Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Прикладные задачи математического анализа

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

«РЕШЕНИЕ ОПТИМИЗАЦИОННЫХ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ СКА MAPLE»

БГУИР КП 1-40 04 01

Студент: гр. 253503 Ярмак В.С.

Руководитель: кандидат физико-математических наук, доцент Рыкова О.В.

Минск 2023

**СОДЕРЖАНИЕ**

[1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОПТИМИЗАЦИИ 3](#_Toc152512127)

[1.1 Понятие оптимизации 4](#_Toc152512128)

[1.2 Оптимизация в СКА Maple 5](#_Toc152512129)

[1.2.1 Библиотека «simplex» 5](#_Toc152512130)

[1.2.2 Библиотека Optimization 6](#_Toc152512131)

[1.3 Понятие и алгоритм симплекс метода 7](#_Toc152512132)

[1.4 Понятие и описание алгоритма Дейкстры 8](#_Toc152512133)

[2 РЕШЕНИЕ ПРАКТИЧЕСКИХ ЗАДАЧ В MAPLE 11](#_Toc152512134)

[2.1 Решение задачи на оптимизацию функцию одной переменной simplex методом. 11](#_Toc152512135)

[2.2 Решение практической задачи для нахождения максимума и минимума для функции двух переменных 13](#_Toc152512136)

[2.3 Решение задачи на сети 16](#_Toc152512137)

[3 ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ О МАТЕМАТИЧЕСКОЙ ОПТИМИЗАЦИИ 18](#_Toc152512138)

[3.1 Градиентный спуск 18](#_Toc152512139)

[3.2 Практический пример 18](#_Toc152512140)

[ЗАКЛЮЧЕНИЕ 20](#_Toc152512141)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc152512142)

**ВВЕДЕНИЕ**

Задачи на оптимизацию не теряют своей актуальности не только в математических науках, но и в экономике, программировании, инженерии и других областях, включая ежедневную задачу каждого человека о распределении собственного времени на день.

В курсовой работе будут рассмотрены основные типы задач на оптимизацию для линейных функций. Так же данные задачи называют задачами линейного программирования, которые удобнее всего будет решить в системе компьютерной алгебры (СКА) Maple.

Maple — это математическое программное обеспечение, которое сочетает в себе самый мощный в мире математический движок с интерфейсом, который позволяет чрезвычайно легко анализировать, исследовать, визуализировать и решать математические задачи. [1]

Оптимизационные задачи играют ключевую роль в различных областях науки и практики, позволяя находить оптимальные решения в условиях ограничений и неопределенности.

Целью данной работы является изучение и применение методов решения оптимизационных задач в Maple для функций одной переменной. Для чего будут изучены теоретические основы оптимизации и математические методы поиска экстремума функций одной переменной, проведен анализ возможностей и особенностей системы Maple в решении оптимизационных задач, разработка примеров практических задач, и их формализация для решения в Maple.

# **1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОПТИМИЗАЦИИ**

Понятие «оптимизация» прочно вошло в нашу жизнь: если у задачи существует несколько решений, то естественно выбрать то из них, которое является оптимальным. Основные задачи на оптимизацию связаны с нахождением экстремумов функции.

Математическое программирование (МП) представляет собой дисциплину, содержание которой составляют теория и методы решения задач о нахождении экстремумов функции на множествах, определяемых линейными и нелинейными ограничениями (равенствами и неравенствами).

Наименование «математическое программирование» связано с тем, что целью решения задач является выбор программы действий. Например, ряд производственных и экономических задач связан с распределением ресурсов, которые, как правило, являются ограниченными. Кроме того, само распределение возможно не единственным образом. Необходимую для выпуска продукцию можно получить различными способами (за счет выбора вида сырья, технологии производства, применяемого оборудования и пр.). При этом каждый способ распределения ресурсов, оцениваемый с т. з. некоторого критерия (например, прибыли), характеризуется определенным его значением. При решении таких задач необходимо найти максимальное (минимальное) значение критерия. Т. о., требуется выбрать такое распределение ресурсов (программу, план), которое доставляло бы экстремальное значение выбранному критерию. Такую программу (план) называют оптимальной.

Общая задача МП формулируется следующим образом: найти вектор *x = (x1, x2, …, xn )*, удовлетворяющий системе ограничений в виде равенств

*gi (x1, x2, …, xn) = bi, i=*1*..k,*

и неравенств

*gj (x1, x2, …, xn) <= bj, j=k+*1*..m,*

и доставляющий экстремум целевой функции *f(x1, x2, …, xn).* Предполагается, что *f(x1, x2, …, xn), gl (x1, x2, …, xn)* и постоянные *bl, l=*1..*m,* известны. Зачастую на некоторые переменные *x1, x2, …, xn* накладываются условия неотрицательности: *xj >=0.* Если

*gj (x1, x2, …, xn) = ijxj, i=*1*..m,*

*f(x1, x2, …, xn) =jxj,*

где *aij, cj –* известные постоянные, то получаем *задачу линейного программирования* (ЗЛП). Если хотя бы одна из функций f(x1, x2, …, xn), gl (x1, x2, …, xn) нелинейна, то соответствующая задача является *задачей нелинейного программирования*.

