МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ УЗБЕКИСТАН

КАРАКАЛПАКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ БЕРДАХА

ОТДЕЛЕНИЕ МАГИСТРАТУРЫ

НЫСАНБАЕВ ШИНГИС ОМИРБАЕВИЧ

Д И С С Е Р Т А Ц И Я

написанная с целью получения звания магистра

Тема: Решение систем конечномерных уравнений методом дифференциального спуска

Научный руководитель:
к.ф-м.н., проф. Отаров А.О.

Нукус-2012г.

СОДЕРЖАНИЕ

Введение

Развитие электронной вычислительной техники, создание алгоритмических языков программирования и обширного математического обеспечения ЭВМ позволяет широко использовать методы вычислительной математики при решении различного рода прикладных задач в науке, технике, производстве. Благодаря этому во многих случаях стало возможным отказаться от приближенной трактовки прикладных вопросов и перейти к решению задач в точной постановке. Это предполагает использование более глубоких специальных разделов математики (нелинейные дифференциальные уравнения, функциональный анализ, теоретико-вероятностные методы и др.).

Огромное быстродействие ЭВМ открывает новые широкие возможности для применения общих математических методов исследования в проблемах физики, механики, химии, астрономии, техники, экономики и многих других областей.

Велика роль ЭВМ для развития самой математики. Они используются для подсчета математических постоянных, для решений алгебраических, трансцендентных и дифференциальных уравнений; для решения сложнейших функциональных неравенств и т.п. Появились новые статистические методы машинного решения задач математической физики, стало возможным экспериментальное решение логических задач и многое другое [3, 8, 16].

Таким образом, создание ЭВМ знаменует решительный скачок по пути прогресса точных и технических наук.

Разумное использование современной вычислительной техники немыслимо без умелого применения методов приближенного и численного анализа. Этим и объясняется чрезвычайно возросший как у нас , так и за рубежом интерес к методам вычислительной математики [3, 4, 8, 12, 14, 15, 18].

Все это сделало весьма актуальным усовершенствованием и развитие численных и приближенных методов решения задач. Дело в том, что машина

способна выполнять очень большое, но конечное число операций. Поэтому точные предельные процессы решения задач, связанные с бесконечным числом операций, при работе на машине по необходимости должны быть заменены приближенными алгоритмами, содержащими лишь конечное число действий. Кроме того, машина обладает конечной памятью и может оперировать с цифрами лишь конечной длины. Поэтому промежуточные результаты округляются; в результате чего даже точный метод с конечным числом действий становится приближенным.

Высокая производительность ЭВМ существенным образом изменила подход к оценке того или иного вычислительного метода. Ценным оказывается тот метод, который является наиболее универсальным и который допускает простую реализацию на машинах. К таким методам относятся методы спуска, один из которых изучается в данной работе.

Актуальность темы работы. Методы приближенного решения задачи Коши для обыкновенных дифференциальных уравнений, как известно, хорошо разработаны [2,3,8,9,19]. Кроме того, современная вычислительная математика позволяет получить с большой точностью решения больших систем обыкновенных дифференциальных уравнений, если задан полный набор начальных условий. В силу этих причин, естественна попытка сведения вычислительных проблем к задачам Коши для обыкновенных дифференциальных уравнений (линейных или нелинейных). Поэтому, как только физическая, техническая или биологическая проблема сведена к задаче Коши для обыкновенных дифференциальных уравнений, можно считать, что ее полное решение близко к завершению. Если раньше стремились так упростить задачи, чтобы получить линейные функциональные уравнения, в настоящее время целью является преобразование вычислительных проблем к задачам Коши для обыкновенных дифференциальных уравнений.

Уровень разработанности задачи. Методы спуска, основанные на идее минимизации некоторого вещественного функционала $\varphi(x) \geq 0$, при

неудачном выборе начального приближения при наличии локальных минимумов $\varphi(x)$, могут не сходиться к решению. Кроме того, на каждом шаге итерационного процесса приходится находить минимум функции одной независимой переменной, что требует выполнения большой вычислительной работы.

В силу того, что в проекционно-дифференциальном методе функционалы $f_i(x)$, в отличии от функционала $\varphi(x) \geq 0$, не подчинены никаким дополнительным условиям, кроме условий, накладываемых теоремой существования и единственности решения задачи Коши (II. 4.2), отмеченные выше недостатки в этом методе отсутствуют.

В данной работе рассматривается новый вариант методов дифференциального спуска – проекционно-дифференциальный метод решения уравнений применительно к системам линейных и нелинейных конечномерных уравнений, обсуждаются вопросы численной реализации метода на современных вычислительных машинах, приводятся результаты численных экспериментов, проведенных с использованием компьютеров.

Цели и задачи магистерской диссертации. Основной целью диссертационной работы является изучение общих принципов построения методов дифференциального спуска, исследование их существующих вычислительных алгоритмов и разработка комплекса программ для их реализации.

В связи с целью диссертации, ее основными задачами являются:

- глубокое изучение теоретических положений методов дифференциального спуска решения уравнений по учебной и специальной литературе;
- руководствуясь общими принципами построения метода дифференциального спуска, усовершенствование известных и разработка новых вычислительных алгоритмов этого метода применительно к системам линейных и нелинейных конечномерных уравнений;

- изучение основ проекционно-дифференциального спуска, различных его вариантов, разработка новых вычислительных алгоритмов метода относительно линейных и нелинейных конечномерных уравнений;
- разработка комплекса программ для современных компьютеров, позволяющих реализовать различные варианты проекционно-дифференциального спуска.

Объект и предмет исследования. Объектом исследований является сравнение вычислительных алгоритмов известных итерационных методов решения систем линейных и нелинейных уравнений (методов простой итерации, Ньютона, спуска и др.) с алгоритмами итерационных методов, полученных на основе метода дифференциального спуска, выявление достоинства и недостатки этих методов друг от друга.

Предметом исследования является разработка новых вариантов метода дифференциального спуска, сходимость которых не зависит от выбора начального приближения к решению. Так применение итерационных методов (например, простых итераций, метода Ньютона, Эйткена-Стеффенсона [2, 3, 9, 11]) наталкивается на ту трудность, что далеко не всегда можно дать простой способ надежного выбора начального приближения. Между тем, от этого выбора зависит сходимость процесса. Поэтому проблема выбора начального приближения, обеспечивающего сходимость итерационного метода имеет первостепенное значение.

Научная новизна исследования. В ходе подготовки материалов и написания диссертационной работы, согласно ее цели и задачи, были усовершенствованы известные и разработаны новые вычислительные алгоритмы проекционно-дифференциального метода применительно к системам линейных и нелинейных конечномерных уравнений. В частности, выполнены научно-исследовательские работы следующего содержания:
- разработан новый вычислительный алгоритм проекционно-дифференциального метода применительно к системам линейных

алгебраических уравнений, имеющих специальную структуру (ленточными, почти треугольными, разреженными и др. матрицами);

- основывается применение проекционно-дифференциального метода для решения систем линейных алгебраических уравнений с плохообусловленными матрицами;
- построены два новые упрощенные вычислительные алгоритмы проекционно-дифференциального метода применительно к решению систем нелинейных уравнений.

Практическое значение исследования и апробация научных результатов. Научные результаты, полученные в диссертационной работе:

- могут служить основанием для дальнейшего продолжения научных исследований по данному направлению;
- могут быть новым материалом для специальных курсов, читаемых для бакалавров по направлению образования "Прикладная математика и информатика" и магистров по специальности "Прикладная математика и информационные технологии";
- могут быть материалом для написания выпускных квалификационных работ бакалаврами и диссертации магистрами;
- могут быть введены в учебники и учебные пособия, а также в специальную литературу по данному направлению.

Основные научные результаты, изученные в ходе выполнения исследований регулярно докладывались на научном семинаре кафедры "Прикладная математика и информатика", на научно-теоретических конференциях, проведенных в 2011-2012 гг. в Каракалпакском государственном университете и в КК отделении АН РУз. По результатам научных исследований также были опубликованы следующие научные статьи:

1. Отаров А.О., Отаров А.А., Нысанбаев Ш.О. Решение СЛАУ методом дифференциального спуска. Журн. "Вестник Каракалпакского государственного университета", №№3-4, 2011.

2. Отаров А.О., Нысанбаев Ш.О. Решение задачи линейного программирования методом дифференциального спуска. Журн. КК филиала АН РУз "Материалы XXII Республиканской научной конференции молодых ученых Каракалпакстана", 2012.

Структура и объем работы. Диссертационная работа состоит из двух глав, разделенных на параграфы, раздела заключения, списка использованной литературы, общий ее объем составляет 78 страниц (без приложений).

Во введении основывается актуальность темы диссертации, освещаются цели и задачи, объект и предмет исследований, излагаются вопросы научной новизны, практического значения исследования и апробации научных результатов.

В первой главе диссертации выводятся дифференциальные уравнения наискорейшего спуска и подъема, исследуются вопросы устойчивости уравнений спуска, излагается основная идея метода дифференциального спуска применительно к системам нелинейных уравнений и основывается применение этого метода для решения таких систем уравнений, а также применение метода для нахождения минимума нелинейных функций, приведено уравнение модифицированного дифференциального спуска, обсуждаются вопросы обусловленности систем нелинейных уравнений.

Во второй главе работы излагаются основные теоретические положения новых вариантов метода дифференциального спуска, разработанных сравнительно в последние годы. Первый метод основан на замене дифференциального уравнения спуска системой дифференциальных уравнений путем перехода от независимой переменной s к переменной $t = \varphi(x(s))$ такой, что решение $x(t)$ полученной системы дает в пределе при $t \to 0$ корень исходной системы нелинейных уравнений. Здесь $\varphi(x) \geq 0$ - такой функционал, что $\varphi(x^*) = 0$ тогда и только тогда, когда x^* - есть корень исходной системы нелинейных уравнений.

Второй метод основан на интегрировании дифференциального уравнения спуска с определенным выбором шага интегрирования.

Эти новые варианты метода дифференциального спуска применяются для нахождения всех действительных корней алгебраического уравнения и решения систем линейных алгебраических уравнений.

Излагаются также основные положения проекционно-дифференциального метода применительно к системе нелинейных уравнений в конечномерных евклидовых пространствах. В отличие от других методов дифференциального спуска, в этом методе дифференциального спуска уравнения нелинейной системы не сворачиваются в функционал. Спуск к решению системы происходит по линии, определяемой решениями задач Коши, соответствующих каждому уравнению этой системы.

Основываются три упрощенные вычислительные алгоритма проекционно-дифференциального метода, применение его для решения экстремальных задач, приводятся результаты численных экспериментов.

В заключении диссертации коротко перечислены ее основные результаты. В список литературы включены названия 24 работ, из которых 3 работы учебная литература, остальные 21 работ монографии и научные статьи.

В приложении к диссертации приведены распечатки комплекса программ, составленного на алгоритмическом языке Delphi и предназначенного для реализации трех новых вариантов алгоритма проекционно-дифференциального метода применительно к системам линейных и нелинейных уравнений, а также численных результатов при реализации этих программ для конкретных систем таких уравнений на компьютере.

I глава. Основные положения метода дифференциального спуска решения конечномерных уравнений

§ 1. Краткий обзор литературы

В самых разнообразных областях современной науки и техники все чаще встречаются задачи, для которых невозможно получить точное решение классическими методами или же оно получается в таком сложном виде, который совершенно неприемлем для практического использования. Так, например, к числу таких задач относятся задачи решения систем алгебраических уравнений с большим числом неизвестных, дифференциальных уравнений, которые не интегрируются в элементарных функциях и т.д. Особенно это относится к нелинейным задачам; даже в сравнительно простых видах таких задач, в большинстве случаев получение аналитического решения невозможно, и надежду на успешное решение задачи дают только численные методы. Поэтому в последние четыре десятилетия приближенные методы получили большое развитие. Одновременно с разработкой уже известных методов был предложен ряд новых.

Основные достоинства приближенных методов состоят в том, что зачастую они являются универсальными и эффективными, так как позволяют находить приближенное решение для широкого класса случаев и при применении требуют простых и вполне осуществимых вычислений.

В настоящее время разработка приближенных методов решения задач в основном протекает в двух направлениях: с одной стороны создаются более эффективные детерминированные способы решения задач, учитывающие специфические особенности вычислительных машин; с другой стороны, в практику успешно внедряются недетерминированные методы (статистические), основанные на случайных испытаниях (метод Монте-Карло [11, 23]).

Возможность осуществления способов решения, связанных с большим объемом вычислений существенным образом изменила подход к оценке того или иного вычислительного метода. Ценным оказывается наиболее универсальный метод, который возник в связи с общими идеями функционального анализа и который допускает простую реализацию на ЭВМ. К таким методам относятся методы спуска.

Исторически первыми среди методов спуска возникли релаксационные методы, основанные на координатном спуске.

Релаксационный метод известен еще со времен Гаусса, которым рекомендован как сам метод, так и многие его остроумные модификации.

Метод наискорейшего спуска применительно к системам линейных алгебраических уравнений предложен еще в 1847 году О. Коши.

Поскольку направление вектора $grad\varphi(x)$ есть направление наибыстрейшего убывания функционала $\varphi(x)$, то предпочтительней именно в этом направлении пытаться существенно уменьшить $\varphi(x)$. Учитывая это обстоятельство, в методе наискорейшего спуска от некоторой начальной точки $x_0(grad\varphi(x) \neq 0)$ осуществляется движение по прямой

$$x = x_0 - \lambda grad\varphi(x_0),\ \lambda > 0$$

до ближайшего к x_0 минимума функции

$$\varphi(x_0 - \lambda_0 grad(\varphi(x_0))$$

которая является функцией только одной вещественной переменной λ.

Если при $\lambda = \lambda_0$ достигается указанный минимум, то точка

$$x_1 = x_0 - \lambda_0 grad\varphi(x_0)$$

принимается за новое приближение и при этом будем иметь $\varphi(x_1) < \varphi(x_0)$. По x_1 можно построить таким же образом x_2 и т.д.

В результате описанного процесса приходим к последовательности $\{x_n\}, n = 1, 2, ...,$, элементы которой суть

$$x_n = x_{n-1} - \lambda_{n-1} grad\varphi(x_{n-1}) \qquad (1)$$

и которая в ряде случаев оказывается минимизирующей. Приведенная конструкция и составляет принципиальную схему метода наискорейшего спуска.

В дальнейшем расширение вычислительных работ, вызванных потребностями практики привело к видоизменениям методов спуска. был предложен ряд вариантов метода спуска.

В работах [5] был исследован один весьма общий вариант метода спуска и его применение к решению уравнений. Многие известные методы, такие, как метод Ньютона, метод Л.В.Канторовича, метод М.А,Красносельского-С.Г.Крейга и т.д. являются частным случаем этого метода [10, 11, 19].

Идея метода заключается в том, что минимизация функционала $\varphi(x) \geq 0$ происходит по линии, определяемой решением задачи Коши [4]

$$x'(t) = -\frac{a(x)}{(\varphi(x), a(x))},\ x_0 = x_0(t_0) \qquad (2)$$

где $t_0 = -\varphi(x_0)$, а $a(x)$ - оператор, действующий из банахова пространства $\bar{B}$ в B, $\varphi' = grad\varphi(x)$, а символом (φ', a) обозначено значение линейного относительно $a(x)$ функционала $\varphi'(x)$ на элементе $a(x) \in B, \varphi'(x) \in B^*$ - сопряженному с B пространству, $x \in B$.

В отличие от других методов дифференциального спуска задача Коши (2) сконструирована таким образом, что при некоторых условиях, накладываемых на $\varphi(x)$ и $a(x)$ (определяемых теоремой существования и единственности решения), имеет решение $x(t)$ на конечном интервале $[0, \bar{0}$ и $\lim_{t \to 0} x(t) = \alpha$ является решением уравнения

$$P(x) = 0 \qquad (3)$$

Функционал $\varphi(x)$ построен на $P(x)$ так, что при $x = \alpha$ он имеет нулевой минимум: $\varphi(\alpha) = 0$. В точке $x = \alpha$ функционал не обязательно предполагается дифференцируемым.

Кроме того, для дифференциального уравнения (2) известен первый интеграл

$$t = -\varphi(x) \qquad (4)$$

Соотношение (4) дает некоторую дополнительную информацию о поведении решения $x(t)$ и может быть использовано для контроля точности движения по рассматриваемой траектории при приближенном решении задачи (2).

Впоследствии в работах [5,14] было исследовано некоторое обобщение задачи (2) применительно к системе нелинейных уравнений в конечномерных евклидовых пространствах. В отличие от метода (2), в этом методе дифференциального спуска уравнения системы не сворачиваются в функционал. Спуск к решению системы происходит по линии, определяемой решениями задач Коши, соответствующих каждому уравнению этой системы.

В настоящей работе рассматривается применение этого варианта дифференциального спуска к решению систем алгебраических и трансцендентных уравнений.

§ 2. Дифференциальные уравнения наискорейшего спуска и подъема

Градиентные методы позволяют свести решения ряда математических задач к поиску локального экстремума функций многих переменных посредством решения системы обыкновенных дифференциальных уравнений. Эта важная особенность градиентного метода делает его применение особо привлекательным при программировании на современных компьютерах.

1. Вывод дифференциальных уравнений. Пусть необходимо найти локальный экстремум функционала (функции многих переменных) $J(x)$, где $x = x_1, x_2, ..., x_j, ..., x_n$. В дальнейшем всюду предполагается, что в области

определения функционала $J(x)$ локальный экстремум $x^* = \{x_1^*, x_2^*, ..., x_j^*, ..., x_n^*\}$ существует и единствен.

Поставленная задача будет решена, если мы укажем способ построения траектории $x(t) = \{x_1(t), x_2(t), ..., x_j(t), ..., x_n(t)\}$, которая обеспечивает попадание в x^* изображающей точки, начавшей движение из произвольной точки $x(0) = x_0 = \{x_{10}, x_{20}, ..., x_{j0}, ..., x_{n0}\}$ области определения $J(x)$.

Для определенности рассмотрим минимизацию функционала $J(x)$, т.е. поиск его минимума. В области определения можно положить бесконечное множество траекторий, соединяющих произвольно взятую точку x_0 с искомой x^*. Выберем из этих траекторий наилучшую и найдем соответствующее ей определяющее дифференциальное уравнение. Под наилучшей траекторией будем понимать ту, для которой dJ/dt - скорость убывания функционала – наибольшая. Выражение для скорости изменения функционала представляет собой скалярное произведение вектора

$$\frac{dJ}{dt} = \sum_{j=1}^{n} \frac{dJ}{dx_j} \frac{dJ}{dt}$$

вектора градиента

$$grad\, J(x) = \left\{ \frac{dJ}{dx_1}, \frac{dJ}{dx_2}, ..., \frac{dJ}{dx_j}, ..., \frac{dJ}{dx_n} \right\}$$

и вектора скорости изображающей точки

$$\frac{dx}{dt} = \left\{ \frac{dx_1}{dt}, \frac{dx_2}{dt}, ..., \frac{dx_j}{dt}, ..., \frac{dx_n}{dt} \right\}.$$

Всякое скалярное произведение принимает наибольшее абсолютное значение, когда сомножители коллинеарны. Это требует, чтобы соответствующие компоненты перемножаемых векторов были пропорциональны друг другу, т.е.

$$dx_j / dt = -\rho \cdot \partial J / \partial x_j, \qquad j = 1, 2, ..., n,$$

или в векторной форме

$$dx / dt = -\rho\, grad\, J(x),$$

где ρ - произвольное положительное число, играющее роль коэффициента пропорциональности компонент. Если к последним выражениям присоединить в качестве начальных условий координаты начальной точки x_0, то получим искомые определяющие дифференциальные уравнения в форме задачи Коши в векторной записи [6, 11]

$$\frac{dx}{dt} = -\rho\, grad\, J(x),\ x(0) = x_0, \tag{1}$$

или в скалярной –

$$\frac{dx_j}{dt} = -\rho \frac{\partial J}{\partial x_j},\ x_j(0) = x_{j0},\ \ j = 1,\ 2,\ ...,\ n\,. \tag{2}$$

Дифференциальные уравнения (1), (2) называют *дифференциальными уравнениями наискорейшего спуска*.

В тех случаях, когда отыскивается локальный максимум функционала $J(x)$, естественно потребовать максимальное возрастание $J(x)$ на траектории, что приводит к получению *дифференциальных уравнений наискорейшего подъема*, отличающихся от дифференциальных уравнений (1) и (2) лишь знаком перед коэффициентом ρ:

$$\frac{dx}{dt} = \rho\, grad\, J(x),\ x(0) = x_0 \tag{3}$$

$$\frac{dx_j}{dt} = \rho \frac{\partial J}{\partial x_j},\ x_j(0) = x_{j0},\ \ j = 1,\ 2,\ ...,\ n \tag{4}$$

Геометрическая интерпретация минимизации функционала $J(x)$ от двух переменных показана на рис. 1. Функционалу $J(x_1, x_2)$ в трехмерном пространстве соответствует поверхность, изображенная на рисунке. На плоскости $x_1 0 x_2$ показаны линии уровня функционала $J(x_1, x_2) = const$ и траектория изображающей точки $x_1(t), x_2(t)$, которая, начинаясь из точки x_0, приводит в точку x^*. Эта траектория для каждого момента времени имеет направление, ортогональное к линиям уровня.

Дифференциальные уравнения наискорейшего спуска (подъема) часто используют в несколько ином обобщенном виде:

$$\frac{dx}{dt} = \pm\rho(t)\, grad\, J(x),\ x(0) = x_0 \qquad (5)$$

или

$$\frac{dx}{dt} = \pm\rho(t,x)\, grad\, J(x),\ x(0) = x_0 \qquad (6)$$

В первом уравнении $\rho(t)$ - некоторая положительная функция, а во втором уравнении $\rho(t,x)$ - соответственно положительная функция, зависящая еще и от местоположения движущейся точки. В выборе этих функций допустим определенный произвол, что позволяет получить большое разнообразие дифференциальных уравнений спуска и отобрать из них те, которые обеспечивают желательный характер движения при достаточно простой реализации на ЭВМ.

Иногда вместо одной функции $\rho(t,x)$ вводят в уравнение (6) даже матрицу функций $\rho(t,x)$.

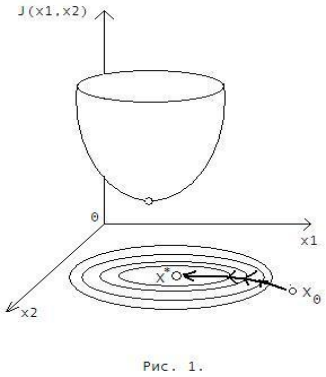


Рис. 1.

В простейшем случае это – диагональная матрица:

$$\begin{bmatrix} \rho_1(t,x) & 0 & \dots & 0 \\ 0 & \rho_2(t,x) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \rho_n(t,x) \end{bmatrix}, \qquad (7)$$

где все $\rho_j(t,x) > 0,\ j = 1,2,\dots,n$.

Дифференциальное уравнение

$$\frac{dx}{dt} = -\rho(t,x)\, grad\, J(x), \qquad (8)$$

строго говоря, уже нельзя называть уравнением наискорейшего спуска из-за нарушения условия коллинеарности векторов $\frac{dx}{dt}$ и $grad\, J(x)$, вызванного введением матрицы $\rho(t,x)$.

2. Устойчивость уравнений спуска. В только что рассмотренном примере решения дифференциальных уравнений спуска оказались асимптотически устойчивыми. Это явилось следствием того, что при выводе

уравнений спуска мы потребовали убывания функционала $J(x)$ вдоль траектории.

Устойчивость решений дифференциальных уравнений спуска (2) устанавливается в общем виде на основе так называемого второго метода Ляпунова. При этом не приходится прибегать к исследованию явного аналитического решения дифференциального уравнения. Метод Ляпунова применим к произвольным системам дифференциальных уравнений. Основу его составляет следующая теорема.

Если для дифференциального уравнения

$$\frac{dx}{dt} = F(t, x)$$

существует дифференциальная функция $V(x)$, *называемая функцией Ляпунова, удовлетворяющая в окрестности начала координат условиям:*

1) $V(x) \geq 0$, *причем* $V(x) = 0$ *лишь при* $x = 0$, *т.е. функция имеет строгий минимум в начале координат;*

2) $\frac{dV}{dt} = grad\, V \cdot F(t, x) \leq 0$,

то решение $x(t) = 0$ *устойчиво.*

Если, сверх того, $\frac{dV}{dt} < 0$, *то решение* $x(t) = 0$ *устойчиво асимптотически.*

Мы не будем здесь приводить строгое доказательство этой теоремы, а наметим лишь идею доказательства в случае асимптотической устойчивости.

