|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Фундаментальные науки»

КАФЕДРА «Вычислительная математика и математическая физика» (ФН-11)

**ОТЧЕТ ПО ПРАКТИКЕ**

Студент *Хаписов Малик Хаписович*

Группа *ФН11-82Б*

Тип практики *Преддипломная*

Название предприятия *НОЦ «СИМПЛЕКС» МГТУ им. Н.Э. Баумана*

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.Х. Хаписов

*подпись, дата*

Руководитель практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Прозоровский

*подпись, дата*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва*

*2024 г.*

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_ВВЕДЕНИЕ) 3

[1 Стохастический метод Галёркина 22](#_1_Стохастический_метод)

[2 Сравнение стохастического метода Галёркина и аппроксимации полиномами Колмогорова-Габора](#_2_Сравнение_стохастического) 26

[3 Применение стохастического метода Галёркина к нахождению плотности эволюции системы 29](#_3_Применение_стохастического)

[3.1 Модель Шлёгля 35](#_3.1_Модель_Шлёгля)

[3.2 Уравнения Фоккера-Планка и нахождение плотности эволюции системы 37](#_3.2_Уравнение_Фоккера-Планка)

[3.3 Система с бистабильным поведением 40](#_3.3_Система_с)

[3.4 Система с моностабильным поведением 46](#_3.4_Система_с)

[4 Задача о затухающем линейном осцилляторе 51](#_4_Задача_о)

[ЗАКЛЮЧЕНИЕ](#_ЗАКЛЮЧЕНИЕ) 52

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ) 53

ПРИЛОЖЕНИЕ А 54

# ВВЕДЕНИЕ

Производственная практика студентов МГТУ им. Н.Э. Баумана является обязательной частью основной образовательной программы высшего образования, одной из форм организации учебного процесса.

Практика – вид учебной работы, направленный на развитие практических навыков и умений, а также формирование компетенций обучающихся в процессе выполнения определенных видов работ, связанных с будущей профессиональной деятельностью.

Основными видами практики студентов Университета, обучающихся по основным образовательным программам высшего профессионального образования, являются:

- учебная;

- производственная.

Производственная практика направлена на формирование у студентов

практических профессиональных умений, приобретение практического опыта по основным видам производственной деятельности для последующего освоения ими общих и профессиональных компетенций по избранной специальности/направлению подготовки. Разновидность производственной практики – практика преддипломная.

Целями преддипломной практики являются:

- ознакомление с реальным производственным процессом на предприятии и участие в нём;

- подготовка материалов для выпускной квалификационной работы.

Задачами преддипломной практики являются:

- участие в производственном процессе на предприятии;

- обзор источников для выпускной квалификационной работы;

- выполнение вычислительной части выпускной квалификационной работы;

- написание программного кода на С++;

- и другое.

Отчёт о практике в электронном виде (формат Word) после получения зачёта направляется студентом в электронный архив кафедры, адрес: archive-fn@mail.ru

# 1 Стохастический метод Галёркина

Метод Галёркина – это интрузивный метод приближенного решения краевой задачи , где – некоторый непрерывный дифференциальный оператор, который может содержать как полные, так и частные производные любого порядка.

Будем искать приближенное решение данной краевой задачи в виде

где функции – это некоторые линейно-независимые функции, удовлетворяющие граничным условиям, наложенным на систему, а – некоторые неопределённые коэффициенты. При этом можно считать, что представляют собой первые функций некоторой полной системы функций.

Пусть теперь является точным решением дифференциального уравнения , то есть пусть . Это требование равносильно ортогональности невязки полученного решения ко всем функциям , так как система функций является полной. Однако, так как мы можем оперировать только первыми функциями , мы можем удовлетворить лишь условий ортогональности, в связи с чем точного решения, в общем случае, мы получить не можем, так как решить систему из бесконечного числа уравнений в общем случае невозможно. Запишем условия ортогональности для уравнений

Из системы уравнений (2) можно получить все коэффициенты и, подставив их в выражение (1), получить приближенное решение данной краевой задачи. [1]

Помимо решения дифференциальных уравнений, метод Галёркина также может быть использован для нахождения параметров регрессионной модели полиномиального хаоса при помощи детерминированных уравнений, определяющих поведение системы. При этом коэффициенты регрессии считаются неизвестными, они находятся путём вычисления стохастических проекций Галёркина. Будем называть данную спецификацию метода Галёркина *стохастическим методом Галёркина*. [2]

Запишем ал горитм стохастического метода Галёркина

1. Записать разложение полиномиального хаоса для входных факторов, учитывая их совместное распределение
2. Записать отклик модели в виде линейной комбинации базисных полиномов, которые бы удовлетворяли наложенным на систему граничным условиям
3. Подставить разложение полиномиального хаоса и отклик модели в детерминированные уравнения
4. Вычислить скалярное произведение левой и правой части уравнения, полученного в пункте 3, с каждым базисным полиномом и получить все необходимые коэффициенты разложения
5. Рассчитать статистические характеристики решения, используя свойства коэффициентов ПХ

Так, для одномерной нормально распределённой выборки коэффициенты можно найти численно, используя общую формулу:

Для численного интегрирования данного выражения используем метод трапеций. Отсортировав нормально распределённую выборку, получим неравномерную сетку, шаг которой определяется как . Тогда введём обозначение

и получим, что

Для двумерной нормально распределённой выборки коэффициенты ищутся аналогичным образом.

Теперь аналогично одномерному случаю введём шаг на двумерной неравномерной сетке и вычислим двойной интеграл в формуле для численно.

Проведём сравнение стохастического метода Галёркина со спектральными методами анализа регрессионных моделей (pseudo-spectral approach). Метод Галёркина – это интрузивный метод, зависящий от внутреннего устройства модели. Коэффициенты в разложении полиномиального хаоса вычисляются путём вычисления скалярного произведения отклика модели с соответствующим базисным полиномом, вследствие чего не возникает дополнительных вычислительных ошибок. Спектральные же методы являются неинтрузивными: модель воспринимается как чёрный ящик, внутреннее устройство которого не влияет на результат аппроксимации. Коэффициенты разложения считаются численно, с помощью, например, метода наименьших квадратов. Таким образом, спектральные методы оказываются гораздо более простыми в использовании, так как являются неинтрузивными и не зависят от используемой модели. В то же время стохастический метод Галёркина является более точным методом и сходится гораздо быстрее. [2]

# 2 Сравнение стохастического метода Галёркина и аппроксимации полиномами Колмогорова-Габора

Пусть входные данные распределены по нормальному закону с параметрами . Длину вектора входных данных возьмём равной 800.

Поскольку входные данные распределены по нормальному закону, в качестве базисных полиномов в методе Галёркина были взяты ортонормированные полиномы Эрмита

где – целая часть числа .

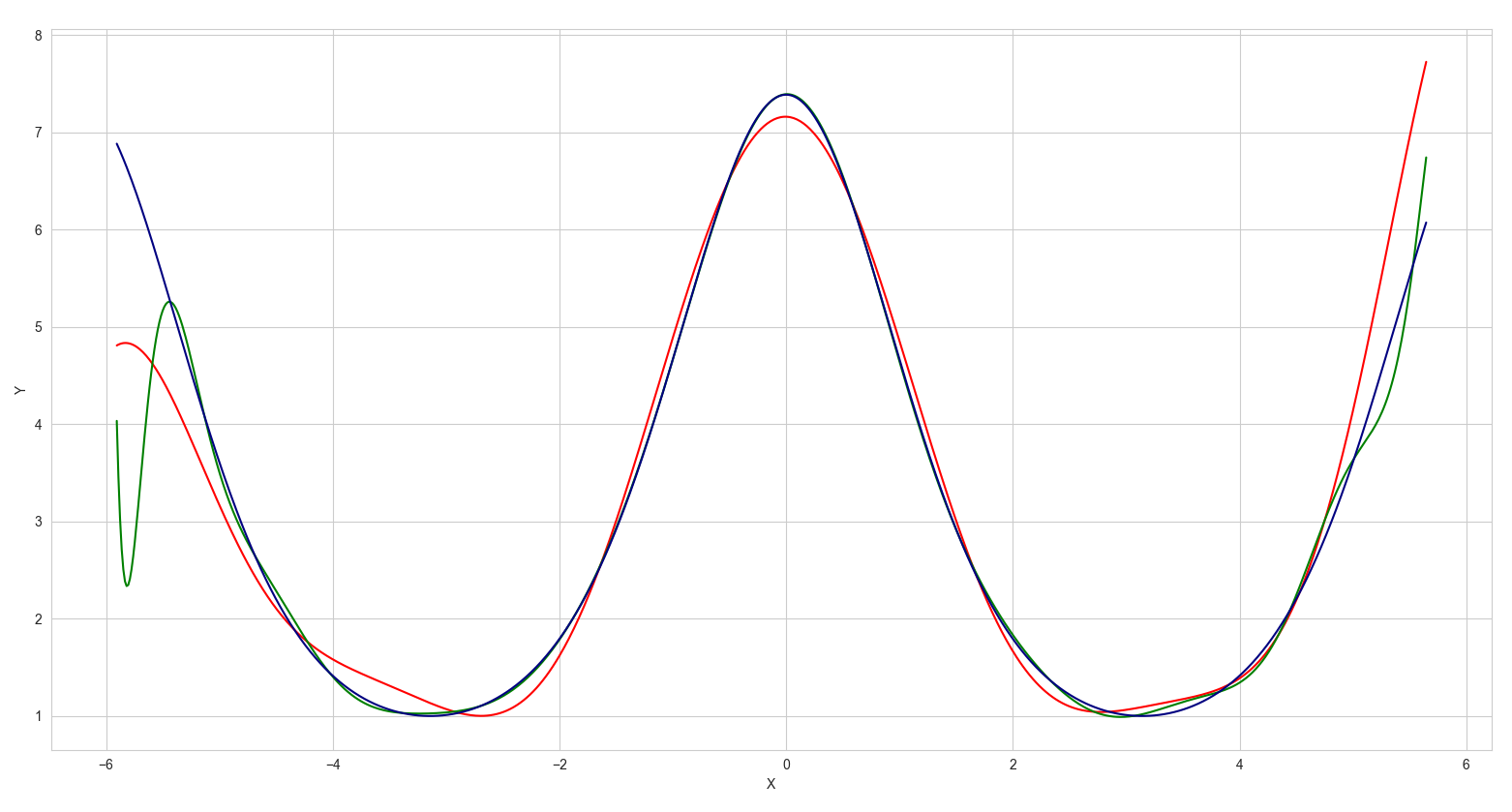
Поскольку методы сходятся с разной скоростью, будем обрывать ряд в том случае, если 6 коэффициентов подряд в разложении ряда оказались меньше 0.1 по своему абсолютному значению.