## **Понятие оптимизации**

Задачами оптимизации называются задачи отыскания точек максимумов и минимумов функций на заданных множествах. Условимся записывать задачу минимизации функции f (x) на множестве X ⊂ Rn в виде

*f (x) → min, x∈ X*

При этом функция f называется целевой функцией, X – допустимым множеством, любая его точка – допустимой точкой задачи (1.1). Точка x ∈ X \* называется:

1) точкой глобального минимума (или глобальным решением задачи (1.1)), если

*f (x\* ) ≤ f (x) ∀x∈ X*

2) точкой локального минимума (локальным решением), если найдется такое число ε > 0, что

f (x\* )≤ f (x) ∀ x ∈ X ∩U (x\*, ε)

где U(x\*,ε) – открытый шар радиуса ε с центром в точке x\* .

Если неравенство в (5.2) или (5.3) выполняется как строгое при x ≠ x\* , то говорят, что x\* – точка строгого минимума (строгое решение) в глобальном или локальном смысле соответственно.

Задача максимизации эквивалентна задаче минимизации функции − f (x) в том смысле, что множества решений этих задач соответственно совпадают. Это позволяет без труда переносить утверждения для задачи минимизации на задачу максимизации и наоборот.

Точки минимума и максимума функции объединяются общим названием; это – точки экстремума. При этом такие задачи оптимизации называют также экстремальными. [6]

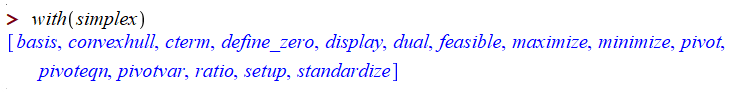
## **1.2 Оптимизация в СКА Maple**

Решение задач на оптимизацию в Maple сильно упрощают библиотеки,

подключаемые к проекту. К ним относятся: Simplex, Optimization, Global Optimization. Но также задачи решаются и классическими алгоритмами.

### **1.2.1 Библиотека «simplex»**

Библиотека «simplex» - предназначена для оптимизации линейных систем с использованием симплексного алгоритма. Особенность ее в том, что имеется возможность выполнять оценки промежуточных этапов, например, определять базисные переменные и т.п.

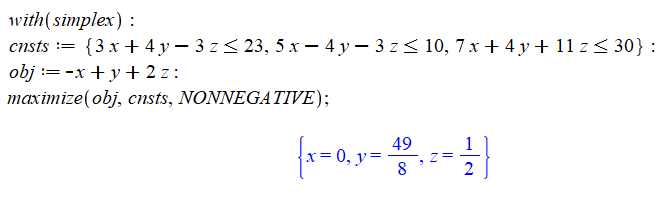
****

После подключения библиотеки командой with(simplex) пользователю становится доступны функции и опции, некоторые из которых указаны в следующей таблице. [4]

Таблица 1.1 – Функции пакета simplex

|  |  |
| --- | --- |
| Функция | Описание |
| NONNEGATIVE | Ключевое слово, указывающее на неотрицательный тип переменных; |
| standardize | Приведение заданной системы ограничений к системе неравенств со знаками  ; |
| dual | Построение двойственной задачи для задачи линейного программирования следующего вида:  Целевая функция максимизируется;  Система ограничений записана в виде системы неравенств со знаками;  Переменные задачи не отрицательны; |
| feasible | Выяснение возможности решения заданной задачи линейного программирования; |
| maximize | Вычисление максимума целевой функции; |
| minimize | Вычисление минимума целевой функции; |

Пример использования функции maximize:



Библиотека называется simplex в виду того, что алгоритм нахождения максимума и минимума функций совпадает с Симплекс методом решения задач линейного программирования. Данный метод будет рассмотрен далее.

### **1.2.2 Библиотека Optimization**

Пакет Optimization представляет собой набор команд для численного решения задач оптимизации, которые предполагают нахождение минимума или максимума объективной функции, возможно, с учетом ограничений. Пакет использует встроенные библиотечные процедуры, предоставленные Numerical Algorithms Group (NAG). [4]

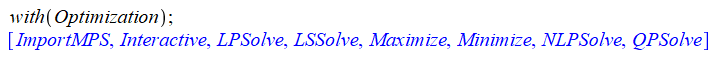
Пакет вводит 8 функций. Две из них это переопределенные функции вычисления максимума Maximize и минимума Minimize. Кроме того, пакет имеет 4 решателя уравнений с заданными ограничениями, реализующих следующие методы:

• LPSolve — линейное программирование;

• LSSolve — улучшенная реализация метода наименьших квадратов;

• QPSolve — квадратичное программирование;

• NLPSolve — нелинейное программирование.



## **1.3 Понятие и алгоритм симплекс метода**

Симплекс метод - это метод последовательного перехода от одного базисного решения (вершины многогранника решений) системы ограничений задачи линейного программирования к другому базисному решению до тех пор, пока функция цели не примет оптимального значения (максимума или минимума). [7]

Симплекс метод был предложен американским математиком Р.Данцигом в 1947 году, с тех пор для нужд промышленности этим методом нередко решаются задачи линейного программирования с тысячами переменных и ограничений.

Перед тем, как перейти к алгоритму симплекс метода, несколько определений.

Всякое неотрицательное решение системы ограничений называется **допустимым решением**.

Пусть имеется система *m* ограничений с *n* переменными (*m < n*).

***Допустимым базисным решением*** является решение, содержащее *m* неотрицательных ***основных (базисных)*** переменных и *n - m* ***неосновных.*** (небазисных, или ***свободных***) переменных. Неосновные переменные в базисном решении равны нулю, основные же переменные, как правило, отличны от нуля, то есть являются положительными числами.

Любые *m* переменных системы *m* линейных уравнений с *n* переменными называются ***основными***, если определитель из коэффициентов при них отличен от нуля. Тогда остальные *n - m* переменных называются ***неосновными*** (или ***свободными***).

