual – максимальное отклонение нормы градиента, время указано в секундах):

Использование Float64:

dim=100 norm residual=1e-300 0.246921085

dim=200 norm residual=1e-300 0.275622797

dim=300 norm residual=1e-300 0.28748395

dim=400 norm residual=1e-300 0.321218262

Использование BigFloat, время указано в секундах:

dim=100 norm residual=1e-300 1.197221184

dim=200 norm residual=1e-300 4.242222756

dim=300 norm residual=1e-300 10.645922959

dim=400 norm residual=1e-300 18.009307385

Вывод: использование BigFloat замедляет работу программы:

- для размерности матрицы Годунова 400, в 57 раз;
- для размерности матрицы Годунова 300, в 32 раза;
- для размерности матрицы Годунова 200, в 16 раз;
- для размерности матрицы Годунова 100, в 5 раз.

3. Реализация различных версий метода сопряженных градиентов

Во время изучения МСГ, было найдено 7 различных версий для реализации метода сопряженных градиентов из 3 различных источников:

- Киреев И.В. «Экономичные критерии останова итераций в методе сопряженных градиентов», 2015 [3];

- Киреев И.В. «Метод сопряженных градиентов» - Красноярск, 2011 [2];
- Презентация Ю. Ю. Ушакова о МСГ.

Необходимо проверить, есть ли среди них одинаковые, и провести сравнение между собой всех версий по скорости работы и максимальному отклонению полученного результата от известного решения.

В качестве критерия остановки в цикле алгоритма МСГ для всех версий используется максимально допустимое отклонение нормы градиента (как указано в алгоритме на рисунке 1). Цикл завершается при достижении нормы градиента не превышающей максимально допустимое отклонение нормы градиента. Дополнительный критерий остановки - ограничение на количество итераций.

В статье [3] описан алгоритм МСГ и на выбор для коэффициентов α и β даны по 2 формулы (рисунок 10):

$$\alpha_n = -\frac{\|\mathbf{r}^{n-1}\|^2}{\langle \mathbf{r}^{n-1}, \mathbf{s}^n \rangle_A} = -\frac{\langle \mathbf{r}^{n-1}, \mathbf{s}^n \rangle}{\|\mathbf{s}^n\|_A^2} \qquad \beta_n = -\frac{\langle \mathbf{r}^n, \mathbf{s}^n \rangle_A}{\|\mathbf{s}^n\|_A^2} = \frac{\|\mathbf{r}^n\|^2}{\|\mathbf{r}^{n-1}\|^2}$$

Рисунок 10 – Фрагмент из статьи И. В. Киреева [3]

Таким образом, комбинируя формулы, получаем 4 алгоритма МСГ.

В презентации Киреева [2] берем только алгоритм из 16 слайда.

Алгоритм из презентации Ушакова о МСГ представлен на рисунке 11. В обозначения этого алгоритма, решается СЛАУ вида $E \cdot x = b$, где E – произвольная матрица размерности N*M, b – вектор размерности N.

В алгоритме Ю. Ю. Ушакова, перед началом работы цикла, матрица A рассчитывается в виде $A = E^T \cdot E$. Не рассчитывая значение матрицы A, подставим $E^T \cdot E$ в алгоритм вместо матрицы A. Тогда, получим ещё одну версию МСГ (рисунок 12).

Метод сопряженных градиентов

$$d_{(0)} = r_{(0)} = c - Ax_{(0)},\ A = E^T E,\ c = E^T b$$

$$\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{d_{(i)}^T A d_{(i)}}$$

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)},$$

$$r_{(i+1)} = r_{(i)} - \alpha_{(i)} A d_{(i)},$$

$$\beta_{(i+1)} = \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}},$$

$$d_{(i+1)} = r_{(i+1)} + \beta_{(i+1)} d_{(i)}.$$

Рисунок 11 – Слайд из презентации Ушакова о МСГ

$$d_0 = r_0 = c - E^T E x_0$$

$$a_{(i)} = \frac{r_{(i)}^T r_{(i)}}{\left(E d_{(i)}\right)^T (E d_{(i)})}$$

$$x_{(i+1)} = x_{(i)} + a_{(i)} d_{(i)}$$

$$r_{(i+1)} = r_{(i)} - a_{(i)} E^T (E d_{(i)})$$

$$\beta_{(i+1)} = \frac{r_{(i+1)}{}^T r_{(i+1)}}{r_{(i)}{}^T r_{(i)}}$$

$$d_{(i)} = r_{(i+1)} + \beta_{(i+1)} d_{(i)}$$

Рисунок 12 – Модифицированный алгоритм МСГ Ушакова

Проведены запуски программ godunovMatrix, отличающихся вариантом алгоритма МСГ, изменяя для каждой программы размерность СЛАУ Годунова, максимально допустимое отклонение нормы градиента и величину мантиссы чисел BigFloat.

Обозначения, используемые в выводе результатов:

- dim — размерность СЛАУ Годунова;
- norm residual — максимально допустимое отклонение нормы градиента (служит критерием остановки в цикле алгоритма МСГ);
- mantissa — количество бит в мантиссе чисел типа BigFloat.

Время работы программ указано после слова “timer:” (указывается в секундах).

Примечание: если целая часть числа равна нулю, нуль опускается. Так, например число 0.24 записывается в виде .24 - особенность записи в файл в Julia.

1)Киреев И.В. «Экономичные критерии останова итераций в методе сопряженных градиентов» [3] использована 1 формула для расчета альфа и 1 формула для расчета бета:

dim=200 norm residual=1e-500 mantissa=2000 timer: 19.085446518

dim=100 norm residual=1e-250 mantissa=1000 timer: 3.933407718

dim=50 norm residual=1e-125 mantissa=500 timer: 1.125106475

dim=25 norm residual=1e-62 mantissa=250 timer: .832668464

2)Киреев И.В. «Экономичные критерии останова итераций в методе сопряженных градиентов» [3] использована 1 формула для расчета альфа и 2 формула для расчета бета:

dim=200 norm residual=1e-500 mantissa=2000 timer: 13.131313187

dim=100 norm residual=1e-250 mantissa=1000 timer: 2.784430227

dim=50 norm residual=1e-125 mantissa=500 timer: 1.01434431

dim=25 norm residual=1e-62 mantissa=250 timer: .820628748

3)Киреев И.В. «Экономичные критерии останова итераций в методе сопряженных градиентов» [3] использована 2 формула для расчета альфа и 1 формула для расчета бета:

dim=200 norm residual=1e-500 mantissa=2000 timer: 19.399853095

dim=100 norm residual=1e-250 mantissa=1000 timer: 3.990158701

dim=50 norm residual=1e-125 mantissa=500 timer: 1.13305135

dim=25 norm residual=1e-62 mantissa=250 timer: .827894873

4)Киреев И.В. «Экономичные критерии останова итераций в методе сопряженных градиентов» [3] использована 2 формула для расчета альфа и 2 формула для расчета бета:

dim=200 norm residual=1e-500 mantissa=2000 timer: 13.295699026

dim=100 norm residual=1e-250 mantissa=1000 timer: 2.760137909

dim=50 norm residual=1e-125 mantissa=500 timer: 1.015523375

dim=25 norm residual=1e-62 mantissa=250 timer: .821944988

5)Презентация Киреев И.В. «Метод сопряженных градиентов» - Красноярск, 2011 [2] — 16 слайд:

dim=200 norm residual=1e-500 mantissa=2000 timer: 6.694421971

dim=100 norm residual=1e-250 mantissa=1000 timer: 1.611667973

dim=50 norm residual=1e-125 mantissa=500 timer: .651669404

dim=25 norm residual=1e-62 mantissa=250 timer: .533816934

6)Слайды Ушакова о МСГ (рисунок 11):

dim=200 norm residual=1e-500 mantissa=2000 timer: 15.10778447

dim=100 norm residual=1e-250 mantissa=1000 timer: 3.959465588

dim=50 norm residual=1e-125 mantissa=500 timer: 2.097025044

dim=25 norm residual=1e-62 mantissa=250 timer: 1.932025239

7)Алгоритм МСГ Ушакова с изменениями (рисунок 12):

dim=200 norm residual=1e-500 mantissa=2000 timer: 9.927693616

dim=100 norm residual=1e-250 mantissa=1000 timer: 2.881636564

dim=50 norm residual=1e-125 mantissa=500 timer: 2.178825141

dim=25 norm residual=1e-62 mantissa=250 timer: 2.080774947

Сравним время работы программ для разных размерностей (цифры указывают на номер программы, нумерация указана выше). Введем обозначение: "1<2" будет обозначать, что время работы программы godunovMatrix под номером 1 меньше времени работы программы под номером 2.

Для каждой размерности получаем:

dim=200 5<7<2<4<6<1<3

dim=100 5<4<2<7<1<6<3

dim=50 5<2<4<1<3<6<7

dim=25 5<2<4<3<1<6<7

Сравнение программ по максимальному модулю отклонения вектора, полученного в результате работы алгоритма МСГ, примененного к СЛАУ Годунова, от вектора решения СЛАУ Годунова, рассчитанного по формулам, с увеличением мантиссы на 5 бит.

Максимальное отклонение указывается после слова “maxDifference:”.

1)Киреев И.В. «Экономичные критерии останова итераций в методе сопряженных градиентов» [3] использована 1 формула для расчета альфа и 1 формула для расчета бета:

dim=200 norm residual=1e-500 mantissa=2000 maxDifference: 8.906e-02

dim=100 norm residual=1e-250 mantissa=1000 maxDifference: 1.417e-220

dim=50 norm residual=1e-125 mantissa=500 maxDifference: 1.147e-110

dim=25 norm residual=1e-62 mantissa=250 maxDifference: 3.455e-56

2)Киреев И.В. «Экономичные критерии останова итераций в методе сопряженных градиентов» [3] использована 1 формула для расчета альфа и 2 формула для расчета бета:

dim=200 norm residual=1e-500 mantissa=2000 maxDifference: 8.906e-02

dim=100 norm residual=1e-250 mantissa=1000 maxDifference: 6.978e-220

dim=50 norm residual=1e-125 mantissa=500 maxDifference: 8.114e-111

dim=25 norm residual=1e-62 mantissa=250 maxDifference: 4.233e-56

3)Киреев И.В. «Экономичные критерии останова итераций в методе сопряженных градиентов» [3] использована 2 формула для расчета альфа и 1 формула для расчета бета:

dim=200 norm residual=1e-500 mantissa=2000 maxDifference: 8.906e-02

dim=100 norm residual=1e-250 mantissa=1000 maxDifference: 7.440e-220

dim=50 norm residual=1e-125 mantissa=500 maxDifference: 1.129e-110

dim=25 norm residual=1e-62 mantissa=250 maxDifference: 3.452e-56

4)Киреев И.В. «Экономичные критерии останова итераций в методе сопряженных градиентов» [3] использована 2 формула для расчета альфа и 2 формула для расчета бета:

dim=200 norm residual=1e-500 mantissa=2000 maxDifference: 8.906e-02

dim=100 norm residual=1e-250 mantissa=1000 maxDifference: 2.404e-219

dim=50 norm residual=1e-125 mantissa=500 maxDifference: 1.340e-110

dim=25 norm residual=1e-62 mantissa=250 maxDifference: 2.956e-56

5)Презентация Киреев И.В. «Метод сопряженных градиентов» - Красноярск, 2011 [2] — 16 слайд:

dim=200 norm residual=1e-500 mantissa=2000 maxDifference: 8.906e-02

dim=100 norm residual=1e-250 mantissa=1000 maxDifference: 1.586e-219

dim=50 norm residual=1e-125 mantissa=500 maxDifference: 7.372e-111

dim=25 norm residual=1e-62 mantissa=250 maxDifference: 2.950e-56

6)Слайды Ушакова о МСГ (рисунок 11):

dim=200 norm residual=1e-500 mantissa=2000 maxDifference: 8.906e-02

dim=100 norm residual=1e-250 mantissa=1000 maxDifference: 4.108e-220

dim=50 norm residual=1e-125 mantissa=500 maxDifference: 1.362e-110

dim=25 norm residual=1e-62 mantissa=250 maxDifference: 5.703e-57

7)Алгоритм МСГ Ушакова с изменениями (рисунок 12):

dim=200 norm residual=1e-500 mantissa=2000 maxDifference: 4.337e-04

dim=100 norm residual=1e-250 mantissa=1000 maxDifference: 8.638e-04

dim=50 norm residual=1e-125 mantissa=500 maxDifference: 1.714e-03

dim=25 norm residual=1e-62 mantissa=250 maxDifference: 3.384e-03

Сравним максимальные отклонения решения СЛАУ Годунова методом сопряженных градиентов от решения СЛАУ Годунова рассчитанного по

формулам (цифры указывают на номер программы, нумерация указана выше).