Пусть $x(0) \neq 0$. Тогда $V(x) > 0$. Поскольку $dV / dt < 0$, следовательно, $V(x)$ убывает по t и стремится к нулю при $t \to \infty$. Однако, если $V(x) = 0$, то и $x = 0$, а решение асимптотически устойчиво.

Для дифференциальных уравнений спуска роль функции Ляпунова $V(x)$ играет минимизируемый функционал $J(x)$. Для уравнения спуска (2) всегда имеет место неположительная скорость изменения функционала вдоль траектории при $t \to \infty$, так как

$$\frac{dJ}{dt} = -\rho(t)\sum_{j=1}^{n}\frac{\partial J}{\partial x_j}\frac{dx_j}{dt} = -\rho(t)\,grad^2\,J \le 0\,.$$

Аналогично для дифференциальных уравнений спуска более общего вида

$$\frac{dx}{dt} = -\,\rho(t,x)\,grad^2\,J\,,$$

где $\rho(t,x)$ - диагональная матрица (7), имеем:

$$\frac{dJ}{dt} = -\sum_{j=1}^{n}\rho_j(t,x)\left(\frac{\partial J}{\partial x_j}\right)^2 \le 0\,.$$

Часто локальные экстремумы отыскиваются для функционалов, которые зависят от времени $J(x,t)$. Такие функционалы называют *нестационарными* в отличие от уже рассмотренных *стационарных* функционалов вида $J(x)$.

При программировании ЭВМ приходится иметь дело именно с нестационарными функционалами. Зависимость функционала от времени геометрически проявляется в том, что поверхность, заданная им, с течением времени меняет свою форму. Это сопровождается изменением места положения локальных экстремумов. Локальные экстремумы перемещаются. Изображающая точка, соответствующая системе дифференциальных уравнений спуска (подъема), следует за перемещающимся экстремумом и достигает его в благоприятных случаях. Когда же изображающая точка отстает от экстремума, говорят, что процесс слежения за экстремумом *неустойчив*.

Найдем условия, при которых минимизируемый функционал $J(x,t)$ уменьшается $(dJ/dt < 0)$ вдоль траектории, заданной уравнениями спуска (5)

$$\frac{dJ}{dt} = -\rho(t)\frac{\partial J}{\partial x_j}, \qquad j = 1,2,...,n\,.$$

Полная производная $J(x,t)$ по t

$$\frac{dJ}{dt} = \sum_{j=1}^{n}\frac{\partial J}{\partial x_j}\frac{dx_j}{dt} + \frac{\partial J}{\partial t} \qquad (10)$$

с учетом (5) будет равна

$$\frac{dJ}{dt} = -\rho(t) \sum_{j=1}^{n} \left(\frac{\partial J}{\partial x_j} \right)^2 + \frac{\partial J}{\partial t},$$

где $\sum_{j=1}^{n} \left(\frac{\partial J}{\partial x_j} \right)^2$ - положительная величина, которая обращается в нуль лишь в локальных экстремумах.

Функционал $J(x,t)$ будет уменьшаться вдоль траектории ($dJ/dt < 0$), если

$$\rho(t) \sum_{j=1}^{n} \left(\frac{\partial J}{\partial x_j} \right)^2 > \frac{\partial J}{\partial t} \tag{11}$$

Однако это условие всегда нарушается, как только изображающая точка оказывается в окрестности экстремума, где величина $\sum_{j=1}^{n} \left(\frac{\partial J}{\partial x_j} \right)^2$ достаточно мала. Размер этой окрестности определяется функциями $\rho(t)$ и $\partial J / \partial t$, и, в частности, может асимптотически стремиться к нулю. в этом случае будем говорить, что процесс оптимизации функционала $J(x,t)$ *асимптотически устойчив*.

§ 3. Основная идея методов спуска применительно к системам нелинейных уравнений.

В самом общем виде задачу решения какой-либо системы линейных или нелинейных уравнений можно представить как частный случай разрешения некоторой системы неявно заданных функций.

Пусть имеется система n уравнений с $n+m$ переменными:

$$\begin{aligned} &F_1(x_1, x_2, ..., x_n, y_1, y_2, ..., y_m) = 0, \\ &F_2(x_1, x_2, ..., x_n, y_1, y_2, ..., y_m) = 0, \\ &\text{.....} \\ &F_n(x_1, x_2, ..., x_n, y_1, y_2, ..., y_m) = 0, \end{aligned} \tag{1}$$

или, в векторной записи,

$$F(x, y) = 0 \qquad (2)$$

Система (1) определяет $x_1, x_2, ..., x_n$ как неявные функции заданных переменных $y_1, y_2, ..., y_m$. Как известно, условия существования совокупности неявных функций предполагают отличие от нуля определителя Якоби системы (1):

$$\begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_n} \end{bmatrix} \neq 0$$

В дальнейшем мы всюду будем считать, что это неравенство имеет место во всей области изменения переменных.

Если заданные переменные $y_1, y_2, ..., y_m$ представляют собой совокупность констант, то (1) превращается в обычную систему n уравнений с n неизвестными, которую можно записать в традиционной форме:

$$\begin{aligned} &F_1(x_1, x_2, ..., x_n) = 0, \\ &F_2(x_1, x_2, ..., x_n) = 0, \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ &F_n(x_1, x_2, ..., x_n) = 0, \end{aligned} \qquad (3)$$

Однако довольно часто, особенно в ряде практических задач управления, $y_1, y_2, ..., y_m$ не являются постоянными числами, а представляют собой некоторые известные функции времени $y_1(t), y_2(t), ..., y_m(t)$, которые характеризуют поведение объекта управления. Эти функции поступают на ЭВМ системы автоматического управления. Задача ЭВМ здесь состоит в воспроизведении функций $x_1(t), x_2(t), ..., x_n(t)$ и передаче их на органы управления объектом.

Далее мы в основном будем рассматривать системы типа (3). Однако изложенные ниже приемы решения (3) в ряде случаев без труда можно будет распространить и на более общую систему (1).

1. Решение систем уравнений общего вида. Как всегда, при программировании ЭВМ, решение исходной задачи стараются свести к решению некоторой эквивалентной системы дифференциальных уравнений. Применительно к системе

$$\begin{aligned} &F_1(x_1, x_2, ..., x_n) = 0, \\ &F_2(x_1, x_2, ..., x_n) = 0, \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ &F_n(x_1, x_2, ..., x_n) = 0, \end{aligned} \tag{1}$$

такой переход можно сделать достаточно разнообразными способами. Важно однако то, чтобы решения систем дифференциальных уравнений асимптотически стремились бы к решению системы (1). В противном же случае систему дифференциальных уравнений нельзя считать эквивалентной (1).

Наиболее эффективный прием решения (1) с точки зрения устойчивости получаемых эквивалентных дифференциальных уравнений базируется на градиентных методах минимизации функционала [6, 11, 23]

$$J(x) = \frac{1}{2}\sum_{i=1}^{n} F_i^2(x) \tag{2}$$

или

$$J(x) = \sum_{i=1}^{n} \left| F_i(x) \right|. \tag{3}$$

Соответствующие им дифференциальные уравнения наискорейшего спуска будут иметь вид:

$$\frac{dx_j}{dt} = -\rho(t) \sum_{i=1}^{n} F_i(x) \frac{\partial F_i}{\partial x_j}, \quad j = 1, 2, ..., n, \tag{4}$$

$$\frac{dx_j}{dt} = -\rho(t) \sum_{i=1}^{n} \frac{\partial F_i}{\partial x_j} sign\, F_i(x), \quad j = 1, 2, ..., n. \tag{5}$$

Выражения (4) и (5) часто оказываются сложными для аппаратурной реализации, что вызвано большим числом слагаемых в выражении функционалов (2) и (3).

В следующем методе, по существу тоже градиентном, вместо одного сложного функционала типа (2) или (3) проводится независимая минимизация n простых функционалов

$$J_i(x) = \frac{1}{2} F_i^2(x), \quad i = 1, 2, ..., n \tag{8}$$

или

$$J_i(x) = \left| F_i(x) \right|, \quad i = 1, 2, ..., n \tag{9}$$

При этом функционал $J_i(x)$ минимизируется по переменной x_j, в результате чего получается система дифференциальных уравнений

$$\frac{dx_j}{dt} = -\rho_j(t) \frac{\partial J_j}{\partial x_j} = -\rho_j(t) \frac{\partial J_j}{\partial x_j} F_j(x), \qquad j = 1, 2, ..., n, \tag{10}$$

или

$$\frac{dx_j}{dt} = -\rho_j(t) \frac{\partial J_j}{\partial x_j} = -\rho_j(t) \frac{\partial F_j}{\partial x_j} \, sign \, F_j(x), \qquad j = 1, 2, ..., n \tag{11}$$

Дифференциальные уравнения (10) и (11) гораздо легче реализовать на ЭВМ, чем (6) и (7). Однако, к сожалению, полученные уравнения могут иметь неустойчивые решения. Поясним причину этой неустойчивости. В процессе минимизации каждого i-го функционала производится разрешение i-го неявного уравнения $F_i(x) = 0$ системы (1) относительно переменной x_i. При этом полностью игнорируется влияние (чувствительность) всех остальных переменных x_j, $j \neq i$ на уравнение $F_i(x) = 0$. Возникающая в процессе минимизации ошибка $\varepsilon_i = F_i(x)$ может слабо изменяться при больших значениях x_i и очень сильно меняться при малых изменениях некоторых из остальных переменных x_j, $j \neq i$. Это и приводит к появлению неустойчивых решений или к возникновению нежелательных «паразитных» колебаний. Более формально в этом можно убедиться, рассмотрев скорость изменения величины функционала $J_i(x)$. Как известно, устойчивость решений (10) или (11) предполагает, что $dJ_i / dt < 0, \; i = 1, 2, ..., n$. Оценим

факторы, влияющие на знак скорости изменения функционала $J_i(x)$. В общем случае скорость можно записать выражением

$$\frac{dJ_i}{dt} = \sum_{j=1}^{n} \frac{\partial J_i}{\partial x_j} \frac{dx_j}{dt} = -\sum_{j=1}^{n} \rho_j(t) \frac{\partial J_i}{\partial x_i} \frac{\partial J_j}{\partial x_j}, \quad i = 1, 2, ..., n,$$

которое запишем в несколько измененном виде:

$$\frac{dJ_i}{dt} = -\rho_i(t) \left(\frac{\partial J_i}{\partial x_i} \right)^2 - \sum_{j \neq i}^{n} \rho_j(t) \frac{\partial J_i}{\partial x_j} \frac{\partial J_j}{\partial x_j}, \quad i = 1, 2, ..., n. \tag{12}$$

Видно, что знак dJ_i / dt не определен. Все зависит от членов, стоящих под знаком суммы в (12). Однако в частном случае, если система (3) оказалась такой, что

$$\rho_i(t) \left(\frac{\partial J_i}{\partial x_i} \right)^2 > \left| \sum_{j \neq i}^{n} \rho_j(t) \frac{\partial J_i}{\partial x_j} \frac{\partial J_j}{\partial x_j} \right|, \quad i = 1, 2, ..., n, \tag{13}$$

то решения систем дифференциальных уравнений (10) и (11) асимптотически стремятся к решению (3). Выражения $\partial J_j / \partial x_j$ пропорциональны частным производным $\partial F_j / \partial x_j$. Поэтому, если есть возможность, то целесообразно j-е уравнение $F_j(x) = 0$ системы (3) разрешить относительно той переменной x_k, по которой частная производная $\partial F_j / \partial x_k$ - наибольшая. Так, скажем, первое уравнение $F_1(x) = 0$ будем разрешать относительно x_2, если $\partial F_1 / \partial x_2 > \partial F_1 / \partial x_j$ для всех $j \neq 2$. Но даже при этом условии не может быть всегда гарантирован успех.

Существует еще один прием сведения решения системы (3) к решению системы дифференциальных уравнений. Поясним его смысл на простом примере. Пусть требуется решить уравнение $x = \exp(-x)$. Запишем его в неявной форме

$$-x + \exp(-x) = 0 \tag{14}$$

и для получения решения на ЭВМ заменим его дифференциальным уравнением

$$\frac{dx}{dt} = -x + \exp(-x). \tag{15}$$

Основанием для перехода от (14) к (15) могут служить следующие соображения. Если решение дифференциального уравнения (15) при некотором произвольно взятом начальном значении $x(0) = x_0$ асимптотически стремится к некоторой постоянной величине x^*, то эта величина и есть искомый корень уравнения (14). Действительно, при этом производная dx/dt стремится к нулю, и следовательно, дифференциальное уравнение (15) превращается в конечное $-x + \exp(-x) = 0$. Главная неприятность этого метода в том, что априори в общем случае нет никакой уверенности в асимптотической устойчивости полученных таким образом дифференциальных уравнений. Действительно, даже в нашем простом примере достаточно было бы уравнение (14) записать в виде $x - \exp(-x) = 0$, как это сразу привело бы к дифференциальному уравнению $dx/dt = x - \exp(-x)$ с неустойчивым решением.

Применительно к системе (3) излагаемый способ приводит к системе дифференциальных уравнений

$$\frac{dx_i}{dt} = \pm F_j(x_1, x_2, ..., x_n), \qquad j = 1, 2, ..., n, \quad i = 1, 2, ..., n. \tag{16}$$

Мы сознательно взяли разные индексы у переменных $x_i(t)$ и уравнений $F_j(x) = 0$ с тем, чтобы подчеркнуть необходимость подбора этих индексов i и j, а также знаков правых частей уравнений (16). Подбор индексов и знаков правых частей осуществляется путем пробных воспроизведений решений системы (16) и отбора из них устойчивых решений. Количество возможных пробных вариантов факториально зависит от $2n$, поэтому число неудачных проб может быть значительным.

Последний метод можно существенно улучшить, если над системой (1) предварительно совершить некоторые преобразования. В ряде задач, связанных с автоматическим управлением, исходную систему (1) предварительно слева умножают на обратную матрицу Якоби этой системы, что приводит к дифференциальному уравнению

$$\frac{dx}{dt} = -\left(\frac{\partial F}{\partial x}\right)^{-1} F(x, y(t)), \qquad (17)$$

где $(\partial F / \partial x)$ – матрица Якоби системы (1).

В задачах автоматического слежения за меняющимися корнями системы (1) благодаря такому преобразованию векторное уравнение (17) по существу распадается на n скалярных линейных практически независимых друг от друга дифференциальных уравнений. Покажем это, для чего рассмотрим установившийся процесс слежения за меняющимися корнями. В установившемся режиме слежения вектор ошибок $\varepsilon = F(x, y(t))$ достаточно мал и приближенно может быть представлен в виде первых двух членов ряда Тейлора

$$\varepsilon = F(x, y(t)) \approx F[x^*, y(t)] + (\partial F / \partial x)\Delta x + ..., \qquad (18)$$

где $\Delta x = x(t) - x^*(t)$ - вектор-ошибка между точным значением и значением, выработанным ЭВМ.

Учитывая, что $F[x^*, y(t)] = 0$, после подстановки (18) в (17) имеем

$$\frac{dx}{dt} = -\left(\frac{\partial F}{\partial x}\right)^{-1}\left(\frac{\partial F}{\partial x}\right)[x(t) - x^*(t)]$$

или

$$dx/dt = -E[x(t) - x^*(t)], \qquad (19)$$

где E - единичная матрица.

Таким образом, мы доказали, что в установившемся режиме слежения за корнями системы (1) уравнение (17) действительно распалось на систему несвязанных между собой устойчивых дифференциальных уравнений. Однако, когда вектор-ошибка $\Delta x(t)$ еще достаточно велика (это бывает в начальные, еще неустановившиеся периоды слежения), процессы, описываемые (17), могут отличаться сильной зависимостью одних переменных от других, т.е. $x_j(t)$ от $x_i(t)$, и, естественно, уравнение (19) не имеет места. Эта взаимосвязанность переменных при больших $\Delta x(t)$ приводит к возникновению колебаний в начальные периоды слежения.

Дифференциальное уравнение (17) интересно еще в одном отношении. Оно представляет собой непрерывный аналог известного *итерационного метода Ньютона* численного решения системы нелинейных уравнений.

2. Решение систем линейных алгебраических уравнений. Система линейных алгебраических уравнений, записанная скалярно:

$$\begin{aligned} F_1(x) &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n - b_1 = 0, \\ F_2(x) &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n - b_2 = 0, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ F_n(x) &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n - b_n = 0 \end{aligned} \qquad (20)$$

или в векторной форме:

$$F(x) = Ax - b = 0, \qquad (21)$$

представляет собой частный и наиболее простой вариант системы (3). Поэтому все изложенные ранее приемы перехода от (3) к системам эквивалентных дифференциальных уравнений применимы и к системе (20).

Но на первый взгляд может показаться, что для получения решения линейной системы алгебраических уравнений (20) вовсе и необязательно переходить к каким-то эквивалентным дифференциальным уравнениям. Может создаться впечатление, что для аппаратурного решения (20) достаточно располагать лишь сумматорами и инвенторами. Число сумматоров должно быть равно n - числу уравнений, а количество необходимых инвенторов определяется числом отрицательных коэффициентов среди a_{ij}. При этом первый сумматор, реализуя первое уравнение $F_1(x) = 0$, разрешает его относительно переменной x_1; второй сумматор разрешает второе уравнение $F_2(x) = 0$ относительно переменной x_2 и т.д. К сожалению, оказывается, что практического значения описанный способ не имеет и в общем случае не приводит к получению желаемого результата. Причина здесь в том, что сумматоры ЭВМ воспроизводят сумму приближенно, и, как было отмечено ранее, выходная переменная сумматора зависит от производной выходной переменной по времени. Поэтому описанный способ решения системы (20) по существу приводит к

воспроизведению решений некоторой системы дифференциальных уравнений с неизвестными малыми параметрами. Эти решения часто оказываются неустойчивыми и характеризуются быстрыми изменениями машинных переменных. Проконтролировать устойчивость решений и, тем более, изменить параметры дифференциальных уравнений для достижения устойчивости не представляется возможным. Замена системы алгебраических уравнений системой эквивалентных дифференциальных уравнений существенно меняет дело. Реализация решений эквивалентной системы дифференциальных уравнений, конечно, не уничтожает малые параметры сумматоров, а лишь уменьшает их влияние и тем сильнее, чем медленнее воспроизводится на ЭВМ решение эквивалентной системы дифференциальных уравнений. Это подтверждается данными практики, а для ряда систем алгебраических уравнений удается доказать строго.

Рассмотрим подробнее основные методы перехода от линейной алгебраической системы (20) к системам эквивалентных дифференциальных уравнений. Самый универсальный метод основан на минимизации функционала

$$J(x) = \frac{1}{2}(Ax - b)^T (Ax - b). \qquad (22)$$

Функционал (22) является векторной записью (4). Символ T означает знак транспонирования. Функционал имеет минимум, равный нулю. Этот минимум соответствует решению исходной системы уравнений. По выражению (22) легко находится дифференциальное уравнение спуска, эквивалентное системе (21):

$$dx/dt = -\rho\, grad\, J(x),$$

где

$$grad\, J(x) = \frac{1}{2}\left[A^T (Ax - b) + (Ax - b)^T A\right] = A^T (Ax - b).$$

Теперь в окончательном виде получаем дифференциальное уравнение

$$\frac{dx}{dt} = -\rho\left[A^T Ax - A^T b\right]. \qquad (23)$$

Это уравнение всегда имеет асимптотически устойчивое решение $x(t)$. Некоторое неудобство при программировании на ЭВМ доставляет необходимость умножения исходной системы (21) на транспонированную матрицу A^T, что может сопровождаться значительными вычислениями.

Покоординатные градиентные методы, основанные на минимизации функционала $J_i(x) = |F_i(x)|$ или $J_i(x) = \frac{1}{2} F_i^2(x),\ i = 1, 2, ..., n$ не требуют предварительных затрат вычислительного труда, но они могут привести к получению неустойчивой системы дифференциальных уравнений.

Остановимся сначала на решении системы уравнений (20) посредством минимизации функционалов вида $J_i(x) = |F_i(x)|,\ i = 1, 2, ..., n$. Этому соответствует система дифференциальных уравнений спуска

$$dx / dt = -\rho_j a_{jj}\, sign\, F_j(x), \quad j = 1, 2, ..., n, \tag{24}$$

где ρ_i - произвольное число. В частности, если $\rho = \frac{1}{|a_{jj}|}$, то получаем удобную для реализации на ЭВМ систему дифференциальных уравнений

$$dx / dt = -sign\, a_{jj} F_j(x), \quad j = 1, 2, ..., n. \tag{25}$$

Устойчивость решений систем дифференциальных уравнений (24) и (25) определяется свойствами матрицы A системы (20). Условия устойчивости достаточно просто установить, если найти отношения, которым должны удовлетворять элементы матрицы A, для того чтобы скорость изменения всех минимизируемых функционалов все время оставалась бы отрицательной, т.е. $dJ_i / dt < 0,\ i = 1, 2, ..., n$.

§4. Обоснования применения метода дифференциального спуска для решения систем нелинейных уравнений

Задачу решения системы нелинейных уравнений $g(x) = d$ можно заменить вариационной задачей на разыскание минимума некоторой

функции n переменных. Существует бесконечно много функций, минимумы которых, или часть из них, достигаются на решениях системы $g(x) = d$. Мы будем рассматривать одну из простейших функций такого типа, а именно, следующую

$$f(x) = \|g(x) - d\|^2 \tag{1}$$

Норму в (1) можно брать любую, но мы, для определенности, рассмотрим только евклидову норму, т.е., если $x = (x_1, x_2, ..., x_n)$ и $y = (y_1, y_2, ..., y_n)$ два вектора, принадлежащие n-мерному евклидовому пространству R^n, то скалярное произведение этих векторов и норма вектора соответственно определяются по формулам [9, 10]:

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i\, y_i\,; \quad \|x\| = \langle x, x \rangle^{1/2} = \left[\sum_{i=1}^{n} x_i^2 \right]^{1/2}$$

Очевидно, что если x^* - решение системы $g(x) = d$, то $f(x)$ в точке $x = x^*$ достигает максимума, равного нулю. Верно и обратное: всякий нулевой минимум функции (1) достигается в точке, являющейся решением системы $g(x) = d$. Но функция (1) наряду с нулевыми может иметь и другие локальные минимумы.

Градиентом функции $f(x)$ назовем вектор, компонентами которого являются частные производные от функции $f(x)$ по переменным $x_1, x_2, ..., x_n$:

$$grad\ f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n} \right)$$

Дифференцируя выражение (1), находим

$$\frac{\partial f}{\partial x_i} = \frac{\partial}{\partial x_i} \langle g(x) - d,\ g(x) - d \rangle = 2 \left(\frac{\partial g}{\partial x_i},\ g(x) - d \right) = \sum_{k=1}^{n} \frac{\partial g_k}{\partial x_i} \left[g(x) - d \right]_k$$

$$(i = 1,\ 2,\ ...,\ n)$$

Отсюда заключаем, что градиент функции $f(x)$ можно представить в следующем виде:

$$grad\ f(x) = \Gamma^*(x) \left[g(x) - d \right] \tag{2}$$

где $\Gamma^*(x)$ - транспонированная матрица Якоби вектор-функции $g(x)$. Поскольку $grad\, f(x)$ ортогональна плоскости к поверхности уровня функции $f(x)$, т.е. к поверхности

$$\|g(x)-d\|^2 = c = const,$$

то направление вектора-градиента является направлением наибыстрейшего изменения функции $f(x)$ в точке x, причем положительное направление вектора $grad\, f(x)$ соответствует возрастанию функции $f(x)$, отрицательное – убыванию этой функции. Очевидно, что всякий вектор y, составляющий с вектором $-grad\, f(x)$ острый угол, будет также направлен в сторону убывания функции $f(x)$. Аналитически это условие запишется так:

$$(y, grad\, f(x)) < 0$$

В частности, можно принять $y = -grad\, f(x)$.

