Университет ИТМО

Методическое пособие по курсу «Вычислительная математика»

Перл О.В.
Весенний семестр, 2023г

Оглавление

Общие положения по организации курса

Список ресурсов по курсу «Вычислительная математика»

1. Группа VK
2. Moodle
3. Журнал в Google
4. Комната Microsoft Teams для занятий
5. Ссылка на форму для записи в очередь на сдачу работ на практике
6. Посмотреть очередь
7. В случае вопросов по лекциям спросить можно тут, и мы разберем вопрос на следующем занятии

Список лекций и организация сдачи лабораторных работ

1. Лекция 1.
    1. Введение.
    2. Clean Code.
2. Лекция 2.
    1. Погрешность.
    2. Матричная алгебра.
3. Лекция 3. Системы линейных алгебраических уравнений.
4. Лекция 4. Системы нелинейных уравнений.
5. Лекция 5. Численное интегрирование.
6. Рубежная работа 1.
7. Лекция 7. Интерполирование и аппроксимирование.
8. Лекция 8. Численное дифференцирование и задача Коши.
9. Лекция 9. Другие более сложные задачи.
10. Рубежная работа 2.

Текущее расписание занятий по дисциплине доступно в журнале курса на вкладке Календарь. Серым в графике выделены выходные дни. В крайнем правом столбце цветом выделены недели, в конце которых крайний срок сдачи лабораторных работ. Цвет недели соответствует цвету лабораторной работы из вкладки Курс для соответствующего крайнего срока.

Все студенты всех групп могут ходить не только на свои занятия, но и на занятия других групп, однако преимущество в очереди отдаётся тем студентам, занятие у которых стоит по расписанию. Для удобства контроля очереди предлагается заполнить короткую форму, которая формирует список, доступный публично.

На сдачу лабораторной работы отводится не более 7 минут, поэтому будьте готовы демонстрировать экран с выполнением лабораторной работой и отвечать на теоретические вопросы, когда наступит Ваша очередь. Крайне настоятельно рекомендуется **внимательно следить за тем, как сдают работы другие** студенты, чтобы заранее проверить себя и свою программу на аналогичные ошибки.

За лабораторную работу Вы получаете 2 оценки: за выполнение и за владение теоретическим материалом. Для этого в системе Moodle вы загружаете zip-архив с исходным кодом Вашей лабораторной работы и PDF-отчёт о результатах выполненной работы. Обе оценки ставятся в процентах, в зависимости от качества выполненной работы и качества проработки теоретического материала. Проценты умножаются на максимально возможный для этого периода балл. Обе оценки ставятся одновременно, частичная сдача работы (только выполнение или только теория) не допускается. **Оба файла** zip-архив с кодом и PDF-отчёт **должны быть загружены** в Moodle **до начала** сдачи лабораторной работы.

Всего по курсу предлагается выполнить 6 лабораторных работ, 5 из которых обязательные, а 6 – зачётная. Зачётную лабораторную работу можно не писать, если Вам хватает количества баллов, и она может быть поставлена автоматом как среднее из предыдущих процентов в случае, если все остальные работы сданы до 08.05.23. Список лабораторных работ:

1. Системы линейный алгебраических уравнений.
2. Системы нелинейных уравнений.
3. Численное интегрирование.
4. Интерполирование и аппроксимация.
5. Решение задачи Коши.
6. Зачётная лабораторная работа из списка других более сложных задач.

Лабораторные работы должны быть сданы своевременно, в противном случае, максимальное количество баллов за работу будет снижено. Максимальное количество баллов за 6 зачётную лабораторную работу никогда не снижается и всегда равно 20 (10 за выполнение и 10 за теорию). Максимальный балл за лабораторную работу рассчитывается как снижение на 10% каждую последующую неделю просрочки.

Таблица максимальных баллов

	Лабораторная работа №					
	1	2	3	4	5	6
06.02.2023	4,00	8,00	12,00	16,00	20,00	20,00
27.02.2023	4,00	8,00	12,00	16,00	20,00	20,00
13.03.2023	3,60	8,00	12,00	16,00	20,00	20,00
27.03.2023	3,24	7,20	12,00	16,00	20,00	20,00

10.04.2023	2,92	6,48	10,80	16,00	20,00	20,00
24.04.2023	2,62	5,83	9,72	14,40	20,00	20,00
08.05.2023	2,36	5,25	8,75	12,96	18,00	20,00
22.05.2023	2,13	4,72	7,87	11,66	16,20	20,00
05.06.2023	1,91	4,25	7,09	10,50	14,58	20

Аналогичная таблица максимальных баллов по выполнению и теории:

	1		2		3		4		5		6	
	code	theory	code	theory	code	theory	code	theory	code	theory	code	theory
06.02.2023	2	2	4	4	6	6	8	8	10	10	10	10
27.02.2023	2	2	4	4	6	6	8	8	10	10	10	10
13.03.2023	1,8	1,8	4	4	6	6	8	8	10	10	10	10
27.03.2023	1,62	1,62	3,6	3,6	6	6	8	8	10	10	10	10
10.04.2023	1,46	1,46	3,24	3,24	5,4	5,4	8	8	10	10	10	10
24.04.2023	1,32	1,32	2,92	2,92	4,86	4,86	7,2	7,2	10	10	10	10
08.05.2023	1,19	1,19	2,63	2,63	4,38	4,38	6,48	6,48	9	9	10	10
22.05.2023	1,07	1,07	2,37	2,37	3,94	3,94	5,84	5,84	8,1	8,1	10	10
05.06.2023	0,96	0,96	2,13	2,13	3,55	3,55	5,25	5,25	7,29	7,29	10	10

Вариант на каждую следующую работу можно получить после демонстрации выполнения предыдущей работы. Даже если Вы не закончили текущую работу или не до конца изучили теоретический материал**, Вы можете продемонстрировать то, что есть на данный момент, чтобы получить вариант и иметь возможность продолжать работать на следующей лабораторной работой.** Демонстрация работы для получения варианта следующей работы не является сдачей части «выполнение» лабораторной работы! Обе части «выполнение» и «теория» всегда сдаются вместе. Во время демонстрации для получения варианта, вы лишь показываете свои наработки, которые даже не тестируются в полной мере, как при защите лабораторной работы.