Введем обозначение: "1<2" будет обозначать что максимальное отклонение программы godunovMatrix под номером 1 меньше максимального отклонения программы под номером 2, знаком "=" обозначим, что максимальные отклонения совпали. Для каждой размерности получаем:

dim=200 7<1=6=2=3=5=4

dim=100 1<6<2<3<5<4<7

dim=50 5<2<3<1<4<6<7

dim=25 6<5<4<3<1<2<7

Вывод:

Проведенное выше исследование показало, что алгоритм под номером 5 затрачивает на своё выполнение меньше времени, чем другие алгоритмы. При этом для некоторых размерностей СЛАУ Годунова алгоритм 5 уступает в точности другим алгоритмам.

Нельзя однозначно сказать, что один алгоритм лучше других одновременно и по быстродействию и по точности нахождения решения.

4. Сравнение времени работы программы godunovMatrix реализованной на julia с программой МСГ реализованной на C++

Разработчики языка программирования Julia утверждают, что Julia практически не уступает в скорости языку C++. Проверим это высказывание на примере сравнения быстродействия программы для решения СЛАУ Годунова методом сопряженных градиентов.

Ю. Ю. Ушаков предоставил программу, реализованную на языке программирования C++, для решения произвольных СЛАУ методом сопряженных градиентов. В этой программе расчет производится с точностью long double.

В программе godunovMatrix, написанной на языке программирования Julia [5], использовался тип данных Float64.

В программы подставлялись СЛАУ Годунова одинаковых размерностей (СЛАУ Годунова представлена в приложении А).

При проведении тестов, сначала запускалась программа Ушакова на C++ и после, в программе godunovMatrix, ограничение на количество итераций выбиралось такое же, сколько потребовалось итераций программе на C++.

Julia использует JIT (Just-In-Time) компилятор, то есть код программы компилируется во время исполнения программы. Это значит, что часть времени после запуска программы уйдет на компиляцию. А при повторном запуске программы (даже с другими параметрами запуска) в одной консольной сессии, не закрывая консоль с Julia, компиляция кода не потребуется, и время работы программы может значительно сократиться.

Далее повторный пересчет подразумевает повторный запуск программы godunovMatrix.jl в той же консоли не закрывая консоль.

Результаты (время указано в секундах):

Размерность 50, количество итераций 36:

c++	0.00111899999999999986
Julia	2.274982477
Julia (при повторном пересчете)	0.020991488

программа на C++ затрачивает в 2049 раз меньше времени и в 18,77 при повторном пересчете.

Размерность 100, количество итераций 36:

c++	0.00233000000000000004
Julia	2.353452287
Julia (при повторном пересчете)	0.03265142

программа на C++ затрачивает в 1010 раз меньше времени и в 14,19 при повторном пересчете.

Размерность 250, количество итераций 36:

c++	0.002700000000000000143
Julia	2.317531849
Julia (при повторном пересчете)	0.032741284

программа на C++ затрачивает в 858 раз меньше времени и в 12 при повторном пересчете.

Размерность 500, количество итераций 36:

c++	0.007844999999999999543
Julia	2.340429235
Julia (при повторном пересчете)	0.048053456

программа на C++ затрачивает в 298 раз меньше времени и в 6,12 при повторном пересчете.

Размерность 1000, количество итераций 36:

c++	0.0107119999999999993
Julia	2.396075568
Julia (при повторном пересчете)	0.102384972

программа на C++ затрачивает в 223 раз меньше времени и в 10,2 при повторном пересчете.

Размерность 5000, количество итераций 36:

c++	0.05133700000000000069
Julia	6.638375584
Julia (при повторном пересчете)	4.168623977

программа на C++ затрачивает в 130 раз меньше времени и в 81,25 при повторном пересчете.

5. Универсальная интерполяция орбит ГЛОНАСС и GPS по данным IGS

Решение задачи определения координат спутников ГЛОНАСС и GPS тесно связано с решением больших систем линейных алгебраических уравнений. В процессе решения задачи только для спутников ГЛОНАСС возникает система и 30000 строк и 6 неизвестными.

В статье С. П. Царева и А. С. Пустошилова [11] показано, что используя только 6 последовательных позиций спутников, с некоторым установленным интервалом времени, можно найти позицию спутников внутри интервала более точно, чем другими интерполяционными методами.

Определение позиции спутников в любой момент времени является фундаментальной частью в спутниковой навигации. Информация о положении спутников обычно описывается в текстовом виде в формате SP3, который содержит координаты спутников. SP3 файлы берутся из центра IGS (International GNSS Service [7]), где координаты рассчитаны с погрешностью не более 2,5 см и представлены с интервалом в 15 мин.

Для нахождения координат спутников в любой момент времени используются интерполяционные методы. Наиболее часто применяется на практике интерполяционный полином Лагранжа. В статье [11] указано, что в этом методе среднеквадратическое отклонение невязки составляет 6,5 мм при использовании 10 узловых точек интерполяции, чего хватает для решения многих навигационных проблем.

Методика, предложенная в статье [11], обладает следующими свойствами:

- Нет необходимости выбирать интерполирующую функцию. Здесь функция автоматически адаптируется к любой орбите или временному ряду;

- Недостаток методики заключается в необходимости предварительного вычисления интерполяционных коэффициентов для каждого интерполяционного шаблона.

В статье [11] показаны результаты расчета для одной целевой точки интерполяционного шаблона.

Методика [11] основывается на известной интерполяционной формуле Лагранжа:

$$L(\hat{t}) = \sum_{i=1}^{n} x(t_i) \cdot a_i(\hat{t}), \qquad (1)$$

где $L(\hat{t})$ - интерполирующая функция для промежуточных моментов времени $\hat{t}$; $x(t_i)$ - позиции спутника в момент времени t_i, i=1.. n; n – количество узлов интерполяции; $a_i(\hat{t}) = \prod_{k=1,k\neq i}^{n} \frac{\hat{t} - t_k}{t_i - t_k}$.

От вида коэффициентов $a_i(\hat{t})$ в формуле (1) зависит, какой способ интерполяции используется (полиномиальная, тригонометрическая или какая-либо другая интерполяция) [4, 6, 9, 13,14].