Идея методов спуска состоит в том, что, задавшись начальной точкой x^0, осуществляют движение от нее по некоторой кривой или ломаной в сторону убывания функции $f(x)$. Движение происходит до тех пор, пока точка не достигнет минимума функции $f(x)$. Если этот минимум нулевой, то тем самым, мы находим решение системы $g(x)-d$, если он положительный, то спуск необходимо производить с другой начальной точки.

Линию, по которой проходит точка x к минимуму функции $f(x)$, будем называть линией спуска. В параметрическом виде уравнение линии спуска запишем так: $x = x(t)$. Нетрудно получить дифференциальные уравнения, определяющие линию спуска. Эти уравнения имеют вид

$$\frac{dx}{dt} = y(x), \qquad (3)$$

где $y(x)$ - любой вектор, удовлетворяющий условию

$$(y(x), grad\, f(x)) = (y(x), \Gamma^*(x)\,|g(x)-d) < 0 \qquad (4)$$

при $grad\, f(x) \neq 0$. Покажем, что действительно, с увеличением t функция $f[x(t)]$ стремится к своему стационарному значению, т.е. к тому значению, где $grad\, f(x) = 0$. Обозначим $x_1 = x(t_1)$, $x_2 = x(t_2)$. Тогда из (3) находим

$$x_2 = x_1 + \int_{t_1}^{t_2} y(x)dt$$

Предположим, что вектор $y(x)$ непрерывен по x. В этом случае имеем

$$f(x_2) = f(x_1 + \int_{t_1}^{t_2} y(x)dt) = f(x_1) + (gradf(x_1), \int_{t_1}^{t_2} y(x)dt) + O(|t_2 - t_1|^2) =$$

$$= f(x_1) + \int_{t_1}^{t_2} (gradf(x_1), y(x))dt + O(|t_2 - t_1|^2).$$

Если $grad\ f(x) \neq 0$, то $(gradf(x_1), y(x)) < 0$, поэтому в силу непрерывности $y(x)$ на промежутке $[t_1, t_2]$ достаточно малой длины будет также выполняться неравенство $(gradf(x_1), y(x)) < 0$. Следовательно, при условии ограниченности вторых производных от функции $f(x)$ по переменным x_i найдется такой достаточно малый промежуток $[t_1, t_2]$, на котором будет справедливо неравенство

$$f(x) < f(x_1) \qquad x = x(t),\ t \in [t_1, t_2] \tag{5}$$

Отсюда заключаем, что функция $f(x)$ с возрастанием t будет стремиться к такому значению, при котором $grad\ f(x) = 0$. Это значение, в частности, может быть минимумом функции $f(x)$. Покажем теперь, что при некоторых дополнительных условиях функция $x(t)$ также стремится к определенному пределу при увеличении t. С этой целью выберем последовательность значений $t_0 < t_1 < ... < t_i < ...$ и соответствующую ей последовательность точек x_i, для которых $f(x_{i+1}) < f(x_i), (i = 0,1,2,...)$ - выбранная последовательность значений t_i, на которых функция $f[x(t_i)]$ стремится к стационарному значению, может быть как ограниченной, так и неограниченной. В первом случае последовательность t_i, как монотонная и ограниченная, имеет предел t^*. Если $\|y(x)\| \le a$, то в этом случае

$$\|x_{m+p} - x_m\| = \left\| \int_{t_m}^{t_{m+p}} y(x)dt \right\| \le a(t_{m+p} - t_m) \tag{6}$$

Поскольку правая часть неравенства (6) имеет своим пределом нуль, то отсюда следует сходимость последовательности точек x_m к некоторому

пределу x^*. В силу непрерывности функции $f(x)$, точка x^* будет ее стационарной точкой, т.е. $grad\, f(x^*) = 0$. Рассмотрим второй случай, когда последовательность t_i неограниченна. Предположим в этом случае, что выполняется неравенство

$$\left| f(x_{m+p}) - f(x_m) \right| \geq a \left\| x_{m+p} - x_m \right\|^{1+\alpha}, \ (\alpha > 0) \tag{7}$$

где x_m, x_{m+p} - два любых элемента последовательности x_i. Так как последовательность $f(x_m)$ сходится, то из (7) следует, что $\left\| x_{m+p} - x_m \right\| \to 0$ при $m \to \infty$, т.е. последовательность x_i также сходится. Предел x^* последовательности x_i является стационарной точкой функции $f(x)$. Таким образом, нами в основных чертах доказана следующая теорема [6].

Теорема. Пусть выполнены условия:

1) поверхность $\|g(x) - d\|^2 = \|g(x0) - d\|^2$ замкнутая и ограничивает область S_0, во внутренних точках которой $f(x) < f(x_0)$;
2) вектор $y(x)$ однозначно определен в каждой точке области S_0 и удовлетворяет условиям

$$(y(x), grad\ f(x)) < 0 \text{ при } gradf(x) \neq 0$$

$$a_1 \|gradf(x)\| \leq \|y(x)\| \leq a_2,$$

$$\|y(x_2) - y(x_1)\| \leq c \|x_1 - x_2\| \quad \text{при} \quad x_1, x_2 \in S_0$$

Тогда существует единственное решение задачи Коши

$$\frac{dx}{dt} = y(x), \ x(0) = x_0 \tag{8}$$

которое определяет линию спуска $x = x(t), t \in [0, t^*)$, соединяющую точку x_0 со стационарной точкой функции $f(x)$. Если $t^* = \infty$, то следует предположить дополнительно выполнение неравенства

$$\left| f(x_2) - f(x_1) \right| \geq b \left\| x_2 - x_1 \right\|^{1+\alpha}, \ (\alpha > 0)$$

при $x_1, x_2 \in S_0$.

Чтобы получить линию спуска с наиболее быстрым убыванием функции $f(x) = \|g(x) - d\|^2$, необходимо положить в (8) $y(x) = -\Gamma^*(x)[g(x) - d]$.

Подчеркнем, что это убывание будет наиболее быстрым только в локальном смысле, т.е. в окрестности текущей точки линии спуска. В целом же может оказаться, что более короткой линией, соединяющей точку x_0 со стационарной точкой функции $f(x)$, будет линия с другим выбором $y(x)$. Информацию, позволяющую сделать выбор такого $y(x)$ можно получить только из исследования поведения функции $f(x)$ во всей области S_0 либо на значительной ее части.

Таким образом, задача разыскания минимума функции сведена нами к задаче численного интегрирования задачи Коши (8), решение которой позволяет определить стационарную точку функции $f(x)$, т.е. ту точку, где выполняются необходимые условия минимума. Поскольку нашей конечной целью является разыскание решения системы $g(x) = d$, а не определение точек минимума, то нам нет необходимости выяснить, будет ли найденная стационарная точка x^* точкой минимума.

Достаточно проверить, обращается ли в нуль функция $f(x)$ в точке x^*. Если $f(x^*) \neq 0$, то следует выбрать новое начальное значение решения задачи Коши в (8), снова построить линию спуска, решая задачу (8).

§5. Применение метода дифференциального спуска для нахождения минимума нелинейных функций

В этом параграфе на основе работы [6] будут изложены основные положения метода дифференциального спуска в n-мерном евклидовом пространстве R^n применительно к задачам нахождения минимума заданной функции и решения систем нелинейных уравнений.

1. Нахождение точки минимума функции. Пусть в области G с границей ∂G евклидова n-мерного пространства R^n задана функция $u(x_1, x_2, ..., x_n)$ класса $C^2(G)$, причем

$$\min_{\partial G} \varphi(x) > \min_{G} \varphi(x) \geq 0 \qquad (1)$$

необходимо найти точки минимума функции внутри области C.

Рассмотрим траекторию системы обыкновенных дифференциальных уравнений [6]

$$\frac{dx}{dt} = -\varphi \frac{grad\,\varphi}{|grad\,\varphi|^2} \qquad (2)$$

Точку x_0 будем, как обычно, называть критической, если $grad\,\varphi|_{x=x_0} = 0$. Так как в некритических точках правая часть системы (2) непрерывно дифференцируема, то через каждую некритическую точку проходит единственное гладкое решение системы (2).

Справедливы следующие теоремы [6]

Теорема 1. Для любого решения $x(t)$ системы (2) выполняется тождество

$$\varphi(x(\tau)) = \varphi(x(t))e^{-(\tau - t)} \qquad (3)$$

Доказательство. Имеем

$$\frac{d\varphi}{dt} = (grad\varphi, \frac{dt}{dx}) = -\frac{\varphi}{|grad\varphi|^2}(grad\varphi, grad\varphi) = -\varphi,$$

откуда следует (3).

Теорема 2. Если во всех точках круга $K_{\rho,x_0} : |x - x_0| < \rho$, принадлежащего G, выполнены неравенства

$$0 \leq \varphi(x) \leq K|grad\varphi|^m \quad (m > 1,\ k - const), \qquad (4)$$

$$\varphi(x_0) \leq H\rho^{m(m-1)};\ H = \left[\frac{m-1}{2mK^{1/m}}\right]^{m/(m-1)} \qquad (5)$$

то, во-первых, функция $\varphi(x)$ и K_{ρ,x_0} имеет точку минимума ξ, причем $\varphi(\xi) = 0$, во-вторых, система (2) имеет решение $x(t)$ при $0 \leq t < \infty$, для которого $x(0) = x_0$ и

$$\lim_{t\to\infty} x(t) = \xi \qquad (6)$$

$$|x(\tau) - x(t)| \leq \frac{\rho}{2}(e^{-\beta t} - e^{-\beta t});\ \ (\beta = \frac{m-1}{m}, \tau > t) \qquad (7)$$

В частности,

$$|x(t)-\xi| \le \frac{\rho}{2} e^{-\beta t} \tag{8}$$

Доказательство. Из (4) следует, что в круге K_{ρ,x_0} во всех критических точках $\varphi(x)=0$, и поэтому все критические точки – точки минимума. Пусть $x(t)$ - траектория системы (2) для которой $x(0)=x_0$ определенная при $0 \le t < T$. Тогда

$$|x(\tau)-x(t)| = \left|\int_t^\tau \frac{dx}{dt}dt\right| \le \int_t^\tau \left|\frac{dx}{dt}\right|dt = \int_t^\tau \frac{\varphi dt}{|grad\varphi|};$$

Используя (3) – (5), получаем

$$|x(\tau)-x(t)| \le K^{1/m}\int_t^\tau \varphi^{1-1/m}dt = K^{1/m}\int_t^\tau \left(\varphi(x_0)e^{-t}\right)^{1-1/m}dt = K^{1/m}\left(\varphi(x_0)\right)^{\beta}\frac{1}{\beta}(e^{-\beta t}-e^{-\beta\tau}).$$

Так как согласно (5)

$$K^{1/m}\left(\varphi(x_0)\right)^{\beta}\frac{1}{\beta} \le \frac{\rho}{2},$$

то

$$|x(\tau)-x(t)| \le \frac{\rho}{2}(e^{-\beta t}-e^{-\beta\tau}).$$

Отсюда следует, что существует $\lim\limits_{t\to T} x(t) = x_T$. Если $T<\infty$, то $\varphi(x_T)=\varphi(x_0)e^{-T}$ и, следовательно, $U(x_T)\ne 0$. Точка некритическая. Поэтому траектория $x(t)$ может быть продолжена для значения параметра $0 \le t < \infty$. При $t\to\infty$ существует

$$\lim_{t\to\infty} x(t) = \xi,\ U(\xi)=0.$$

Неравенство (8) следует из (7). Функция $\varphi(x)$ может иметь точки минимума, в которых значение функции положительно.

Следствие. Если во всех точках некоторого круга K_{ρ,x_0} выполнены неравенства

$$0 \le \varphi(x) - C \le K\left|grad\,\varphi\right|^m;\quad m>1 \tag{9}$$

$$\varphi(x_0) - C \le H\rho^{m/(m-1)};\quad H=\left[\frac{m-1}{2mK^{1/m}}\right] \tag{10}$$

то в K_{ρ,x_0} функция $\varphi(x)$ имеет точку минимума $\varphi(\xi)=C$.

Решение $x(t)$ $(0\le t<T)$ системы (2), удовлетворяющее начальному условию $x(0)=x_0$, стабилизируется к ξ, т.е.

$$\lim_{t\to T} x(t)=\xi,\ \ |x(t_2)-x(t_1)|\le\frac{\rho}{2}(e^{-\beta S1}-e^{-\beta S2}) \tag{11}$$

для $0<t_1<t_2<T$. Здесь

$$S=\ln\frac{U_0-C}{U_0e^{-t}-C};\ \ \beta=\frac{m-1}{m};\ \ T=\ln\frac{U_0}{C}; \tag{12}$$

Доказательство. На траектории $x(t)$ с началом в точке x_0 введем новый параметр S по формуле (12) и заменим функцию $v(x)=\varphi(x)-C$. Тогда система (2) перейдет в систему

$$\frac{dx}{dS}=-v\frac{grad\,v}{|grad\,v|^2}. \tag{13}$$

В круге K_{ρ,x_0} функция $v(x)$ удовлетворяет условиям (4) и (5). Поэтому для нее справедливы утверждения теоремы 2.

2. Решение систем нелинейных уравнений. Пусть дана система нелинейных уравнений

$$\begin{cases} f_1(x_1,...,x_n)=0 \\ ----------\\ f_n(x_1,...,x_n)=0 \end{cases} \quad \text{или} \quad F(x)=0 \tag{14}$$

Предположим, что функции $f_1,...,f_n$ определены и принадлежат к классу $C^2(G)$ в области G. Необходимо найти решение системы (14) внутри области G.

Рассмотрим функцию

$$\varphi(x)=|F(x)|^2 \tag{15}$$

и составленную для нее систему дифференциального спуска

$$\frac{dx}{dt}=-2|F|^2\frac{grad\,|F|^2}{\left|grad\,|F|^2\right|^2}$$

или

$$\frac{dx_i}{dt} = -\left(\sum_{j=1}^{n} f_j^2 \frac{\sum_{j=1}^{n} f_j \partial f_j / \partial x_j}{\sum_{k=1}^{n} \left(\sum_{j=1}^{n} f_j \partial x_i / \partial x_k \right)^2} \right) \tag{16}$$

Имеет место следующая теорема [6]

Теорема 3. Для любого решения $x(t)$ системы (16) справедливо тождество

$$|F(x(\tau))| = |F(x(t))| e^{-(\tau - t)} \tag{17}$$

Если в круге K_{ρ, x_0} выполнены неравенства

$$|F(x)| \le K |grad |F||^m \text{ при } |F(x)| \ne 0,\ n > 1, \tag{18}$$

$$|F(x_0)| \le H \rho^{m/(m-1)};\ \ H = \left[\frac{m-1}{2mK^{1/m}} \right]^{m/(m-1)} \tag{19}$$

то в круге K_{ρ, x_0} система (14) имеет решение ξ. Существует предел решения $x(t)$ $(0 \le t < \infty)$ системы (16), проходящего через точку

$$\lim_{t \to \infty} x(t) = \xi \tag{20}$$

и

$$|x(\tau) - x(t)| \le \frac{\rho}{2} \left(e^{-\beta t} - e^{-\beta \tau} \right) \text{ при } \beta = \frac{m}{m-1};\ \tau > t \tag{21}$$

В частности,

$$|x(t) - \xi| \le \frac{\rho}{2} e^{-\beta t}. \tag{22}$$

Теорема доказывается аналогично теореме 2.

3. Модифицированный спуск. Для ускорения сходимости иногда удобно рассмотреть более общее уравнение

$$\frac{dx}{dt} = -[W(\cdot)]^{-1} (grad\ \varphi(x)] \tag{23}$$

в котором $W(\cdot)$ - положительно определенный матричный оператор, вообще говоря, нелинейный.

Справедлива следующая [ ]

Теорема 4. Для решения системы дифференциальных уравнений (23) справедливо тождество

$$\int_t^{\tau}\left(W\left(\frac{dx}{dt};\frac{dx}{dt}\right)\right)dt=\varphi(x(t))-\varphi(x(\tau)) \qquad (24)$$

т.е. траектория $x(t)$ является кривой строго спуска для функции $\varphi(x)$.

Действительно,

$$\frac{d\varphi(x(t))}{dt}=(grad\,\varphi(x),\frac{dx}{dt})=-(W(\cdot)\frac{dx}{dt},\frac{dx}{dt}). \qquad (25)$$

откуда следует (24). В частном случае теорема справедлива для $\varphi(x)=|F(x)|^2$.

Практическое вычисление траекторий системы (2) или (16) на ЭВМ проводится хорошо известными алгоритмами численного интегрирования систем обыкновенных дифференциальных уравнений (методами Рунге-Кутта, Адамса и т.д. [2, 3, 8, 9, 19]). При этом тождества (3) или (17) служат для контроля точности и движения по траектории.

4. Обусловленность нелинейных систем. Пусть ξ - невырожденный простой корень системы (14). В окрестности ξ для функции $\varphi(x)=|F(x)|^2$ справедливо общее представление

$$\varphi(x)=(H\eta,\eta)+o(|\eta|^2),$$

где $\eta=x-\xi$ - гессиан функция $\varphi(x)$ в точке ξ

$$H=\left\|\frac{1}{2}\frac{\partial^2\varphi}{\partial x_i\partial x_j}\right\|(\xi)$$

Если матрица H положительно определенная, то собственные значения этой матрицы

$$0<\lambda_1\le\ldots\le\lambda_n.$$

Поверхности уровня $(H\eta,\eta)=const$ (заметим, что $U(x)\approx(f\eta,\eta)$) являются эллипсоидами с полуосями, направленными по собственным векторам, и по величине обратно пропорциональными $\sqrt{\lambda_i}$. В нашем случае

$$\frac{1}{2}\frac{\partial^2\varphi}{\partial x_i\partial x_j}=\sum_{k=1}^{n}\frac{\partial F_k}{\partial x_i}-\frac{\partial F_k}{\partial x_j},$$

т.е.

$$H = D^T D_\xi$$

где $D_\xi = \left\| \frac{\partial f_j}{\partial x_j} \right\| (\varphi)$ - матрица Якоби отображения $F(x)$, а D_ξ^T - матрица, транспонированная к D_ξ.

Определение. Коэффициентом обусловленности системы $F(x) = 0$ в невырожденном простом корне ξ называется отношение

$$H = \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}$$

где $\lambda_{\max}$ и $\lambda_{\min}$ - наибольшее и наименьшее собственные значения матрицы $H = D_\xi^T = D_\xi$

В случае нелинейной системы это определение совпадает с определением H - числа матрицы D_ξ (см. [11, 12, 23])

Как известно [11, 12] коэффициент обусловленности системы уравнений определяет скорость сходимости метода. При $H > 1$ система (14) плохо обусловлена, т.е. малому изменении. Исходных данных соответствуют большие изменения решения.

Особый интерес в связи с этим представляют такие методы решения систем линейных и нелинейных уравнений, которые являются универсальными, т.е. пригодными и для плохо обусловленных задач. Именно к таким методам относится метод дифференциального спуска [4, 6]. Таким образом, помимо своего прямого назначения – решения хорошо обусловленных задач – дифференциальный спуск применяется и для решения плохо обусловленных систем уравнений, т.е. имеет более широкую область применения.

II глава. Новые варианты метода дифференциального спуска.

§1. Об одном новом варианте метода дифференциального спуска.

Пусть имеется система нелинейных уравнений

$$f(x) = 0, \tag{1}$$

где $x = (x_1, x_2, ..., x_n)$ - n-мерный вектор, а $f = (f_1, f_2, ..., f_n)$ – заданная вектор функция. Предполагается, что функции $f_1, f_2, ..., f_n$ имеют достаточное число частных производных и уравнение (1) в некоторой области ω имеет единственное решение $x = \alpha$.

Как известно, при градиентном методе решение системы (1) сводится к нахождению в области ω точки, дающей минимум функционалу $\varphi(x)$, построенного из функции $f(x)$. Идеальным градиентным методом являлось бы движение к этой точке, совпадающей с решением $x = \alpha$ системы (1), по линии наискорейшего спуска, которая определяется решением системы дифференциальных уравнений [4]

$$\frac{dx}{ds} = \frac{\varphi_x(x)}{\sqrt{(\varphi_x(x), \varphi_x(x),)}} \tag{2}$$

с начальными условиями $s_0 = 0, x_0 = (x_1^0, x_2^0, ..., x_n^0), \quad x_0 \in \omega$ - начальное приближение к решению уравнения (1), s - дуга линии наискорейшего спуска и $\varphi_x(x) = grad \ \varphi(x)$.

Так как получение решения системы (2) в замкнутом виде в большинстве случаев невозможно, то используют приближенное решение $x_0(s)$. Двигаясь по линии $x_0(s)$, отыскивают на этой линии точку x_1, если $x_1(s)$ - приближенное решение системы (2) с начальными условиями $s_1 = 0, x_1$, то, двигаясь по линии $x_1(s)$, находят на этой линии точку x_2, в которой $\varphi(x_1(s))$ имеет минимальное значение.

Если уже найдено k-е приближение к корню $x = \alpha$, то $(k+1)$-е приближение x_{k+1} суть точка на линии $x_k(s)$, в которой $\varphi(x_k(s))$ имеет

минимальное значение. $x_k(s)$ - приближенное решение системы (2) с начальными условиями $s_k = 0$, x_k.

Как видно из вышеизложенного, применение обычного метода скорейшего спуска требует на i-м шаге $(i = 0,1,2,...)$ выполнения большой вычислительной работы при нахождении минимума функции $\varphi(x_i(s))$.

В данной работе рассматриваются три метода, которые не требуют нахождения на i-м шаге $(i = 0,1,2,...)$ нахождения минимума функции $\varphi(x_i(s))$.

Первый метод основан на замене системы (2) системой дифференциальных уравнений путем перехода к системе (2) от независимой переменной s к переменной t такой, что решение $x(t)$ полученной системы дает в пределе при $t \to 0$ корень уравнения (1).

Второй метод основан на интегрировании системы (2) с определенным выбором шага интегрирования.

Эти методы могут быть использованы для решения систем линейных алгебраических уравнений.

Перейдем к рассмотрению первого метода. Пусть

$$t = \varphi(x(s)),$$

где $\varphi(x) \geq 0$ - такой функционал, что $\varphi(x^*) = 0$ тогда и только тогда, когда $x^* = \alpha; x(s) -$ решение системы (2). В дальнейшем во всех параграфах рассматриваются только такие функционалы.

З а м е ч а н и е 1. Построение функционала в явном виде, вообще говоря, не является обязательным. Например, можно задать функционал

$$f(x_0(t)) = f(x).$$

Тогда, если условия теоремы о неявной функции выполнены, этот функционал определяет движение к решению уравнения (1) по линии, которая является решением системы

$$\frac{dx}{dt} = I^{-1}(x)\, f\,(x_0)$$

с начальными условиями $t_0 = 1,\ x_0$, где x_0 - начальное приближение к решению системы (1), $I^{-1}(x)$ - матрица обратная матрице Якоби. Решение $x(t)$ этого дифференциального уравнения при $t = 0$ дает корень системы (1).

З а м е ч а н и е 2. Пусть уравнение $f(x) = \theta$ имеет в некоторой области ω единственное решение $x = \alpha$. Предполагается, что E и F - дав банаховых пространства и f - достаточное число раз непрерывно дифференцируемое в ω отображение $\omega \subset E$ в $\tilde{\omega} \subset F$, а θ - нулевой элемент $\tilde{\omega}$.

Если условия теоремы о неявной функции выполнены, то функционал

$$f(x_0(t)) = f(x),$$

заданный в неявном виде, определяет дифференциальное уравнение

$$\frac{dx}{dt} = (f'(x))^{-1} \cdot f(x_0)$$

с начальными условиями $t_0 = 1,\ x_0$, где $x_0 \in \omega$ - начальное приближение к корню $x = \alpha$, $f'(x)$ - производная Фреше, $(f'(x))^{-1}$ - операция обратная операции $f'(x)$. Решение $x(t)$ этого дифференциального уравнения при $t = 0$ дает корень $x = \alpha$, и различные способы его интегрирования дают возможность построить различные итерационные методы решения уравнения $f(x) = \theta$.