Распределение баллов по курсу

Элемент курса	Максимальный балл
Лабораторные работы: Лабораторная работа 1 Лабораторная работа 2 Лабораторная работа 3 Лабораторная работа 4 Лабораторная работа 5	60 4 8 12 16 20
Зачётная лабораторная работа 6	20
Рубежная работа 1	10
Рубежная работа 2	10
Итого	100

Литература и прочие материалы по курсу

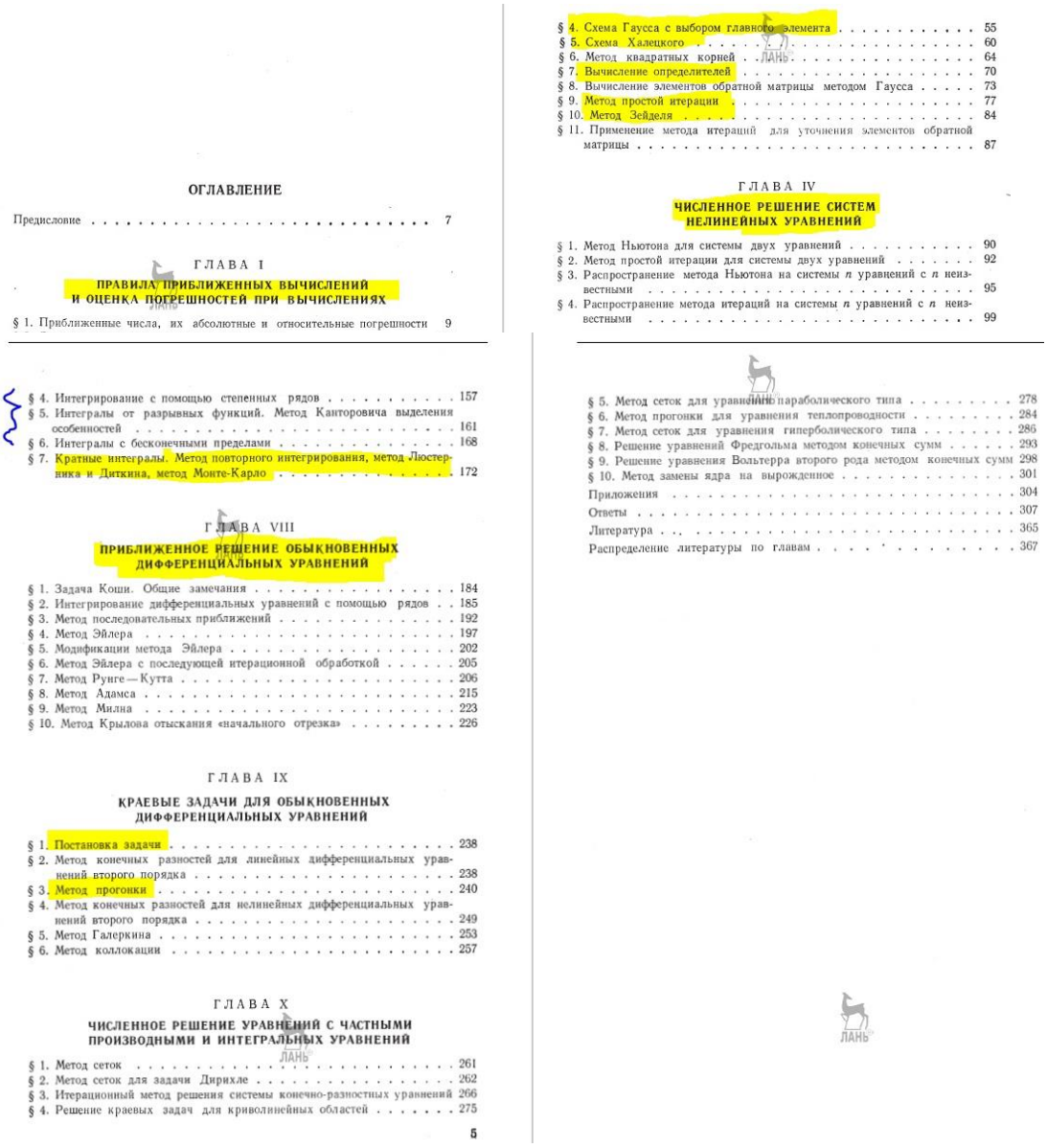
Ссылка на статью с рекомендациями литературы по курсу:
https://vk.com/@computationalmathematics-books

Для начала заметим, на всякий случай, что вычислительная математика и численные методы — это одна и та же область науки, поэтому подбор литературы не должен опираться только на одно из названий.

Список основной литературы, полезной для изучения курса и подготовки лабораторных работ:

Обратите внимание, что на многие книги приведена ссылка с доступом на https://e.lanbook.com/ , а значит, что они бесплатно доступны вам после регистрации в библиотеке ИТМО.

Для примера оглавление второй книги из списка. Жёлтым отмечены разделы, которые непосредственно относятся к выполнению лабораторных работ, синим — полезные для прочтения главы, желательные для более глубокого понимания. Последняя глава не рассматривается в данном курсе, но для заинтересованных в решении дифференциальных уравнений в частных производных, могу порекомендовать отличный курс видео с подробными разборами от преподавателей MIT.



ОГЛАВЛЕНИЕ

1. Демидович Б. П., Марон И. А. - Основы вычислительной математики — отличная книга с подробными разъяснениями и подробными выкладками.
2. Копченова Н.В., Марон И.А. - Вычислительная математика в примерах и задачах — замечательная книга, с объяснениями, разобранными примерами и задачами (с ответами в конце книги). Прекрасно дополняет первую книгу.
3. Волков Е. А. - Численные методы — Эта книга написана более простым языком, по сравнению с предыдущими. Автор даёт подробные указания о различных методах, некоторые из них могут быть пропущены в предыдущих книгах.
4. Демидович Б.П., Марон И.А., Шувалова Э.З. - Численные методы анализа. Приближение функций, дифференциальные и интегральные уравнения — Будет полезной находкой для тех, кто стремиться лучше разобраться в решении ОДУ (лабораторная работа №5), интегрировании (лабораторная работа №3) и приближении функций (лабораторная работа №4).
5. Турчак Л.И., Плотников П.В. - Основы численных методов — Книга даёт простые и дельные объяснения большинства методов, используемых в качестве вариантов для лабораторных работ.
6. Форсайт Дж., Малькольм М., Моулер К. - Машинные методы математических вычислений — Книга предлагает альтернативный взгляд на численные методы.
7. Фаддеев М.А., Марков - Основные методы вычислительной математики — Скорее дополнительная книга, которая помогает посмотреть на изучаемую тему с разных сторон.