Приведём таблицу сравнения среднеквадратичных погрешностей

разных методов (обозначим за среднеквадратичную погрешность стохастического метода Галёркина и за – среднеквадратичную погрешность аппроксимации полиномами Колмогорова-Габора) и количество слагаемых в разложении ряда (аналогично – степень многочлена, представляющего собой усечение ряда полиномов Эрмита, – ряда полиномов Колмогорова-Габора) в зависимости от аппроксимируемой функции , среднеквадратичного отклонения входных данных .

Таблица 1 – сравнение среднеквадратичных погрешностей стохастического метода Галёркина и аппроксимации полиномами Колмогорова-Габора на тестовых примерах

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |

 Рисунок 1 – графики аппроксимируемой функции (синий цвет), аппроксимации методом Галёркина (красный цвет) и аппроксимации полиномами Колмогорова-Габора (зелёный цвет) для случая, представленного 4-ой строкой таблицы 1.

Таким образом, хоть и при малом среднеквадратичном отклонении входных данных аппроксимация полиномами Колмогорова-Габора и является гораздо более точной, при повышении среднеквадратичного отклонения погрешность аппроксимации полиномами Колмогорова-Габора очень сильно возрастает, в то время как погрешность метода Галёркина растёт гораздо медленнее.

# 3 Применение стохастического метода Галёркина к нахождению плотности эволюции системы

## **3.1 Модель Шлёгля**

Рассмотрим в качестве реальной системы упрощённую версию модели Шлёгля, поскольку она считается простейшей возможной одномерной бистабильной системой. Она описывается следующей схемой взаимодействий.

Эта система содержит только один тип частиц; каждое взаимодействие происходит с заданной интенсивностью . Значение соответствует интенсивности появление новой частицы , гибель частицы, соответствует интенсивности преобразования 2 частиц в 3 частицы того же типа, аналогично соответствует интенсивности преобразования 3 частиц в 2 частицы того же типа.

Проанализируем теперь детерминированную модель для модели Шлёгля. Уравнение детерминированной модели выглядит следующим образом

где – количество молекул вещества [3].

Примем новые обозначения . Тогда в новых обозначениях выпишем дискриминант многочлена

При система бистабильна с одной нестабильной фиксированной точкой между двумя стабильными фиксированными точками, при система моностабильна. Переход от моностабильности к бистабильности происходит через бифуркацию седлового узла.

Детерминированный подход хорошо описывает поведение данного процесса при большом количестве молекул, что даёт возможность пренебречь стохастическими флуктуациями [4].

## **3.2 Уравнение Фоккера-Планка и нахождение плотности эволюции системы**

Детерминированный подход хорошо подходит для тех случаев, когда размеры молекул достаточно большие и стохастическими флуктуациями можно пренебречь. В том случае, когда всё-таки нужно учитывать стохастические эффекты, удобно бывает записать для системы уравнение Фоккера-Планка. Уравнение Фоккера-Планка – это дифференциальное уравнение в частных производных, описывающее временную эволюцию функции плотности вероятности системы. Частным случаем уравнения Фоккера-Планка является уравнение Эйнштейна-Смолуховского, впервые полученное при описании броуновского движения.

Уравнение Фоккера-Планка записывается следующим образом

С физической точки зрения уравнение Фоккера-Планка описывает обобщённый диффузионный процесс, где вектор называется вектором сноса, – тензором диффузии.

В одномерном случае уравнение Фоккера-Планка записывается в следующем виде

причём функция связана с функциями и следующим стохастическим дифференциальным уравнением [5]

Рассмотрим теперь уравнение Фоккера-Планка (2.25) для модели Шлёгля (2.7).

где

При уравнение Фоккера-Планка становится стационарным и не зависит от времени, поэтому можно записать

Решая это уравнение, получим

где – константа, нормализующее распределение. Можно показать, что это решение является точным для систем с гауссовым белым шумом [4].

## 3.3 Система с бистабильным поведением

Рассмотрим частный случай модели Шлёгля, когда система имеет бистабильное поведение. Исходя из анализа детерминированной модели (2.13), система имеет бистабильное поведение тогда и только тогда, когда дискриминант многочлена

где .

Возьмём некоторые конкретные значения интенсивностей переходов в схеме (10), удовлетворяющие (21). Пусть, например, интенсивности заданы через особые точки дифференциального уравнения следующим образом

где .

Применим стохастический метод Галёркина к данной системе.

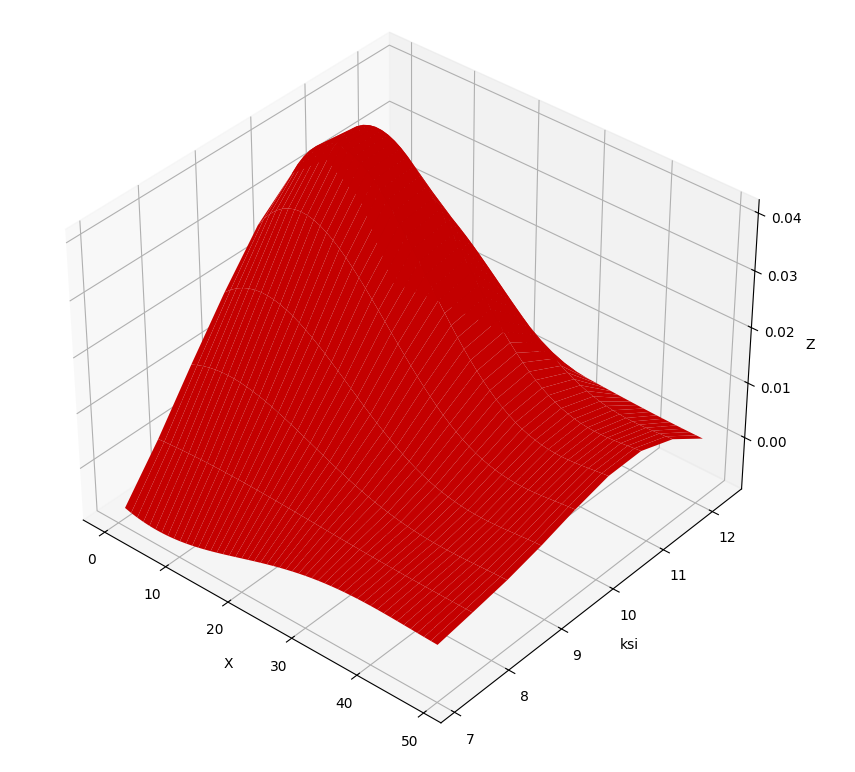


Рисунок 2 – аппроксимация плотности эволюции системы (22) стохастическим методом Галёркина.

Разложение проводилось до 18-го порядка; среднеквадратичная ошибка аппроксимации в данном случае оказалась равна .