Если перейти в системе (2) от переменной s к переменной t, то получим систему дифференциальных уравнений

$$\frac{dx}{dt} = \frac{\varphi_x(x)}{(\varphi_x(x), \varphi_x(x))} \tag{3}$$

с начальными условиями $t_0 = \varphi(x_0),\ x_0$. Решение $x(t)$ системы (3) при $t \to 0$ дает корень α системы (1), т. Е. $\lim\limits_{t \to 0} x(t) = \alpha$.

Интегрируя систему (3) приближенными способами, можно получить различные итерационные методы решения системы (1). Например, метод Эйлера с шагом $h_k = -\varphi(x_k)$ (уравнение решается с начальными условиями $t_k = \varphi(x_k),\ x_k$) дает итерационный метод

$$x_{k+1} = x_k - \frac{\varphi_x(x_k)\varphi(x_k)}{(\varphi_x(x_k), \varphi_x(x_k))}, \ k = 0,1,2,..., \qquad (4)$$

а метод Рунге-Кутта с тем же шагом и имеющий порядок ошибки h_k^3 дает итерационный метод

$$x_{k+1} = x_k - \frac{\varphi_x(x_k + 0.5\tau_k) \cdot \varphi_x(x_k)}{(\varphi_x(x_k + 0.5\tau_k), \varphi_x(x_k + 0.5\tau_k))}, \ k = 0,1,2,...,$$

где

$$\tau_k = \frac{\varphi_x(x_k) \cdot \varphi(x_k)}{(\varphi_x(x_k), \varphi_x(x_k))}.$$

В одномерном случае, если $\varphi(x) = |f(x)|$, то формула (4) дает метод Ньютона и, если интегрировать (3) методом Рунге-Кутта с шагом $h_k = -|f(x_k)|$, то можно получить итерационные методы высоких порядков, предложенные в [6, 11, 24].

Система (3) разрешена относительно производных и поэтому обладает некоторыми преимуществами по сравнению с системами, приведенными, например, в работах [6, 10].

§2. Последовательное нахождение всех действительных корней алгебраического уравнения

Прежде, чем рассматривать второй метод, остановимся подробно на способе последовательного вычисления всех действительных корней одного уравнения, полученном Л. М. Рыбаковым [18].

Т е о р е м а 1. Пусть непрерывная и дифференцируемая на отрезке $[a, b]$ функция $f(x)$ удовлетворяет условиям

$$f(\alpha_i) = 0, \ i = 1,2,...,\nu, \ a < \alpha_1 \le \alpha_2 \le ... \le \alpha_\nu < b,$$

$$|f'(x)| \le M \ \text{для всех} \ x \in [a, b],$$

тогда итерационный метод

$$x_{k+1} = x_k + |\beta\ f(x_k)|,\ \beta = \frac{1}{M},\ k = 0,1,2,... \qquad (5)$$

(соответственно, итерационный метод

$$x_{k+1} = x_k - |\beta\ f(x_k)|,\ \beta = \frac{1}{M},\ k = 0,1,2,...\ ) \qquad (6)$$

сходится к ближайшему справа от $x_0 \in [a,b]$ корню α_j (соответственно к ближайшему слева от $x_0 \in [a,b]$ корню α_p) уравнения

$$f(x) = 0,$$

если начальное приближение x_0 выбрано так, что корень α_j (соответственно корень α_p) существует.

Доказательство [18]. Так как $x_{k+1} - x_k = |\beta f(x_k)| \geq 0,$ то последовательность $\{x_k\}$ является возрастающей (соответственно так как $x_{k+1} - x_k = -|\beta f(x_k)| \leq 0,$ то последовательность $\{x_k\}$ является убывающей).

Доказательство ограниченности последовательности $\{x_k\}$ т. Е. $x_k \leq \alpha_j$ для всех $k = 0,1,2,...$ (соответственно $x_k \geq \alpha_p$ для всех $k = 0,1,2,...$) проведем по индукции, т.е. сначала докажем, что $x_1 \leq \alpha_j$ (соответственно $x_1 \geq \alpha_p$), а затем покажем, что из $x_k \leq \alpha_j$ следует $x_{k+1} \leq \alpha_j$ (соответственно из $x_k \geq \alpha_p$ следует $x_{k+1} \geq \alpha_p$).

Используя теорему о среднем, получим

$$|\beta f(\alpha_j) - \beta f(x_0)| = |\beta f(x_0)| = |\beta f'(\xi)|(\alpha_j - x_0) \leq \alpha_j - x_0,$$

где $x_0 < \xi < \alpha_j$ (соответственно $|\beta f(x_0)| \leq x_0 - \alpha_p$).

Из (5) имеем $x_1 \leq \alpha_j$ (соответственно из (6) имеем $x_1 \geq \alpha_p$).

Точно так же доказывается, что из $x_k \leq \alpha_j$ следует $x_{k+1} \leq \alpha_j$ (соответственно из $x_k \geq \alpha_p$ следует $x_{k+1} \geq \alpha_p$).

Следовательно, последовательность $\{x_k\}$ имеет предел, т.е. $\lim\limits_{k\to\infty} x_k = \alpha_j^*$ (соответственно $\lim\limits_{k\to\infty} x_k = \alpha_p^*$). Так как вследствие сходимости

последовательности $\{x_k\}$ $\lim_{k\to\infty}|x_{k+1}-x_k|=0$, то, учитывая непрерывность $f(x)$, из (5) и (6) имеем

$$\lim_{k\to\infty}|f(x_k)|=\left|f(\lim_{k\to\infty}x_k)\right|=0.$$

Последнее равенство имеет место тогда и только тогда, когда $\lim_{k\to\infty}x_k=\alpha_j^*=\alpha_j$ (соответственно $\lim_{k\to\infty}x_k=\alpha_p^*=\alpha_p$), что и требовалось доказать.

Вернемся теперь к методу скорейшего спуска.

Если $x(s)$ - решение системы (2), то корень s^* уравнения

$$\varphi(x(s))=0 \qquad (7)$$

определяет корень $x(s^*)=\alpha$ системы (1).

Так как получение решения системы (2) в замкнутом виде во многих случаях невозможно, то при численном методе интегрирования системы (2) необходимо выбирать шаг интегрирования так, чтобы «не проскочить» искомый минимум функционала $\varphi(x)$.

Метод (5) дает возможность выбирать шаг интегрирования системы (2) и строить итерационные методы.

Из (2) и (7) с помощью метода (5) последовательные шаги интегрирования определяются формулой

$$\Delta s_k=|\beta\varphi(x_k)|,\quad \beta=\frac{1}{M},\quad k=0,1,2,\ldots,$$

где $M\ge\sqrt{(\varphi_x(x),\varphi_x(x))}$ для всех $x\in\omega$; ω - область, в которой ищется решение системы (1).

Если при некотором $k=\mu$ окажется, что $x_\mu\overline{\in}\,\omega$, то необходимо вновь подобрать постоянную β для новой области ω', для которой $x_k\in\omega'$ при новой постоянной β и любом $k=0,1,2,\ldots$. Решая систему (2) методом Эйлера и методом Рунге-Кутта, имеющем порядок ошибки $(\Delta s_k)^3$, получим соответствующие итерационные методы

$$x_{k+1} = x_k - \frac{\varphi_x(x_k)\left|\beta\varphi(x_k)\right|}{\sqrt{(\varphi_x(x_k),\varphi_x(x_k))}},\ x_k \in \omega,\ k = 0,1,2,\ldots; \tag{8}$$

$$x_{k+1} = x_k - \frac{\varphi_x(x_k + 0.5\tau_k)\cdot\left|\beta\varphi(x_k)\right|}{\sqrt{(\varphi_x(x_k + 0.5\tau_k),\varphi_x(x_k + 0.5\tau_k))}},\ x_k \in \omega,\ \ k = 0,1,2,\ldots, \tag{9}$$

где

$$\tau_k = \frac{\varphi_x(x_k)\cdot\left|\beta\varphi(x_k)\right|}{\sqrt{(\varphi_x(x_k),\varphi_x(x_k))}}.$$

Рассмотрим теперь вопрос определения постоянной β для алгебраических уравнений.

При отыскании корней алгебраического уравнения

$$f(x) = x^m + a_1 x^{m-1} + \ldots + a_{m-1}x + a_m = 0 \tag{10}$$

на отрезке $[a,b]$ постоянную β для методов (5) и (6) можно принимать равной

$$\beta = (\left|mc^{m-1}\right| + \left|a_1(m-1)c^{m-2}\right| + \ldots + \left|a_{m-1}\right|)^{-1},$$

где $c = \max\{|a|, |b|\}$. Например, для уравнения

$$f(x) = x^4 - 0{,}234375\,x^2 - 0{,}01953125\,x + 0{,}005859375 = 0$$

при отыскании его корней на отрезке $[-0{,}75,\ 1]$ постоянную β можно принять равной $\frac{1}{4{,}5}$. В работе [18] для этого уравнения постоянная β принята равной $\frac{1}{8}$.

Так как в итерационных методах (5), (6) постоянные β могут быть весьма малыми, что вызывает слабую сходимость этих методов, то можно предложить соответственно следующие модификации этих методов:

$$x_{k+1} = x_k + \left|\beta_k f(x_k)\right|, \quad k = 0,1,2,\ldots; \tag{5′}$$

$$x_{k+1} = x_k - \left|\beta_k f(x_k)\right|, \quad k = 0,1,2,\ldots, \tag{6′}$$

где β_k зависят от номера k.

Если принять $x_0 = a$ (или соответственно $x_0 = b$), то отрезок, на котором находятся корни уравнения (10), с каждым номером k уменьшается и,

следовательно, β_k образуют возрастающую последовательность т.е. $\beta_0 \le \beta_1 \le \beta_2 \le ....$ Модифицированным методом $(5')$ (соответственно методом $(6')$) можно пользоваться при нахождении отрицательных (соответственно положительных) корней уравнения (10), при этом полагая $x_0 = a$ (соответственно $x_0 = b$) и

$$\beta_k = (\left|mx_k^{m-1}\right| + \left|a_1(m-1)x_k^{m-2}\right| + ... + \left|a_{m-1}\right|)^{-1}.$$

Так как при приближении к корню величины β_k в методах $(5')$,$(6')$ могут все же оставаться, то при подходе к корню можно переходить на классические итерационные методы высоких порядков.

В случае перехода на метод Ньютона критерием подхода может служить неравенство

$$h = N^2 L\delta \le \frac{1}{2}, \qquad (11)$$

где

$$N \ge (\left|f'(x_k)\right|)^{-1}, \quad L \ge \max_{a \le x \le b} \left|f''(x)\right|, \quad \delta \ge \left|f(x_k)\right|.$$

Если неравенство (11) удовлетворяется при некотором x_k, то, начиная с этого x_k, метод Ньютона сходится к корню уравнения (10). Подробное доказательство этого можно найти в работе [7]. Если система (1) – система алгебраических уравнений, а область, в которой ищется решение, определяется неравенствами

$$a_i \le x_i \le b_i, \quad i = 1, 2, ..., n,$$

то при определении постоянной β можно поступать так же, как для одного алгебраического уравнения.

§3. Применение метода для решения систем линейных алгебраических уравнений.

Рассмотрим применение метода (4) для решения систем линейных алгебраических уравнений.

Пусть имеется система n линейных уравнении с n неизвестными

$$Ax = b, \tag{12}$$

где A - вещественная симметрическая матрица n-го порядка; b - вектор-столбец свободных членов; x - вектор-столбец неизвестных.

Предполагается, что матрица A имеет собственные значения, отличные от нуля.

Примем $\varphi(x) = (b - Ax, b - Ax) = (r, r),\ b - Ax = r -$ вектор невязки.

Тогда, так как $\varphi_x(x) = -2Ar,$ система (3) и метод (4) примут соответственно вид

$$\frac{dx}{dt} = -\frac{Ar}{2(Ar, Ar)}; \tag{13}$$

$$x_{k+1} = x_k + \frac{Ar_k(r_k, r_k)}{2(Ar_k, Ar_k)},\ \ r_k = b - Ax_k,\ \ k = 0,1,2,.... \tag{14}$$

Т е о р е м а 2 [4]. Последовательные приближения $x_0, x_1, x_2,.,$ метода (14) сходятся к решению системы (12) с быстротой геометрической прогрессии, если $\left(\frac{\rho}{4} + \frac{1}{4\rho}\right)^2 < 1$ (ρ - число обусловленности Тодда).

Доказательство. Покажем сначала, что если условия теоремы выполнены, то

$$\frac{\varphi(x_{k+1})}{\varphi(x_k)} = \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)} = \frac{(r_k, r_k)(A^2 r_k, A^2 r_k)}{4(Ar_k, Ar_k)^2} < 1.$$

Пусть $\lambda_i,\ i = 1,2,....,n$ - собственные значения матрицы A; $U_i,\ i = 1,2,...,n$ - принадлежащие им собственные векторы, ортогональные друг другу и нормированные так, что $(U_i, U_i) = 1$ при $i = 1,2,...,n$. Так как A вещественная симметрическая матрица, все собственные значения

λ_i, $i=1,2,....,n$ вещественны. Пусть $|\lambda_1| \le |\lambda_2| \le ... \le |\lambda_n|$. Пусть далее $r_k = c_1U_1 + ... + c_nU_n$, причем не все c_i равны нулю. Тогда

$$Ar_k = c_1\lambda_1U_1 + ... + c_n\lambda_nU_n;$$
$$A^2r_k = c_1\lambda_1^2U_1 + ... + c_n\lambda_n^2U_n.$$

Следовательно,

$$(r_k, r_k) = \sum_{i=1}^{n} c_i^2;$$
$$(Ar_k, Ar_k) = \sum_{i=1}^{n} c_i^2\lambda_i^2;$$
$$(A^2r_k, A^2r_k) = \sum_{i=1}^{n} c_i^2\lambda_i^4,$$

и

$$\frac{(r_k, r_k)(A^2r_k, A^2r_k)}{4(Ar_k, Ar_k)^2} = \frac{\sum_{i=1}^{n} c_i^2 \cdot \sum_{i=1}^{n} c_i^2\lambda_i^4}{4(\sum_{i=1}^{n} c_i^2\lambda_i^2)^2}. \qquad (15)$$

Введем обозначения

$$a_i = c_i^2\lambda_i^2, \ \gamma_i = \frac{1}{\lambda_i^2}.$$

Тогда равенство (15) примет вид

$$\frac{(r_k, r_k)(A^2r_k, A^2r_k)}{4(Ar_k, Ar_k)^2} = \frac{\sum_{i=1}^{n} \gamma_i a_i \cdot \sum_{i=1}^{n} \frac{a_i}{\gamma_i}}{4(\sum_{i=1}^{n} a_i)^2}.$$

Для оценки последнего отношения применим неравенство

$$\frac{\sum_{i=1}^{n} \gamma_i a_i \cdot \sum_{i=1}^{n} \frac{a_i}{\gamma_i}}{4(\sum_{i=1}^{n} a_i)^2} \le \left(\sqrt{\frac{M}{m}} + \sqrt{\frac{m}{M}}\right)^2,$$

справедливое при условии, что все числа $a_i > 0$, а числа γ_i удовлетворяют неравенствам $0 < m \le \gamma_i \le M$. Обоснование этого неравенства можно найти в работе [6]. В нашем случае:

$$M = \frac{1}{|\lambda_1|^2}, \quad m = \frac{1}{|\lambda_n|^2}.$$

Поэтому имеем

$$\frac{\varphi(x_{k+1})}{\varphi(x_k)} = \frac{(r_k, r_k)(A^2 r_k, A^2 r_k)}{4(Ar_k, Ar_k)^2} \le \frac{1}{16}\left(\frac{|\lambda_n|}{|\lambda_1|} + \frac{|\lambda_1|}{|\lambda_n|}\right)^2 = \left(\frac{\rho}{4} + \frac{1}{4\rho}\right)^2, \qquad (16)$$

где $\rho = \left|\frac{\lambda_n}{\lambda_1}\right|$ - число обусловленности Тодда. Итак,

$$\varphi(x_{k+1}) \le \left(\frac{\rho}{4} + \frac{1}{4\rho}\right)^2 \varphi(x_k)$$

и, следовательно,

$$\varphi(x_{k+1}) \le \left(\frac{\rho}{4} + \frac{1}{4\rho}\right)^{2(k+1)} \varphi(x_0).$$

Таким образом, так как $\left(\frac{\rho}{4} + \frac{1}{4\rho}\right)^2 < 1$, $\varphi(x_{k+1}) \to 0$ при $k \to \infty$ и потому $x_{k+1} \to \alpha$, где α точное решение системы (12). Сменим теперь длину вектора ошибки, т.е. вектора $y_k = \alpha - x_k$. Так как

$$\varphi(x_k) = (Ay_k, Ay_k) = (A^2 y_k, y_k) \ge |\lambda_1|^2 (y_k, y_k),$$

то

$$y_k = \sqrt{(y_k, y_k)} \le \frac{\sqrt{\varphi(x_0)}}{|\lambda_1|}\left(\frac{\rho}{4} + \frac{1}{4\rho}\right)^k.$$

Тем самым доказано, что $|y_k|$ стремится к нулю со скоростью геометрической прогрессии со знаменателем $q = \left(\frac{\rho}{4} + \frac{1}{4\rho}\right) < 1$.

Метод (14) получен с помощью интегрирования системы (13) методом Эйлера с шагом $h_k = -(r_k, r_k)$. Так как начальное приближение x_0 может быть выбрано достаточно далеко от корня α, то величина шага $h_k = -(r_k, r_k)$ может стать настолько большой. Что приближенное решение системы (13) при $t = 0$ не будет иметь ничего общего с точным решением уравнения (12). Поэтому рассмотрим вопрос выбора шага интегрирования системы (13) таким образом, чтобы, интегрируя (13) методом Эйлера, можно было прийти к точному решению системы (12) с любого начального приближения.

Естественно принимать шаг интегрирования равным $h_k = -\dfrac{(r_k, r_k)}{\gamma}$, где γ - некоторая постоянная, удовлетворяющая условию $\gamma \geq 1$.

На вопрос о том, как выбрать постоянную γ, дает ответ следующая теорема.

Т е о р е м а 3 [4]. Последовательные приближения $x_0, x_1, x_2, \ldots$ метода

$$x_{k+1} = x_k + \frac{Ar_k (r_k, r_k)}{2\gamma (Ar_k, Ar_k)}, \quad r_k = b - Ax_k, \quad k = 0, 1, 2, \ldots \tag{17}$$

сходятся к решению системы (12) с быстротой геометрической прогрессии с любого начального приближения x_0, если

$$\gamma > \left(\frac{\rho}{4} + \frac{1}{4\rho} \right)^2 \text{ при } \left(\frac{\rho}{4} + \frac{1}{4\rho} \right)^2 \geq 1$$

$$\gamma = 1, \text{ если } \left(\frac{\rho}{4} + \frac{1}{4\rho} \right)^2 < 1,$$

где ρ - число обусловленности Тодда.

Доказательство. Покажем прежде всего, что если условия теоремы выполнены, то

$$\frac{\varphi(x_{k+1})}{\varphi(x_k)} = \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)} = 1 - \frac{1}{\gamma} + \frac{1}{\gamma^2} \cdot \frac{(r_k, r_k)(A^2 r_k, A^2 r_k)}{4(Ar_k, Ar_k)^2} \leq q^2 < 1, \tag{18}$$

откуда

$$\gamma > \frac{(r_k, r_k)(A^2 r_k, A^2 r_k)}{4(Ar_k, Ar_k)^2}.$$

Рассмотрим случай, когда $\gamma > \left(\dfrac{\rho}{4} + \dfrac{1}{4\rho} \right)^2$ и $\left(\dfrac{\rho}{4} + \dfrac{1}{4\rho} \right)^2 \geq 1$. Тогда, учитывая неравенство (16), получим

$$\gamma > \left(\frac{\rho}{4} + \frac{1}{4\rho} \right)^2 \geq \frac{(r_k, r_k)(A^2 r_k, A^2 r_k)}{4(Ar_k, Ar_k)^2}.$$

Далее рассмотрим случай, когда $\gamma = 1$ и $\left(\dfrac{\rho}{4} + \dfrac{1}{4\rho} \right)^2 < 1$.

Опять, учитывая неравенство (16), получим

$$\gamma = 1 > \left(\frac{\rho}{4} + \frac{1}{4\rho}\right)^2 \geq \frac{(r_k, r_k)(A^2 r_k, A^2 r_k)}{4(Ar_k, Ar_k)^2}.$$

Отсюда видно, что в обоих случаях удовлетворяется неравенство (18). Итак,

$$\varphi(x_{k+1}) \leq q^2 \varphi(x_k)$$

и

$$\varphi(x_{k+1}) \leq q^{2(k+1)} \varphi(x_0), \ q < 1.$$

Следовательно, $\varphi(x_{k+1}) \to 0$ при $k \to \infty$ и потому $x_{k+1} \to \alpha$, где α - точное решение системы (12).

Оценим теперь длину вектора ошибки, т.е. $y_k = \alpha - x_k$. Так как

$$\varphi(x_k) = (Ay_k, Ay_k) = (A^2 y_k, y_k) \geq |\lambda_1|^2 (y_k, y_k),$$

то

$$|y_k| = \sqrt{(y_k, y_k)} \leq \frac{\sqrt{\varphi(x_0)}}{|\lambda_1|} \cdot q^k$$

и $|y_k|$ стремится к нулю со скоростью геометрической прогрессии со знаменателем

$$q = \sqrt{1 - \frac{1}{\gamma} + \frac{1}{\gamma^2}\left(\frac{\rho}{4} + \frac{1}{4\rho}\right)^2} < 1.$$

Что и требовалось доказать.

Для собственных значений λ_1 и λ_n можно принять оценки [10, 11]

$$|\lambda_1| \geq |d| \sqrt[n]{\left(\frac{n-1}{n}\right)^{n-1}};$$

$$|\lambda_n| \leq \sqrt{s - \sqrt[n]{s^n - nd^2(n-1)^{n-1}}},$$

где d - определитель A; s - след AA'.

Таким образом, выбирая шаг интегрирования, зависящий от обусловленности матрицы A, можно, интегрируя (13) методом Эйлера, прийти к точному решению системы (12) с любого начального приближения.

§4. Проекционно-дифференциальный метод решения систем нелинейных уравнений.

Вопросы решения систем нелинейных уравнений в литературе разработаны совершенно недостаточно. Обычно каждая нелинейная система должна рассматриваться как специальная задача [6, 11, 12, 23, 24]. Такие системы решают практически только приближенными методами. Наиболее распространенными методами для решения систем нелинейных уравнений являются метод итераций, метод Ньютона, метод скорейшего спуска и некоторые их модификации [2, 3, 9, 11, 19] .

Одной из наиболее сложных задач при применении многих итерационных методов (например, простая итерация, метод Ньютона, метод Эйткена – Стефенсона [2, 3] и т.д.) является выбор достаточно близкого к решению уравнения начального приближения, обеспечивающего сходимость итерационного процесса. Но поскольку решение системы нелинейных уравнений заранее неизвестно и не всегда можно указать даже область небольшого диаметра, в которой может находиться решение, то проблема выбора начального приближения, обеспечивающего сходимость итерационного процесса, имеет существенное значение для применения упомянутых методов. Часто удается применить метод случайного выбора. Во многих случаях данное уравнение заменяют «близким» и точное решение этого близкого уравнения рассматривают, как начальное приближение к решению исходного уравнения. Общие рецепты построения приемлемых начальных приближений, конечно, не могут быть даны. Однако могут быть описаны приемы, применимые при решении широких классов уравнений [10,11].

Итерационные методы, построенные с помощью приближенных способов решения задачи Коши [5, 14], в значительной степени решают проблему выбора начального приближения. Конкретизируем вычислительную схему метода дифференциального спуска [5, 14]

применительно к системе нелинейных уравнений в конечномерных евклидовых пространствах.

Коротко изложим суть метода дифференциального спуска [4, 14].

Основные обозначения. E^n - n-мерное евклидово пространство; (x, y) - скалярное произведение элементов $x = (x_1, x_2, ..., x_n),\ y = (y_1, y_2, ..., y_n)$; α_{nj} - элемент, принадлежащий решению α_i первых i уравнений системы (1) и соответствующий начальному условию $x_n,\ n = 0,1,2,...,$ для рекуррентной последовательности задач (2); x_{ni} - приближение к α_{ni}; n_i - число делений отрезка $[f_i(x_{ni-1}), 0]$; $h_i = \dfrac{1}{n_i} f_i(x_{ni-1})$ – шаг интегрирования при приближенном решении задачи (2).