Книги по математике, чтобы лучше понимать смысл численных методов и терминов, упоминаемых в других книгах

1. Справочник по математике для научных работников и инженеров. Определения, теоремы, формулы : пер. с англ. / Г. А. Корн, Т. М. Корн .— Изд. 6-е, стер. — СПб. [и др.] : Лань, 2003 .— 831 с. : ил. — (Учебники для вузов. Специальная литература) .— Библиогр.: с. 796-800 .— Предм. указ.: с. 804-831. — Отличная книга, которую я правда использую как настольный справочник и в университете и дома
2. Беклемишев, Д.В. Курс аналитической геометрии и линейной алгебры. — Есть задачки и можно свериться с тем, что выдаёт Ваша программа, например, в лабораторной работе №1.
3. Глейк Джеймс. Хаос. Создание новой науки — ***ОЧЕНЬ полезная книга***. Она не даст Вам ответов на то, как реализовать метод в лабораторной работе, но вполне **способна изменить Вашу жизнь и взгляды на мир вокруг.**
4. Любая другая книга по математике, которая Вам нравится. :)

Книги про программирование и вычислительную математику

1. Devpractice Team - Линейная алгебра на Python. – Довольно базовая книга для начинающих в Python и кому нужно начать работать с линейной алгеброй.
2. Svein Linge, Hans Petter Langtangen - Programming for Computations: Python. – Отличная книга, которая может быть использована одновременно и как пособие для начинающих в Python и с наглядными примерами и объяснениями того, что и как реализуется в вычислительной математике.

3. Paul Orland — Math for Programmers. Книга немного с другим фокусом, но она даёт хорошее описание того, как вообще можно работать с математикой в программировании, какие возникают задачи и как их принято решать. Примеры в основном на Python.
4. Ещё огромное количество книг про программирование и математику.

Другие ресурсы для освоения навыков программирования на основе задач и, в том числе, математики

Часто студентам для реализации того или иного алгоритма не хватает именно навыка программирования и решения разного рода алгоритмических задач, включая математические задачи. Ниже приведён список ресурсов, на которых можно этот навык отточить и закрепить.

Иные из них часто используются и для проведения собеседований, так что потренироваться на них заранее будет очень даже полезно.

1. Project Euler — сайт представляет собой длинную череду задач, решение которых помогает освоиться как в математике, так и в программировании. Есть различные сайты с переводами задач на русский язык, например этот.
2. Hackerrank — на сайте представлены различные курсы и задачи в них, которые помогут Вам прокачаться в различных областях, будь то DevOps, SQL или Java, Python, что угодно ещё. Решать задачи и запускать код можно на огромном количестве языков и прямо на сайте запускать. Также содержит задачки на математику.
3. binarysearch — сайт, который предоставляет отличную возможность решать задачи как самостоятельно, так и с друзьями. Есть дружелюбное русскоязычное комьюнити, а раз в неделю там проводят конкурсы.
4. codewars — сайт, похожий на Hackerrank и binarysearch, предлагающий решение задачек по различным языкам и технологиям.
5. Kaggle — отличный сайт для тех, кто хочет освоиться и попрактиковаться в Data Science! Куча практики, полезное общение и знакомство с другими участниками, которые пытаются решить те же задачи, а ещё возможность выиграть реальные и весьма немаленькие деньги.
6. codingame — вариант для тех студентов, которые пошли учиться на программистов и сутками не вылезают из компьютерных игр. На сайте программирование разных задачек является частью игры, при чём жанр игры можно выбрать самостоятельно и это вовсе не обязательно унылая графика и скучный сюжет. А ещё есть соревнования, где можно состязаться с другими участниками.
7. codebattle.hexlet.io — это, мягко выражаясь, настоящее «рубилово», PvP от мира программирования. Можно соревноваться с ботом, друзьями или со случайными игроками.

Прочие книги по курсу на английском языке

1. Numerical Recipes: The Art of Scientific Computing by William H. Press
2. Numerical Recipes in C: The Art of Scientific Computing by William H. Press
3. Numerical Recipes: Example Book C by William T. Vetterling
4. Numerical Recipes in FORTRAN 77: The Art of Scientific Computing by William H. Press

5. Numerical Recipes in FORTRAN 90: The Art of Parallel Scientific Computing (FORTRAN Numerical Recipes, Volume 2) by William H. Press
6. Numerical Recipes: Example Book FORTRAN by William T. Vetterling
7. Numerical Methods That Work by Forman S. Acton
8. Numerical Methods by Germund Dahlquist
9. Numerical Methods and Software by David Kahaner
10. An Introduction to Numerical Methods in C++ by Brian Hilton Flowers
11. Introduction to Precise Numerical Methods [With CD] by Oliver Aberth
12. Precise Numerical Methods Using C++ [With CDROM] by Oliver Aberth
13. Introduction to Numerical Programming: A Practical Guide for Scientists and Engineers Using Python and C/C++ by Titus Adrian Beu
14. Numerical Linear Algebra by Lloyd N. Trefethen
15. Applied Numerical Linear Algebra by James W. Demmel
16. Numerical Methods for Scientists and Engineers by Richard Hamming
17. Numerical Methods and Optimization: A Consumer Guide by Eric Walter
18. Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics by Justin Solomon
19. Numerical Optimization by Jorge Nocedal
20. Elementary Numerical Mathematics for Programmers and Engineers (Compact Textbooks in Mathematics) by Gisbert Stoyan
21. Applied Numerical Methods in C by Shoichiro Nakamura
22. Numerical Methods in Economics by Kenneth L. Judd
23. Compact Numerical Methods for Computers by John C. Nash
24. Computer Solution of Linear Algebraic Systems by George E. Forsythe
25. Computer Methods for Mathematical Computations by George E. Forsythe
26. A First Course on Numerical Methods by Uri M. Ascher
27. Numerical Analysis with Algorithms and Programming by Santanu Saha Ray
28. Numerical Analysis for Statisticians by Kenneth Lange
29. Numerical Linear Algebra for Applications in Statistics by James E. Gentle
30. An Introduction to Numerical Methods and Analysis by James F. Epperson
31. Numerical Methods and Optimization: An Introduction by Sergiy Butenko
32. Computing for Numerical Methods Using Visual C++ by Shaharuddin Salleh
33. Numerical Linear Algebra: An Introduction by Holger Wendland
34. Introduction to Numerical Linear Algebra and Optimisation by Philippe G. Ciarlet