Построим сечение данного графика для . Выборка параметра состоит из 10 значений и выглядит так:

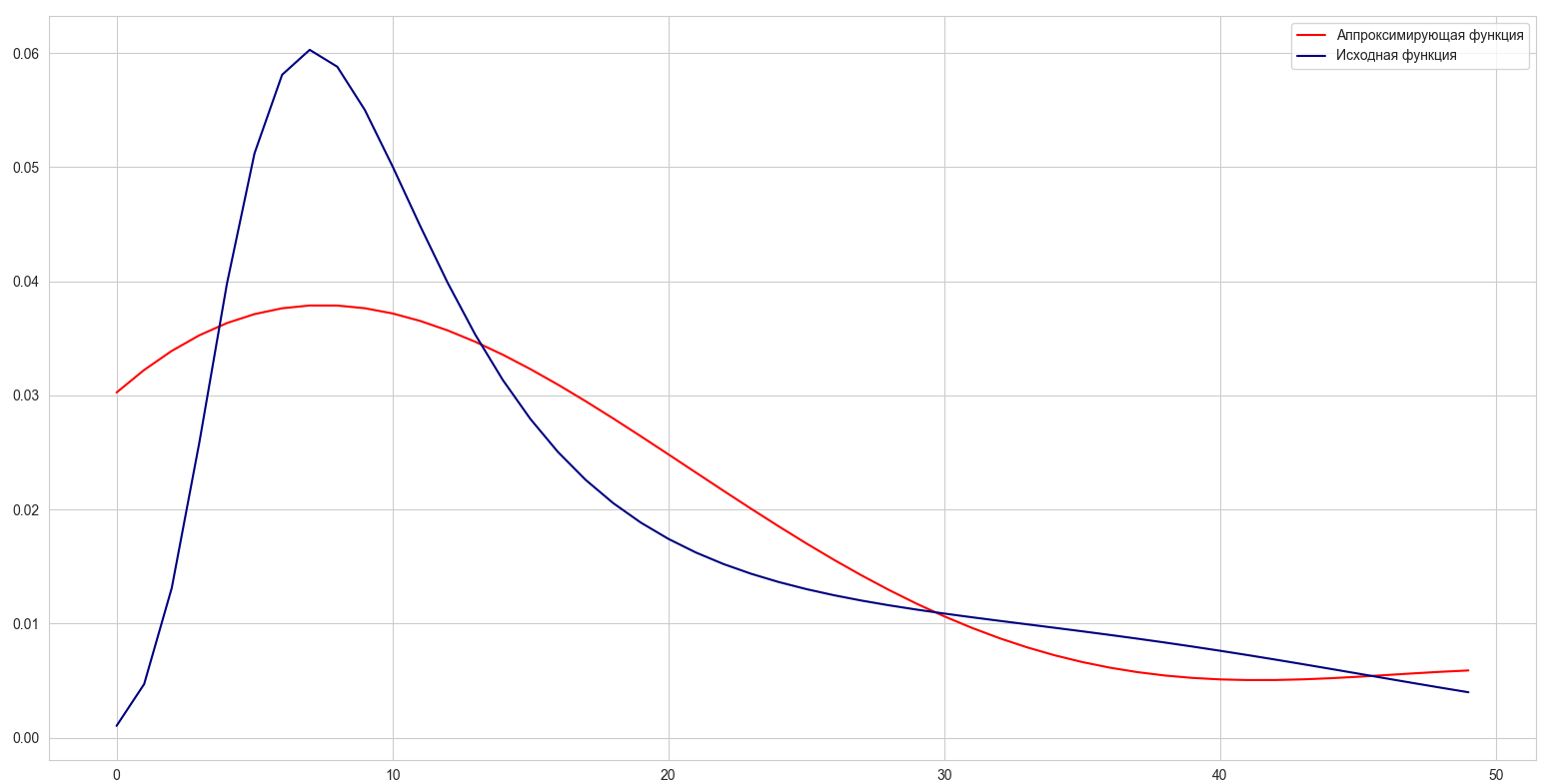
**

Рисунок 3 – сечение аппроксимирующей функции, построенной с помощью стохастического метода Галёркина (синий цвет – исходная функция, красный цвет – аппроксимация)

Среднеквадратичная ошибка аппроксимации, посчитанная для данного сечения, составила .

Как видно из рисунка 4, аппроксимирующая функция сильно отличается по значениям в точках от аппрокисмируемой. Однако можно заметить, что средние значения на рассматриваемом нами множестве функций довольно близки: взяв интеграл по рассматриваемому нами множеству от обоих функций, мы получаем довольно близкие значения:

где – детерминированная плотность вероятности эволюции системы при , – аппроксимирующая функция при .

Зададим следующие обозначения

Нетрудно видеть, что – не что иное, как функция распределения вероятности эволюции системы, а – аппроксимация этой функции

Проведём сравнение этих двух функций

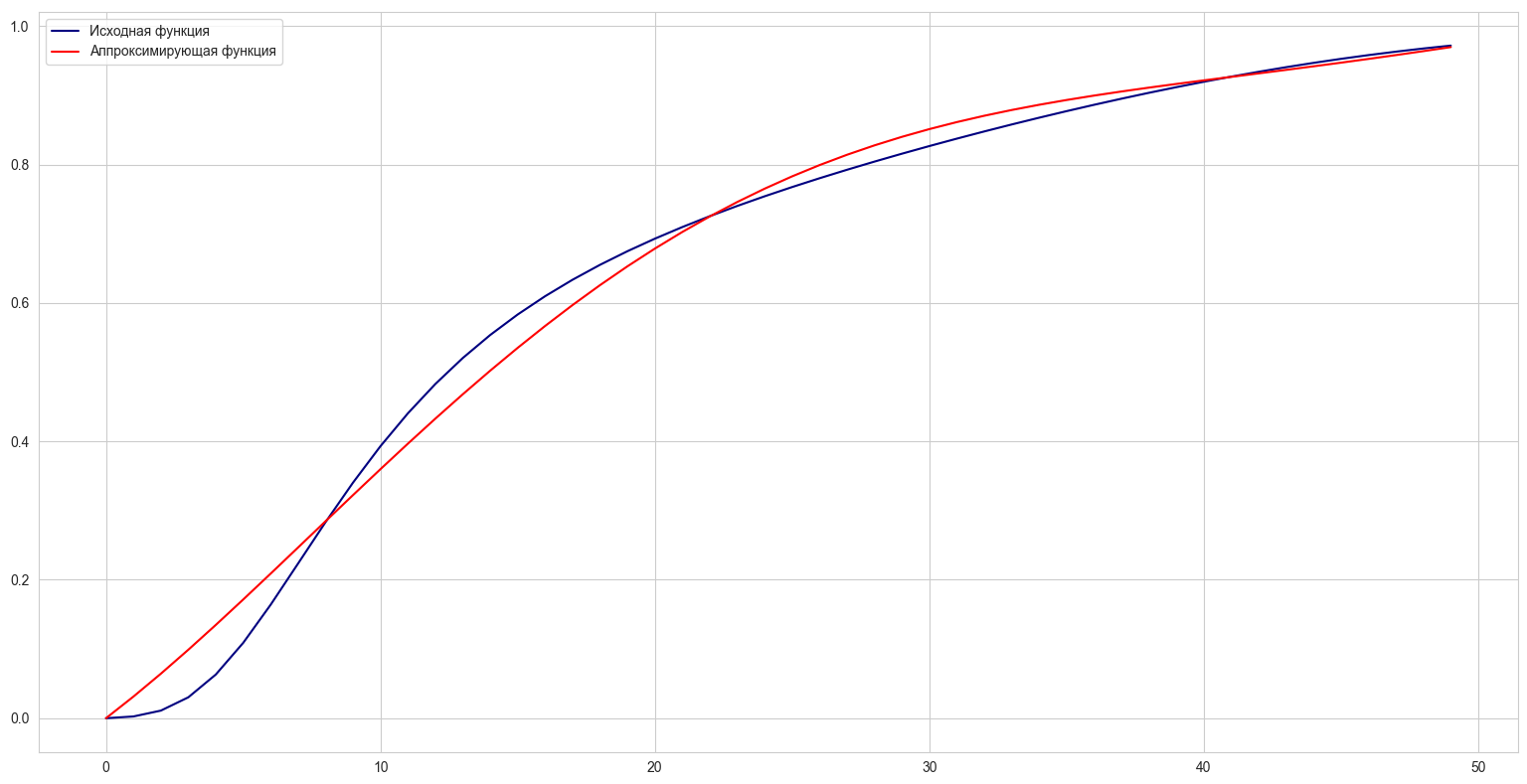


Рисунок 4 – графики функций (синий цвет) и (красный цвет)

Среднеквадратичная ошибка аппроксимации в данном случае составила .

Таким образом, стохастический метод Галёркина хорошо подходит для тех случаев, когда важна не близость значений функций в определённых точках, а схожесть средних значений функций в определённых областях.