Как показано в [11], фиксируем n последовательных моментов времени t_i (i=1..n) из SP3 файлов и целевой момент времени $\hat{t} = t_j$ (момент времени в котором будет происходить расчет координаты) в интервале между t_i. Тогда получим фиксированный *интерполяционный шаблон* из n точек t_i и целевой точки t_j. С учетом фиксированного интерполяционного шаблона, для коэффициентов $a_i(\hat{t})$ можно опустить зависимость от $\hat{t}$ и тогда, внеся в (1) $\varepsilon(t_j)$ - величину характеризующую невязку для момента времени t_j, (1) запишем в виде:

$$\underline{x}(t_j) = \sum_{i=1}^{n} a_i \cdot x(t_i) + \varepsilon(t_j), \qquad (2)$$

где a_i неизвестные коэффициенты, $x(t_i)$ – координата спутника из SP3 файла в момент времени t_i; $\underline{x}(t_j)$ - целевая координата (в которой находится спутник в целевой момент времени t_j); $\varepsilon(t_j)$ - неизвестная величина, характеризующая невязку интерполяции для момента времени t_j.

Для нахождения коэффициентов a_i и невязки $\varepsilon(t_j)$ понадобится, как минимум, n+1 уравнение. Как указано в статье [11], сделаем сдвиг во времени для интерполяционного шаблона с шагом 900 секунд. При этом мы считаем, что коэффициенты a_i не изменяются. Совершив m таких сдвигов, получим систему уравнений:

$$\begin{cases} \underline{x}(t_j) = \sum_{i=1}^{n} a_i \cdot x(t_i) + \varepsilon(t_j) \\ \cdots \\ \underline{x}(t_j + 900m) = \sum_{i=1}^{n} a_{i+m} \cdot x(t_{i+m}) + \varepsilon(t_j + 900m) \end{cases}. \qquad (3)$$

А. С. Пустошилов предоставил программу (на языке программирования Julia [5]), которая считывает координаты из файлов, содержащих координаты спутников, и подготавливает их к последующему решению системы (3) относительно неизвестных a_i методом наименьших квадратов. Расчет производит для узлов интерполяции с шагом 15 мин., а целевая точка отсчитывается с шагом 5 минут от 1 точки интерполяционного шаблона. Программа была переписана так, чтобы она считывала координаты из файлов с шагом 1 сек. Теперь отсчет целевой точки происходит с шагом 1 сек., для шага узлов интерполяции введена переменная и добавлен цикл для перебора целевых точек. Подставив файлы с координатами спутников в программу, программа считывает их, подготавливает данные с координатами

к решению системы (3), решает её и создает текстовой файл, содержащий набор коэффициентов, максимум от невязки и среднеквадратическое отклонение (СКО) невязки для каждой целевой точки.

Один из центров IGS (CODE) предоставляет координаты с шагом 1 сек. В программу были подставлены файлы, содержащие координаты для спутников ГЛОНАСС за период 10.03.2017 – 14.03.2017 с шагом 1 сек. Шаг для узлов интерполяции выбран 900 сек. В результате работы программы получен файл, содержащий СКО невязки для каждой целевой точки интерполяционного шаблона. При помощи Julia был построен график зависимости величины СКО невязки от целевой точки (Рисунок 13).

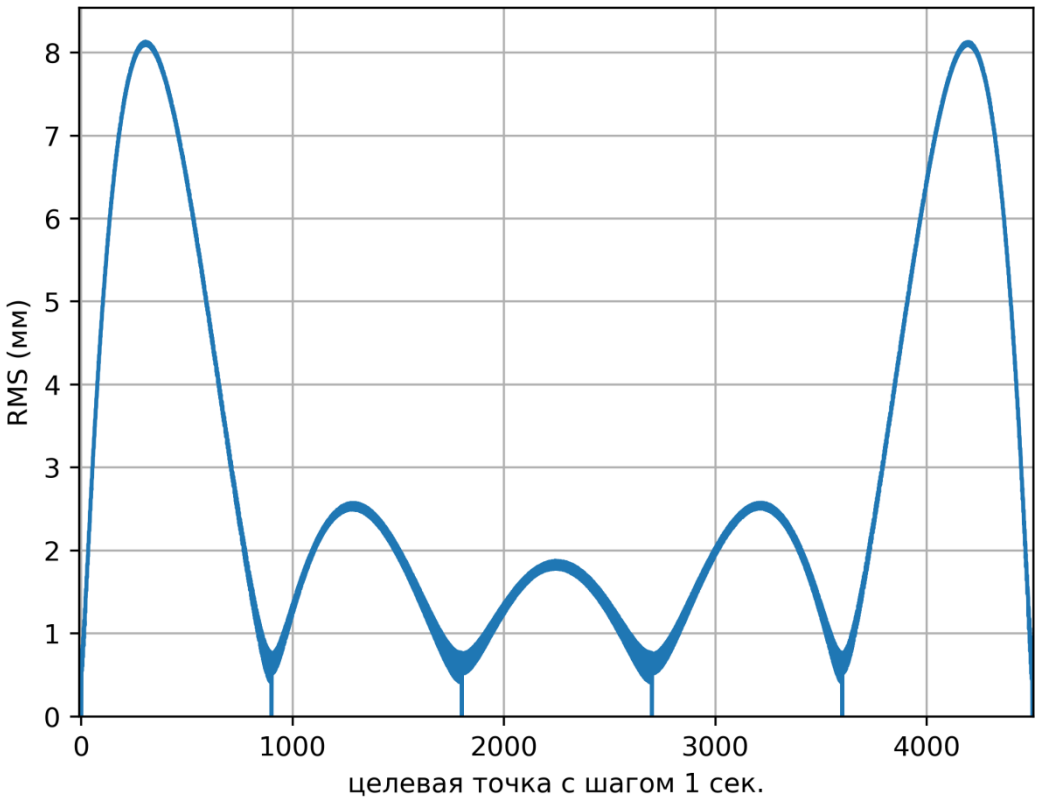

Рисунок 13 – Среднеквадратическое отклонение (RMS) невязки для спутников ГЛОНАСС по координатам из файлов центра IGS CODE

На рисунке 13 на границах интерполяционного шаблона наблюдается увеличение СКО невязки в 3,1 раза по сравнению с максимумами СКО

невязок в середине шаблона. Это явление описывается феноменом Рунге. Наибольшее СКО невязки в середине равно 2,6 мм.

Рисунок 13 подтверждает, что метод, указанный в статье [11], позволяет по 6 точкам находить промежуточные координаты с высокой точностью (в статье [11] СКО невязки в середине шаблона составляет 3.88 мм для спутников ГЛОНАСС).