35. C++ and Object-Oriented Numeric Computing for Scientists and Engineers by Daoqi Yang
36. Numerical Methods and Optimization in Finance by Manfred Gilli
37. Computational Methods of Linear Algebra (Undergraduate Mathematics Books) by D.K. Faddeev
38. Computational Methods of Linear Algebra (3rd Edition) by Granville Sewell
39. Guide To Scientific Computing by Peter R. Turner
40. Guide to Scientific Computing in C++ by Joe Pitt-Francis
41. Practical Numerical Methods: Algorithms And Programs by Michael Kohn
42. Practical Numerical Algorithms for Chaotic Systems by Thomas S. Parker
43. Numerical Methods For Unconstrained Optimization And Nonlinear Equations by J.E. Dennis
44. Accuracy and Stability of Numerical Algorithms by Nicholas J. Higham
45. A First Course in Numerical Analysis by Anthony Ralston
46. Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and Their Implementation by George E.M. Karniadakis
47. Functions of Matrices by Nicholas J. Higham
48. Scientific Computing by Michael T. Heath
49. Quadpack: A Subroutine Package For Automatic Integration by R. Piessens
50. Elementary Numerical Analysis: An Algorithmic Approach by Samuel Daniel Conte
51. A Survey of Numerical Mathematics: In Two Volumes - Volume I by David M. Young
52. A Survey of Numerical Mathematics: In Two Volumes - Volume II by David M. Young
53. Numerical Computation 1: Methods, Software, and Analysis by Christoph Uberhuber
54. Numerical Computation 2: Methods, Software, and Analysis by Christoph Uberhuber
55. Computational Integration by Arnold R. Krommer
56. Elementary Numerical Methods and C++ by Melvin R. Corley
57. Gaussian Quadrature Formulas by A.H. Stroud
58. Afternotes on Numerical Analysis by G.W. Stewart
59. Afternotes Goes to Graduate School: Lectures on Advanced Numerical Analysis by G.W. Stewart
60. Numerical Algorithms with C by Frank Uhlig
61. Numerical Algorithms With Fortran by Gisela Engeln-Mullges
62. Numerical Methods of Mathematics Implemented in Fortran by Sujit Kumar Bose
63. Numerical Optimization Of Computer Models by Hans-Paul Schwefel
64. Numerical Linear Algebra For High Performance Computers by Jack Dongarra
65. Gnu Scientific Library Reference Manual - Third Edition by Brian Gough
66. A Numerical Library in C for Scientists and Engineers by Hang Tong Lau

67. A Numerical Library In Java For Scientists And Engineers by Hang Tong Lau

Методические указания по изучению математики и вычислительной математики в частности

Почему математика важна?

- Математика — это язык формального, унифицированного описания мира.
- Математика обладает достаточной мерой абстракции, чтобы развивать фантазию, воображение и образ мышления.

Качества и навыки программиста, развиваемые математикой

1. Умение мыслить абстрактно, выделять общее и частное;
2. Умение анализировать структуры и методы;
3. Умение формально описывать сущности и пути достижения желаемого;
4. Умение видеть и создавать красоту;
5. Умение упрощать методы и находить альтернативные пути решения;
6. Навыки стратегического мышления.

Первое — это способность мыслить абстрактно, различать общее и частное. Например, когда вы создаете несколько классов и вам следует создать иерархию, вы можете создать несколько абстрактных классов и поместить в них общую часть, а конкретные части поместить в дочерние классы. Другой пример - когда вы работаете над каким-то требованием к продукту и вам следует указать общие и конкретные варианты использования.

Второе — это способность анализировать структуры и методы. Не уверена, что я должна это разъяснять, надеюсь, вы понимаете: когда мы обучены работать со сложными математическими структурами, способны доказывать некоторые теоремы и т. д. Это помогает вам ежедневно делать то же самое в области программирования: создавать структуры и методы, которые содержат правильные цели.

Третье – умение формально описать суть и способы достижения желаемого. Например, когда вы работаете над каким-либо требованием к продукту, вы должны разработать и понять, какие дополнительные функции вы должны охватить. Опишите поведение пользователей вашего продукта и их потребности. То же самое, что вы делаете, когда доказываете некоторые теоремы по математике.

Четвертое — это умение видеть и создавать Красоту. Это очень важный навык для программистов. Я была очень рада, что на моей первой работе в моей команде было много высококлассных специалистов со значительным опытом разработки программного обеспечения. Их код в любом случае был прекрасен, в нем было очень легко ориентироваться и понимать его. В то же время большое количество молодых программистов, в основном студентов, любят писать очень сложный код, чтобы показать, что другие, кто может его просмотреть, дураки, “если они не могут понять мой гениальный код”. Самое важное правило разработки программного обеспечения заключается в том, что ваш код должен быть легок для понимания другими программистами. Тот же навык вы тренируете в математике: вы должны использовать самый простой из применимых методов, в другом случае вам и другим будет трудно проверить ваше решение, и это увеличит вероятность ошибок.

Пятое – способность упрощать методы и находить альтернативные решения. Я хорошо знаю, что легко сделать что-то более сложное, но чрезвычайно трудно сделать что-то более простое. Например, когда вы решаете какое-то уравнение, вы упрощаете обе его стороны (левую и правую), а затем применяете некоторые операции. Существует множество способов решения одного уравнения, но если вы хорошо подготовлены и хорошо разбираетесь в математике, то вы выберете самый простой способ решения этого конкретного уравнения. И если у вас возникнут какие-то трудности при решении, вы можете просто сменить выбранный метод на другой. И те же способности, которые вам нужны для написания кода, или для разработки каких-то функций, или для чего-то еще. В каждый конкретный момент, как и в нашей повседневной жизни, математика помогает вам упростить сложную задачу, которую вам нужно решить, и найти альтернативные способы решения этой проблемы.