## **3.4 Система с моностабильным поведением**

Аналогично рассмотрим частный случай модели Шлёгля, когда система имеет моностабильное поведение. В этом случае дискриминант (21) многочлена .

Возьмём некоторые конкретные значения интенсивностей переходов в схеме (2.7), удовлетворяющие этому неравенству. Пусть, например, интенсивности заданы следующим образом

где .

Применим стохастический метод Галёркина к данной системе.

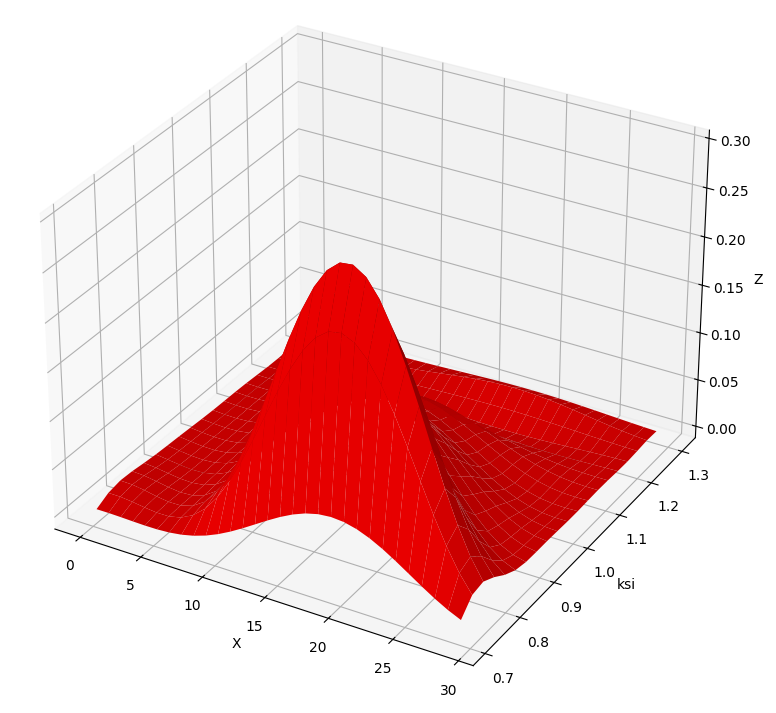


Рисунок 5 – аппроксимация плотности эволюции системы с интенсивностями (25) стохастическим методом Галёркина.

Разложение проводилось до 18-го порядка. Среднеквадратичная ошибка аппроксимации составила в данном случае .

Построим сечение данного графика для . Выборка параметра состоит из 20 значений и выглядит так:

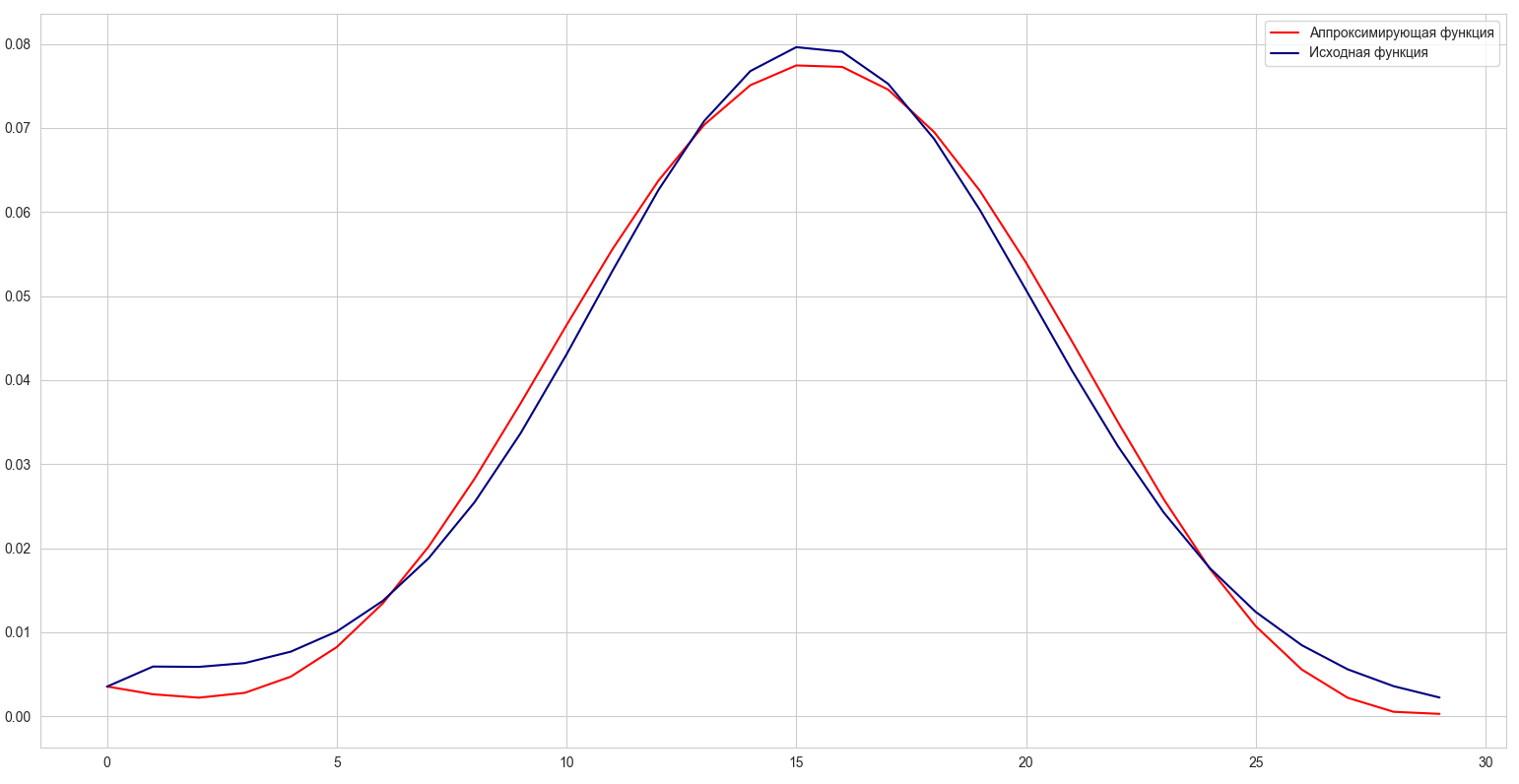


Рисунок 6 – сечение аппроксимирующей функции, построенной с помощью стохастического метода Галёркина (синий цвет – исходная функция, красный цвет – аппроксимация)

Среднеквадратичная ошибка аппроксимации, посчитанная для данного сечения, составила .