В программу были подставлены файлы, которые предоставил мне Ю. Ю. Ушаков, рассчитанные по его методике [12] для спутника GPS G01 за 2013 год. Координаты из файлов имеют шаг 1 сек. Шаг для узлов интерполяции был установлен 900 сек. В результате работы программы, получен файл, содержащий СКО невязки для каждой целевой точки. График зависимости величины СКО от целевой точки изображен на рисунке 14.

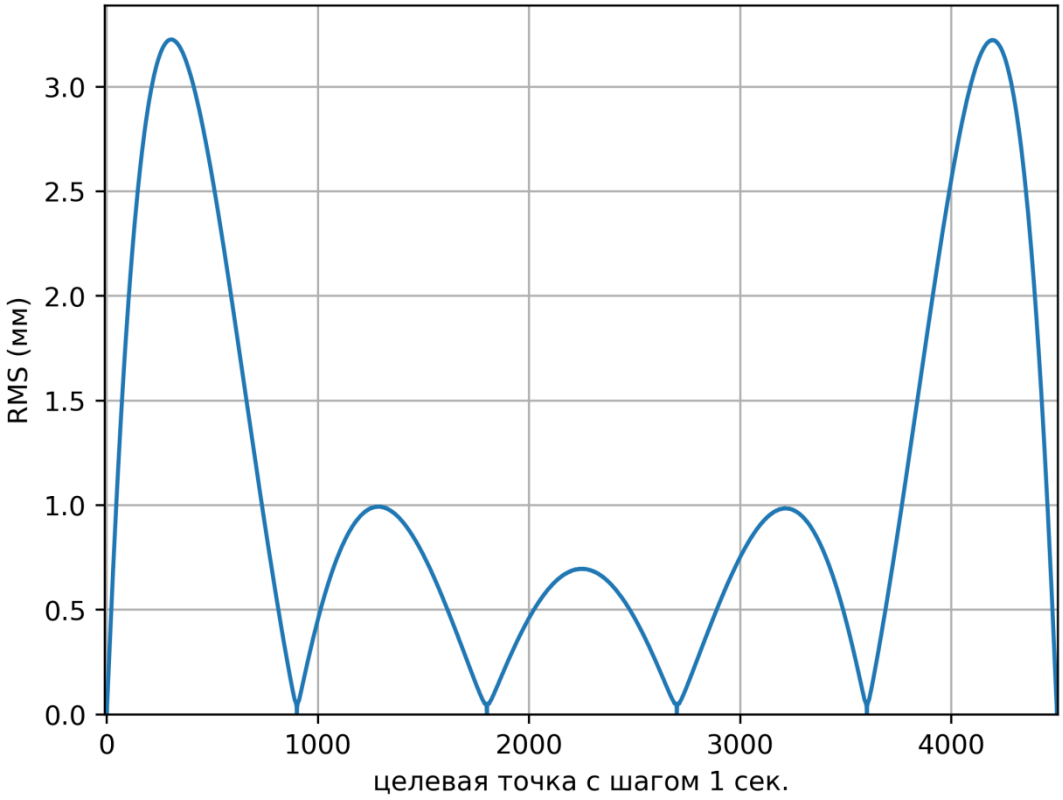

Рисунок 14 – Среднеквадратическое отклонение (RMS) невязки для спутника GPS G01 по координатам из файлов Ю. Ю. Ушакова

На рисунке 14, на границах интерполяционного шаблона СКО невязки возрастает до 3,4 мм. Это явление описывается феноменом Рунге. Наибольшее значение СКО невязки в середине шаблона равно 1 мм.

В статье [11] расчет производится только для одной целевой точки в середине интерполяционного шаблона. В моей работе целевая точка перебирается с шагом 1 сек. по всему интерполяционному шаблону. Таким образом, по рисункам 13 и 14 можно определить СКО невязки для каждой точки интерполяционного шаблона, а не только для середины.

6. Чтение и запись файлов в Julia

В процессе работы с программой для расчета невязки в некоторой целевой точке интерполяционного шаблона по методике [11], было замечено, что большую часть времени работы программы занимает считывание координат из файлов. Для каждого запуска программы нужно ждать окончания считывания файлов. Для ускорения процесса считывания, было вынесено предложение, записывать все считанные координаты в один файл, чтобы при последующем запуске программы на считывание уходило гораздо меньше времени. Для этого, прежде всего, нужно разобраться, каким методом лучше производить чтение и запись в Julia.

6.1 Сравнение методов чтения и записи в Julia

Сравним 4 варианта чтения и записи в языке программирования Julia:

- write(stream, value) – записывает бинарное представление value в файл указанный в stream. value может быть числом, строкой, массивом (далее в этой работе записывается массив). Для чтения полученного файла использовалась команда read!(stream, mas) – считывает бинарный файл, указанный в stream, и заполняет массив mas.

- writedlm(file, A, *delim='\t'; opts*) – запись матрицы A в файл file в форме текста с разделителем delim=’\t’ по умолчанию. Считывание производится командой readdlm(stream).
- JLD [8] – модификация HDF5 (Hierarchical Data Format**,** название формата файлов, разработанного для хранения большого количества цифровой информации).
- serialize(stream, value) – записывает value в файл указанный в stream. Для чтения полученного файла используется команда deserialize(stream) – возвращает записанное значение из файла. Чтение может производиться некорректно, если запись происходила на другой версии языка программирования Julia.

Прежде чем работать с модулем JLD, его требуется установить. Для установки модуля JLD в Julia используется команда:

```
Pkg.add("JLD")
```

При этом требуется, чтобы в системе был установлен HDF5.

Чтобы задействовать модуль JLD :

```
using JLD
```

Реализованы 4 программы для записи массива размерностью 30000x6 в файлы разными методами записи и 4 программы для чтения массива из созданных файлов. Коды программ расположены в приложении B.

В результате запуска каждой из программ, создавались текстовые файлы, содержащие время записи или чтения (в зависимости от программы).

Результаты:

r – чтение, w – запись (в секундах).

write r - 0.01994581 сек, w - 0.026957968 сек, size – 1440000 бит

writedlm r - 2.137002756 сек, w - 0.313572262 сек, size – 3468779 бит

JLD r - 1.845941324 сек, w - 0.767692832 сек, size – 1444456 бит

Serialize r - 0.074389281 сек, w - 0.204687758 сек, size – 1440015 бит

Наиболее быстрое запись-чтение файла достигается при использовании команд “read” и “write”. Эти команды не позволяют записывать и считывать сложные типы данных, поэтому требуется реализовать специальные функции для записи и чтения.