И шестое. Математика улучшает ваши навыки стратегического мышления. Как вы, возможно, знаете, в армии много математики используется, как бы это сказать, для реальной стратегии. Если кто-то хочет иметь армию и эффективно управлять ею, вы должны знать математику и понимать, как она работает. Та же ситуация с экономикой или, в конце концов, с компьютерными играми. Но для нас, как для инженеров-программистов, также очень важно обладать навыками стратегического мышления. Вы должны уметь понимать цену своих решений. Например, в вашей программе есть какая-то ошибка. Что вы выберете: решите это быстро, но тогда у вас возникнут некоторые проблемы с архитектурой в будущем; или исправьте это медленно и очень тщательно, но тогда вам придется отложить дату выпуска и, возможно, дату, когда вы начнете получать деньги за свой продукт; или вы пропустите эту ошибку сейчас, потому что она чрезвычайно мала и может быть легко исправлена в будущем, но не сейчас, когда у вас будет больше информации, например, но этот путь приведет вас в технический долг, список проблем, которые вам придётся решать в будущем вместо разработки новых функций. Что вы выберете? И это становится чрезвычайно важным, если вы создаете свой собственный стартап, свой продукт, чтобы зарабатывать деньги. Низкое качество, поздняя реализация, технический долг и очень длинный список одних и тех же вещей. Вы должны выбрать, и вы решаете. В этом случае вам необходимо обладать навыками стратегического мышления, потому что это поможет вам сделать оценку, прогноз, чтобы предсказать, что поможет дальше и что вы можете сделать сейчас, чтобы улучшить это. Есть много фильмов, например, о том, как это работает и почему математика в этом так важна. Один из них - “Игра на понижение" 2015 года.

Как изучать математику?

Здесь я привожу кратко некоторые заметки о том, как программисту стоит изучать математику. Если у вас будут свои замечания по этому поводу я с удовольствием их обсужу. В дополнение рекомендую книгу «Думай как математик», Барбары Оакли.

- Выделять структуры и модули в разбираемом математическом тексте.
- Изучать связные и используемые определения и термины. Рекурсивно искать термины и понятия, встретившиеся ранее.
- Референтный подход: по заданной теме изучать не один, а несколько источников. Дополнительные источники могут быть найдены отдельно или из списка литературы первого.
- Де-абстрагирование: мысленное применение абстрактных математических структур и понятий к решаемой или гипотетически решаемой задаче.

Методические указания по выполнению лабораторных работ

Общие требования к выполнению лабораторных работ

- Язык программирования: Java, C++ или Python, иное обговаривается лично с преподавателем. Язык программирования от работы к работе может отличаться. Это может быть важно, т. к. начиная с 4 работы необходимо будет строить графики.
- Конечному пользователю должна быть представлена интуитивно понятная программа, без значительных дефектов (плавающие поля, бесконечно множащиеся ячейки, обработанный ввод некорректных данных и т. д.).
- В лабораторных, предполагающих отображение графиков - график должен полностью отображать весь заданный интервал (с запасом).

В программе численный метод должен быть в виде отдельной подпрограммы (метод, функция, процедура) или класса, в который исходные данные передаются в качестве параметров, выходные - тоже (либо возвращаемое значение). Ввода-вывода в классе (подпрограмме), где реализован сам численный метод, быть не должно (учимся писать код так, чтобы можно было повторно использовать). Например, код численного метода из лабораторной работы 1 вполне можно будет использовать в лабораторной работе 4 и 6.

Например, если Вам необходимо реализовать метод для интегрирования чисел, то в качестве аргументов ему будут переданы подынтегральная математическая функция (как lambda-функция, объект функции или указатель на функцию или хотя бы номер функции) и границы интегрирования. Численный метод не должен знать о существовании пользователя или быть зависимым от других внешних источников. Численный метод не должен запрашивать значения у пользователя или писать в консоль результат или сообщения об ошибках.

Аналогичные ограничения распространяются и на класс, если он реализует численный метод. О любых внештатных ситуациях можно сообщить пользователю иными средствами выбранного Вами языка программирования, например, механизмом исключений. Класс, работающий со структурами данных для численного метода, может содержать метод, подготавливающий вывод объекта, например метод to_string().

Состав отчёта по лабораторной работе

1. Описание метода, расчетные формулы; (Материал пишется самостоятельно по результатам изученного в ходе подготовки к лабораторной работе материала, а не копируется с различных источников, в том числе сайтов и книг)
2. Блок-схема численного метода; (см. требования к оформлению блок-схем)
3. Листинг реализованного численного метода программы; (весь листинг писать нет необходимости, только численный метод)
4. Примеры и результаты работы программы на разных данных; (3-4 примера)
5. Вывод.

Обратите внимание на порядок следования элементов отчёта и отсутствия раздела «Цель». Порядок элементов определён порядком выполнения самой работы: сначала изучается сам метод и его теоретическая основа, на основе чего строится блок-схема. Только после этого можно приступать к написанию кода на основе блок-схемы. После написания кода происходит самое важное: вы запускаете численный метод на разных данных и анализируете результат (это и есть цель лабораторной работы, а вовсе не написание кода как таковое). Как раз на основе этого вы пишете результат о том, каким является исследованный вами метод.

Раздел «Цель» отсутствует в отчёте, так как в большинстве случаев является бездумным копированием темы работы и не несет смысловой нагрузки ни для преподавателя, ни для студента.

Как составить блок-схему численного метода

Блок-схема должна строиться до написания кода, в тот момент, когда у вас ещё нет никаких переменных и программных функций. Поэтому блок-схема должна содержать в себе математические обозначения используемых понятий.

Например, если мы работаем с матрицей, как в лабораторной работе 1, нам необходимо использовать математическую запись элементов матрицы x_{ij}, а не x[i][j].

Сама блок-схема должна строиться на основе принципов построения блок-схем программ. Об основах можно прочитать подробнее: https://ru.wikipedia.org/wiki/Блок-схема.

Для построения блок-схем допустимо использовать любые подходящие для этого программы, например:

1. Draw.io
2. Visio
3. Dia
4. Встроенные средства Word или LibreOffice Draw

Как написать вывод по лабораторной работе

Вывод должен содержать анализ выполненной работы:

- результаты запуска реализованного метода на **различных** данных; (не сам факт, что запускается, а что это значит и почему работает или не работает именно так)
- сравнение с другими методами;
- анализ применимости метода;
- анализ алгоритмической сложности метода;
- анализ численной ошибки самого численного метода и пр.

Вывод содержащий текст "Я реализовал ... я молодец" не засчитываются как вывод к лабораторной работе.