Аналогично бистабильному случаю, рассмотрим функции распределения

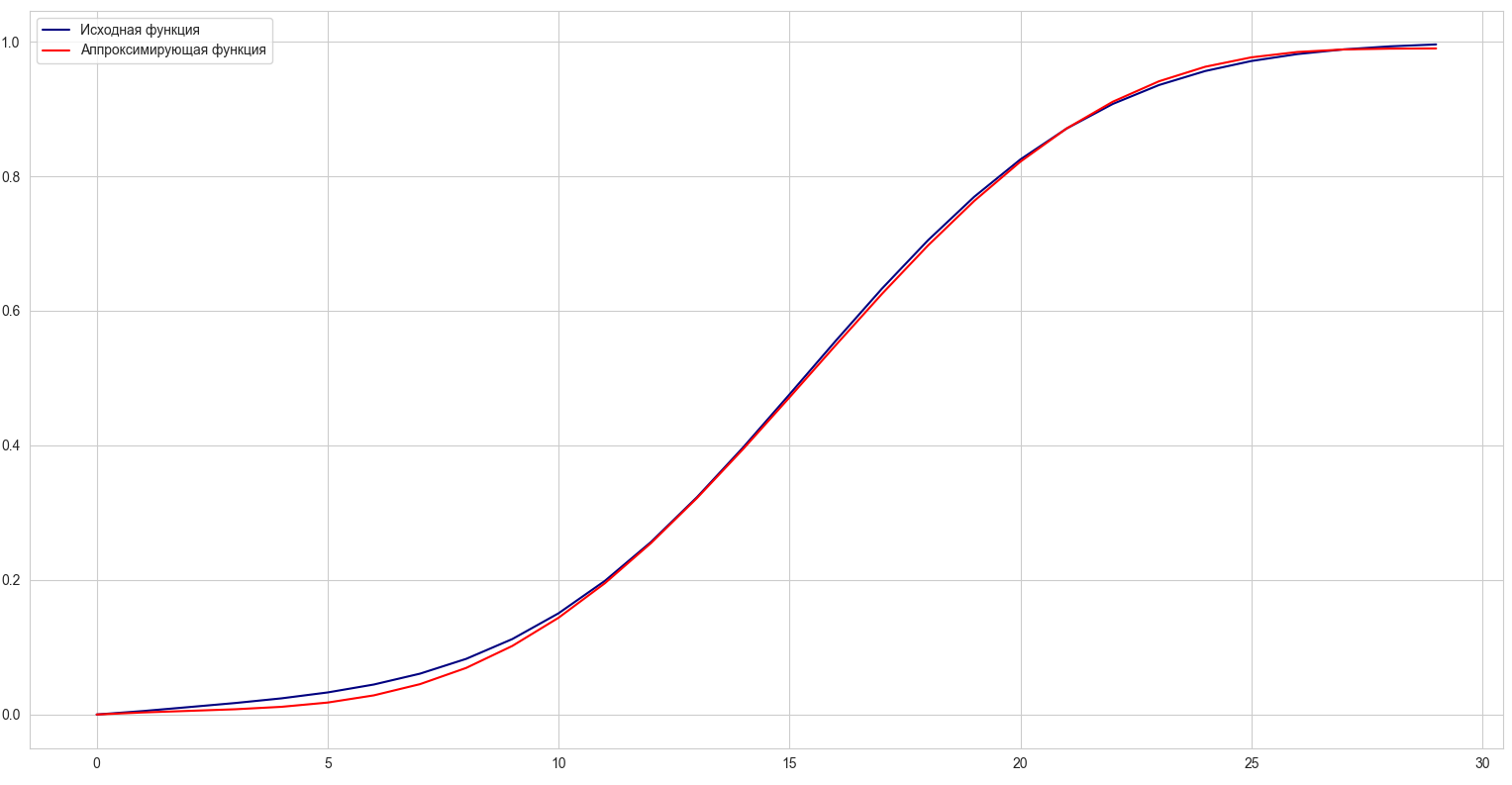


Рисунок 7 – графики функций (синий цвет) и (красный цвет)

Среднеквадратичная ошибка аппроксимации в данном случае составила .

Выпишем также интегралы по всей рассматриваемой области и убедимся, что получаемые значения оказываются довольно близкими

Таким образом, в данной работе было показано, что стохастический метод Галёркина наиболее хорошо подходит для оценки интегральных параметров функции, а не значений функции в отдельных точках.

# 4 Задача о затухающем линейном осцилляторе

Рассмотрим модель затухающего линейного осциллятора

где – масса тела, – коэффициент затухания колебаний, – жёсткость пружины, – амплитуда вынуждающей силы, – циклическая частота вынуждающей силы, – начальное положение, – начальная скорость. Примем также следующие ограничения на параметры системы

где – собственная частота колебаний системы, .

При данных ограничениях в системе будет установлен колебательный режим без эффектов резонанса.

Зададим некоторые конкретные значения характеристик системы

Поскольку модель (28) является динамической, коэффициенты в разложении полиномиального хаоса являются скалярными функциями от времени

Решим данную задачу методом Галёркина. Для этого перепишем задачу (28) в несколько ином виде.

где имеет физический смысл скорости.

Пусть . Тогда получаем следующее соотношение

Теперь представим функции и в виде разложения полиномиального хаоса

Подставим при этом начальные условия, получим следующие соотношения

Заметим, при этом, что, с одной стороны

а с другой стороны

Отсюда следует, что

Вычислим также скалярное произведение левой и правой части уравнения (32) с каждым базисным полиномом .

И, таким образом, приходим к выражению

Подставив во второе уравнение системы (32) разложение полиномиального хаоса для скорости (34), получим

Вычислим скалярное произведение левой и правой частей уравнения (2.84) с каждым базисным полиномом . Для этого распишем это произведение отдельно для каждого слагаемого.

Таким образом, имеет место следующее уравнение

Наконец, приходим к следующей системе уравнений

где

Таким образом, задача Коши для двух обыкновенных дифференциальных уравнений (ОДУ) первого порядка, разрешённых относительно производной, при добавлении в систему стохастических эффектов сводится к системе из ОДУ первого порядка.

Решим задачу Коши (2.91). Для этого представим её в несколько ином виде

где – неизвестная вектор-функция, зависящая от времени,

– гладкая вектор-функция

Зададим функцию как

А начальное условие как

Представим вектор-функцию в следующем виде

где – невырожденная квадратная матрица размера ,

– вектор-функция, зависящая от времени, причём

Матрица является блочной и имеет следующую структуру

Интегрирование системы (2.92) производилось методом Рунге-Кутта третьего порядка. При этом сложность алгоритма составила , где – размер равномерной сетки.

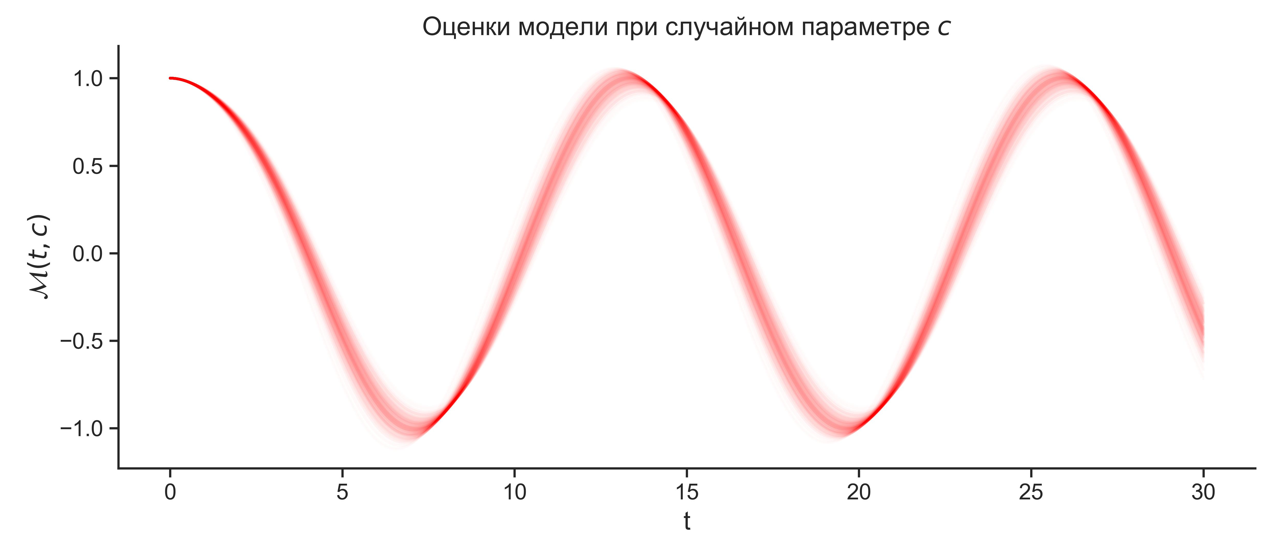


Рисунок 8 – Набор траекторий при вариации коэффициента затухания колебаний.

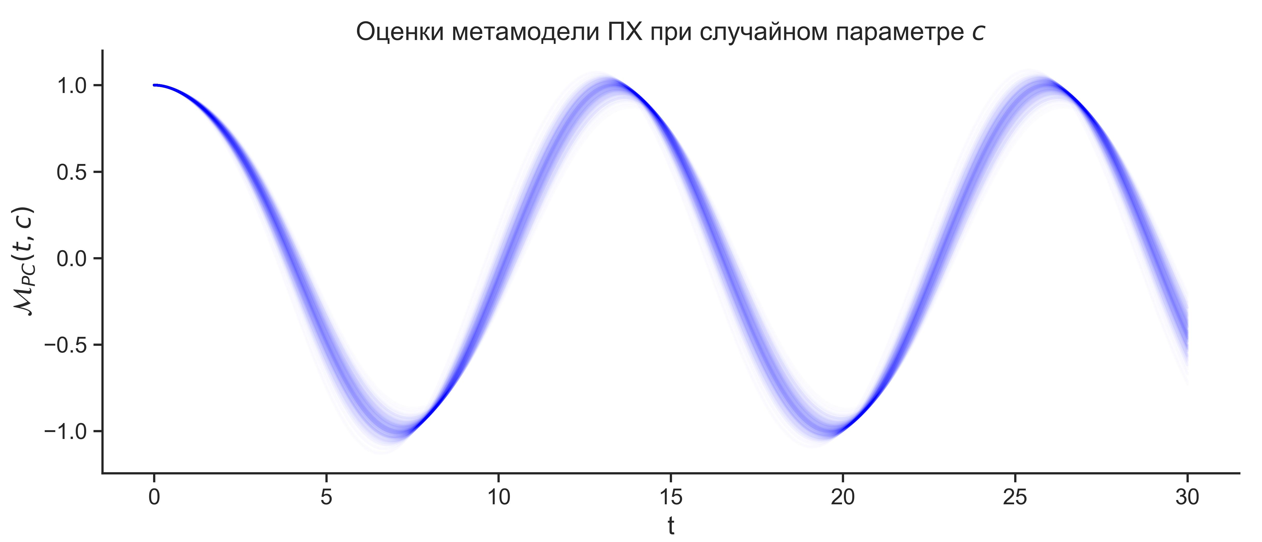
**

Рисунок 9 – Набор траекторий при вариации коэффициента затухания колебаний.