6.2 Метод записи и чтения бинарного файла в Julia

Программа для расчета невязки в некоторой целевой точке интерполяционного шаблона по методике [11] затрачивает большую часть времени на считывание координат спутников из файлов IGS (Для файлов IGS CODE спутников ГЛОНАСС время чтения составляет 370 секунд). Это объясняется тем, что считывание происходит командой readdlm. Выше было показано, что команда readdlm считывает в 200 раз медленнее, чем команда read. Лучше всего использовать команду write - для записи и read - для чтения.

read(stream, Type, n) – считывает из файла, указанного в stream, количество бит равное числу бит необходимых для хранения переменной типа Type и распознает, как значение типа Type. Такое считывание повторяется n раз. При n>1 функция read(stream, Type, n) возвращает массив типа Type из n элементов.

Разработан метод записи и чтения массива имеющего тип array{onefile}, то есть массив из данных типа onefile. Тип onefile не из стандартных типов данных, его структура представлена на рисунке 15.

```
1 Type onefile{String,T}
2   name::String # имя файла с которого считываются координаты
3   values::T    # Массив координат спутника
4   TS_time      # Массив хранящий время спутника
5 End
```

Рисунок 15 – Структура типа данных onefile

Для записи массива array{onefile} предложена функция binwritefiles представленная на рисунке 16.

```
1  function binwritefiles(file, allfiles)  # запись содержимого allfiles в
 2                                          # файл file.
 3   n = length(allfiles)  # n – количество элементов в массиве allfiles.
 4                         # allfiles имеет тип array{onefile}.
 5   write(file, n::Int)   # в файл записывается число n.
 6   for i in 1:1:n
 7     name = allfiles[i].name
 8     len = length(name)           # len - кол-во символов в названии.
 9     write(file, len)             # в файл записывается len.
10     write(file, name)            # в файл записывается строка name.
11     values = allfiles[i].values
12     len = length(values)         # len - кол-во значений в массиве values.
13     write(file, len)             # в файл записывается len.
14     write(file, values           # в файл записывается массив values (тип
15                                 #  элементов - Float64).
16     write(file, allfiles[i].TS_time) # записывается массив TS (тип
17                            #элементов Int) (кол-во элементов массива = len).
18   end
19 end
```

Рисунок 16 – Структура функции binwritefiles

Функция binwritefiles записывает массив типа Array{onefile} в файл в бинарном виде.

Для чтения файла, созданного функцией binwritefiles, предложена функция binreadfiles представленная на рисунке 17.

```
function binreadfiles(file)
  count_files = read(file,Int)      # из файла считывается кол-во байт
                                    # необходимое для записи числа типа Int.
  allfiles = Array{onefile}(count_files)  # создается массив типа onefile
                                          # из count_files элементов.
  for i in 1:1:count_files
    len = read(file, Int)            # из файла считывается число типа Int
                                     #(указывает на кол-во символов в имени.
    name = read(file, len)           # считывается len байт.
    name = String(name)       # считанные len байт переводятся в тип String.
    len = read(file, Int)      # из файла считывается число типа Int
                               #(указывает на кол-во элементов в массиве).
    values = read(file, Float64, len)    # из файла считывается len чисел
                                         # типа Float64.
    TS_time = read(file, Int, len)       # из файла считывается len чисел
                                         # типа Int.
    allfiles[i]=onefile(name, values, TS_time)
  end
  return allfiles          # функция возвращает массив типа Array{onefile}.
end
```

Рисунок 17 – Структура функции binreadfiles

Подставив в программу файлы IGS CODE для спутников ГЛОНАСС:

- Время записи Array{onefile} в файл: 1.673159742 секунд.
- Время чтения из бинарного файла: 0.56589274 секунд.

Получаем, что время считывания при использовании разработанного бинарного метода чтения - записи сократилось в 660 раз.

7. Сравнение времени работы алгоритма метода сопряженных градиентов и оператора “A\B” в Julia

В программе для расчета невязки в некоторой целевой точке интерполяционного шаблона по методике [11], при подстановке файлов IGS CODE, спутников ГЛОНАСС, формируется матрица размером 26004x6.

Для решения системы в программе использовался встроенный в Julia оператор “A\B” для решения систем вида A*x=B, где A – матрица m*n, B – вектор столбец размерности m, x – искомый вектор размерности n.

Оператор “A\B” работает только для плотных матриц. Он не срабатывает для разряженных матриц (заданных командой sparse). В прикладных задачах космической навигации часто возникают большие разряженные системы. В качестве альтернативы оператору “A\B” рассматривается МСГ.

В теории, МСГ приводит к точному решению за количество итераций равное количеству неизвестных в СЛАУ. Но, из-за неточности представления чисел в ЭВМ, точное решение системы может быть не получено за это количество итераций. При этом среднеквадратическое отклонение невязки может уменьшиться при последующих итерациях. В программе решается система с шестью неизвестными. По этому, ограничение на количество итераций для МСГ выбрано равное 6. Дополнительно проведем вычисления с количеством итераций равным 8.

Ещё один способ уменьшить среднеквадратическое отклонение невязки – повысить точность представления чисел. Точность представления чисел в ЭВМ зависит от мантиссы числа с плавающей запятой. В числах произвольной точности количество бит в мантиссе числа можно изменять.

Программа была модифицирована путем добавления метода сопряженных градиентов в программу для решения системы, имеющую плотную матрицу.

Сравним время работы оператора “A\B” с временем работы алгоритма МСГ, и сравним среднеквадратические отклонения невязки для найденных решений систем.

Для сравнения, в качестве алгоритмов МСГ, были взяты алгоритмы, представленные в источниках:

- презентация Киреев И.В. «Метод сопряженных градиентов» - Красноярск, 2011 [2] ;
- Презентация Ю. Ю. Ущакова (алгоритм приведен на рисунке 11).

В качестве критерия остановки использовался только критерий ограничения количества итераций.

В ходе экспериментов в программе изменялись количество итераций и величина мантиссы BigFloat.

Результаты представлены в таблице 1.