Алгоритм симплекс метода

1 Привести задачу линейного программирования к канонической форме. Для этого перенести свободные члены в правые части (если среди этих свободных членов окажутся отрицательные, то соответствующее уравнение или неравенство умножить на - 1) и в каждое ограничение ввести дополнительные переменные (со знаком "плюс", если в исходном неравенстве знак "меньше или равно", и со знаком "минус", если "больше или равно").

2 Если в полученной системе m уравнений, то m переменных принять за основные, выразить основные переменные через неосновные и найти соответствующее базисное решение. Если найденное базисное решение окажется допустимым, перейти к допустимому базисному решению.

3 Выразить функцию цели через неосновные переменные допустимого базисного решения. Если отыскивается максимум (минимум) линейной формы и в её выражении нет неосновных переменных с отрицательными (положительными) коэффициентами, то критерий оптимальности выполнен и полученное базисное решение является оптимальным - решение окончено. Если при нахождении максимума (минимума) линейной формы в её выражении имеется одна или несколько неосновных переменных с отрицательными (положительными) коэффициентами, перейти к новому базисному решению.

4 Из неосновных переменных, входящих в линейную форму с отрицательными (положительными) коэффициентами, выбирают ту, которой соответствует наибольший (по модулю) коэффициент, и переводят её в основные. Переход к шагу 2.

Важные условия

a) Если допустимое базисное решение даёт оптимум линейной формы (критерий оптимальности выполнен), а в выражении линейной формы через неосновные переменные отсутствует хотя бы одна из них, то полученное оптимальное решение - не единственное.

б) Если в выражении линейной формы имеется неосновная переменная с отрицательным коэффициентом в случае её максимизации (с положительным - в случае минимизации), а во все уравнения системы ограничений этого шага указанная переменная входит также с отрицательными коэффициентами или отсутствует, то линейная форма не ограничена при данной системе ограничений. В этом случае её максимальное (минимальное) значение записывают в виду *Fmax = ∞.* [7]

## **1.4 Понятие и описание алгоритма Дейкстры**

Рассмотрим нагруженный граф на рисунке1.4.1. Он может представлять, например, длины дорог в милях между шестью деревнями. Поскольку количество вершин в этом графе невелико, то перебрать все возможные пути между любой парой заданных вершин нам вполне по силам. При этом, естественно, мы найдем наиболее короткий путь, соединяющий соответствующие деревни. В реальной задаче, возникающей в профессиональной деятельности, число вершин, как правило, настолько велико, что такой упрощенный подход к поиску кратчайшего пути слишком неэффективен.

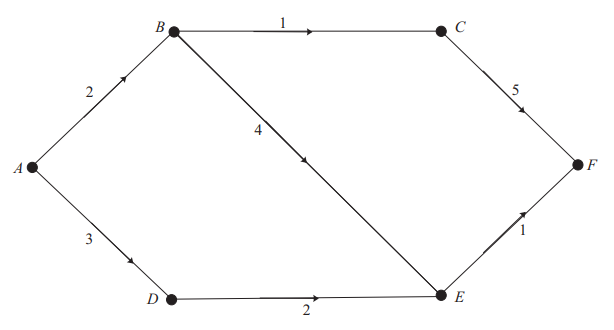
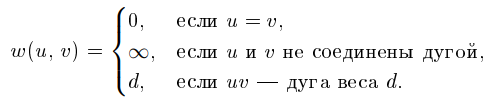
****

Рисунок 1.1 – Нагруженный граф

Существует множество алгоритмов поиска кратчайшего пути, но рассмотрен будет только один, алгоритм Дейкстры.

Допустим, что нужно найти кратчайший путь от вершины A к любой другой вершине орграфа. Кратчайший путь – это путь минимального общего веса, соединяющий выбранные вершины. Общий вес, по определению, равен сумме весов всех дуг, составляющих путь. Общий вес кратчайшего пути, ведущего из вершины u в вершину v, называют расстоянием от u до v.

Определим весовую матрицу w, чьи элементы w(u,v) задаются формулой.



Для нашего графа весовая матрица представлена на рисунке 1.2.

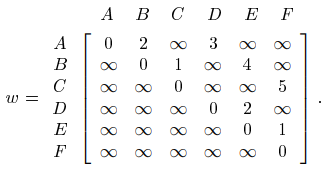


Рисунок 1.2 – Весовая матрица для графа

В течение работы алгоритма каждой вершине v орграфа присваивается число *d[v],* равное расстоянию от вершины *A* до *v*. Перед началом работы *d[v]* совпадает с весом дуги *(A, v),* если такая существует, или равно ∞ в противном случае. Мы будем проходить вершины орграфа и уточнять значения *d[v].*

На каждом шагу алгоритма отмечается одна вершина *u*, до которой уже найден кратчайший путь от *А* и расстояние *d[u]* до нее. Далее полученное значение *d[u]* отмеченной вершины и не меняется. Для оставшихся, неотмеченных вершин *v*, число *d[u]* будет меняться с учетом того, что искомый кратчайший путь до них от *А* будет проходить через последнюю отмеченную вершину *u.* Алгоритм завершится в тот момент, когда все возможные вершины будут отмечены и получат свои окончательные значения *d[u].*

Для каждого шага алгоритма, описанного ниже, в соответствующую строку рисунке 1.3 заносится отмеченная вершина, текущие значения *d[u]* и оставшиеся неотмеченные вершины. При этом полужирным шрифтом выделяется наименьшее из значений *d[v]* среди неотмеченных вершин. Соответствующую Вершину следует отметить. Кроме того, в таблице все значения *d[v]* для уже отмеченных вершин отделены от остальных ломаной линией.