Лабораторная работа 1. Системы линейных алгебраических уравнений

Вам будет выдан один из следующих вариантов:

- Прямые методы:
  - Метод Гаусса
  - Метод Гаусса с выбором главного элемента
- Итерационные методы:
  - Метод простых итераций
  - Метод Гаусса-Зейделя

В лабораторной работе Вам предлагается реализовать один из методов, который бы позволял находить столбец неизвестных для системы линейных алгебраических уравнений. Размер системы предполагается ограниченным 20, это означает, что Вашу программу после реализации необходимо будет проверить на матрице размером 20 x 20 + 20 элементов (количество неизвестных в матрице A и столбце B). Исходя из этого условия матрицы такого размера должны иметь возможность каким-то образом попадать в Ваш численный метод. Например, ввод данных в ручном формате может быть затруднительным для 420 элементов системы с 20 неизвестными. Традиционно предлагается сделать следующие методы ввода данных в программу:

- Пользовательский ввод;
- Ввод данных из файла;
- Генерация случайных матриц.

Пользовательский ввод данных позволяет протестировать программу относительно просто и не требует дополнительных усилий со стороны разработчика, однако, по этой же причине вполне может быть заменён исключительно вводом из файла. С другой стороны, если такой способ ввода уже существует, то почему бы не сделать его приятным для пользователя?

Например, в случае, если пользователь вводит некорректные данные (буквы вместо чисел, две запятые или точку вместо запятой в вещественных чисел, пробел между числом и знаком минуса – всё, что не может быть распознано как число; не правильное количество данных; матрицу, определитель которой будет свидетельствовать об отсутствии решений; отрицательное значение размера матрицы и пр.) пользовательский ввод необходимо прервать с просьбой повторного ввода как можно раньше, а не продолжать запрашивать данные, чтобы сообщить об ошибке в самом конце. Особо необходимо отметить, что все подобные сценарии ошибок пользовательского ввода должны быть обработаны, при чём не только для ввода данных через терминал, но и для, например, ввода данных из файла.

Ввод данных из файла представляется хорошим способом проверки корректности работы численного метода, особенно когда речь идёт о матрицах большого размера. При вводе данных из терминала вполне возможно ошибиться, а при генерации случайной матрицы результат трудно проверяем. При работе с файлами данных необходимо продумать формат данных, с которыми будет проводиться работа: какие данные содержатся в самом файле, а какие вводятся отдельно пользователем. Например, для итерационных методов файл с данными будет содержать размер матрицы и саму матрицу, а точность приближения будет задавать пользователь. Так же необходимо продумать вопрос локализации: какой знак для вещественных чисел Вы будете использовать: точку или запятую. Для самопроверки численного метода в системе Moodle и в группе VK загружены несколько файлов с решениями. При защите лабораторной работы преподаватель может попросить Вас продемонстрировать работу Вашей программы на каком-то ином файле для того, чтобы проверить, действительно-ли Ваш реализованный численный метод считает результат достаточно точно.

Генерация случайных матриц может быть реализована двумя основными способами:

- Простая генерация матриц A, удовлетворяющей условиям применимости метода, и столбца B;
- Генерация столбца неизвестных и генерация матрицы A, удовлетворяющей условиям применимости метода, затем расчёт правой части уравнений и «забывание» столбца неизвестных. В таком случае можно сравнить рассчитанные и «загаданные» значения неизвестных.

Обратите внимание, что в обоих случаях имеет смысл генерация только матриц, имеющих решение реализуемым Вами численном методе. Чем больше будет размер матрицы, тем меньше будет вероятность сгенерировать матрицу, которая будет подходить под условия применимости метода! Особенно это актуально для итерационных методов. Обратите внимание, что условия сходимости итерационных процессов для методов простой итерации и для метода Гаусса-Зейделя – отличаются.

В отчёт по лабораторной работе в раздел с примерами нужно будет вставить в том числе пример с матрицей некоторого размера большего, чем обычно считают, например системы с 5-6 неизвестными. Вывод матриц систем с 20 неизвестными можно продемонстрировать на защите работы, однако в отчёте они как правило помещаются плохо.

При анализе реализованного численного метода Вам необходимо запускать его на различных наборах данных, а также попытаться понять работает ли программа на этом наборе данных правильно, если не работает, то должна ли работать и если нет, то почему. Так же предлагается проанализировать скорость работы численного метода. Сделать это можно путём сравнения аналитического значения алгоритмической сложности метода (по составленной ранее блок-схеме)

и путём реальных замеров скорости работы численного метода различными инструментами выбранного вами языка программирования. Например, в языке Python это можно сделать до и после вызова численного метода использовать команду `datetime.datetime.now()`, а затем вычесть полученные значения. Аналогично в Java можно использовать `System.currentTimeMillis()`.

Можно также использовать различные средства для анализа потребляемой памяти. Например, для Java можно использовать VisualVM для просмотра динамики работы JVM, либо запрашивать текущее состояние памяти через запуск в JVM команд: `Runtime.getRuntime().freeMemory()`. Аналогично можно поступать и для других языков программирования по Вашему желанию.

Для начала тестирования Вашей программы предлагаю использовать два простейших набора данных:

- Все элементы матрицы A равны 0, а значения столбца B – любые не одинаковые числа;
- Все элементы матрицы A равны 1, а значения столбца B – любые не одинаковые числа.
- Любая матрица, определитель которой для самопроверки Вы можете рассчитать при помощи метода Крамера.

Обратите также внимание, что для прямых и итерационных методов вывод в консоль должен отличаться.

Для прямых методов (метод Гаусса и метод Гаусса с выбором главных элементы) должно быть реализовано:

- Вычисление определителя;
- Вывод треугольной матрицы (включая преобразованный столбец B);
- Столбец неизвестных;
- Столбец невязок.

Для итерационных методов (метод простой итерации и метод Гаусса-Зейделя) должно быть реализовано:

- Точность задается пользователем;
- Проверка диагонального преобладания для соответствующего метода
  (В случае, если диагональное преобладание в изначальной матрице отсутствует - предлагается сделать перестановку строк/столбцов до тех пор, пока преобладание не будет достигнуто. В случае невозможности достижения диагонального преобладания - выводить сообщение.);
- Столбец неизвестных;
- Количество итераций, за которое было найдено решение;
- Столбец погрешностей.