Таблица 1 – Сравнение оператора «A\B» с двумя алгоритмами МСГ

Итераций	Мантисса	A\B		МСГ Киреева [4]		МСГ Ушакова	
		СКО невязки	Время (сек)	СКО невязки	Время (сек)	СКО невязки	Время (сек)
6	300	9,53E-07	3,86	9,53E-07	2,49	9,53E-07	2,50
6	200	9,53E-07	3,60	9,53E-07	2,26	9,53E-07	2,40
6	100	9,53E-07	3,45	5,03E-06	2,01	5,03E-06	2,38
8	300	9,53E-07	3,65	9,53E-07	2,99	9,53E-07	2,47
8	200	9,53E-07	3,20	9,53E-07	2,65	9,53E-07	2,46
8	100	9,53E-07	3,51	5,03E-06	2,60	5,03E-06	2,36

Для сравнения точности решения полученного МСГ решим СЛАУ оператором “A\B” с мантиссой 1000 бит и решим СЛАУ методом сопряженных градиентов с мантиссой 300 бит.

При 8 итерациях максимум модуля разности векторов решений СЛАУ, полученных в результате применения МСГ и оператора “A\B” составляет:

- Для МСГ Киреева [2] - 2.92e-67;
- Для МСГ Ушакова (алгоритм приведен на рисунке 11) - 2.66e-65.

При 6 итерациях максимум модуля разности векторов решений СЛАУ, полученных в результате применения МСГ и оператора “A\B” составляет:

- Для МСГ Киреева [2] - 1.16e-51;
- Для МСГ Ушакова (алгоритм приведен на рисунке 11) - 1.52e-52.

Заключение

У метода сопряженных градиентов есть различные вариации алгоритма, но нельзя точно определить, какой работает быстрее и точнее других одновременно. МСГ работает приемлемо, но он требователен к точности представления чисел. Рекомендуется использовать числа большой точности, что замедляет работу программы. Например, для размерности СЛАУ Годунова равной 400 время работы программы при использовании чисел большой точности увеличилось в 57 раз. Но, как показывает практика, для системы размерностью 26004x6 время работы алгоритма МСГ меньше в 1,5 раза, чем время работы встроенного в Julia оператора “A\B” при одинаковой точности представления чисел.

Среди всех возможных вариантов чтения-записи в Julia определен наиболее быстрый. Разработана методика быстрого чтения-записи файлов, хранящих координаты, время и названия спутников. Эта методика ускоряет время считывания в 660 раз.

Список сокращений

МСГ – метод сопряженных градиентов;

СКО – среднеквадратическое отклонение;

СЛАУ – система линейных алгебраических уравнений;

ЭВМ – электронно-вычислительная машина;

CODE – Center for Orbit Determination in Europe;

GNSS – Global Navigation Satellite System;

IGS – International GNSS Service.

Список использованных источников

1. Годунов, С.К. Решение систем линейных уравнений - Новосибирск: Наука, 1980.

2. Киреев, И.В. Метод сопряженных градиентов. Красноярск, 2011.

3. Киреев, И.В. Экономичные критерии останова итераций в методе сопряженных градиентов / И.В. Киреев // Вычислительные технологии. – Красноярск, 2015. – Т. 20, № 2.

4. Feng Y. and Zheng Y. Efficient interpolations to GPS orbits for precise wide area applications. *GPS Solutions*, 2005, vol. 9, pp 67-72 .

5. Julia documentation [Electronic resource] / Bezanson J., Karpinski S., Shah V.B. – Mode of access: https://docs.julialang.org/en/stable.

6. Horemuz, M. and Andersson, J. V. Polynomial interpolation of GPS satellite coordinates. *GPS Solutions*, 2006, vol. 10, pp 67-72 .

7. Official site of IGS [Electronic resource] / IGS. – Mode of access: http://www.igs.org/products.

8. Saving and loading variables in Julia Data format [Electronic resource] / Holy T. E., Hinsen K., Short T., Kornblith S. – Mode of access: https://github.com/JuliaIO/JLD.jl.

9. Schenewerk, M. A brief review of basic GPS orbit interpolation strategies. *GPS Solutions*, 2003, vol. 6, pp 265-267.

10. Shewchuk J. R. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. *Technical report, School of Computer Science, Carnegie Mellon University*, 1994, vol. CMU-CS-94-125.

11. Tsarev, S. P., Pustoshilov, A. S. Universal Coefficients for Precise Interpolation of GNSS Orbits from Final IGS SP3 Data. 2017 International Siberian Conference on Control and Communications, *SIBCON 2017 - Proceedings 2017.*

12. Ushakov Yu. Evaluation of acceleration and motion model parameters of a space vehicle from IGS final orbits. 2013 International Siberian Conference on

Control and Communications, *SIBCON 2013 - Proceedings 2013*, Article number 6693594.

13. Xue S. and Yang Y. Recursive algorithm for fast GNSS orbit fitting. *GPS solutions,* 2016, vol. 20, 151-157.

14. Yousif H. and El-Rabbany A. Assessment of Several Interpolation Methods for Precise GPS Orbit. *The journal of navigation,* 2007, vol. 60, pp 443–455.

ПРИЛОЖЕНИЕ А

Система линейных алгебраических уравнений Годунова

Система линейных уравнений Годунова [1] представляет собой систему N уравнений вида:

$$A_N \cdot x = B_N$$

Где A_N - двухдиагональная матрица (рисунок А.1):

$$A_N = \begin{pmatrix} \frac{7}{5} & \frac{11}{3} & & & 0 \\ & \frac{7}{5} & \frac{11}{3} & & \\ & & \ddots & \ddots & \\ & & & \frac{7}{5} & \frac{11}{3} \\ 0 & & & & \frac{7}{5} \end{pmatrix}$$

Рисунок А.1 – матрица левой части СЛАУ Годунова

Вектор B_N (рисунок А.2):

$$B_N = \begin{pmatrix} \frac{6}{5} \\ \vdots \\ \frac{152k+118}{15(2k+1)(2k+3)} \\ \vdots \\ \frac{152N-34}{15(2N-1)(2N+1)} \\ \frac{7}{5(2N+1)} \end{pmatrix}$$

Рисунок А.2 – правая часть СЛАУ Годунова

Решение системы уравнений Годунова можно найти по формулам:

$x_1 = \frac{1}{3}, x_2 = \frac{1}{5}, \dots, x_k = \frac{1}{2k+1}, \dots, x_{N-1} = \frac{1}{2N-1}, x_N = \frac{1}{2N+1}.$