1. Пусть имеется система нелинейных уравнений

$$f(x) = 0, \tag{1}$$

где $x \in E^n,\ f(x) \in E^n, m \le n.$ Предполагается, что функции $f_i(x),\ i = 1,2,...,m$ не имеют непрерывные частные производные до второго порядка включительно и уравнение (1) в некоторой области $\omega \in E^n$ имеет единственное решение $x = \alpha$.

Спуск к решению $x = \alpha$ уравнения (1) осуществляется вдоль траектории, составленной из решений $x_i(t_i),\ i = 1,2,...,m$ рекуррентной последовательности задач Коши:

$$\frac{dax_i}{dat_i} = \frac{a_i^{i-1}(x)}{f', a_{i-1i}^{i}},\ x_i^0 = x_i(t_i = t_i^0) = x_{i-1}(t_i = 0) = \alpha_{0i=1},$$
$$t_i^0 = f_i(\alpha_{0i-1}),\ \alpha_{00} = x_0, i = 1,2,...,m. \tag{2}$$

Вектор-функция $a_i^{j-1}(x) \in E^m$ построен из функций $f_j(x),\ j = 1,2,...,i \le m$ так, что

$$(f_j', a_i^{i-1}) = 0,\ j \le i-1, \tag{3}$$

$$(f_j', a_i^{i-1}) \ne 0,\ j = i,\ f_j' = grad\ f_j(x). \tag{4}$$

При некоторых условиях, накладываемых на $f_i(x)$ и $a_i^{i=1}(x)$ теоремой существования и единственности решения [5, 14], уравнение (2) имеет решение $x_i(t_i)$, определенное на конечном интервале $[t^0, 0]$ и

$$\lim_{t_i \to 0} x_i(t_i) = \alpha_{0i}.$$

Для задач (2) известны первые интегралы

$$f_j(x_i(t_i)) = 0,\ j \le i-1,\ ti - f_i(x_i(t_i)) = 0,$$

с помощью которых осуществляется контроль точности спуска к решению системы (1) по траекториям, определяемым решениями задач (2).

Практическое использование свойства уравнения (2) состоит в следующем. Чтобы определить решение уравнения (1), интегрируем уравнение (2) одним из методов численного интегрирования обыкновенных дифференциальных уравнений первого порядка, разбив при этом интервал $[f_i(x_{0i-1}), 0]$ на достаточное число n_i подынтервалов длины $h_i, i = 1, 2, ..., n_i$. В результате получим приближенное решение $x_{0i}(t_i)$ уравнения (2). Значение $x_{0m}(t_m)$ при $t_m = 0$ дает приближенное значение x_1 корня $x = \alpha_1$, т.е. $x_1 = x_{0m}(0) \approx \alpha_{0m}$

Если полученное решение x_i еще не является с заданной степенью точности решением уравнения (1), то указанный выше процесс повторяем, беря за исходное приближение x_1, т.е. вновь решаем уравнение (2) с начальными условиями $t_{1i} = f_i(x_{1i-1}), x_{1i-1}(x_{10} = x_1)$ одним из методов (не обязательно тем же) численного интегрирования обыкновенных дифференциальных уравнений. В результате получим новое приближение x_2 к корню уравнения (1): $x_2 = x_{1m}(0) \approx \alpha_{1m}$.

Повторяя описанный процесс, получим последовательность $\{x_{nm}(0)\}, n = 0, 1, 2, ...,$ элементы которой рассматриваются как приближенные решения уравнения (1). Такой прием построения приближенного решения уравнения (1) и составляет принципиальную схему этого варианта метода дифференциального спуска.

Заметим, что при различных построениях указанного процесса (т.е. при различных $n = 0, 1, 2, ...$) и методы, и шаги численного интегрирования

уравнения (2) можно выбирать различными с таким расчетом, чтобы требуемый результат получался наиболее «дешевым» путем.

Применяя к уравнению (2) различные методы численного интегрирования обыкновенных дифференциальных уравнений, можно получить новым путем как многие известные, так и новые итерационные методы. Тем самым многие известные методы можно описать аналитически и подчинить единой схеме.

Если (1) – система линейных алгебраических уравнений, то правая часть уравнения (2) не зависит от x, и интегрируя (2) методом Эйлера для решения $\alpha = \alpha_{0m}$ такой системы имеем

$$\begin{aligned}
&\alpha_{01} = \alpha_{00} - \frac{f_1(\alpha_{00})}{(f_1', a_1^0)} a_1^0, \\
&\alpha_{02} = \alpha_{01} - \frac{f_2(\alpha_{01})}{(f_2', a_2^1)} a_2^1, \\
&- - - - - - - - - - - - - \\
&\alpha_{0m} = \alpha_{0m-1} - \frac{f_m(\alpha_{0m-1})}{(f_m', a_m^{m-1})} a_m^{m-1},
\end{aligned} \tag{5}$$

где $\alpha_{00} \in \omega$ - произвольное начальное приближение. Таким образом, в этом случае имеем метод (5), дающий на m-м шаге решение системы алгебраических уравнений. Если вычисления приближенные, то процесс можно сделать итерационным.

Вектор-функции $a_i^{i-1}(x), i = 1, 2, ..., m$ со свойствами (3), (4) могут быть построены, например, по формуле [20]

$$a_i^j = a_i^{j-1} - a_j^{j-1} \frac{(f', a_i^{j-1})}{(f_j', a_j^{j-1})}, \quad j < i \tag{6}$$

где i, j - целые числа. Они также могут быть построены с помощью известных алгоритмов ортогонализации, A- ортогонализации и биортогонализации [20, 24].

2. Нередко оказывается возможным данное уравнение (1) заменить на эквивалентное ему уравнение, состоящее из линейной и нелинейной части. Для решения последнего можно воспользоваться следующим приемом.

Пусть данное уравнение имеет вид:

$$f(x) \equiv P(x) + R(x) = 0, \tag{7}$$

где $x \in E^n$, $f(x) = (P_1(x) + R_1(x), P_2(x) + R_2(x), ..., P_m(x) + R_m(x))$ - вектор-функция со значениями в E^m, а $P(x)$ - линейная, $R(x)$ - нелинейная вектор-функции.

Уравнение (7) эквивалентно системе уравнений

$$\begin{aligned} P(x) + S(x) &= 0 \\ R(x) - S(x) &= 0, \end{aligned} \tag{8}$$

где $S(x) \in E^m$ - новая неизвестная вектор-функция.

Поскольку метод является точным для линейных алгебраических уравнений, то решая рекуррентную последовательность задач Коши (2) $i = 1, 2, ..., 2m$, соответствующую системе (8), каким-нибудь приближенным методом, при точных вычислениях получим точное решение первого линейного уравнения системы (8). В силу алгоритма метода, находясь на решении линейного уравнения системы (8), должны находить решение нелинейного уравнения системы (8). Поэтому в дальнейшем вся вычислительная работа будет направлена на получение с требуемой точностью решения нелинейного уравнения системы (8).

Таким образом, если в данном уравнении можно выделить линейную часть, то можно построить алгоритм метода так, чтобы последовательные приближения итерационного процесса для нелинейной части принадлежали решению линейной части. Это значительно уменьшает объем вычислений, особенно при небольшой нелинейной части уравнения (7).

§5. Упрощенные алгоритмы проекционно-дифференциального метода

Основные обозначения. E^n - n-мерное евклидово пространство; (x, y) - скалярное произведение элементов $x = (x_1, x_2, ..., x_n)$, $y = (y_1, y_2, ..., y_n) \in E^n$; α_0^i - вектор, принадлежащий решению α_i первых i уравнений системы (1) и

соответствующий начальному условию $x^{(0)} = \alpha_{00}$ для рекуррентной последовательности задач Коши (2).

1. Рассмотрим систему уравнений (вообще говоря, нелинейных)

$$f(x) = 0, \tag{1}$$

где $x \in E^n$, $f(x) \in E^m$, $m \le n$. Предполагается, что функции $f_j(x)$ ($j = \overline{1,m}$) имеют непрерывные частные производные до второго порядка включительно и уравнение (1) в некоторой области $\omega \in E^n$, имеет единственное решение $x = \alpha$.

Согласно [5, 14] спуск к решению $x = \alpha$ уравнения (1) осуществляется вдоль траектории, составленной из решений $x_i(t_i)$ ($i = \overline{1,m}$) задач Коши:

$$\frac{dx_i}{dt_i} = \frac{a_i^{i-1}}{(f_i', a_i^{i-1})}, \quad x_i^{(0)} = x_i(t_i = t_i^0) = x_{j-1}(t_{j-1} = 0) = \alpha_{0\,j-1},$$
$$t_i^0 = t_i(\alpha_{0i-1}),\ x(_{00}) = x^{(0)} \qquad (i = \overline{1,m}) \tag{2}$$

Вектор-функции $a_i^{i-1}(x)$ строятся из функций $j = \overline{1,m}$ так, чтобы

$$(f_j^1, a_i^{i-1}) = 0, \quad j \le i-1, \tag{3}$$

$$(f_j^1, a_i^{i-1}) \ne 0, \quad j = i,\ f_j^1 = grad\ f_j(x) \tag{4}$$

При некоторых ограничениях на $f_i(x)$ и a_i^{i-1} уравнение (2) имеет решение $x_i(t_i)$, определенное на конечном интервале $[f_i(x_{0i-1}), 0]$ и $\lim\limits_{t_i \to 0} x_i(t_i) = \alpha_{0i}$.

Для задач (2) известны первые интегралы

$$f_j(x_i(t_i)) = 0,\ j \le i-1 \tag{5}$$

$$t_i - f_i(x_i(t_i)) = 0 \tag{6}$$

Если (1) – система нелинейных уравнений, то и соответствующие ей уравнения дифференциального спуска (2) также являются нелинейными. Поэтому для интегрирования уравнения (2) обычно применяется какая-нибудь формула численного интегрирования обыкновенных дифференциальных уравнений [2, 3, 8, 9, 19], т.е. спуск к решению уравнения (1) происходит по линиям, определяемым приближенными решениями задач

(2). При этом соотношения (5), (6) используются для контроля точности приближенного решения задачи (2) .

Рассматриваемый вариант метода дифференциального спуска требует построения вектор-функций $\{a_i^{j-1}(x)\}_1^m$ со свойствами (3), (4). Они могут быть построены из начальной линейно-независимой системы векторов $\{a_i^0\}_1^m$, например, по формуле [20]

$$a_j^j = a_i^{j-1} - a_j^{j-1} \frac{(a_j^1, a_i^{j-1})}{(a_j^1, a_j^{j-1})} \tag{7}$$

где i, j - целые числа. В частности, если $\{a_i^1\}_1^m$ - система линейно независимых векторов, то можно принять $\{a_i^1 = a_i^0\}_1^m$.

Для построения вектор-функций $\{a_i^{j-1}(x)\}_1^m$ могут быть использованы также известные алгоритмы ортогонализации, A-ортогонализации [20, 24].

Неоднозначностью в построении вектор-функций $\{a_i^{j-1}(x)\}_1^m$ можно воспользоваться так, чтобы упростить процедуру их построения.

В связи с этим, ниже даны некоторые новые вычислительные схемы метода применительно к системе уравнений (1), основанные на предварительных преобразованиях этой системы.

2. Преобразуем уравнение (1) к виду

$$F(x) = 0,$$

где

$$\begin{gathered} F_1(x) = f_1(x) = 0, \\ F_k(x) = f_k(x) - \sum_{i=1}^{k-1} \frac{(f_k^1, F_1^1)}{(F_1^1, F_1^1)} F_i(x) = 0 \\ (k = 2, 3, \ldots, m) \end{gathered} \tag{8}$$

В предположении линейной независимости системы векторов $\{f_i^1\}_1^m$ по индукции легко можно показать эквивалентность систем уравнений (1) и (8), а также выполнение условий:

$$(F_i^1, F_j^1) = \begin{cases} 0 \text{ при } j = 1, 2, \ldots, i-1, \\ A_{ij}(x) > 0 \text{ при } j = i \end{cases} \tag{9}$$

Поэтому в этом случае можно принять

$$a_i^{i-1}(x) = F_1^i(x) \qquad (i = \overline{1,m}) \tag{10}$$

а системе уравнений (9) соответствует следующая рекуррентная последовательность задач Коши:

$$-\frac{dx_i}{dt_i} = \frac{F_i^1}{(F_i, F_j)},\ x_i^0 = x_i(t_i = t_i^0) = x_{i-1}(t_{i-1} = 0) = \alpha_{0i-1},$$
$$t_i^0 = F_i(\alpha_{0i-1}),\ \alpha_{00} = x^{(0)} \quad (i = \overline{1,m}) \tag{11}$$

В частности, если (1) есть система линейных алгебраических уравнений: $Ax = b$ с матрицей A размера $m \times n$, то её предварительно перепишем в виде

$$f_i(x) = (A_i, x) - b_i = 0 \qquad (i = \overline{1,m}) \tag{12}$$

где $A_i = (a_{i1}, a_{i2}, ..., a_{in})$ - вектор-строка, b_i - компоненты вектора b правых частей, x - вектор неизвестных.

Тогда, после интегрирования задачи Коши (11) методом Эйлера, решение $\alpha = \alpha_{0m}$ системы уравнений (12) получим по формулам:

$$\alpha_{0i} = \alpha_{0i-1} - \frac{F_i(\alpha_{0i-1})}{(F_i', F_i')} F_i' \qquad (i = \overline{1,m}) \tag{13}$$

где $F_i(x)$ определяются по формулам (8).

Таким образом, путем предварительного эквивалентного преобразования переходим от системы (1) к системе уравнений (8), для которой F_i' удовлетворяют условиям (9). В этом случае в силу (10) не требуется построение вектор-функций $a_i^{i-1}(x)_{i=1}^{m}$ со свойствами (3), (4) по формуле (7).

2. Из соотношений (5), (6) следует, что задача Коши (2) может быть использована для обращения в нуль невязки i- го уравнения системы (1) при нулевых невязках $(i-1)$-ых уравнений этой системы, т.е. находясь на решении первых $i-1$ уравнений системы (1), находится решение ее i- го уравнения. В свою очередь, для получения решения первых i уравнений системы (1) необходимо последовательно

находить приближенными методами решение первых i задач Коши из семейства (2).

Предлагаемая ниже модификация метода основана на предварительном переходе от системы уравнений (1) к такой эквивалентной системе, что произвольно выбранное начальное приближение $x^{(0)}$ является решением её первых $m-1$ уравнений. В этом случае в силу свойств метода дифференциального спуска (отмеченных в начале п.3), для получения решения эквивалентной системы, достаточно решить её только последнее m -ое уравнение. Поэтому в предлагаемой модификации метода, для получения решения системы уравнений (1), нет необходимости в последовательном решении всех задач Коши из семейства (2), а достаточно получить решение только последней m -ой задачи Коши с требуемой точностью.

Суть этой модификации метода дифференциального спуска состоит в следующем.

С помощью невырожденной матрицы $\|f_i(x_{0i-1}), 0\|$ от системы уравнений (1) переходим к эквивалентной ей системе уравнений

$$\Phi(x) = Bf(x) = 0 \tag{14}$$

такой, что $\Phi_i(x^{(0)}) = 0$ $(i = \overline{1, m-1})$ и $\Phi_m(x^{(0)}) \neq 0$, где $x^{(0)} \in \omega$ - произвольное начальное приближение к α .

Например, в качестве матрицы B можно взять матрицу:

$$B = \begin{bmatrix} f_m^0 & 0 & 0 & \dots & 0 & -f_1^0 \\ 0 & f_m^0 & 0 & \dots & 0 & -f_2^0 \\ - & - & - & - & - & - \\ 0 & 0 & 0 & \dots & f_m^p & -f_{m-1}^0 \\ 0 & 0 & 0 & \dots & 0 & -1 \end{bmatrix} \tag{15}$$

Здесь f_i $(i = \overline{1, m})$ - невязка уравнений системы (1) в начальной точке $x^{(0)}$. Очевидно, что для соблюдения эквивалентности систем (1) и (14) должно быть: $f_i^0 \neq 0$ $(i = \overline{1, m})$.

При использовании матрицы (15) система (14) имеет вид:

$$\begin{aligned}&\Phi_i(x) \equiv f_i(x)\cdot f_m^0 - f_m(x)\cdot f_i^0 = 0 \quad (i=\overline{1,m-1})\\&\Phi_m(x) \equiv f_m(x) = 0\end{aligned} \tag{16}$$

Более удобнее переписать матрицу B в виде:

$$B=\begin{bmatrix}1\,0\,0\ldots 0 & -f_1^0\\0\,1\,0\ldots 0 & -f_2^0\\---------- & -\\0\,0\,0\ldots 1 & -f_{m-1}^0\\0\,0\,0\ldots 0 & -1\end{bmatrix} \tag{15′}$$

В случае матрицы (15′) система (14) принимает вид:

$$\begin{aligned}&\Phi_i(x) \equiv f_i(x) - \Phi_m(x)\cdot f_i^0 = 0 \quad (i=\overline{1,m-1})\\&\Phi_m(x) \equiv \frac{f_m(x)}{f_m(x^{(0)})} = 0 \quad (f_m(x^{(0)})\neq 0)\end{aligned} \tag{16′}$$

Матрица (15′) легко может быть обращена простым изменением всех знаков при f_i, т.е.

$$B^{-1}=\begin{bmatrix}1\,0\,0\ldots 0 & f_1^0\\0\,1\,0\ldots 0 & f_2^0\\---------- & -\\0\,0\,0\ldots 1 & f_{m-1}^0\\0\,0\,0\ldots 0 & 1\end{bmatrix}$$

Поскольку при произвольном $x^{(0)}$ удовлетворяются первые $m-1$ уравнений систем (16) и (16′), то для нахождения их решения необходимо решить последние m-ые уравнения, находясь на решениях первых $m-1$ уравнений этих систем. Как было отмечено выше, последняя задача может быть решена методом дифференциального спуска.

Таким образом, задача решения систем уравнений (16), (16′) свелась к нахождению методом дифференциального спуска решения их последних m-ых уравнений. А для этой цели, в свою очередь, необходимо интегрировать только одно m-е уравнение в последовательности задач Коши (2):

$$\begin{aligned}&\frac{dx_m}{dt_m} = \frac{a_m^{m-1}}{(\Phi_m^1, b_m^{m-1})};\quad x_m^{(0)} = x_m(t_m = t_m^0) = x_{m-1}(t_{m-1}=0) = x^{(0)},\\&t_m^0 = \Phi_m(x^{(0)})\end{aligned} \tag{17}$$

При этом вектор-функция $a_m^{m-1}(x)$ строится из функций $\Phi_j(x)$ $(j=\overline{1,m})$, удовлетворяющим условиям типа (3), (4), т.е.

$$\begin{aligned}(\Phi_j', a_m^{m-1}) &= 0, \quad j \le m-1,\\ (\Phi_m', a_m^{m-1}) &\ne 0,\end{aligned}$$

где $\Phi_j' \equiv grad\,\Phi_j(x)$ $(j=\overline{1,m})$.

Если дана система линейных алгебраических уравнений (12), то после интегрирования задачи Коши (17) методом Эйлера, для решения получим следующую формулу:

$$\alpha_{0m} = x^{(0)} - \frac{(f_m(x^{(0)}))}{(f_m', a_m^{m-1})} a_m^{m-1} \tag{18}$$

где $x^{(0)} \in \omega$ - произвольное начальное приближение к решению системы (12) такое, что $f_i(x^{(0)}) \ne 0$ $(i=\overline{1,m})$.

В частности, если $x^{(0)}$ - нулевой вектор, то $f_i^0 = f_i(x^{(0)}) = -b_i$, $(b_i \ne 0,\ i=\overline{1,m})$, система (16) принимает вид

$$\begin{aligned}&\Phi_i(x) \equiv f_m(x)\cdot b_i - f_m(x)\cdot b_m = 0 \quad (i=\overline{1,m-1})\\ &\Phi_m(x) \equiv f_m(x) = (A_m, x) - b_m = 0\end{aligned}$$

и по формуле (18) получаем

$$\alpha_{0m} = -\frac{b_m}{(A_m, a_m^{m-1})} a_m^{m-1} \tag{18′}$$

В случае системы (16′) $f_m = \Phi_m(x^{(0)}) = 1$ и формула (18) принимает еще более простой вид

$$\alpha_{0m} = -\frac{a_m^{m-1}}{(A_m, a_m^{m-1})} \tag{18″}$$

Непосредственной проверкой убеждаемся, что формулы (13), (18), (18′), (18″) определяют точные решения системы линейных алгебраических уравнений (12). Если вычисления по этим формулам приближенные, то процесс можно сделать итерационным.

Заметим, что в силу вышеназванных свойств рассмотренной модификации метода ($x^{(0)}$ есть точное решение первых $m-1$ уравнений системы (1) и для получения решения её достаточно интегрировать только

одну m-ю задачу Коши в (2)), объем необходимых вычислений по формулам (18), (18′), (18″) значительно сокращается по сравнению с вычислительной схемой метода, приведенной в [5, 14] и обеспечивает более высокую точность решения.

В заключение отметим, что непосредственное построение систем уравнений (16), (16′) не требуется, так как в процессе вычислений они не используются.

§6. Применение проекционно-дифференциального метода для решения экстремальных задач

Численные методы решения оптимизационных задач с ограничениями представляют собой различные способы воплощения двух подходов к оптимизации поиска условного экстремума. Первый состоит в том, чтобы непосредственно контролируя соблюдение ограничений задачи, двигаться к её оптимуму по последовательности допустимых или «почти допустимых» точек с монотонно убывающими либо монотонно возрастающими значениями целевой функции (это зависит от того, что ищется – минимум или максимум). Соответствующие методы называются методами спуска [1, 17, 21].

Второй подход заключается в сведении задачи на экстремум при наличии ограничений к последовательности задач безусловной оптимизации конструируемых специальным образом вспомогательных функций. Эти функции принято называть штрафными, а использующие их алгоритмы – методами штрафных функций [1, 7, 21].

Методы последовательной безусловной оптимизации выгодно отличаются от методов спуска простотой реализации и сильными свойствами сходимости. Поэтому на их разработку и разного рода усовершенствования направлялись и направляются в настоящее время значительные усилия [1, 3, 17, 21].

В данной работе существенно используется идея методов последовательной безусловной оптимизации. В связи с этим сначала коротко изложим основную идею этих методов.

Рассмотрим задачу:

$$\min\ f(x) \tag{1}$$

при ограничениях

$$\varphi_i(x) \geq 0, i = 1, 2, \ldots, m, \tag{2}$$

где $x \in E^n$, $m \leq n$, $E^n - n$ - мерное евклидово пространство. Далее везде предполагается, что $f(x)$, $\varphi_i(x)$, $i = \overline{1, m}$ непрерывно дифференцируемыми функциями (вообще говоря, нелинейными).

В методах последовательной безусловной оптимизации общий вид вспомогательной функции для задачи (1)-(2) таков [1, 13, 21]:

$$P(x, r) = f(x) + \Phi(x, r) \tag{3}$$

где r - вектор управляющих параметров, а $\Phi(x, r)$ - вещественная функция, «штрафующее воздействие» которой регулируется вектором r. Точку безусловного локального минимума функции $P(x, r)$ по x будем обозначать через $x(r)$. Различные методы преобразования отличаются друг от друга выбором функции $\Phi(x, r)$ и способами задания управляющих параметров r, позволяющими найти решение задачи (1) – (2) посредством безусловной минимизации $P(x, r)$ по x [1, 13, 17, 21].

Основная идея методов последовательной безусловной минимизации состоит в следующем. Выбирается некая последовательность положительных чисел $\{r_k\}$ и решаются задачи безусловной минимизации функции $P(x, r)$ по x для $k = 1, 2, \ldots$. В результате получается последовательность точек $\{x(r_k)\}$ и $\lim\limits_{x \to \infty} x(r_k) = x^*$, где x^* - точка минимума задачи (1) – (2).

Если функции, определяющие задачу (1)-(2), достаточно гладкие, то желательно, чтобы функция $P(x, r)$ обладала той же степенью гладкости, так

как в этом случае для минимизации можно использовать методы, обладающие высокой скоростью сходимости.