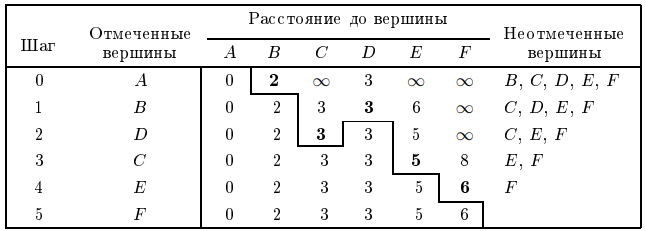


Рисунок 1.3 – Таблица нахождения кратчайшего пути

Рассмотрим пошаговый алгоритм.

1Поскольку мы интересуемся кратчайшими путями от вершины *А*, мы отмечаем ее и использует первую строку весовой матрицы w для определения начальных значений *d[v].* Таким образом получается первая строка таблицы. Наименьшее число из всех *d[u]* для неотмеченных вершин это *d[B]* = 2.

2 Отмечаем вершину В, так как она является ближайшей к *A.* Вычисляем длины путей, ведущий от *А* к неотмеченным вершинам через вершину *B*. Если новые значения *d[u]* оказываются меньше старых, то меняем последние на новые. Итак, при этом проходе цикла путь *ABC* имеет вес 3, а путь *ABE* – 6, в то время как старые расстояния до этих вершин от *А* были ∞. Следовательно, заполняя вторую строку таблицы, мы заменим *d[C]* на 3 и *d[E]* на 6.

3 Из оставшихся неотмеченными, вершины *С* и *D* находится ближе всех к *А.* Отметить можно любую из них. Возьмем вершину *D.* Так как длина пути *ADE* равна 5, текущее значение *d[E]* следует уменьшить до 5. Теперь можно заполнить третью строчку таблицы. Наименьшее значение *d[v]* среди неотмеченных к этому моменту вершин оказывается у вершины С.

4 Отмечаем вершину *С* и подправляем значения *d[v].* Теперь можно дойти и до вершины *F,* следуя путем *ABCF*. Его длина, а стало быть, и значение *d[F],* равны 8. К этому моменту остались неотмеченными две вершины: *E* и *F.*

5 Мы отмечаем вершину *Е*, что позволяет на уменьшить величину *d[F]* с 8 до 6.

6 Отмечаем вершину *F*. [9]

# **2 РЕШЕНИЕ ПРАКТИЧЕСКИХ ЗАДАЧ В MAPLE**

## **2.1 Решение задачи на оптимизацию функцию одной переменной simplex методом.**

В пакете MAPLE для решения задач линейного программирования нужно загрузить библиотеку SIMPLEX. Для загрузки библиотек служит команда with(<имя библиотеки>).

Ограничения вводятся в виде списка линейных неравенств. Матричную запись введённой системы можно получить при помощи команды display из того же пакета.

После того, как ограничения введены, применяется команда minimize(f, е), если нужно найти минимум целевой функции, либо команда maximize(f, е), если нужно найти ее максимум, где f – целевая функция, а е – список условий.

Задача о производстве стульев (см. рисунок 2.1). Мебельная фабрика может выпускать стулья двух типов, стоимостью 8000 и 12000 рублей. Имеются следующие ресурсы: 440 погонных метров досок, 65 кв.м. обивочной ткани и 320 человеко-часов трудовых ресурсов. На изготовление одного стула требуются следующее количество ресурсов:

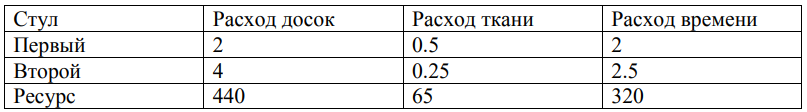
****

Рисунок 2.1 – Условие задачи

Требуется так спланировать производство стульев, чтобы общая цена продукции была максимальной. [10]

Перейдем к математической формулировке задачи. Обозначим через х количество стульев первого типа, через у – количество стульев второго типа. Тогда условия задачи сводятся к следующему:

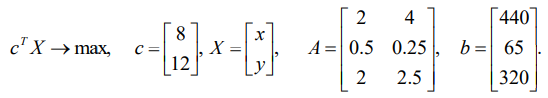
8x +12y → max – оптимизируемый критерий;