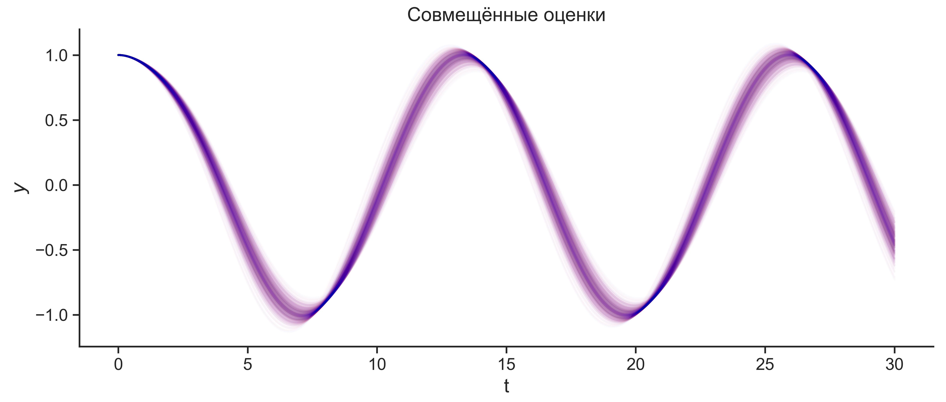
**

Рисунок 10 – Совмещённые траектории модели и метамодели ПХ при вариации коэффициента затухания колебаний.

Приведём также зависимость математического ожидания и стандартного отклонения траекторий от времени.

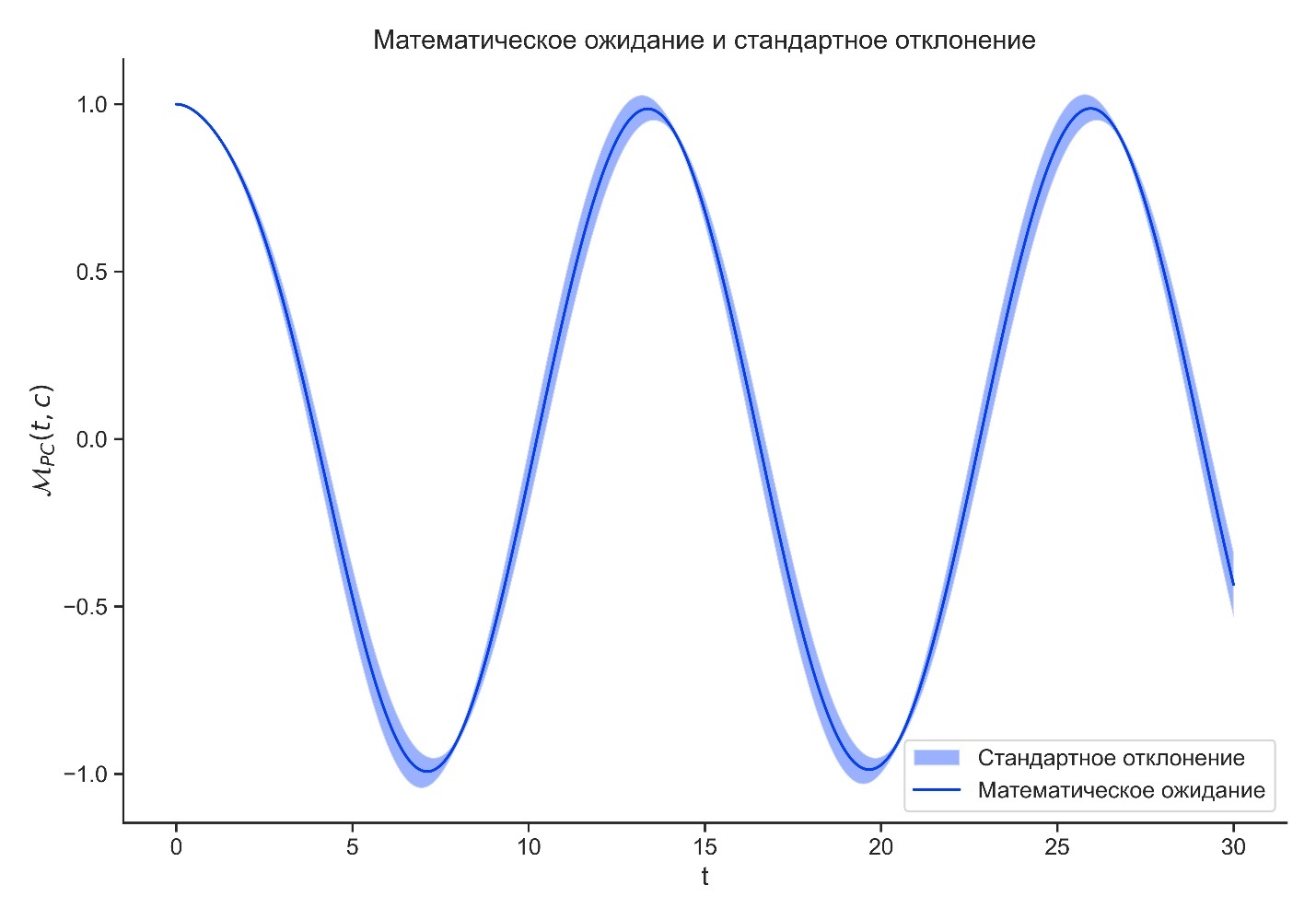
**

Рисунок 11 – Математическое ожидание и стандартное отклонение траекторий метамодели ПХ.

Ошибка аппроксимации при данном подходе составила .

Таким образом, стохастический метод Галёркина можно очень эффективно использовать для решения дифференциальных уравнений в том случае, если некоторые характеристики данного уравнения являются случайными величинами, которые можно выразить как функции от случайных величин с известным ортонормированным базисом. Более того, стохастический метод Галёркина остаётся достаточно точным даже в том случае, когда среднеквадратичное отклонение оказывается достаточно большим.

# ЗАКЛЮЧЕНИЕ

В данной работе был подробно разобран стохастический метод Галёркина. Он сравнивался с аппроксимацией методом наименьших квадратов, где в качестве ортонормированного базиса были выбраны полиномы Колмогорова-Габора, на некоторых тестовых примерах. Также стохастический метод Галёркина был применён к задаче нахождения плотности вероятности эволюции системы для модели Шлёгля в том случае, когда интенсивности взаимодействий зависят от некоторой нормально распределённой случайной величины. Наконец, в данной работе был использован стохастический метод Галёркина и для решения дифференциальных уравнений, а именно для решения задачи о затухающем линейном осцилляторе, причём коэффициент затухания колебаний являлся случайной величиной, распределённой по нормальному закону. В данной работе была доказана эффективность стохастического метода Галёркина для нахождения интегральных параметров системы, а также показана его сравнительная эффективность в тех случаях, когда среднеквадратичное отклонение ошибок оказывается сравнительно велико. Таким образом, в данной работе были определены границы применимости данного метода.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Канторович Л.В., Крылов В.И. Приближённые методы высшего анализа. — 5-е изд. — Л.-М., 1962.
2. Neckel T. Lecture 7, Polynomial Chaos Approximation 2: The stochastic Galerkin approach. – Algorithms for Uncertainty Quantification, Technische Universität München, 2018.
3. А. В. Калинкин, Схемы взаимодействий: детерминированные и стохастические модели: Методические указания к выполнению типового расчета по курсу «Дополнительные главы теории случайных процессов». М.: Изд-во МГТУ им. Н.Э. Баумана, 2009. – 44 с.
4. Falk J, Mendler M, Drossel B, A minimal model of burst-noise induced bistability, PLoS ONE 12(4): e0176410, 2017. – 15 с.
5. Gardiner C. Stochastic Methods – A Handbook for the Natural and Social. Springer; 2009.

# ПРИЛОЖЕНИЕ А

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Кафедра «Вычислительная математика и математическая физика» (ФН-11)

**ЗАДАНИЕ**

**на прохождение производственной практики**

на предприятии НОЦ «СИМПЛЕКС» МГТУ им. Н.Э. Баумана

Студент группы ФН11-82Б Хаписов Малик Хаписович

Во время прохождения производственной (преддипломной) практики студент должен:

1. провести обзор источников по теме заданной выпускной квалификационной работы и составить список литературы, при этом необходимо реализовать программный код на языке С++,

2. выполнить вычислительную часть выпускной квалификационной работы,

3. электронную версию готового отчёта по практике (формат Word) выслать в электронный архив кафедры – на адрес электронной почты archive-fn@mail.ru

Дата выдачи задания « 13 » мая 2024 г.