Можно взять, например, [1, 13, 17, 21]:

$$P(x,r) = f(x) - r\sum_{i=1}^{m} \ln \varphi_i(x) \tag{4}$$

или

$$P(x,r) = f(x) - r^2\sum_{i=1}^{m} \frac{1}{\varphi_i(x)}. \tag{5}$$

Различные способы преобразования задачи с ограничениями в задачу безусловной минимизации приведены в [1, 13, 21].

При использовании функции (4), (5) из условия минимума функции $P_k(x,r) = P(x,r_k)$ соответственно имеем следующие системы нелинейных уравнений:

$$f_i(x) \equiv \frac{\partial P_k}{\partial x_i} = \frac{\partial f}{\partial x_i} - \sum_{j=1}^{m} \frac{r_k}{\varphi_j(x(r_k))} \cdot \frac{\partial \varphi_j}{\partial x_i} = 0, \tag{6}$$

$$f_i(x) \equiv \frac{\partial P_k}{\partial x_i} = \frac{\partial f}{\partial x_i} - \sum_{j=1}^{m} \frac{r_k^2}{\varphi_j^2(x(r_k))} \cdot \frac{\partial \varphi_j}{\partial x_i} = 0, \tag{7}$$

$$(i = 1,\ 2,\ ...,\ n)$$

Системы уравнений (6), (7) могут быть решены одним из приближенных методов [2, 3, 9, 11, 23]. Однако решение этих систем уравнений связано со свойственными им принципиальными затруднениями. Дело в том, что с увеличением (в методах внешних штрафных функций [1, 21]) или уменьшением (в методах внутренних штрафных функций [1, 21]) параметра r, функция $P(x,r)$ приобретает ярко выраженную овражную структуру, что сильно затрудняет поиск минимума [1, 21]. В связи с этим для решения систем уравнений вида (6), (7) линейных или нелинейных возникает необходимость привлечения быстросходящихся итерационных процессов, не требующих знания хорошего начального приближения к решению или разработки принципиально новых эффективных методов.

Итерационные процессы, построенные с помощью приближенных способов интегрирования задачи Коши [2, 3, 8, 9] обладают указанными

свойствами. В данной работе для решения системы уравнений (6) и (7) при различных значениях параметра r_k, $k = 1, 2, \ldots$ (или, что то же самое, для отыскания стационарной точки функции $P(x, r_k)$) применяется метод дифференциального спуска [5, 14].

Основные обозначения соответствуют обозначениям, принятым в [5, 14].

Согласно [5, 14] системам уравнений (6), (7) соответствует следующая рекуррентная последовательность задач Коши:

$$\frac{dx_i}{dt_i} = \frac{a_i^{i-1}}{(f_i', a_i^{i-1})},\ x_i^0 = x_i(t_i = t_i^0) = x_{i-1}(t_{i-1} = 0) = \alpha_{0i-1},$$
$$t_i^0 = f_i(\alpha_{0i-1}),\ \alpha_{00} = x_0,\ i = 1, 2, \ldots, n. \tag{8}$$

Вектор-функции $a_i^{i-1}(x) \in E^n$, $i = 1, 2, \ldots, n$ построены из $f_i(x)$, $i = 1, 2, \ldots, n$ так, чтобы

$$\begin{aligned} (f_j', a_i^{i-1}) &= 0,\ j \le i-1, \\ (f_j', a_i^{i-1}) &\ne 0,\ j = i. \end{aligned} \tag{9}$$

При некоторых условиях, накладываемых на $f_i(x)$ и $a_i^{i-1}(x)$ теоремой существования и единственности решения [5, 14], уравнение (8) имеет решение $x_i(t_i)$, $i = 1, 2, \ldots, n$, определенное на конечном интервале $[0, \bar{0}]$ и $\lim\limits_{t_i \to 0} x_i(t_i) = \alpha_{0i}$.

Для задач (8) известны первые интегралы

$$f_j(x_i(t_i)) = 0,\ j \le i-1,\ t_i - f_i(x_i(t_i)) = 0$$

с помощью которых осуществляется контроль точности движения к решению систем (6), (7) по траекториям, определяемым решениями задач (8).

Некоторые способы построения вектор-функции $a_i^{i-1}(x)$ со свойствами (9) приведены в [16].

Системам нелинейных уравнений (6), (7) соответствует нелинейное дифференциальное уравнение спуска (8). Поэтому в этих случаях получение решения уравнения (8) в замкнутом виде невозможно, и для интегрирования уравнения (8) применяется какая-нибудь формула численного

интегрирования. Таким образом, на практике спуск к решениям систем (6), (7) происходит по линиям, определяемым приближенными решениями задач (8).

Интегрируя задачи Коши (8) приближенными способами, можно получить различные итерационные методы решения систем (6), (7). Например, метод Эйлера [2, 3] с шагом $h_i = \frac{f_i(x_{ni-1})}{n_i}$, где n_i - число делений отрезка $[f_i(x_{ni-1}), 0]$ (уравнения (8) решаются с начальными условиями $t_{ni-1} = f_i(x_{ni-1}), x_{ni-1}$) дает итерационный метод

$$x_{s+1i} = x_{si} - \frac{a_i^{i-1}(x_{si})h_i}{(f_i(x_{si}), a_i^{i-1}(x_{si}))}, \quad (i = 1,2,...,n;\ s = 0,1,2,...), \tag{10}$$

а метод Рунге-Кутта [2, 3] с тем же шагом и имеющий порядок погрешности на одном шаге h_i^4 дает итерационный метод

$$x_{s+1i} = x_{si} - \frac{1}{6}(k_{si}^1 + 4k_{si}^2 + k_{si}^3), \tag{11}$$

где

$$k_{si}^1 = \frac{a_i^{i-1}(x_{si-1})h_i}{(f_i'(x_{si-1}, a_i^{i-1}(x_{si-1}))}, \quad k_{si}^2 = \frac{a_i^{i-1}(x_{si-1} + 0{,}5k_{si}^1)h_i}{(f_i'(x_{si-1} + 0{,}5k_{si}^1, a_i^{i-1}(x_{si-1} + 0{,}5k_{si}^1))},$$

$$k_{si}^3 = \frac{a_i^{i-1}(x_{si-1} - k_{si}^1 + 2k_{si}^2)h_i}{(f_i'(x_{si-1} - k_{si}^1 + 2k_{si}^2, a_i^{i-1}(x_{si-1} - k_{si}^1 + 2k_{si}^2))}, \quad (i = 1,2,...,n;\ s = 0,1,2,...).$$

Многообразие способов построения вектор-функции $a_i^{i-1}(x)$ со свойствами (9) порождает многообразие рекуррентных последовательностей задач Коши (8) для каждого конкретного уравнения. А в свою очередь, различные способы интегрирования задач Коши (8) порождает целый класс итерационных процессов, каждый из которых определяет приближенное решение заданного уравнения и имеет свою область сходимости. Поэтому имеются возможности варьировать последовательностями задач Коши (8) и приближенными способами их решения, если тот или иной итерационный процесс не дает желаемого результата.

Некоторые вопросы численной реализации данного метода при решении нелинейных уравнений в конечномерных евклидовых пространствах обсуждаются в [14].

§7. Результаты численных экспериментов.

В этом параграфе, описанные выше различные вычислительные алгоритмы метода дифференциального спуска применяются для решения конкретных плохообусловленных систем линейных алгебраических уравнений и достаточно сложной системы трансцендентных уравнений. Для реализации трех новых вариантов метода дифференциального спуска применительно к системам линейных и нелинейных уравнений разработан комплекс программ на алгоритмическом языке Delphi 7.0. Они приведены в приложении к диссертационной работе. На основе этого комплекса программ выполнены многочисленные численные эксперименты с целью проверки работоспособности и эффективности различных вычислительных алгоритмов метода дифференциального спуска. Результаты их подтвердили высокую эффективность всех различных вариантов вычислительного алгоритма метода. Ниже опишем коротко основные результаты некоторых численных экспериментов, выполненных с использованием трех новых вычислительных алгоритмов метода дифференциального спуска.

Пример 1. Классический пример плохообусловленных матриц дают матрицы Гильберта [12, 24], элементы которых являются точными рациональными числами:

$$H_n = \begin{bmatrix} 1 & 1/2 & \dots & 1/n \\ 1/2 & 1/3 & \dots & 1/(n+1) \\ \dots & \dots & \dots & \dots \\ 1/n & 1/(n+1) & \dots & 1/(2n-1) \end{bmatrix}$$

С увеличением n эти матрицы становятся все более плохообусловленными. Насколько плохообусловленность этих матриц, можно проиллюстрировать

таким примером. При $n=10$ $\|H_{10}\|=1{,}75$, $\|H_{10}^{-1}\|=9{,}5\cdot 10^{12}$, число обусловленности $\mu(H_{10})=1{,}60\cdot 10^{13}$, где $\|H_{10}\|$ - евклидова норма матрицы H_{10}.

Приведем один из случаев, в которых матрицы Гильберта появляются. Пусть на интервале $0\le x\le 1$ дана непрерывная функция $f(x)$ и требуется аппроксимировать ее полиномом степени $n-1$. Запишем этот полином в форме

$$P_{n-1}(x)=\sum_{i=1}^{n} c_i x^{i-1},$$

определяем ошибку аппроксимации как

$$R(f)=\int_0^1 [\sum_{i=1}^{n} c_i x^{i-1} - f(x)]^2 dx.$$

Коэффициенты c_i должны быть определены из условия минимальности величины $R(f)$. Так как ошибка является дифференцируемой функцией неизвестных c_i, то минимум достигается при $\partial R/\partial c_i = 0, i=1,2,...,n$. Расписывая эти произведения приходим к условиям:

$$\frac{\partial R}{\partial c_i}=2\int_0^1 [\sum_{i=1}^{n} c_i x^{i-1} - f(x)]x^{i-1}dx=0,\ i=1,2,...,n$$

Меняя порядок суммирования и интегрирования, получаем

$$\sum_{j=1}^{n}\left[\int_0^1 x^{i+j-2}\,dx\right]c_j=\int_0^1 f(x)x^{i-1}\,dx,\quad i=1,2,...,n \qquad (1)$$

Этим n уравнениям удовлетворяют n неизвестных c_i. Если положим

$$h_{ij}=\int_0^1 x^{i+j-2}\,dx=\frac{1}{i+j-1},\ b_i=\int_0^1 f(x)x^{i-1}\,dx,\quad i=1,2,...,n$$

то уравнения (1) могут быть записаны в виде

$$\sum_{j=1}^{n} h_{ij}c_j=b_i,\ i=1,2,...,n$$

Таким образом, коэффициенты $c=(c_1,c_2,...,c_n)$ полинома $P_{n-1}(x)$ можно найти, решая систему из n уравнений с n неизвестными

$$H_n c=b \qquad (2)$$

где матрица H_n имеет элементы

$$h_{ij} = \frac{1}{i + j - 1}, \quad i, j = 1, 2, ..., n, \tag{3}$$

а вектор $b = (b_1, b_2, ..., b_n)'$ определяется по данной функции $f(x)$ (штрих означает вектора-столбца). Матрица H_n, определенная формулой (3), называется $(n \times n)$-матрицей Гильберта [12, 24].

Плохая обусловленность матриц Гильберта может быть связана с проблемой аппроксимации, которую мы использовали для введения этих матриц. На интервале $0 \le x \le 1$ функции x^i $(i = 0, 1, 2, ..., n-1)$ почти линейно зависимы. Это означает, что строки гильбертовой матрицы почти линейно зависимы, т.е. матрица почти вырождена. В таких случаях небольшие возмущения в исходных данных вызывают изменения в результате. В исходной задаче небольшие ошибки в значениях функции $f(x)$ или ошибки округления при ее вычислении могут привести к большим изменениям в коэффициентах c_i. Поэтому задача аппроксимации не является «хорошо поставленной», если она приводит к матрицам, подобным гильбертовым.

При $n = 10$ для функции $f(x) = \sqrt{x}$ система (2) имеет вид

1.00	0.50	0.33	0.25	0.20	0.17	0.14	0.13	0.11	0.10	-0.666667
0.50	0.33	0.25	0.20	0.17	0.14	0.13	0.11	0.10	0.09	-0.400000
0.33	0.25	0.20	0.17	0.14	0.13	0.11	0.10	0.09	0.08	-0.285714
0.25	0.20	0.17	0.14	0.13	0.11	0.10	0.09	0.08	0.08	-0.222222
0.20	0.17	0.14	0.13	0.11	0.10	0.09	0.08	0.08	0.07	-0.181818
0.17	0.14	0.13	0.11	0.10	0.09	0.08	0.08	0.07	0.07	-0.153846
0.14	0.13	0.11	0.10	0.09	0.08	0.08	0.07	0.07	0.06	-0.133333
0.13	0.11	0.10	0.09	0.08	0.08	0.07	0.07	0.06	0.06	-0.117647
0.11	0.10	0.09	0.08	0.08	0.07	0.07	0.06	0.06	0.06	-0.105263
0.10	0.09	0.08	0.08	0.07	0.07	0.06	0.06	0.06	0.05	-0.095238

(4)

Детерминант матрицы Гильберта порядка n имеет величину [24]

$$\det H_n = \frac{[1!2!3!...(n-1)!]^3}{n!(n+1)!...(2n-1)!},$$

которая быстро стремится к нулю. В частности, отсюда при $n = 10$ имеем $\det H_{10} = 4.01568069\,003321\text{E} - 0054$, т.е. матрица системы (4) почти вырождена.

Поэтому следует ожидать, что погрешность вычислений будет достаточно большой величиной при решении этой системы.

Плохообусловленная система (4) была решена с применением трех различных вычислительных формул (5.13),(5.17), (6.10) метода дифференциального спуска, исходя из нулевого начального приближения $\alpha_{00} = (0,0,0,...,0)'$. На первом шаге выполнения вычислений по этим формулам невязки всех уравнений системы (4) равнялись к нулю: h_{ij}, т.е. получено машинное точное решение плохообусловленной системы (4).

Таким образом, полином $P_9(x)$ наилучшего среднеквадратичного приближения для функции $f(x) = \sqrt{x}$ имеет вид (сохранены десять десятичных знаков после запятой)

$$P_9(x) = 0{,}12149863\,36 + 1{,}8725910028\,x - 0{,}38040981\,46\,x^2 - 6{,}57275566\,94\,x^3 +$$
$$+ 12{,}9746763\,305\,x^4 - 10{,}4761300\,488\,x^5 + 8{,}016225147\,8\,x^6 - 5{,}947694207\,9\,x^7 -$$
$$- 1{,}077286506\,x^8 + 2{,}479411089\,1\,x^9$$

Результаты решения системы (4) с тремя вычислительными алгоритмами метода дифференциального спуска приведены в таблицах №№1-3.

Таблица № 1

k	i	$x^{(k)}$	$f_i(x^{(k)})$	$\|x^{(k)}\|$	$\|f(x^{(k)})\|$	$\max\limits_{1\le i\le 10}\|f_i(x^{(k)})\|$
0	1	0.0000000000	-0.6666666667	0,0000000000	0,9186705281	0.6666666667
	2	0.0000000000	-0.4000000000			
	3	0.0000000000	-0.2857142857			
	4	0.0000000000	-0.2222222222			
	5	0.0000000000	-0.1818181818			
	6	0.0000000000	-0.1538461538			
	7	0.0000000000	-0.1333333333			
	8	0.0000000000	-0.1176470588			
	9	0.0000000000	-0.1052631578			
	10	0.0000000000	-0.0952380952			
1	1	0,1214986336	0.0000000000	4,2286588472	0,0000000000	0.00000000000
	2	1,8725910028	0.0000000000			
	3	-0,3804098146	0.0000000000			
	4	-6,5727556694	0.0000000000			
	5	12,9746763305	0.0000000000			
	6	-10,4761300488	-0.0000000000			
	7	8,0162251478	-0.0000000000			
	8	-5,9476942079	0.0000000000			
	9	-1,077286506	0.0000000000			
	10	2,4794110891	0.0000000000			

Таблица № 2

k	i	$x^{(k)}$	$f_i(x^{(k)})$	$\|x^{(k)}\|$	$\|f(x^{(k)})\|$	$\max_{1\le i\le 10}\|f_i(x^{(k)})\|$
0	1	0.0000000000	-0.6666666667	0,0000000000	0,9186705281	0.6666666667
	2	0.0000000000	-0.4000000000			
	3	0.0000000000	-0.2857142857			
	4	0.0000000000	-0.2222222222			
	5	0.0000000000	-0.1818181818			
	6	0.0000000000	-0.1538461538			
	7	0.0000000000	-0.1333333333			
	8	0.0000000000	-0.1176470588			
	9	0.0000000000	-0.1052631578			
	10	0.0000000000	-0.0952380952			
1	1	0,1214986336	0.0000000000	4,2286588472	0,0000000000	0.00000000000
	2	1,8725910028	0.0000000000			
	3	-0,3804098146	0.0000000000			
	4	-6,5727556694	0.0000000000			
	5	12,9746763305	0.0000000000			
	6	-10,4761300488	-0.0000000000			
	7	8,0162251478	-0.0000000000			
	8	-5,9476942079	0.0000000000			
	9	-1,077286506	0.0000000000			
	10	2,4794110891	0.0000000000			

Таблица № 3

k	i	$x^{(k)}$	$f_i(x^{(k)})$	$\|x^{(k)}\|$	$\|f(x^{(k)})\|$	$\max_{1\le i\le 10}\|f_i(x^{(k)})\|$
0	1	0.0000000000	-0.6666666667	0,0000000000	0,9186705281	0.6666666667
	2	0.0000000000	-0.4000000000			
	3	0.0000000000	-0.2857142857			
	4	0.0000000000	-0.2222222222			
	5	0.0000000000	-0.1818181818			
	6	0.0000000000	-0.1538461538			
	7	0.0000000000	-0.1333333333			
	8	0.0000000000	-0.1176470588			
	9	0.0000000000	-0.1052631578			
	10	0.0000000000	-0.0952380952			
1	1	0,1214986336	0.0000000000	4,2286588472	0,0000000000	0.00000000000
	2	1,8725910028	0.0000000000			
	3	-0,3804098146	0.0000000000			
	4	-6,5727556694	0.0000000000			
	5	12,9746763305	0.0000000000			
	6	-10,4761300488	-0.0000000000			
	7	8,0162251478	-0.0000000000			
	8	-5,9476942079	0.0000000000			
	9	-1,077286506	0.0000000000			
	10	2,4794110891	0.0000000000			

Сравним теперь значения построенного полинома $P_9(x)$ с соответствующими значениями функции $f(x)=\sqrt{x}$ в некоторых точках отрезка $x\in$ ▮,1¯. Результаты вычислений приведены в следующей таблице

x_i	$P_9(x_i)$	$f(x_i)=\sqrt{x_i}$	$P_9(x_i)-f(x_i)$
0.0	0,1214986336	0.0000000000	0,1214986336
0.1	0,2995809996	0.3162277660	-0,0166467665
0.2	0,4460609363	0.4472135955	-0,0011526592
0.3	0,5557337192	0.5477225575	0,0080111617
0.4	0,6369229522	0.6324555320	0,0044674201
0.5	0,7040570356	0.7071067812	-0,0030497456
0.6	0,7696847599	0.7745966692	-0,0049119094
0.7	0,8371853782	0.8366600265	0,0005253516
0.8	0,8985930662	0.8944271910	0,0041658752
0.9	0,9460199582	0.9486832981	-0,0026633399
1.0	1,0101259571	1.0000000000	0,0101259571

Как видно из таблицы, несмотря на то, что рассмотренная задача является очень "плохо поставленной", функция $f(x)=\sqrt{x}$ достаточно хорошо аппроксимирована построенным полиномом $P_9(x)$.

Пример 2. Вычислить с точностью $\varepsilon=10^{-10}$ один из действительных корней системы трансцендентных уравнений:

$$\begin{cases} e^{x_1+x_2}+sh(\dfrac{x_3+x_4}{x_1})-1=0 \\ e^{x_1-x_2}+sh(\dfrac{x_3-x_4}{2x_2})-2.5=0 \\ x_1x_2^2-ch(\dfrac{x_1+x_2+x_3}{x_4})+1=0 \\ 3x_1^2+x_4^3-th(\dfrac{x_2-x_1x_4}{x_3})=0 \end{cases} \qquad (5)$$

Эта система уравнений была исследована и решена методом вариации параметров [22] (стр. 404-405) и получено ее следующее приближенное решение:

$$x_1^*=1{,}146238,\ x_2^*=-0.561509,\ x_3^*=0.606558,\ x_4^*=-1{,}441442;$$
$$f_1(x_1^*,x_2^*,x_3^*,x_4^*)=0{,}000003,\quad f_2(x_1^*,x_2^*,x_3^*,x_4^*)=0{,}000001,$$
$$f_3(x_1^*,x_2^*,x_3^*,x_4^*)=0{,}000000,\ f_4(x_1^*,x_2^*,x_3^*,x_4^*)=0{,}000000$$

В найденном решении все цифры значащие.

Для решения системы уравнений (7.5) нами применены три различные вычислительные алгоритмы (5.13), (5.17), (6.10) метода дифференциального

спуска. Во всех этих формулах в качестве начального приближения к решению системы (5) был принят вектор $\alpha_{00} = (0,0,0,0)$. Выполнение итераций прекратилось по признаку

$$\max_{1 \le i \le 4} \left| f_i(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}) \right| \le \varepsilon = 10^{-12}$$

где k - номер итерации.

По формуле (5.13) приближенное решение системы (5) с заданной точностью получено при $k = 6$, по формуле (5.17) – при $k = 7$, а по формуле (6.10) – при $k = 5$. Результаты вычислений по указанным формула м приведены в таблицах №№ 4-6.

Как известно, что применение многих итерационных методов (например простая итерация, метод Ньютона, метод Эйткена-Стеффенсона [2, 3]) наталкивается на ту трудность, что далеко не всегда можно дать простой способ надежного выбора начального приближения к решению. Между тем, от этого выбора зависит сходимость итерационного процесса. Поэтому проблема выбора начального приближения, обеспечивающего сходимость итерационного метода имеет первостепенное значение.

Рассмотренный в диссертации метод дифференциального спуска и его различные модификации в значительной степени решает эту проблему. Решение этой проблемы (т.е. расширение области выбора начального приближения) осуществляется за счет многообразия способов построения последовательности задач Коши и методов их решения (имеется большое число хороших программ решения обыкновенных дифференциальных уравнений) для каждого заданного уравнения.