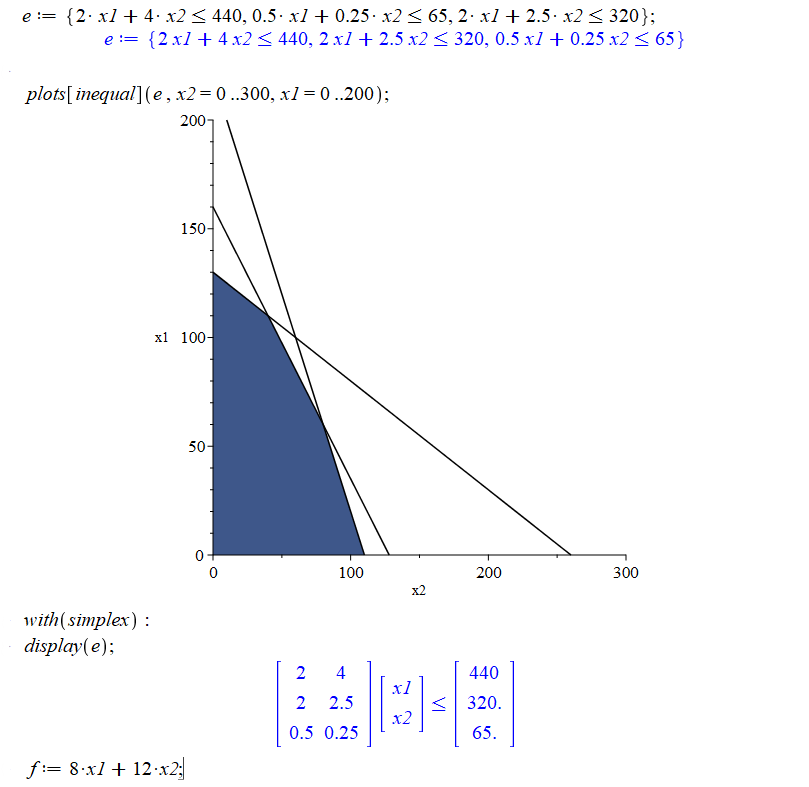
2x + 4y ≤ 440 – ограничение по расходу досок;

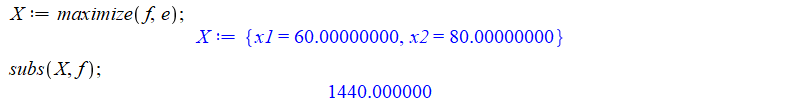
0.5x + 0.25y ≤ 65 – ограничение по расходу ткани;

2x + 2.5y ≤ 320 – ограничение по расходу времени.

Матричная форма записи:





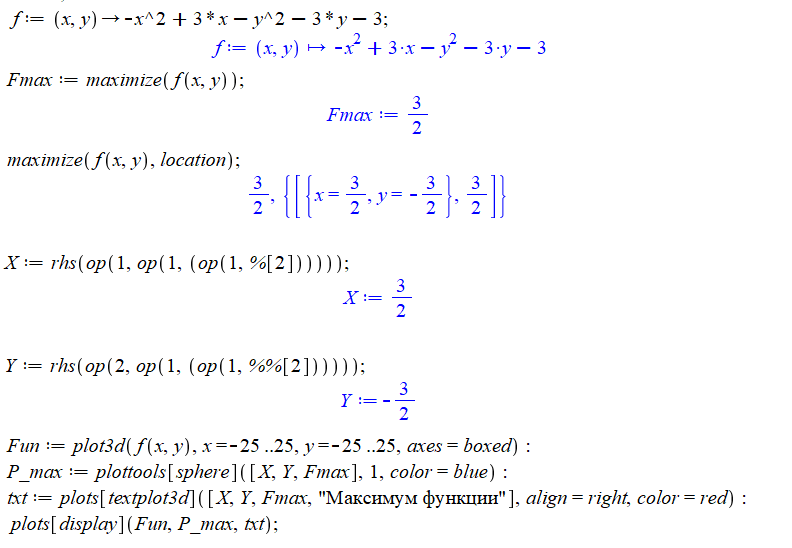


## **2.2 Решение практической задачи для нахождения максимума и минимума для функции двух переменных**

Часто бывает необходимо найти минимум или максимум заданной функции. Для поиска минимума или максимума функции как одной, так и нескольких переменных можно использовать стандартные средства Maple – ***minimize*** и***maximize***.

С помощью параметров можно задавать дополнительные данные для поиска, так, например, ограничить область поиска или, используя параметр ***location***, включить расширенный вывод результатов – выводить не только значение минимума (или максимума), но и значения переменных в этой точке. [10]

Рассмотрим следующий пример, представленный на листинге кода из Maple (см. рисунок 2.2).



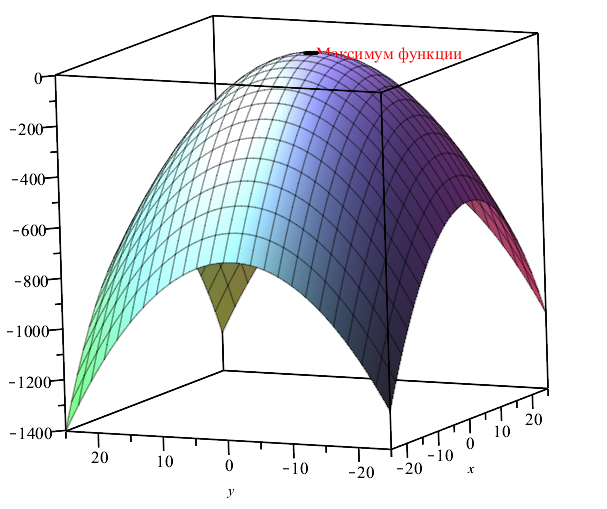
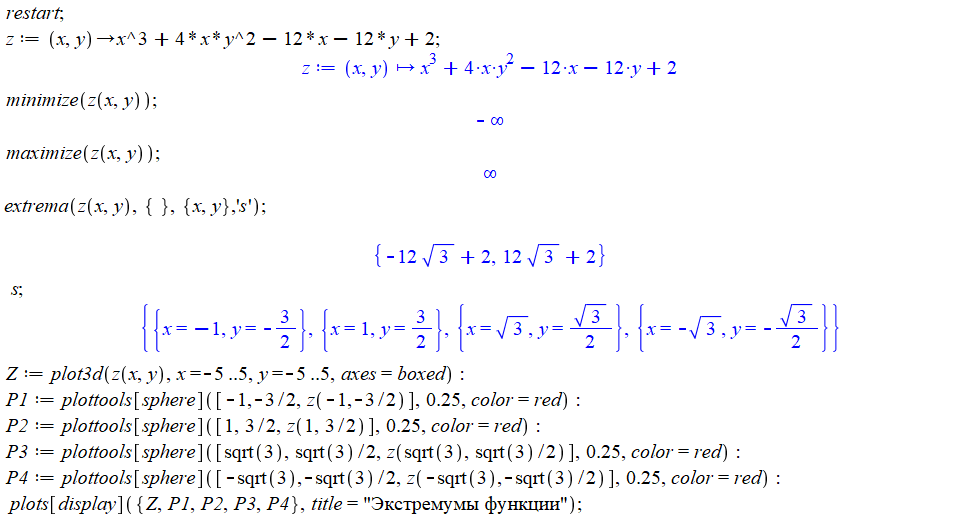