Обращаю ваше внимание, что для итерационных методов в пользовательском выводе вашей программы должны быть только значения неизвестных, полученные на последней выполненной итерации, а не на всех выполненных итерациях. Аналогично со столбцом погрешностей – они должны соответствовать лишь последней итерации.

Для прямых методов часто представляет собой трудность расчёт невязки. Невязка представляет собой разницу между левой и правой частью уравнения. Таким образом для каждого уравнения в системе будет рассчитываться собственное значение невязки. Если правая часть уравнения у нас уже есть – это соответствующий элемент матрицы B, то левую часть нужно рассчитать, подставив на места неизвестных x рассчитанные при помощи численного метода значения, а затем произведя

операции умножения и сложения. Невязка – это то, на сколько правая часть уравнения не равна левой, в то время как в уравнении по определению обе части должны быть равны.

При расчёте определителя для прямых методов и в особенности для метода Гаусса с выбором главного элемента, необходимо помнить и учитывать свойства определителя.

В выводе помимо анализа самого метода необходимо сравнить его со вторым методом той же категории (например, для метод простой итерации сравнить с методом Гаусса-Зейделя), а также прямые методы в целом сравнить с итерационными.

Лабораторная работа 2. Системы нелинейных уравнений

В лабораторной работе 2 Вам необходимо будет реализовать три численных метода. Два из них будут решать нелинейные уравнения и ещё один будет решать систему нелинейных уравнений. Вариант будет выдан в виде 1аб, где цифра относится к методу решения систем нелинейных уравнений, а буквы – к решению нелинейных уравнений:

Решение нелинейных уравнений:

а метод деления пополам

б метод хорд

в метод касательных

г метод простой итерации

Решение систем нелинейных уравнений:

1 метод Ньютона

2 метод простой итерации

В связи с тем, что для решения нелинейных уравнений вам предлагается реализовать 2 метода, имеет смысл сравнить полученный результат между друг другом. Например, решением уравнения по методу деления пополам мы получили 5,1, а методом хорд получили 4,9, тогда разница между двумя методами будет равна 0,2. Соответственно, вычисления двух методов для одного уравнения должно производиться **одновременно**, без уточнения у пользователя каким именно численным методом он хотел бы рассчитать уравнение.

Нелинейные уравнения и системы нелинейных уравнений студенту предлагается выбрать самостоятельно. Тем не менее, необходимо учитывать области допустимых значений. Для примеров можно выбрать как нелинейные алгебраические, так и не алгебраические нелинейные уравнения.

Лабораторная работа 3. Численное интегрирование

Среди методов численного интегрирования вам будет предложено реализовать один из следующих методов:

1. Метод прямоугольников;
2. Метод трапеций;
3. Метод Симпсона (метод парабол).

При этом, если Вам было задано реализовать метод прямоугольников, то должно быть реализовано **все три** его варианта:

- Метод левых прямоугольников;
- Метод правых прямоугольников;

- Метод средних прямоугольников;

В этом случае пользователю нужно лишь единожды задать параметры, для которых он хочет получить результат, а применено должно быть все три варианта, без уточнения каким именно вариантом метода прямоугольника ему следует рассчитать интеграл.

При расчёте интеграла необходимо учитывать ОДЗ функции и необходимые и достаточные условия существования определённого интеграла. Если на интервале интегрирования существует устранимый разрыв первого рода, то его следует устранить одним из способов (с уточнением пользователю, каким именно способом Вы собираетесь его устранять) и выполнить расчёт интеграла. Например, может быть выполнен расчёт левой части интеграла от разрыва и правой в отдельности. Альтернативным методом устранения разрыва будет принять алгоритмическое среднее от значения от двух соседних точках функции $f(x-\varepsilon), f(x+\varepsilon)$, где ε – наперёд заданная малая постоянная, которая может быть зависима от текущего шага разбиения интеграла. В обоих случаях Вы должны продумать стратегию алгоритма устранения разрыва при попадании его на самую границу интервала.

Важным вопросом, который надо рассмотреть в данной лабораторной работе – погрешность квадратурных формул интегрирования, соответственно обозначаемых как r_i на каждом разбиении интеграла и R на глобальном интервале $[a; b]$, на котором производится интегрирование функции. На основании того факта, что интеграл по определению является пределом суммы бесконечно малых разбиений, без уточнений как именно это разбиение будет производиться, то в пределе абсолютно все методы численного интегрирования будут давать верный результат при правильном их применении. Поэтому для сравнения методов между собой следует анализировать максимальную величину шага разбиения, либо количество разбиений (особенно для методов с динамическим, а не постоянным шагом). В прямую это можно сделать при помощи сравнения полученного значения интеграла с его аналитическим значением, рассчитанным, например, по формуле Ньютона-Лейбница или табличных значений интеграла. Однако, для глобальных выводов необходимо также анализировать погрешности самих квадратурных формул R, которые также дают ответ на то, какие функции следует интегрировать при помощи тех или иных численных методов интегрирования. Ваши рассуждения, наблюдения и результаты анализа и должны составлять основу вывода в данной лабораторной работе.

Лабораторная работа 4. Интерполяция и аппроксимация

В лабораторной работе 4 будет предложено реализовать наиболее известный метод аппроксимации: аппроксимация методом наименьших квадратов и методы интерполяции:

- Метод интерполяции полиномом Ньютона;
- Метод интерполяции полиномом Лагранжа;
- Метод интерполяции кубическими сплайнами.

Обратите внимание, что начиная с данной лабораторной работы Вам необходимо строить графики функций. На графике необходимо отобразить исходные точки и точки, полученные при помощи разработанного численного метода, согласно варианту.

Во всех случаях вам необходимо сгенерировать набор входных точек, на которых вы будете тестировать свою программу. Набор точек должен представлять собой некоторое количество пар (x; y) полученных из какой-то существующей функции. Чтобы сгенерировать такой набор данных рекомендуется использовать подготовленную функцию, поведение которой вам уже известно. Предпочтительно иметь несколько наборов входных данных из нескольких функций. Также желательно добавить шум (отклонения значений y от математического). Имеет смысл также отобразить на графике функцию, на основе которой был получен набор данных.

Для варианта аппроксимации следует применить метод аппроксимации дважды: второй раз метод должен быть запущен на укороченном списке входных точек, при чём исключена должна быть точка, имеющая наибольший квадрат отклонения после первого запуска. Иначе говоря, алгоритм, следующий:

1. Задается произвольный (полученный ранее) набор значений пар $(x; y)$.