Таблица № 4

k	$\bar{i}$	$x^{(k)}$	$f_i(x^{(k)})$	$\|x^{(k)}\|$	$\|f(x^{(k)})\|$	$\max_{1\le i\le 10}\|f_i(x^{(k)})\|$
0	1	0,000000000000000	-0,000000000000000	0,000000000000000	2,575829102905580	0,666666666666667
	2	0,000000000000000	1,484197241278563			
	3	0,000000000000000	-2,059935129392445			
	4	0,000000000000000	0,434420739665387			
1	1	0,457776656894492	0,001695384902792	0,676528439353959	1,628632485268860	1,628631601694020
	2	-0,493979621983397	-0,000060882177894			
	3	0,053078029943388	0,000000000000000			
	4	-0,036029397159463	1,628631601694020			
2	1	0,025927940665665	-1,432645541075746	20,947323588885000	3122,360018155560000	3122,355414854352130
	2	-3,443671242822942	-5,166607480652238			
	3	14,604619523094114	-0,000000000000000			
	4	-14,616289208836960	-3122,355414854352130			
3	1	1,254628620686833	0,010159876062948	1,890366340039500	3,211112118478300	3,210968777353583
	2	-0,386705224506894	0,028588900241331			
	3	-0,047789444714907	-0,000000000000228			
	4	-1,359252427630987	3,210968777353583			
4	1	1,114470784376910	-0,000107487925974	2,076378205527110	0,249335257644108	0,249335234312937
	2	-0,602911469038366	-0,000008995132732			
	3	0,760588503363780	-0,000000000042572			
	4	-1,458528097847678	-0,249335234312937			
5	1	1,146056331172300	0,000000072511953	2,018945227965150	0,001583499759408	0,001583499755959
	2	-0,561759108321373	-0,000000075256996			
	3	0,607493190574071	-0,000000000000004			
	4	-1,441552443349628	-0,001583499755959			
6	1	1,146236840374456	0,000000000000003	2,018616549032250	0,000000000066802	0,000000000066802
	2	-0,561508870174897	-0,000000000000003			
	3	0,606556268926263	-0,000000000000000			
	4	-1,441440654045956	0,000000000066802			

Таблица № 5

k	$\bar{i}$	$x^{(k)}$	$f_i(x^{(k)})$	$\|x^{(k)}\|$	$\|f(x^{(k)})\|$	$\max_{1\le i\le 10}\|f_i(x^{(k)})\|$
0	1	0,000000000000000	0,001915124222788	0,000000000000000	1,628184684860250	0,666666666666667
	2	0,000000000000000	-0,000036975909006			
	3	0,000000000000000	-0,000000000000000			
	4	0,000000000000000	1,628183558124047			
1	1	0,034798376603049	1,492276907400987	14,422134632540600	991,8088763209910 00	991,8042922387494 80
	2	-2,816007320018943	-2,620335003864985			
	3	10,029836257484053	-0,000000000000002			
	4	-9,973426886169629	-991,804292238749480			
2	1	1,652634895392684	1,228423284101271	2,836881218610430	2,568644605311440	1,866947173957269
	2	-0,532852588159430	-1,266262055471302			
	3	0,828289151187570	-0,000000000081521			
	4	-2,084873653810174	-1,866947173957269			
3	1	1,263629252820551	0,023119710879774	1,865583489609440	3,523362199307530	3,516890879121877
	2	-0,363984877518697	0,212191448525747			
	3	-0,149027714775816	-0,000000000000334			
	4	-1,314894907316134	3,516890879121877			
4	1	1,063417928095462	-0,000365055616328	2,161794467118720	0,552856027277828	0,552855895726613
	2	-0,659548621639566	-0,000110418042776			
	3	0,965252235210105	-0,000000000000000			
	4	-1,475053004602044	-0,552855895726613			
5	1	1,144717652669088	0,000015826290519	2,021419148092280	0,013375578810953	0,013375560391251
	2	-0,563621863623831	-0,000015565235242			
	3	0,614473889351094	0,000000000000000			
	4	-1,442396998781142	-0,013375560391251			
6	1	1,146239067006788	0,000000000096967	2,018612497372660	0,000019538820081	0,000019538820080
	2	-0,561505783157581	-0,000000000094434			
	3	0,606544710984354	-0,000000000069957			
	4	-1,441439275472264	0,000019538820080			
7	1	1,146236840373882	0,000000000000000	2,018616549033290	0,000000000057973	0,000000000057973
	2	-0,561508870175692	-0,000000000000000			
	3	0,606556268929236	0,000000000000000			
	4	-1,441440654046308	0,000000000057973			

Таблица № 6

k	$\bar{i}$	$x^{(k)}$	$f_i(x^{(k)})$	$\|x^{(k)}\|$	$\|f(x^{(k)})\|$	$\max_{1\le i\le 10}\|f_i(x^{(k)})\|$
0	1	0,000000000000000	-0,000000000000672	0,000000000000000	2,500310014587090	2,434721958680290
	2	0,000000000000000	0,534750953053461			
	3	0,000000000000000	-2,434721958680290			
	4	0,000000000000000	0,194217844630509			
1	1	1,650196667491419	1,246505289747832	3,059289659880020	3,504224162843040	3,039668541865953
	2	-0,607891502440316	-1,219108978588154			
	3	1,247127599779241	-0,000000000000000			
	4	-2,170540221414664	-3,039668541865953			
2	1	1,262393084698435	0,023300618002652	1,865390838016150	3,507626512000980	3,501139261211146
	2	-0,366745955362109	0,211954482138967			
	3	-0,139141197007819	-0,000000000000398			
	4	-1,316124541896000	3,501139261211146			
3	1	1,066351879145991	-0,000355706945411	2,157305677003400	0,538077669947293	0,538077543040939
	2	-0,656589758387807	-0,000100217279018			
	3	0,954832446252547	-0,000000000000000			
	4	-1,474464764799934	-0,538077543040939			
4	1	1,144865959994951	0,000012031643775	2,021140277506130	0,012059634351794	0,012059622512797
	2	-0,563414307858579	-0,000011865385019			
	3	0,613695155684661	-0,000000000099701			
	4	-1,442301122717389	-0,012059622512797			
5	1	1,146236840422338	0,000000000059944	2,018616549075920	0,000000000099022	0,000000000059944
	2	-0,561508870156191	-0,000000000051249			
	3	0,606556268885478	-0,000000000000342			
	4	-1,441440654093493	0,000000000059879			

Заключение

Исследования по усовершенствованию метода дифференциального спуска решения конечномерных уравнений и разработка его новых вычислительных алгоритмов, позволяют сделать следующие выводы.

1. Применение многих итерационных методов (например, метод простых итераций, метод Ньютона, метод Эйткена-Стеффенсона и др.[2, 3, 8, 9, 12] наталкивается на ту трудность , что далеко не всегда можно дать простой способ надежного выбора начального приближения. Между тем, от этого зависит сходимость итерационного процесса. Поэтому проблема выбора начального приближения, обеспечивающего сходимость итерационного метода имеет первостепенное значение. Метод дифференциального спуска и его новые варианты в значительной степени решает эту проблему.

2. При реализации обычной схемы метода спуска (например, метода наискорейшего спуска [2, 3, 9, 10]), на каждом шаге итерационного процесса приходится находить минимум функции одной независимой переменной, что требует выполнения большой вычислительной работы. Рассмотренные в

работе различные методы дифференциального спуска избавляет вычислителя от этой трудности и отличается большей алгоритмичностью.

3. Методы спуска, основанные на идее минимизации некоторого вещественного функционала $\varphi(x) \geq 0$, при неудачном выборе начального приближения при наличии локальных минимумов $\varphi(x)$, могут не сходиться к решению. А в методах дифференциального спуска никакая связь решаемого уравнения с вариационными проблемами не используется. Поэтому функционалы $f_i(x)$ метода дифференциального спуска не связаны вовсе с вариационными проблемами и указанное выше явление, характерное для известных методов спуска, отсутствует в методах дифференциального спуска, в частности, в проекционно-дифференциальном методе.

4. Многообразие способов построения вектор-функции $a_i^{i-1}(x)$ порождает многообразность рекуррентных последовательностей задач Коши для каждой системы линейных и нелинейных уравнений. а в свою очередь, различные способы приближенного интегрирования задач Коши дифференциального метода порождает целый класс итерационных процессов, каждый из которых определяет приближенное решение заданной системы уравнений и имеет свою область сходимости. Поэтому имеются возможности варьировать последовательностями задач Коши и приближенными способами их решения, если тот или иной итерационный процесс не дает желаемый результат.

5. Принципиальное отличие проекционно-дифференциального метода от проекционных методов (например, методов Галеркина-Петрова, Ритца, наименьших квадратов [2, 3, 9, 10]) заключается в том, что последовательные приближения к решению получаются не в априорно выбранной форме, а в форме, определяемой самой задачей, т.е. аппроксимируется не все пространство, в котором ищется решение, а лишь только само решение.

В проекционных методах две части задачи - получение системы линейных уравнений и их решение – независимы. А в рассматриваемом методе эти части сливаются в единый процесс последовательных

приближений, когда номер i последней итерации заранее неизвестен, а находится во время счета путем последовательного расширения исходного базиса.

Литература

1. Базара М., Шетти К. Нелинейное программирование. Теория и алгоритмы. – М.: Мир, 1982.
2. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. – М.: Наука, 1987.
3. Березин Н. С. и Жидков Н. П. Методы вычислений, т. 1., т.2. – М.: Физматиз, 1962.
4. Галанов Б.А. О некоторых итерационных методах решения нелинейных уравнений // В сб. "Вычислительная и прикладная математика", вып.3, Киев: изд-во КГУ, 1967
5. Галанов Б. А. О сходимости одного непрерывного метода и его аппроксимаций. Сб. "Математические методы в кибернетической технике", вып. 7, Киев, изд-во АН УССР, 1970.
6. Давиденко Д. Ф. О приближенном решении систем нелинейных уравнений. // Укр. мат. журн. т. 5, № 2, 1953.
7. Загускин В. Л. Вычислительная схема для ускоренного вычисления всех корней уравнения. // Математическое просвещение, вып. 6, 1961.
8. Исроилов М. Хисоблаш методлари, 2-кисм. – Тошкент: IQTISOD-MOLIYA, 2008.
9. Калиткин Н. Н. Численные методы. – М.: Наука, 1978.
10. Канторович Л. В. и Акилов Г. П. Функциональный анализ в нормированных пространствах. – М.: Физматгиз, 1959.
11. Ортега Дж., Рейнболдт В. Итерационные методы решения нелинейных систем уравнений со многими неизвестными. – М.: Мир, 1975.
12. Ортега Дж., Пул У. Введение в численные методы решения дифференциальных уравнений. – М.: Наука, 1986.

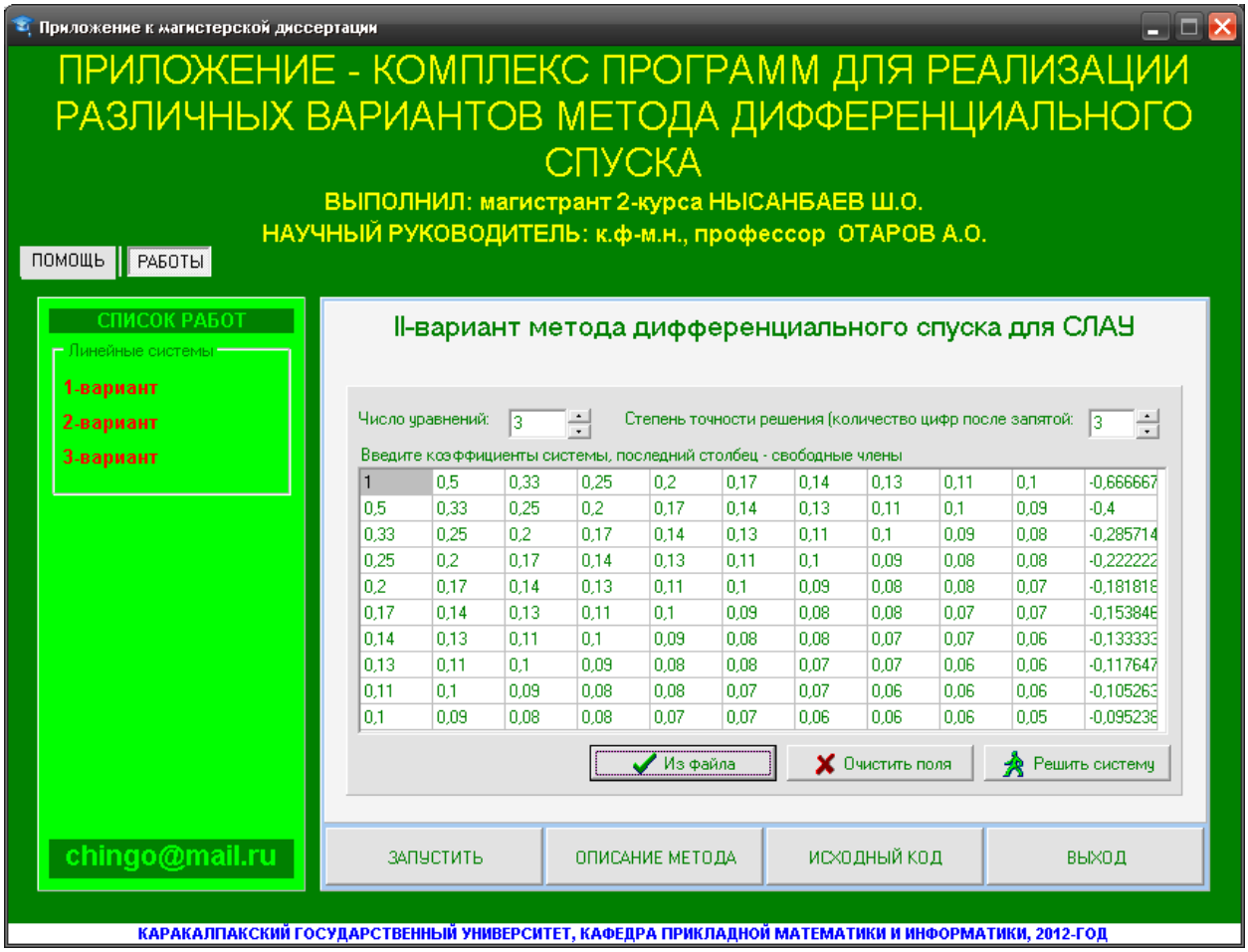
13. Отаров А.О. Метод дифференциального спуска в задачах на условный экстремум // Журн. "Вестник КК филиала АН УзССР", №1, 1983.
14. Отаров А. О. Численная реализация одного метода дифференциального спуска при решении нелинейных уравнений. // Журн. "Вестник КК филиала АН УзССР", № 2, 1983.
15. Отаров А.О. Упрощенные алгоритмы метода дифференциального спуска // Журн. "Вестник КК филиала АН УзССР", №3, 1987.
16. Отаров А.О. Некоторые способы построения правых частей рекуррентной последовательности задач Коши в методе дифференциального спуска // Журн. "Вестник КК филиала АН УзССР", №4, 1994.
17. Пшеничный Б. Н., Данилин Ю. М. Численные методы в экстремальных задачах. – М.: Наука, 1975.
18. Рыбаков Л. М. Метод последовательного вычисления всех действительных корней уравнения. // Математическое просвещение, вып. 6, 1961.
19. Самарский А.А., Гулин А.В. Численные методы. – М: Наука, 1989.
20. Фаддеев Д. К. и Фаддеева В. Н. Вычислительные методы линейной алгебры. – М.: Физматгиз, 1963.
21. Фиакко А., Мак – Кормик Г. Нелинейное программирование. Методы последовательной безусловной минимизации. – М.: Мир, 1972.
22. Фильчаков П.Ф. Справочник по высшей математике. – Киев: Наукова думка, 1974.
23. Форсайт Д., Молер К. Численное решение систем линейных алгебраических уравнений. – М.: Мир, 1969.
24. Хемминг Р.В. Численные методы. – М.: Наука, 1972.

ПРИЛОЖЕНИЯ

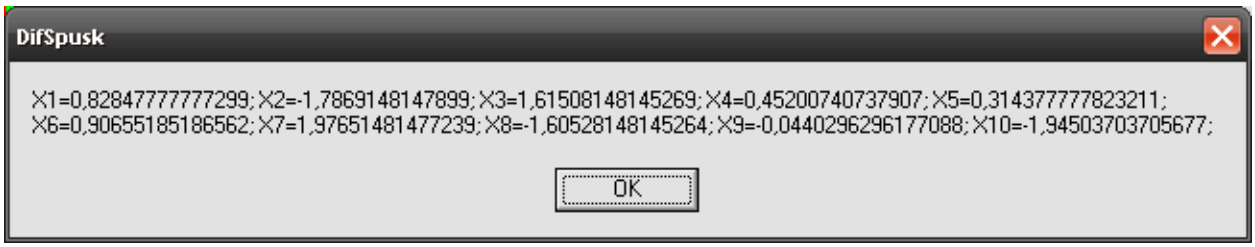
Программный комплекс, составленный на алгоритмическом языке Delphi для реализации различных вариантов метода дифференциального спуска.

I-часть. Программы, реализующие различные варианты метода дифференциального спуска применительно к системам линейных алгебраических уравнений

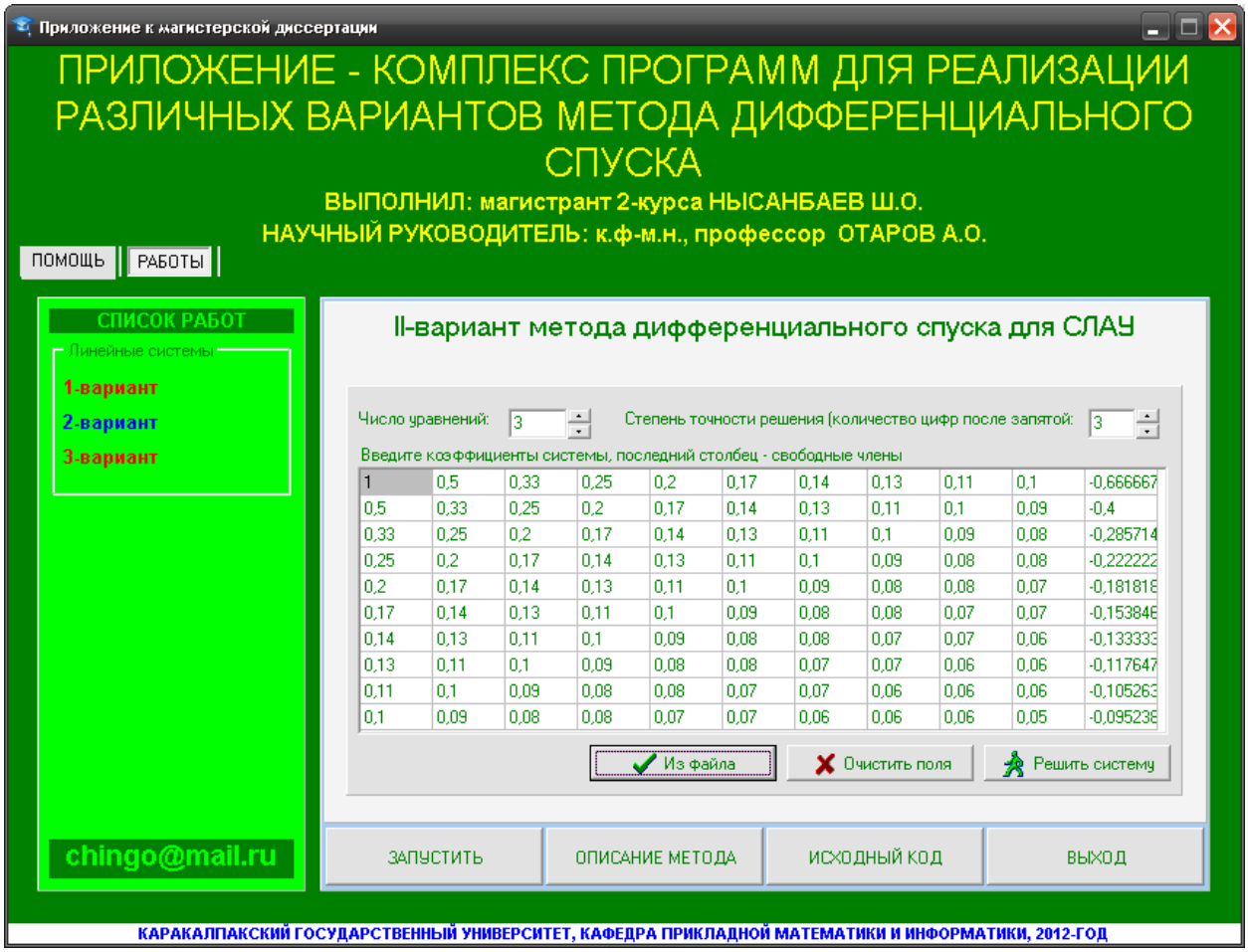
1) Программа реализации 1 варианта метода дифференциального спуска



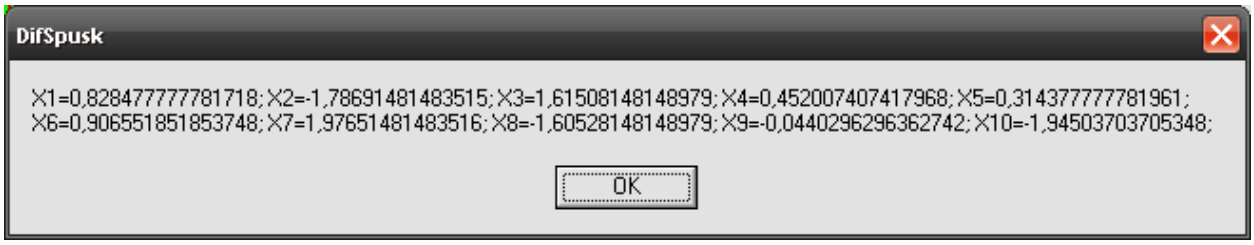
Результат работы программы:



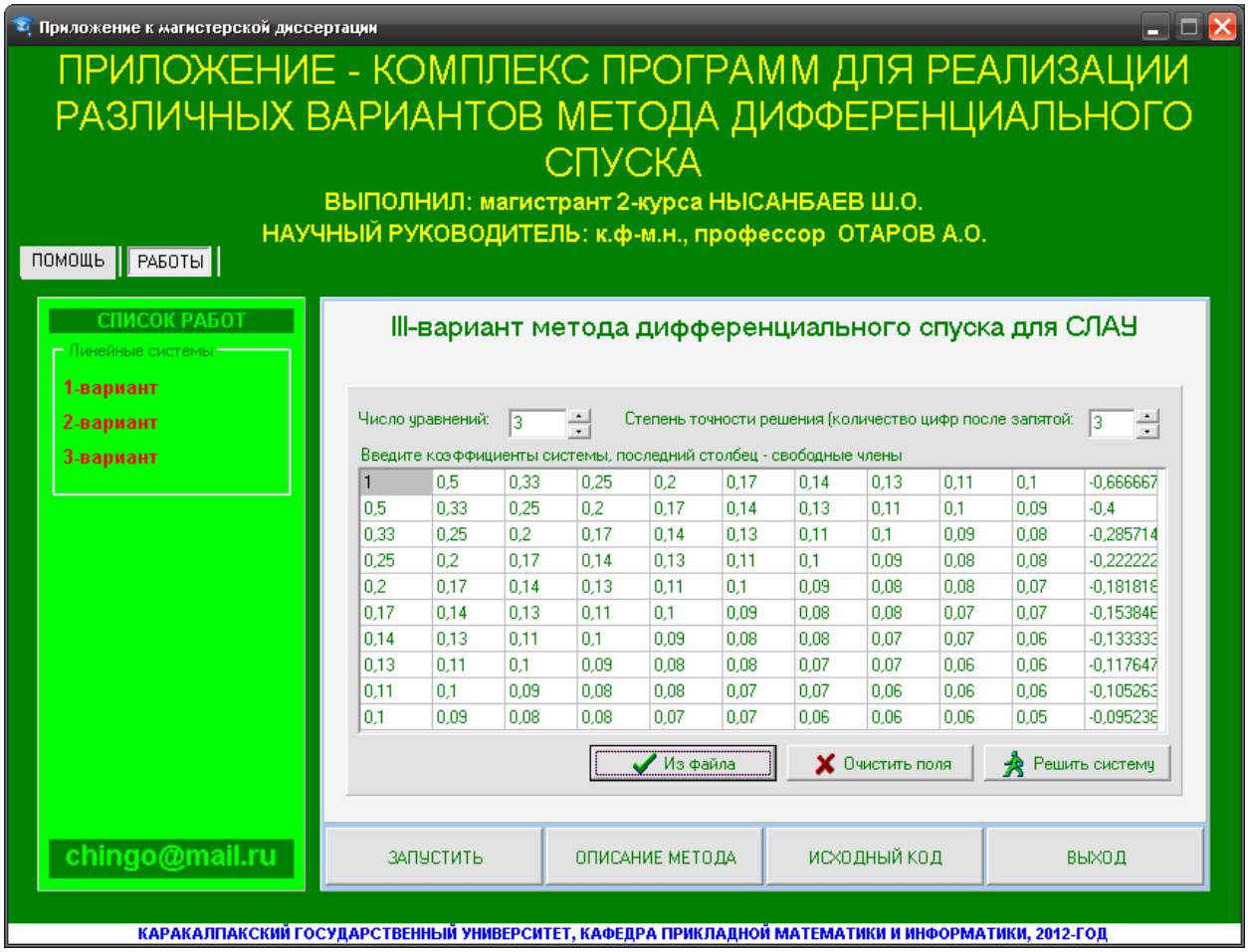
2) Программа реализации 2 варианта метода дифференциального спуска