Рисунок 2.2 – Результат выполнения программы

Функция ***extrema*** позволяет найти экстремумы (как максимумы, так и минимумы) выражения при ограничениях или без них. При отсутствии ограничений вместо них записывается пустое множество {}. Найденные координаты точек экстремума присваиваются переменной `s`. Далее приведем пример применения функции***extrema*** (см. рисунок 2.3).



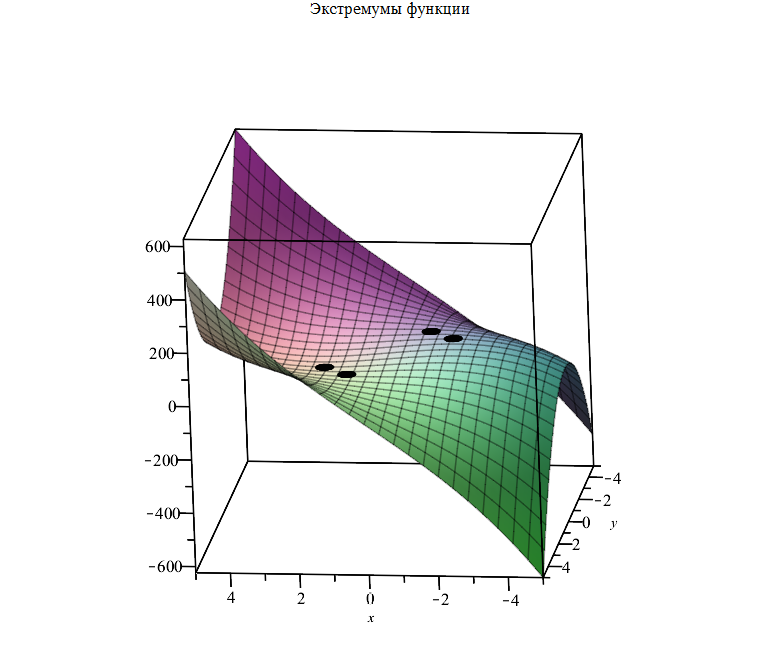


Рисунок 2.3 – Результат выполнения кода программы

Функции ***minimize*** и ***maximize*** определили минимум и максимум функции как -∞ и ∞, в то время как функция ***extrema*** нашла локальные экстремумы.

## **2.3 Решение задачи на сети**

Задача. Дана сеть (рисунок 2.4), дуги которой помечены числами, равными их длинам. Требуется найти кратчайшие пути, ведущие от узла 1 к каждому из остальных узлов сети. [9]

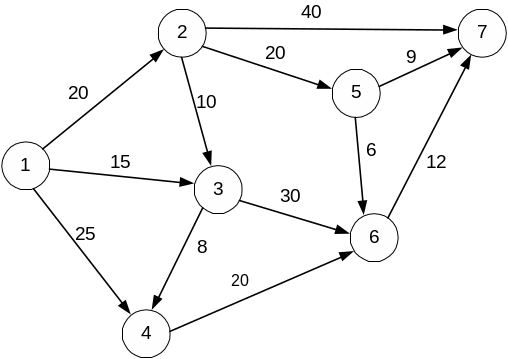
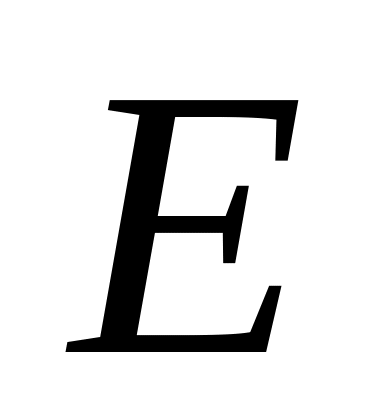
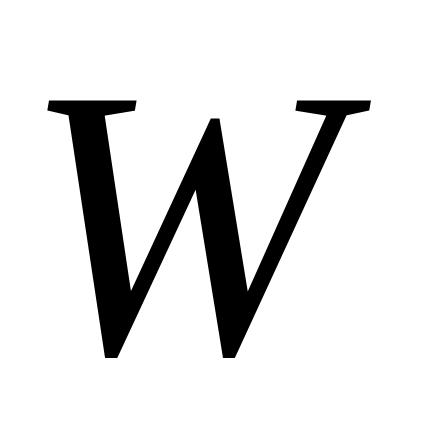
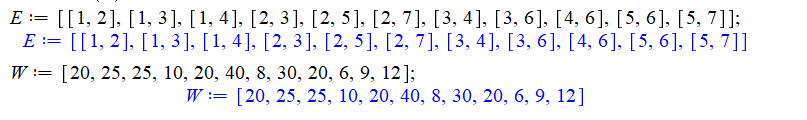


Рисунок 2.4 – Исходный граф

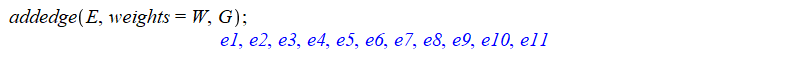
*Решение.* Задача нахождения кратчайшего пути в графе в Maple решается с помощью алгоритма Дейкстры, описанным ранее (гл.1). Опишем сеть, изображенную на рисунке 2.4. Загрузим пакет и зададим граф, содержащий семь изолированных узлов (не соединенных ребрами). Обозначим его идентификатором *G*.