ПРИЛОЖЕНИЕ Б

Код модуля readFile.jl программы godunovMatrix

```
file = open(string(path, "/config.txt"),"r") # открытие файла config.txt
strMas = readlines(file)     # в strMas записывается всё содержимое файла
                             # config.txt
global norm_residual    # переменные, которые будут использоваться в файле
global limit            # godunovMatrix.jl, объявляются глобальными
global Godunov_matrix_dim
global mantissa
global mantissa_out
global pathResult
# считывается значение переменной limit
for i in 1:1:length(strMas)  # перебираются строки массива strMas
  if search(strMas[i], "limit")!=0:-1 # Если подстрока “limit” найдена
    limit = split(strMas[i],"= ")[2] # строка i разделяется на массив
                         # строк через символ “= ” и переменной limit
                         # присваивается только 2 элемент этого массива.
    limit = parse(Int64,limit) # Строка limit конвертируется в Int64.
    break            # цикл for прерывается при нахождении ключевого
                     # слова.
  end
end

for i in 1:1:length(strMas)    # Считывается значение Godunov_matrix_dim
                               # Аналогично считыванию значения limit.
  if search(strMas[i], "Godunov_matrix_dim")!=0:-1
    Godunov_matrix_dim = split(strMas[i],"= ")[2]
    Godunov_matrix_dim = parse(Int64,Godunov_matrix_dim)
    break
  end
end

for i in 1:1:length(strMas)     # максимально допустимое отклонение нормы
                                # градиента.
                                # Аналогично считыванию значения limit.
  if search(strMas[i], "norm_residual")!=0:-1
    norm_residual = split(strMas[i],"= ")[2]
    norm_residual = parse(BigFloat,split(norm_residual,"\n")[1])
    break
  end
end

for i in 1:1:length(strMas)      # Мантисса
                                 # Аналогично считыванию значения limit.
```

Рисунок Б.1 – Код модуля readFile.jl программы godunovMatrix, лист 1

```
43     if search(strMas[i], "mantissa")!=0:-1
44     mantissa = split(strMas[i],"= ")[2]
45     mantissa = parse(Int64,mantissa)
46     break
47   end
48 end
49
50 for i in 1:1:length(strMas)      # Мантисса для записи в файлы
51                                  # Аналогично считыванию значения limit.
52   if search(strMas[i], "mantissa_out")!=0:-1
53     mantissa_out = split(strMas[i],"= ")[2]
54     mantissa_out = parse(Int64,mantissa_out)
55     break
56   end
57 end
58 for i in 1:1:length(strMas)
59   if search(strMas[i], "pathResult")!=0:-1
60     pathResult = split(strMas[i],"= ")[2]
61     pathResult = split(pathResult, "\n")[1]
62     break
63   end
64 end
65 close(file)  #закрытие файла config.txt
```

Рисунок Б.1, лист 2

ПРИЛОЖЕНИЕ В

Реализация различных методов чтения и записи в Julia

```
M = 30000
n = 6

mas = rand(m, n)
println("запись массива $m x $n write")
tic()

file = open("write_method_matrix","w")
write(file, mas)
close(file)
write_time = toc()
println("запись завершена $write_time секунд")

file = open("log_write_w","w")
write(file, "размерность массива $m x $n, время записи: $write_time")
close(file)
```

Рисунок В.1 – Запись массива 30000x6 в файл функцией write

```
mas = zeros(Float64,30000,6)
println("команда read! считывает массив записанный командой write")

tic()
file = open("write_method_matrix","r")
read!(file, mas)
close(file)
read_time = toc()

println("чтение завершено $read_time секунд")

file = open("log_write_r","w")
write(file, "$read_time")
close(file)
```

Рисунок В.2 – Чтение массива функцией read!

```
m = 30000
n = 6
mas = rand(m, n)
println("запись массива $m x $n writedlm")
tic()
file = open("writedlm_method_matrix","w")
writedlm(file, mas)
close(file)
write_time = toc()
println("запись завершена $write_time секунд")

file = open("log_writedlm_w","w")
write(file, "размерность массива $m x $n, время записи $write_time")
close(file)
```

Рисунок В.3 – Запись массива 30000x6 в файл функцией writedlm

```
println("команда readdlm считывает массив записанный командой writedlm")
tic()
mas = readdlm("writedlm_method_matrix")
read_time = toc()
println("чтение завершено $read_time секунд")

file = open("log_writedlm_r","w")
write(file, "$read_time")
close(file)
```

Рисунок В.4 – Чтение массива функцией readdlm

```
using JLD

m = 30000
n = 6
mas = rand(m, n)
println("запись массива $m x $n JLD")
tic()

save("JLD_method_matrix.jld", "mas", mas)

write_time = toc()
println("запись завершена $write_time секунд")

file = open("log_JLD_w","w")
write(file, "матрица размерности $m x $n, время записи: $write_time")
close(file)
```

Рисунок В.5 – Запись массива 30000x6 в файл при помощи JLD

```
using JLD

println("команда load считывает массив записанный командой save")
tic()
mas = load("JLD_method_matrix.jld","mas")
read_time = toc()
println("чтение завершено $read_time секунд")

file = open("log_JLD_r","w")
write(file, "время чтения: $read_time")
close(file)
```

Рисунок В.6 – Чтение массива при помощи JLD

```
m = 30000
n = 6
mas = rand(m, n)
println("запись массива $m x $n serialize")
tic()
file = open("write_method_matrix","w")
serialize(file, mas)
close(file)
write_time = toc()
println("запись завершена $write_time секунд")
file = open("log_serialize_w","w")
write(file, "матрица размерности $m x $n, время записи: $write_time")
close(file)
```

Рисунок В.7 – Запись массива 30000x6 в файл функцией serialize

```
println("команда deserialize считывает массив записанный командой
serialize")
tic()
file = open("write_method_matrix","r")
mas = deserialize(file)
read_time = toc()
println("чтение завершено $read_time секунд")
file = open("log_deserialize_r","w")
write(file, "время чтения: $read_time")
close(file)
```

Рисунок В.8 – Чтение массива функцией deserialize

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Кафедра прикладной математики и компьютерной безопасности

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. А. Кытманов
подпись
«16» 06 2017 г.

БАКАЛАВРСКАЯ РАБОТА

01.03.04 – Прикладная математика

Повышение точности определения траектории спутника системы GPS
по его навигационным сигналам

Руководитель _____ профессор, д.ф.-м.н. С. П. Царев
подпись, дата

Выпускник _____ 13.06.2017 студент группы КИ13-19б А. С. Тетерятников
подпись, дата

Красноярск 2017