Руководитель практики от кафедры\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А.А. Прозоровский /

(подпись, дата)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /** М.Х. Хаписов **/**

(подпись, дата)

# ПРИЛОЖЕНИЕ Б

**Программный код на языке Python**

galerkin.py:

import numpy as np

import matplotlib.pyplot as plt

class Solver:

@staticmethod

def He(n, x):

def factorial(num):

res = 1

for k in range(2, num + 1):

res \*= k

return res

S = 0

for j in range(int(0.5 \* n) + 1):

S += (-0.5) \*\* j \* np.sqrt(factorial(n)) / (factorial(j) \* factorial(n - 2 \* j)) \* x \*\* (n - 2 \* j)

return S

def draw(self):

pass

class Solver\_1D(Solver):

def \_\_init\_\_(self, f, X, \*, acc=0, p=0):

self.f = f

self.X = X

self.X.sort()

self.m = np.mean(self.X)

self.s = np.sqrt(np.var(self.X))

self.acc = acc

self.p = p

def hermite\_approximation(self):

def a(j):

def h(k):

return self.X[k + 1] - self.X[k]

def I(k):

return self.f(self.X[k]) \* He(j, (self.X[k] - self.m) / self.s) \* np.exp(-(self.X[k] - self.m) \*\* 2 / (2 \* self.s \* self.s))

S = 0

for k in range(len(self.X) - 1):

S += 0.5 \* h(k) \* (I(k) + I(k + 1))

return S / (np.sqrt(2 \* np.pi) \* self.s)

A = []

j = 0

if self.p > 0:

while j <= self.p:

A.append(a(j))

j += 1

elif self.acc > 0:

k = 0

while k < 6:

x = a(j)

A.append(x)

if np.abs(x) < self.acc:

k += 1

else:

k = 0

j += 1

else:

raise ValueError("p или acc должны быть положительным числом")

print("Коэффициенты в разложении полиномиального хаоса: ", A, end='\n\n')

print(f"Разложение производится до {len(A) - 1} порядка", end='\n\n')

def Fest(x):

S = 0

for j in range(len(A)):

S += A[j] \* Solver.He(j, (x - self.m) / self.s)

return S

return Fest

def kolmogorov\_approximation(self):

if self.p == 0:

flag = False

N = 6

while not flag:

M = [[sum([self.X[i] \*\* (k + j) for i in range(len(self.X))]) for k in range(N + 1)] for j in range(N + 1)]

V = [sum([self.f(self.X[i]) \* self.X[i] \*\* j for i in range(len(self.X))]) for j in range(N + 1)]

A = np.linalg.solve(M, V)

if max(map(abs, A[-6:])) < self.acc:

flag = True

else:

flag = False

N += 1

else:

M = [[sum([self.X[i] \*\* (k + j) for i in range(len(self.X))]) for k in range(self.p + 1)] for j in range(self.p + 1)]

V = [sum([self.f(self.X[i]) \* self.X[i] \*\* j for i in range(len(self.X))]) for j in range(self.p + 1)]

A = np.linalg.solve(M, V)

print("Коэффициенты ряда: ", A, end='\n\n')

print(f"Разложение производится до {len(A) - 1} порядка", end='\n\n')

def Fest(x):

S = 0

for k in range(len(A)):

S += A[k] \* x \*\* k

return S

return Fest

def compare(self):

plt.grid(True)

D = np.linspace(self.m - 3 \* self.s, self.m + 3 \* self.s, len(self.X))

hermite = self.hermite\_approximation()

kolmogorov = self.kolmogorov\_approximation()

plt.plot(D, [hermite(x) for x in D], color="red")

plt.plot(D, [kolmogorov(x) for x in D], color="green")

plt.plot(D, [self.f(x) for x in D], color="navy")

plt.xlabel("X")

plt.ylabel("Y")

err\_hermite = sum([(self.f(x) - hermite(x)) \*\* 2 for x in D]) / len(D)

print(f"Среднеквадратичная ошибка аппроксимации при использовании полиномов Эрмита равна {err\_hermite}", end='\n\n')

err\_kolmogorov = sum([(self.f(x) - kolmogorov(x)) \*\* 2 for x in D]) / len(D)

print(f"Среднеквадратичная ошибка аппроксимации при использовании полиномов Колмогорова-Габора равна {err\_kolmogorov}", end='\n\n')

plt.show()

def draw(self):

plt.grid(True)

D = np.linspace(self.m - 3 \* self.s, self.m + 3 \* self.s, len(self.X))

hermite = self.hermite\_approximation()

plt.plot(D, [hermite(x) for x in D], color="red")

plt.plot(D, [self.f(x) for x in D], color="navy")

plt.xlabel("X")

plt.ylabel("Y")

err\_hermite = sum([(self.f(x) - hermite(x)) \*\* 2 for x in D]) / len(D)

print(f"Среднеквадратичная ошибка аппроксимации при использовании полиномов Эрмита равна {err\_hermite}", end='\n\n')

plt.show()

class Solver\_2D(Solver):

def \_\_init\_\_(self, f, X, Y, \*, acc=0, p=0):

self.f = f

self.X = X

self.X.sort()

self.Y = Y

self.Y.sort()

self.mx = np.mean(self.X)

self.my = np.mean(self.Y)

self.sx = np.sqrt(np.var(self.X))

self.sy = np.sqrt(np.var(self.Y))

self.nx = len(self.X)

self.ny = len(self.Y)

self.acc = acc

self.p = p

def hermite\_approximation(self):

def acalc(k):

def a(i, j):

def hx(kx):

return self.X[kx + 1] - self.X[kx]

def hy(ky):

return self.Y[ky + 1] - self.Y[ky]

def I(kx, ky):

return self.f(self.X[kx], self.Y[ky]) \* Solver.He(i, (self.X[kx] - self.mx) / self.sx) \* Solver.He(j, (self.Y[ky] - self.my) / self.sy) \* np.exp(-(self.X[kx] - self.mx) \*\* 2 / (2 \* self.sx \* self.sx)) \* np.exp(-(self.Y[ky] - self.my) \*\* 2 / (2 \* self.sy \* self.sy))

S = 0

for kx in range(self.nx - 1):

for ky in range(self.ny - 1):

S += hx(kx) \* hy(ky) \* 0.25 \* (I(kx, ky) + I(kx + 1, ky) + I(kx, ky + 1) + I(kx + 1, ky + 1))

return S / (2 \* np.pi \* self.sx \* self.sy)

return [[a(i, j) for j in range(k + 1)] for i in range(k + 1)]

A = []

L = 0

if self.p > 0:

A = acalc(self.p)

L = self.p

elif self.acc != 0:

k = 0

while k < 6:

data = acalc(L)

if max(map(abs, [elem for elem in sum(data, []) if elem not in sum(A, [])])) < self.acc:

k += 1

else:

k = 0

A = data

L += 1

else:

raise ValueError("p или acc должны быть положительным числом")

print(f"Разложение производится до {max([len(M) for M in A]) - 1} порядка", end='\n\n')

def Fest(x, y):

S = 0

for i in range(L):

for j in range(L):

S += A[i][j] \* Solver.He(i, (x - self.mx) / self.sx) \* Solver.He(j, (y - self.my) / self.sy)

return S

return Fest

def draw(self):

fig = plt.figure()

axes = fig.add\_subplot(projection='3d')

Dx = np.arange(self.mx - 3 \* self.sx, self.mx + 3 \* self.sx, 6 \* self.sx / (self.nx - 1))

Dy = np.arange(self.my - 3 \* self.sy, self.my + 3 \* self.sy, 6 \* self.sy / (self.ny - 1))

xgrid, ygrid = np.meshgrid(Dx, Dy)

hermite = self.hermite\_approximation()

axes.plot\_surface(xgrid, ygrid, hermite(xgrid, ygrid), color="red")

axes.plot\_surface(xgrid, ygrid, self.f(xgrid, ygrid), color="navy")

axes.set\_xlabel("X")

axes.set\_ylabel("Y")

axes.set\_zlabel("Z")

err\_hermite = sum([(self.f(x, y) - hermite(x, y)) \*\* 2 for x in Dx for y in Dy]) / (len(Dx) \* len(Dy))

print(f"Среднеквадратичная ошибка аппроксимации при использовании полиномов Эрмита равна {err\_hermite}", end='\n\n')

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

np.random.seed(100000)

def f1(x):

return np.sin(2 \* x)

X = [np.random.normal(0, 16) for \_ in range(800)]

s1 = Solver\_1D(f1, X, p=9)

s1.compare()