2. Задается аппроксимирующая функция.

3. Рассчитываются программой коэффициенты аппроксимирующей функции.

4. Производится поиск точки с наибольшим отклонением от значения, полученного при помощи аппроксимирующего многочлена.

5. Найденная точка исключается и производится пересчёт коэффициентов аппроксимирующего многочлена (см. п.3).

6. Строится график, содержащий в себе две функции (1 - до исключения, 2 - после исключения и пересчёта) и набор заданных изначально точек (пар значений (x, y)).

7. Помимо этого, отдельно на экран выводятся полученные значения аппроксимирующих коэффициентов.

В качестве аппроксимирующей функции следует выбрать несколько наиболее популярных аппроксимирующих функций: линейная, квадратичная, логарифмическая, тригонометрическая и т. п.

Для расчёта СЛАУ рекомендуется использовать методы решения СЛАУ из лабораторной работа 1.

При анализе методов в лабораторной работе следует обратить внимание на сценарии их применения, основываясь на особенностях методов.

Лабораторная работа 5. Численное дифференцирование и задача Коши

В лабораторной работе 5 вам будет предложено реализовать один из методов решения задачи Коши – решения дифференциального уравнения по заданным начальным условиям.

Одношаговые методы:

- Метод Эйлера
- Усовершенствованный метод Эйлера (не путать с улучшенным методом Эйлера)
- Метод Рунге-Кутты 4-го порядка

Многошаговые методы:

- Метод Милна
- Метод Адамса (не путать с методом Адамса-Башфорта)

В лабораторной работе Вам предлагается подготовить некоторое количество дифференциальных уравнений, которые сможет решать пользователь, вводя собственные начальные условия. Помимо этого, предполагается, что пользователь сможет выбирать как далеко нужно решить дифференциальное уравнение от начального условия. При анализе и демонстрации лабораторной работы рекомендуется исследовать не только различные дифференциальные уравнения, но и различные начальные условия для одного дифференциального уравнения – график решения при этом вероятно будет меняться.

На графике необходимо показать начальное условие и полученное решение дифференциального уравнения. Полезно также отобразить аналитическое решение дифференциального уравнения, чтобы иметь возможность сравнить и проанализировать накопление или не накопление ошибки. Аппроксимацию в обоих случаях (для аналитического и для полученного решения) необходимо выполнять по методу из лабораторной работы 4 (если в лабораторной работе 4 вашим вариантом была аппроксимация, то удваивать количество графиков – до и после удаления точек, как в лабораторной работе 4 - рисовать не нужно, метод должен быть применён только один раз).

В анализе лабораторной работы следует обратить внимание на понятие устойчивости решения в терминах решения дифференцирования – будет ли ошибка накапливаться и возрастать по мере удаления от начального условия, а также на сценариях применения, основанных на особенностях методов.

Зачётная лабораторная работа 6. Другие более сложные вопросы

В лабораторной работе 6 предполагается 5 основных вариантов:

1. Обращение симметричной положительно определенной матрицы методом квадратного корня (метод Холецкого).
2. Вычисление определенного интеграла по прямоугольной области. Под интегралом стоит функция $f(x, y)$, а интегрирование ведется по dx, dy. Интегрирование можно производить любым или некоторыми из методов: метод кратного интегрирования (любым из методов: прямоугольников, трапеций, методом Симпсона), методом Монте-Карло, методом Люстерника-Диткина (любым из методов, но преимущественно прямоугольников, потому что подразумевается интегрирование прямоугольной области).
3. Приближение сложной функции с помощью интерполяционного полинома Лагранжа. В этой постановке узлы интерполяции определяются как корни полинома Чебышева на заданном интервале приближения.
4. Решение краевой задачи для линейного ДУ второго порядка вида $y'' + p(x)y' + q(x)y = f(x)$ разностным методом (метод прогонки).
5. Решение краевой задачи для ОДУ второго порядка методом пристрелки.

Все основные требования, предъявляемые к предыдущим аналогичным работам, сохраняются. Для вариантов 3, 4 и 5 необходимо построить графики функций. При анализе методов необходимо особое внимание уделить достоинствам и недостаткам методов для решаемой задачи, границам применимости метода и сценариям применения метода.

Подготовка к рубежным работам

В курсе предполагается 2 рубежных работы, за каждую из которых можно получить до 10 баллов. Обе рубежные работы будут проводиться на платформе Moodle. Рубежная работа 1 будет проводиться по материалам лекций 1-5 включительно, а рубежная работа 2 по материалам лекций 7-9. Основное правило для подготовки к рубежным работам – это усвоение материала лекционных и рубежных работ. Помимо теоретических заданий, в работах будут присутствовать и задания на вычисления каких-либо значений. Аналогичные задачи будут решаться и разбираться на лекционных занятиях. Решение задач на лекциях не оценивается дополнительно, однако своевременная проработка задач поможет при подготовке курса. Поэтому посещение лекционных занятий, а также активная работа на занятиях и выполнение соответствующих домашних заданий существенно улучшит ваши результаты при написании рубежных работ.

Рубежные работы не содержат в себе заданий на написание кода по изученным численным методам, однако владеть ими всё-таки придётся, потому что некоторые задания требуют

вычисления по конкретным формулам численных методов и вычисление по другим формулам вероятнее всего приведёт вас к неправильному результату.

Полезным при подготовке к рубежным работам также будет повторение материалов лекций. Презентации по лекциям будут выложены в материалы группы Teams. Если Вам необходимы материалы на английском языке, Вы можете обратиться за ними к преподавателю.

Рубежная работа 1

Рубежная работа 1 состоится на лекционном занятии в среду 29 марта 2022 года. Рубежная работа будет включать в себя вопросы по следующим темам:

1. Введение;
2. Чистый код;
3. Погрешности и приближённые числа;
4. Алгебра матриц;
5. Системы линейных алгебраических уравнений и численные методы их решений;
6. Системы нелинейных уравнений и численные методы их решений;
7. Интегрирование и методы численного интегрирования.

Рубежная работа 2

Рубежная работа 2 состоится на лекционном занятии в среду 24 мая 2023 года. Рубежная работа будет включать в себя вопросы по следующим темам:

1. Аппроксимация и интерполяция;
2. Численное дифференцирование и решение задачи Коши;
3. Решение других более сложных задач.