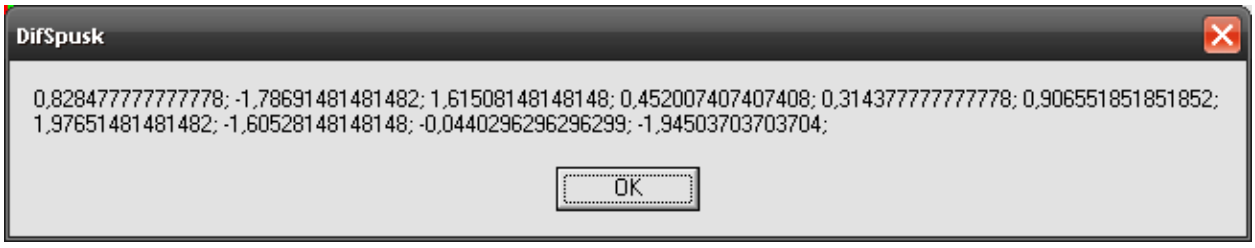
Результат работы программы:



3) Программа реализации 3 варианта метода дифференциального спуска



Результат работы программы:



II-часть

Программы, реализующие различные варианты метода дифференциального спуска применительно к системам нелинейных уравнений

1) Программа, реализующая I-вариант метода дифференциального спуска

```
program Project1;
{$APPTYPE CONSOLE}
uses
  SysUtils;
label 1,2,11,22,33,44;
var
h,x:array[1..4] of extended;
ind,i,k:integer;
eps,s,s1,s2:extended;
f:array[1..4,1..4] of extended;
a:array[1..10,1..4] of extended;
fl:textfile;
function sh(e:extended):extended;
var ee:extended;
begin
 ee:=(exp(e)-exp(-e))/2;
 sh:=ee;
end;
function ch(e:extended):extended;
var ee:extended;
begin
 ee:=(exp(e)+exp(-e))/2;
 ch:=ee;
end;
function summ(i:integer):integer;
var v,l:integer;
begin
 l:=0;
 for v:=0 to i do l:=l+v;
 summ:=l;
end;
procedure calc;
var i2,j2:integer;
begin
 f[1,1]:=exp(x[1]+x[2])-(x[3]+x[4])*ch((x[3]+x[4])/x[1])/sqr(x[1]);
 f[1,2]:=exp(x[1]+x[2]);
```

```
 f[1,3]:=ch((x[3]+x[4])/x[1])/x[1];
 f[1,4]:=ch((x[3]+x[4])/x[1])/x[1];
 f[2,1]:=exp(x[1]-x[2]);
 f[2,2]:=-exp(x[1]-x[2])-2*(x[3]-x[4])*ch((x[3]-x[4])/(2*x[2]))/(4*sqr(x[2]));
 f[2,3]:=ch((x[3]-x[4])/(2*x[2]))/(2*x[2]);
 f[2,4]:=-ch((x[3]-x[4])/(2*x[2]))/(2*x[2]);
 f[3,1]:=sqr(x[2])-sh((x[1]+x[2]+x[3])/x[4])/x[4];
 f[3,2]:=2*x[1]*x[2]-sh((x[1]+x[2]+x[3])/x[4])/x[4];
 f[3,3]:=-sh((x[1]+x[2]+x[3])/x[4])/x[4];
 f[3,4]:=(x[1]+x[2]+x[3])*sh((x[1]+x[2]+x[3])/x[4])/sqr(x[4]);
 f[4,1]:=6*x[1]+x[4]/(x[3]*sqr(ch((x[2]-x[1]*x[4])/x[3])));
 f[4,2]:=-1/(x[3]*sqr(ch((x[2]-x[1]*x[4])/x[3])));
 f[4,3]:=(x[2]-x[4]*x[1])/(sqr(x[3])*sqr(ch((x[2]-x[1]*x[4])/x[3])));
 f[4,4]:=12*sqr(x[4])+x[1]/(x[3]*sqr(ch((x[2]-x[1]*x[4])/x[3])));
 {assignfile(fl,'temp.txt');
 rewrite(fl);
 for i2:=1 to 4 do
  begin
   for j2:=1 to 4 do write(fl,f[i2,j2]:0:2,' ');
   writeln(fl);
  end;
 closefile(fl);
 assignfile(fl,'temp.txt');
 reset(fl);
 for i2:=1 to 4 do
  begin
   for j2:=1 to 4 do read(fl,f[i2,j2]);
   readln(fl);
  end;
 closefile(fl);}
end;
procedure find_a;
var i1,j1,n1:integer;
function summ(i:integer):integer;
var v,l:integer;
begin
 l:=0;
 for v:=0 to i do l:=l+v;
 summ:=l;
end;
begin
 for i1:=1 to 4 do
 for j1:=1 to 4 do
 a[summ(i1-1)+1,j1]:=f[i1,j1];
 for i1:=1 to 4 do
```

```
 if i1-1>0 then
  begin
   ind:=summ(i1-1)+1;
   for j1:=1 to i1-1 do
    begin
     inc(ind);
     s1:=0;
     s2:=0;
     for n1:=1 to 4 do s1:=s1+f[j1,n1]*a[ind-1,n1];
     for n1:=1 to 4 do s2:=s2+f[j1,n1]*a[summ(j1),n1];
     s:=s1/s2;
     for n1:=1 to 4 do a[ind,n1]:=a[ind-1,n1]-s*a[summ(j1),n1]
    end;
  end;
{ assignfile(fl,'temp.txt');
 rewrite(fl);
 for i1:=1 to 10 do
  begin
   for j1:=1 to 4 do write(fl,a[i1,j1]:0:2,' ');
   writeln(fl);
  end;
 closefile(fl);
 assignfile(fl,'temp.txt');
 reset(fl);
 for i1:=1 to 10 do
  begin
   for j1:=1 to 4 do read(fl,a[i1,j1]);
   readln(fl);
  end;
 closefile(fl);}
end;
Begin
 eps:=0.0000000001;
 for i:=1 to 4 do x[i]:=1 ;
k:=0;
 11:
 inc(k);
 repeat
  calc;
  find_a;
  s:=0;
  h[1]:=exp(x[1]+x[2])+sh((x[3]+x[4])/x[1])-1;
  h[2]:=exp(x[1]-x[2])+sh((x[3]-x[4])/(2*x[2]))-2.5;
  h[3]:=x[1]*sqr(x[2])-ch((x[1]+x[2]+x[3])/x[4])+1;
  h[4]:=3*sqr(x[1])+x[4]*sqr(x[4])-sh((x[2]-x[1]*x[4])/x[3])/ch((x[2]-x[1]*x[4])/x[3]);
```

```
 for i:=1 to 4 do s:=s+f[1,i]*a[1,i];
 for i:=1 to 4 do x[i]:=x[i]-h[1]*a[1,i]/s;
 writeln('h1 = ',h[1]);
until (abs(h[1])<=eps) ;
if (abs(h[1])<=eps) and (abs(h[2])<=eps) and (abs(h[3])<=eps)  and (abs(h[4])<=eps)  then goto 22;
assignfile(fl,'rezult.txt');
rewrite(fl);
writeln(fl,'1');
writeln(fl,'-------------------------------------');
for i:=1 to 4 do writeln(fl,h[i]:0:20);
writeln(fl,'-------------------------------------');
writeln(fl);
closefile(fl);
repeat
 calc;
 find_a;
 h[1]:=exp(x[1]+x[2])+sh((x[3]+x[4])/x[1])-1;
 h[2]:=exp(x[1]-x[2])+sh((x[3]-x[4])/(2*x[2]))-2.5;
 h[3]:=x[1]*sqr(x[2])-ch((x[1]+x[2]+x[3])/x[4])+1;
 h[4]:=3*sqr(x[1])+x[4]*sqr(x[4])-sh((x[2]-x[1]*x[4])/x[3])/ch((x[2]-x[1]*x[4])/x[3]);
 s:=0;
 for i:=1 to 4 do s:=s+f[2,i]*a[summ(2),i];
 for i:=1 to 4 do x[i]:=x[i]-h[2]*a[summ(2),i]/s;
 writeln('h2 = ',h[2]:0:20,'; h1= ',h[1]:0:20);
until  abs(h[2])<=eps;
if (abs(h[1])<=eps) and (abs(h[2])<=eps) and (abs(h[3])<=eps)  and (abs(h[4])<=eps)  then goto 22;
assignfile(fl,'rezult.txt');
append(fl);
writeln(fl,'2');
writeln(fl,'-------------------------------------');
for i:=1 to 4 do writeln(fl,h[i]:0:20);
writeln(fl,'-------------------------------------');
writeln(fl);
closefile(fl);
repeat
 calc;
 find_a;
 h[1]:=exp(x[1]+x[2])+sh((x[3]+x[4])/x[1])-1;
 h[2]:=exp(x[1]-x[2])+sh((x[3]-x[4])/(2*x[2]))-2.5;
 h[3]:=x[1]*sqr(x[2])-ch((x[1]+x[2]+x[3])/x[4])+1;
 h[4]:=3*sqr(x[1])+x[4]*sqr(x[4])-sh((x[2]-x[1]*x[4])/x[3])/ch((x[2]-x[1]*x[4])/x[3]);
 s:=0;
 for i:=1 to 4 do s:=s+f[3,i]*a[summ(3),i];
 for i:=1 to 4 do x[i]:=x[i]-h[3]*a[summ(3),i]/s;
 writeln('h3 = ',h[3]);
```

```
 until abs(h[3])<=eps;
 if (abs(h[1])<=eps) and (abs(h[2])<=eps) and (abs(h[3])<=eps)  and (abs(h[4])<=eps)  then goto 22;
 assignfile(fl,'rezult.txt');
 append(fl);
 writeln(fl,'3');
 writeln(fl,'-------------------------------------');
 writeln(fl);
 for i:=1 to 4 do writeln(fl,h[i]:0:20);
 writeln(fl);
 closefile(fl);
 repeat
  calc;
  find_a;
  h[1]:=exp(x[1]+x[2])+sh((x[3]+x[4])/x[1])-1;
  h[2]:=exp(x[1]-x[2])+sh((x[3]-x[4])/(2*x[2]))-2.5;
  h[3]:=x[1]*sqr(x[2])-ch((x[1]+x[2]+x[3])/x[4])+1;
  h[4]:=3*sqr(x[1])+x[4]*sqr(x[4])-sh((x[2]-x[1]*x[4])/x[3])/ch((x[2]-x[1]*x[4])/x[3]);
  s:=0;
  for i:=1 to 4 do s:=s+f[4,i]*a[summ(4),i];
  for i:=1 to 4 do x[i]:=x[i]-h[4]*a[summ(4),i]/s;
  writeln('h4 = ',h[4]:0:20);
 until ( abs(h[4])<=eps);
 if (abs(h[1])<=eps) and (abs(h[2])<=eps) and (abs(h[3])<=eps)  and (abs(h[4])<=eps)  then goto 22;
 if (abs(h[1])>eps) or (abs(h[2])>eps) or (abs(h[3])>eps) then goto 11;
22:
 assignfile(fl,'rezult.txt');
 append(fl);
 writeln(fl,'4');
 writeln(fl,'-------------------------------------');
 writeln(fl);
 for i:=1 to 4 do writeln(fl,h[i]:0:20);
 writeln(fl);
 writeln(fl,'X');
 writeln(fl,'-------------------------------------');
 writeln(fl);
 for i:=1 to 4 do writeln(fl,x[i]:0:20);
 writeln(fl,'-------------------------------------');
 writeln(fl,k,' итераций.');
 closefile(fl);
End.
```

Результаты работы программы:

Значения X:
1.1462368403744560800
-0.5615088701748974500
0.6065562689262625500
-1.4414406540459555800
Проверяем:
0.0000000000000026800
-0.0000000000000025000
-0.0000000000000001000
0.0000000006680155000
Число итераций: 6;
Точность вычислений: 1^-10;

2) Программа, реализующая II-вариант метода дифференциального спуска

```
{$N+}
uses crt;
var
tst:array[1..10] of real;
gr:array[1..4,1..4] of real;
m:array[1..10,1..11] of real;
a:array[1..55,1..10] of real;
x:array[1..4] of real;
i,j,k,l,n:integer;
alpha:array[0..10,1..10] of real;
eps,s,s1,s2:real;
flag:boolean;
f:text;
procedure find_gr;
function sh(e:extended):extended;
var ee:extended;
begin
 ee:=(exp(e)-exp(-e))/2;
 sh:=ee;
end;
function ch(e:extended):extended;
var ee:extended;
begin
 ee:=(exp(e)+exp(-e))/2;
 ch:=ee;
end;

begin
```

```
 gr[1,1]:=exp(x[1]+x[2])-(x[3]+x[4])*ch((x[3]+x[4])/x[1])/sqr(x[1]);
 gr[1,2]:=exp(x[1]+x[2]);
 gr[1,3]:=ch((x[3]+x[4])/x[1])/x[1];
 gr[1,4]:=ch((x[3]+x[4])/x[1])/x[1];
 gr[2,1]:=exp(x[1]-x[2]);
 gr[2,2]:=-exp(x[1]-x[2])-2*(x[3]-x[4])*ch((x[3]-x[4])/(2*x[2]))/(4*sqr(x[2]));
 gr[2,3]:=ch((x[3]-x[4])/(2*x[2]))/(2*x[2]);
 gr[2,4]:=-ch((x[3]-x[4])/(2*x[2]))/(2*x[2]);
 gr[3,1]:=sqr(x[2])-sh((x[1]+x[2]+x[3])/x[4])/x[4];
 gr[3,2]:=2*x[1]*x[2]-sh((x[1]+x[2]+x[3])/x[4])/x[4];
 gr[3,3]:=-sh((x[1]+x[2]+x[3])/x[4])/x[4];
 gr[3,4]:=(x[1]+x[2]+x[3])*sh((x[1]+x[2]+x[3])/x[4])/sqr(x[4]);
 gr[4,1]:=6*x[1]+x[4]/(x[3]*sqr(ch((x[2]-x[1]*x[4])/x[3])));
 gr[4,2]:=-1/(x[3]*sqr(ch((x[2]-x[1]*x[4])/x[3])));
 gr[4,3]:=(x[2]-x[4]*x[1])/(sqr(x[3])*sqr(ch((x[2]-x[1]*x[4])/x[3])));
 gr[4,4]:=12*sqr(x[4])+x[1]/(x[3]*sqr(ch((x[2]-x[1]*x[4])/x[3])));
end;

function sk(i1,i2,l:integer):real;
var
 s:real;
 ii:integer;
begin
 s:=0;
 if l=0 then
  begin
   s:=0;
   for ii:=1 to 4 do s:=s+m[i1,ii]*a[i2,ii];
   sk:=s;
  end
  else if l=1 then
  begin
   s:=0;
   for ii:=1 to 4 do s:=s+m[i1,ii]*m[i2,ii];
   sk:=s;
  end
 else if l=2 then
  begin
   s:=0;
   for ii:=1 to 4 do s:=s+m[i1,ii]*alpha[i2,ii];
   s:=s+m[i1,5];
   sk:=s;
  end;

end;
```

```
procedure calc;
var
 ii,jj:integer;ss:real;
begin
 for ii:=1 to 4 do
  begin
   ss:=0;
   for jj:=1 to 4 do ss:=ss+m[ii,jj]*alpha[4,jj];
   ss:=ss+m[ii,5];
   tst[ii]:=ss;
  end;
  ss:=tst[1];
  for ii:=1 to 4 do if abs(tst[ii])>=abs(ss) then ss:=abs(tst[ii]);
  if ss<= eps then flag:=true else flag:=false;
  writeln(ss:0:20);
end;

function summ(i:integer):integer;
var
s,ii:integer;
begin
 s:=0;
 for ii:=1 to i do s:=s+ii;
 summ:=s;
end;
begin
 flag:=false;
 eps:=0.0000000001;
 for i:=1 to 4 do x[i]:=1;
 find_gr;
 for i:=1 to 4 do
 for j:=1 to 4 do m[i,j]:=gr[i,j];
 m[1,5]:=-1;
 m[2,5]:=-2.5;
 m[3,5]:=1;
 m[4,5]:=0;
 for i:=1 to 4 do
 for j:=1 to 4 do
 a[summ(i-1)+1,j]:=m[i,j];
 for i:=1 to 4 do
 if i-1>0 then
  begin
   k:=summ(i-1)+1;
   for j:=1 to i-1 do
    begin
```

```
    inc(k);
    s1:=0;
    s2:=0;
    for n:=1 to 4 do s1:=s1+m[j,n]*a[k-1,n];
    for n:=1 to 4 do s2:=s2+m[j,n]*a[summ(j),n];
    s:=s1/s2;
    for n:=1 to 4 do a[k,n]:=a[k-1,n]-s*a[summ(j),n]
   end;
 end;
assign(f,'mass_A.txt');
rewrite(f);
for i:=1 to 10 do
 begin
  for j:=1 to 10 do write(f,a[i,j]:0:5,'; ');
  writeln(f);
 end;
close(f);
for i:=1 to 4 do alpha[0,i]:=0;
repeat
for i:=1 to 4 do
for j:=1 to 4 do alpha[i,j]:=alpha[i-1,j]-a[summ(i),j]*sk(i,i-1,2)/sk(i,summ(i),0);
calc;
if flag=false then for i:=1 to 4 do alpha[0,i]:=alpha[4,i];
until flag or keypressed;
assign(f,'alpha0.txt');
rewrite(f);
begin
  for j:=1 to 4 do
   begin
    write(f,alpha[4,j]:0:6,' ',tst[j]:0:20);
    writeln(f);
   end;
end;
close(f);

end.
```

Результаты работы программы:

Значения X:
1.1462368403758610132
-0.56150887017489701124
0.60655626892626254497
-1.44144065404595549713
Проверяем:

0.00000000000001268000
-0.00000000000000254000
-0.00000000000000000100
0.00000000006680147000
Число итераций: 12;
Точность вычислений: 1^-10;

3) Программа, реализующая III-вариант метода дифференциального спуска

```
{$N+}
program ot;
uses crt;
label 1;
var
h,x:array[1..4] of extended;
a:array[1..10,1..4] of extended;
mas:array[1..4,1..5] of extended;
gr:array[1..4,1..4] of extended;
alpha:array[0..4,1..4] of extended;
tst,znach:array[1..4] of extended;
razn:array[1..3] of extended;
eps:extended;
sss,s,s1,s2:extended;
index,i,j,l,kk,m,n,ind:integer;
c:text;
stroka1,stroka2:string;
procedure find_gr;
function sh(e:extended):extended;
var ee:extended;
begin
 ee:=(exp(e)-exp(-e))/2;
 sh:=ee;
end;
function ch(e:extended):extended;
var ee:extended;
begin
 ee:=(exp(e)+exp(-e))/2;
 ch:=ee;
end;
begin
 gr[1,1]:=exp(x[1]+x[2])-(x[3]+x[4])*ch((x[3]+x[4])/x[1])/sqr(x[1]);
 gr[1,2]:=exp(x[1]+x[2]);
 gr[1,3]:=ch((x[3]+x[4])/x[1])/x[1];
 gr[1,4]:=ch((x[3]+x[4])/x[1])/x[1];
 gr[2,1]:=exp(x[1]-x[2]);
 gr[2,2]:=-exp(x[1]-x[2])-2*(x[3]-x[4])*ch((x[3]-x[4])/(2*x[2]))/(4*sqr(x[2]));
 gr[2,3]:=ch((x[3]-x[4])/(2*x[2]))/(2*x[2]);
 gr[2,4]:=-ch((x[3]-x[4])/(2*x[2]))/(2*x[2]);
 gr[3,1]:=sqr(x[2])-sh((x[1]+x[2]+x[3])/x[4])/x[4];
 gr[3,2]:=2*x[1]*x[2]-sh((x[1]+x[2]+x[3])/x[4])/x[4];
 gr[3,3]:=-sh((x[1]+x[2]+x[3])/x[4])/x[4];
 gr[3,4]:=(x[1]+x[2]+x[3])*sh((x[1]+x[2]+x[3])/x[4])/sqr(x[4]);
 gr[4,1]:=6*x[1]+x[4]/(x[3]*sqr(ch((x[2]-x[1]*x[4])/x[3])));
```

```
 gr[4,2]:=-1/(x[3]*sqr(ch((x[2]-x[1]*x[4])/x[3])));
 gr[4,3]:=(x[2]-x[4]*x[1])/(sqr(x[3])*sqr(ch((x[2]-x[1]*x[4])/x[3])));
 gr[4,4]:=12*sqr(x[4])+x[1]/(x[3]*sqr(ch((x[2]-x[1]*x[4])/x[3])));
 for i:=1 to 4 do
 for j:=1 to 4 do mas[i,j]:=gr[i,j];
 mas[1,5]:=-1;
 mas[2,5]:=-2.5;
 mas[3,5]:=1;
 mas[4,5]:=0;
end;
procedure find_h;
function sh(e:extended):extended;
var ee:extended;
begin
 ee:=(exp(e)-exp(-e))/2;
 sh:=ee;
end;
function ch(e:extended):extended;
var ee:extended;
begin
 ee:=(exp(e)+exp(-e))/2;
 ch:=ee;
end;
begin
  h[1]:=exp(x[1]+x[2])+sh((x[3]+x[4])/x[1])-1;
  h[2]:=exp(x[1]-x[2])+sh((x[3]-x[4])/(2*x[2]))-2.5;
  h[3]:=x[1]*sqr(x[2])-ch((x[1]+x[2]+x[3])/x[4])+1;
  h[4]:=3*sqr(x[1])+x[4]*sqr(x[4])-sh((x[2]-x[1]*x[4])/x[3])/ch((x[2]-x[1]*x[4])/x[3]);
end;

procedure find_a;
function summ(i:integer):integer;
var v,l:integer;
begin
 l:=0;
 for v:=0 to i do l:=l+v;
 summ:=l;
end;

begin
 for i:=1 to 4 do
 for j:=1 to 4 do
 a[summ(i-1)+1,j]:=gr[i,j];
 for i:=1 to 4 do
 if i-1>0 then
```

```
 begin
  ind:=summ(i-1)+1;
  for j:=1 to i-1 do
   begin
    inc(ind);
    s1:=0;
    s2:=0;
    for n:=1 to 4 do s1:=s1+gr[j,n]*a[ind-1,n];
    for n:=1 to 4 do s2:=s2+gr[j,n]*a[summ(j),n];
    s:=s1/s2;
    for n:=1 to 4 do a[ind,n]:=a[ind-1,n]-s*a[summ(j),n]
   end;
 end;

end;
procedure calc;
function summ(i:integer):integer;
var v,l:integer;
begin
 l:=0;
 for v:=0 to i do l:=l+v;
 summ:=l;
end;
var
sum,sum1,sum2:extended;
ii,jj:integer;
begin
   for ii:=1 to 4 do
    begin
     find_h;
     sum1:=0;
     for jj:=1 to 4 do sum1:=sum1+gr[ii,jj]*x[jj];
     sum1:=sum1+mas[ii,5];
     sum2:=0;
     for jj:=1 to 4 do sum2:=sum2+gr[ii,jj]*a[summ(ii),jj];
     sum:=sum1/sum2;
     for jj:=1 to 4 do x[jj]:=x[jj]-h[jj]*sum*a[summ(ii),jj];
    end;
end;
function summ(i:integer):integer;
var v,l:integer;
begin
 l:=0;
 for v:=0 to i do l:=l+v;
 summ:=l;
```

```
end;


begin
 eps:=0.000000000001;
 m:=0;
 for i:=1 to 4 do x[i]:=1;
1: find_gr;
   find_a;
 assign(c,'d:\otarov\sem_masA.txt');
 rewrite(c);
 for i:=1 to 10 do
  begin
   for j:=1 to 4 do write(c,a[i,j]:0:5,'; ');
   writeln(c);
  end;
 close(c);
 calc;
 repeat
  for i:=1 to 4 do h[i]:=abs(h[i]);
   sss:=h[1];
   for i:=1 to 4 do if sss<=h[i] then sss:=h[i];
   if sss>eps then  goto 1;
 until (sss<=eps) or keypressed;
 assign(c,'d:\otarov\sem_Alfa.txt');
 rewrite(c);
 for j:=1 to 4 do writeln(c,'Alpha[',j,']= ',x[j]:0:10,' ');
 close(c);
 assign(c,'d:\otarov\Test.out');
 rewrite(c);
 for j:=1 to 4 do writeln(c,h[j]:0:10,' ');
 close(c);
 writeln('Ok');
end.
```

Результаты работы программы:

Значения X:
1.1462368403758576514
-0.56150887017489700121
0.60655626892626200122
-1.44144065404595476013
Проверяем:
0.00000000000001266690
-0.00000000000000254100
-0.00000000000000001100
0.00000000006680147500
Число итераций: 11;
Точность вычислений: 1^-10;