Зададим списки дуг  и соответствующих им весов  . В качестве замечания отметим, что в ориентированном графе дуга окаймляется квадратными скобками – [1,2], а в неориентированном графе ребро окаймляется фигурными – {1,2}.



Получим сеть, добавив в граф списки дуг и соответствующих им весов:



Определимдерево кратчайших путей сети с помощью функции *shortpathtree***,** обозначим егоидентификатором *T* . И нарисуем его.

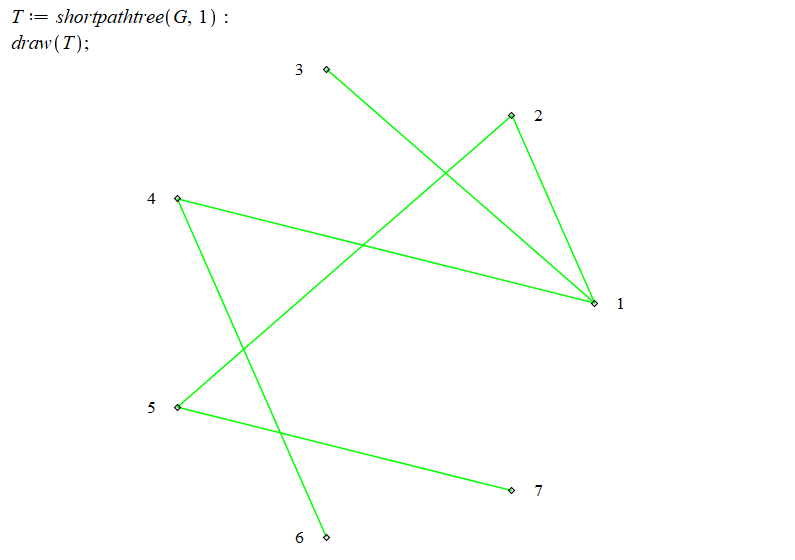
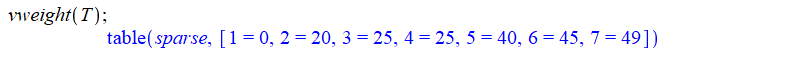


Рисунок 2.5 – Результат выполнения программы

Из выведенного на экран графического объекта (см. рисунок 2.5) мы можем определить, через какие узлы сети пролегают кратчайшие пути, ведущие от узла 1 к каждому из остальных узлов сети. То же самое можно сделать с помощью функции *path*. Определим пути, ведущие от узла 1 к остальным узлам дерева. Чтобы не повторять однотипную операцию несколько раз, организуем цикл с помощью функции *seq*:



Найдем расстояния от узла 1 до каждого из узлов 2, 3, 4, 5, 6, 7 дерева с помощью функции *vweight*.



Результат выполнения – таблица, из которой видно, что вершине 1 присвоена метка (вес) 0, вершине 2 – метка 20, следовательно, длина кратчайшего маршрута от узла 1 к узлу 2 равна 20. Аналогично имеем: длина кратчайшего маршрута от узла 1 к узлу 3 равна 25, от узла 1 к узлу 4 – 15, от узла 1 к узлу 5 – 40, от узла 1 к узлу 6 – 35, от узла 1 к узлу 7 – 49.

# **3. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ О МАТЕМАТИЧЕСКОЙ ОПТИМИЗАЦИИ**

## **3.1 Градиентный спуск**

Градиентный спуск - это способ обучения и совершенствования модели машинного обучения. Он делает это, постоянно пытаясь лучше предсказать правильный ответ, корректируя свое "мышление". Для определения, в каком направлении необходимо осуществлять наискорейший спуск, необходимо найти антиградиент функции . Процесс повторяется много раз, пока алгоритм не сможет предсказать ответ настолько хорошо, насколько это возможно.

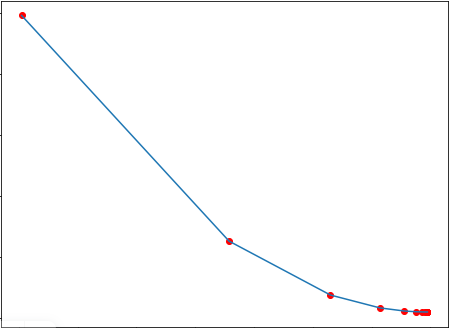
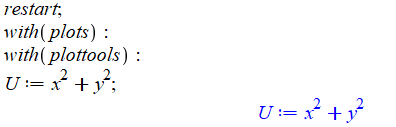


Рисунок 3.1 – Визуальное представление градиентного спуска

## **3.2 Практический пример**

Пусть задана функция U=x2+y2, ɛ=0.0001, λ=0.1, начальное приближение равно x=1, y=2. [10]

Подключим необходимые библиотеки, введем все начальные условия, а также выведем график заданной функции (см. рисунок 3.2).