FP.py:

from galerkin import Solver\_2D

import numpy as np

import matplotlib.pyplot as plt

import scipy.integrate as integrate

class Solver\_FP:

def \_\_init\_\_(self, f, xmax, m, s, n, acc=0, p=0):

self.xmax = xmax

self.arrX = [x for x in range(xmax + 1)]

self.arrK = [np.random.normal(m, s) for \_ in range(n)]

print(f"ksi = {self.arrK}", end='\n\n')

N = 1 / integrate.quad(lambda x: f(x, np.mean(self.arrK)), 0, np.inf)[0]

print(f"N = {N}", end='\n\n')

def P(x, ksi):

return N \* f(x, ksi)

self.P = P

solver = Solver\_2D(P, self.arrX, self.arrK, acc=acc, p=p)

self.hermite = solver.hermite\_approximation()

def check(self, \*, acc=0.01):

mksi = np.mean(self.arrK)

for x in self.arrX:

if self.hermite(x, mksi) < 0:

return False

I = lambda x: integrate.quad(lambda y: self.hermite(y, mksi), 0, x)

for x in self.arrX:

if I(x) < 0 or I(x) > 1:

return False

I1 = integrate.quad(lambda x: self.P(x, mksi), 0, self.xmax)[0]

I2 = integrate.quad(lambda x: self.hermite(x, mksi), 0, self.xmax)[0]

return np.abs(I2 - I1) < acc

def draw(self):

err\_hermite = sum([(self.hermite(x, ksi) - self.P(x, ksi)) \*\* 2 for ksi in self.arrK for x in self.arrX]) / (len(self.arrX) \* len(self.arrK))

print(f"Среднеквадратичная ошибка аппроксимации при использовании полиномов Эрмита равна {err\_hermite}", end='\n\n')

Dx = np.arange(self.arrX[0], self.arrX[-1], (self.arrX[-1] - self.arrX[0]) / (len(self.arrX) - 1))

mksi = np.mean(self.arrK)

sksi = np.sqrt(np.var(self.arrK))

print(f"mksi = {mksi}", end='\n\n')

print(f"sksi = {sksi}", end='\n\n')

Dk = np.arange(mksi - 3 \* sksi, mksi + 3 \* sksi, 6 \* sksi / (len(self.arrK) - 1))

xgrid, kgrid = np.meshgrid(Dx, Dk)

fig = plt.figure()

axes = fig.add\_subplot(projection='3d')

axes.plot\_surface(xgrid, kgrid, self.hermite(xgrid, kgrid), color="red")

#axes.plot\_surface(xgrid, kgrid, self.P(xgrid, kgrid), color="navy")

axes.set\_xlabel("X")

axes.set\_ylabel("ksi")

axes.set\_zlabel("Z")

plt.show()

plt.grid(True)

plt.plot(Dx, self.hermite(Dx, mksi), color="red", label="Аппроксимирующая функция")

plt.plot(Dx, [self.P(x, mksi) for x in Dx], color="navy", label="Исходная функция")

err2 = sum([(self.hermite(x, mksi) - self.P(x, mksi)) \*\* 2 for x in self.arrX]) / len(self.arrX)

print(f"Среднеквадратичная ошибка аппроксимации при использовании полиномов Эрмита (2) равна {err2}", end='\n\n')

plt.legend(loc='best')

plt.show()

I1 = integrate.quad(lambda x: self.P(x, mksi), 0, self.xmax)[0]

I2 = integrate.quad(lambda x: self.hermite(x, mksi), 0, self.xmax)[0]

print(f"Интеграл от исходной функции равен {I1}", end='\n\n')

print(f"Интеграл от аппроксимации равен {I2}", end='\n\n')

plt.show()

I1 = lambda x, ksi: integrate.quad(lambda y: self.P(y, ksi), 0, x)[0]

I2 = lambda x, ksi: integrate.quad(lambda y: self.hermite(y, ksi), 0, x)[0]

err3 = sum([(I2(x, mksi) - I1(x, mksi)) \*\* 2 for x in self.arrX]) / len(self.arrX)

print(f"Среднеквадратичная ошибка аппроксимации при использовании полиномов Эрмита (3) равна {err3}", end='\n\n')

plt.plot(Dx, [I1(x, mksi) for x in Dx], color="navy", label="Исходная функция")

plt.plot(Dx, [I2(x, mksi) for x in Dx], color="red", label="Аппроксимирующая функция")

plt.legend(loc='best')

plt.show()

class Solver\_BFP(Solver\_FP):

def \_\_init\_\_(self, X, k4, \*, xmax, m, s, n, acc=0, p=0):

G = lambda ksi: 2 \* X[0](ksi) \* (X[0](ksi) + X[1](ksi)) \* (X[0](ksi) + X[2](ksi)) / ((X[0](ksi) - X[1](ksi)) \* (X[0](ksi) - X[2](ksi)))

H = lambda ksi: 2 \* X[1](ksi) \* (X[0](ksi) + X[1](ksi)) \* (X[1](ksi) + X[2](ksi)) / ((X[0](ksi) - X[1](ksi)) \* (X[1](ksi) - X[2](ksi)))

M = lambda ksi: 2 \* X[2](ksi) \* (X[0](ksi) + X[2](ksi)) \* (X[1](ksi) + X[2](ksi)) / ((X[0](ksi) - X[2](ksi)) \* (X[2](ksi) - X[1](ksi)))

print(f"G(1) \* m = {G(1) \* m}", end='\n\n')

print(f"H(1) \* m = {H(1) \* m}", end='\n\n')

print(f"M(1) \* m = {M(1)\* m}", end='\n\n')

def B(x, ksi):

return k4 \* (x + X[0](ksi)) \* (x + X[1](ksi)) \* (x + X[2](ksi))

def f(x, ksi):

return np.exp(2 \* (G(ksi) \* np.log(1 + x / X[0](ksi)) - H(ksi) \* np.log(1 + x / X[1](ksi)) - M(ksi) \* np.log(1 + x / X[2](ksi)) - x)) / B(x, ksi)

super().\_\_init\_\_(f, xmax, m, s, n, acc, p)

class Solver\_MFP(Solver\_FP):

def \_\_init\_\_(self, K, \*, xmax, m, s, n, acc=0, p=0):

def f(x, ksi):

A = lambda y: K[0](ksi) - K[1](ksi) \* y + K[2](ksi) \* y \* y - K[3](ksi) \* y \*\* 3

B = lambda y: K[0](ksi) + K[1](ksi) \* y + K[2](ksi) \* y \* y + K[3](ksi) \* y \*\* 3

g = lambda y: 2 \* integrate.quad(lambda u: A(u) / B(u), 0, y)[0]

return np.exp(g(x)) / B(x)

super().\_\_init\_\_(f, xmax, m, s, n, acc, p)

if \_\_name\_\_ == "\_\_main\_\_":

seed = 110005

np.random.seed(seed)

def x1(ksi):

return ksi

def x2(ksi):

return 1.5 \* ksi

def x3(ksi):

return 6 \* ksi

X = [x1, x2, x3]

k4 = 0.001

fp = Solver\_BFP(X, k4, xmax=100, m=10, s=1, n=20, p=18)

print(f"seed = {seed}", end='\n\n')

fp.draw()

deterministic.py:

import numpy as np

import matplotlib.pyplot as plt

m = 1

c = 0.8

k = 1.16

f = 0.9881

p = 0.5

x0 = 1

x1 = 0

if c \* c >= 4 \* k \* m:

raise ValueError("Квадрат коэффициента затухания колебаний не должен превышать или быть равным произведению массы тела и жёсткости пружины, умноженному на 4")

w = np.sqrt(4 \* k \* m - c \* c) / (2 \* m)

C1 = f \* (k - m \* p \* p) / (c \* c \* p \* p + (k - m \* p \* p) \*\* 2)

C2 = c \* f \* p / (c \* c \* p \* p + (k - m \* p \* p) \*\* 2)

A = x0 - C1

B = (x1 + c / (2 \* m) \* A - C2 \* p) / w

def f(t):

return np.exp(-c / (2 \* m) \* t) \* (A \* np.cos(w \* t) + B \* np.sin(w \* t)) + C1 \* np.cos(p \* t) + C2 \* np.sin(p \* t)

def g(t):

return -c / (2 \* m) \* np.exp(-c / (2 \* m) \* t) \* (A \* np.cos(w \* t) + B \* np.sin(w \* t)) + np.exp(-c / (2 \* m) \* t) \* w \* (B \* np.cos(w \* t) - A \* np.sin(w \* t)) + p \* (C2 \* np.cos(p \* t) - C1 \* np.sin(p \* t))

plt.grid(True)

T = np.linspace(0, 30, 1024)

plt.plot(T, f(T), color="navy", label="Перемещение x(t)")

plt.plot(T, g(T), color = "red", label="Скорость x'(t)")

plt.legend(loc='best')

plt.show()