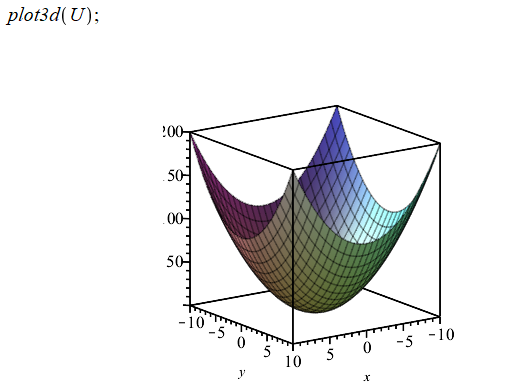
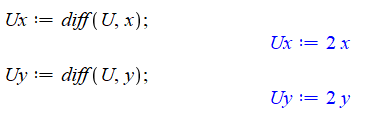
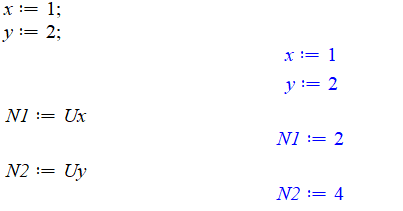


Рисунок 3.2 – График исходной функции

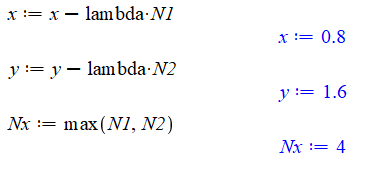
Находим градиент функции:



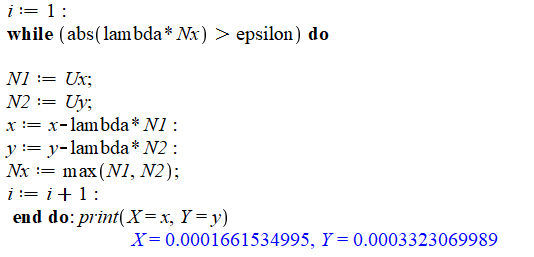
Задаем начальное приближение и находим значение градиента функции в заданном начальном приближении:

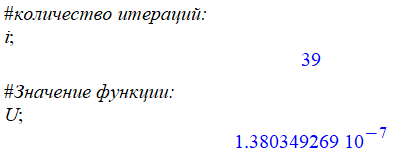


Выполняем первое приближение по заданной формуле и находим наибольшее из значений градиента:



Используя итерационную последовательность, находим значения x и y c заданной степенью точности:





Необходимое значение функции найдено с заданной точностью за 39 итераций.

# **ЗАКЛЮЧЕНИЕ**

Существует огромное количество методов оптимизации различных функций, однако в данной работе я постаралась рассмотреть самые известные и наиболее применимые в различных областях.

В ходе проведения данной курсовой работы были рассмотрены несколько методов математической оптимизации. Симплекс-метод использовался для решения линейных задач на многогранниках, в то время как поиск безусловного экстремума с матрицей Гессе применялся к нелинейным функциям с известной матрицей вторых производных. Алгоритм Дейкстры был использован для нахождения кратчайших путей в графах. Метод Градиентного спуска применялся при оптимизациифункции итерационным методом. Каждый из методов обладает своими характеристиками и находит применение в различных областях, в зависимости от требований конкретной задачи.

Все рассмотренные методы были подкреплены задачами, которые в свою очередь были решены в СКА Maple. Данная программа имеет несколько библиотек, которые позволяют очень быстро и с высокой точностью решить все рассмотренные задачи. Откуда можно сделать вывод, что Maple это отличная программа для решения сложных оптимизационных задач, с широким функционалом.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Maple Documentation [Электронный ресурс]. – Режим доступа:<https://www.maplesoft.com/support/help/>.

[2] Калугина, М. А. Математический анализ. Лабораторный практикум в системе Maple: учеб.-метод. пособие / М. А. Калугина. – Минск, БГУИР, 2018.

[3] Высшая математика. Часть 3. / Жевняк Р. М., Карпук М. А. – Минск, Вышэйшая школа, 1985.

[4] Maple 9.5/10 в математике, физике и образовании / Дьяконов В.П. - Солон-Пресс, 2006.

[5] Высшая математика на компьютере в программе Maple 14 / Касюк С.Т., Логвинова А.А., Челябинск: Южно-Уральский государственный университет (ЮУрГУ), 2011.

[6] Maple 6. Решение задач высшей математики и механики / Матросов А.В., BHV-Санкт-Петербург, 2001.

[7] Компьютерное моделирование задач оптимизации / Мироновский Л.А., Петрова К.Ю., Санкт-Петербург: Санкт-Петербургский государственный университет аэрокосмического приборостроения (СПбГУАП), 2005.

[8] Применение системы Maple для решения задач оптимизации / Харьковский национальный университет имени В.Н. Казарина, Харьков – 2008.

[9] Оптимизационные алгоритмы теории графов в программировании / Ломакина Л.С., Суркова А.С., Уваров П.И. – Нижний Новгород, НГТУ, 2007.

[10] Методы оптимизации. Лабораторный практикум : учеб.-метод. Пособие / Шевченко А.С. – Рубцовск, Рубцовский институт (филиал) АлтГУб 2016.

[11] Далингер В.А., Симонженков С.Д. Решение уравнений и оптимизация на компьютере: учеб. пособие. – Омск: – ООО ИПЦ «Сфера», 2011.