1. **Метод Ньютона (касательных)**

**Суть:** Используем касательную в точке xnx​, чтобы найти следующую точку

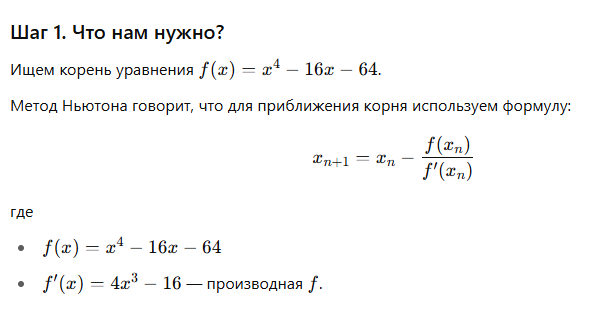
**Что делает:** ищет корень уравнения f(x)=0, используя касательную к графику функции.

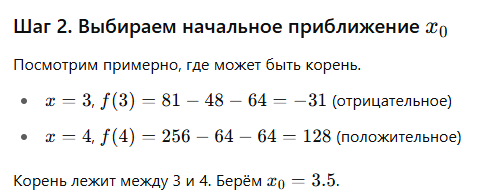
**Как работает:**  
Берёшь точку, считаешь касательную к функции, находишь, где касательная пересекает ось X — это новое приближение. Повторяешь, пока не найдёшь ответ.

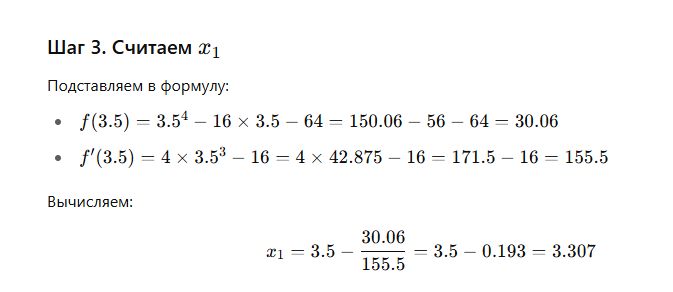
**Пример**

**Уравнение вида**

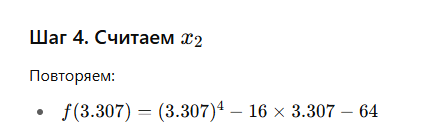
****

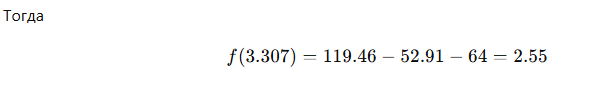
****

****



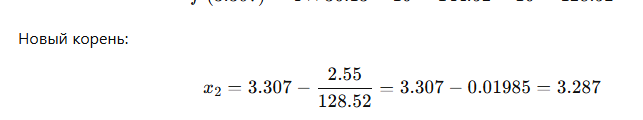
**3.307 – Используется для подсчета х2**

****

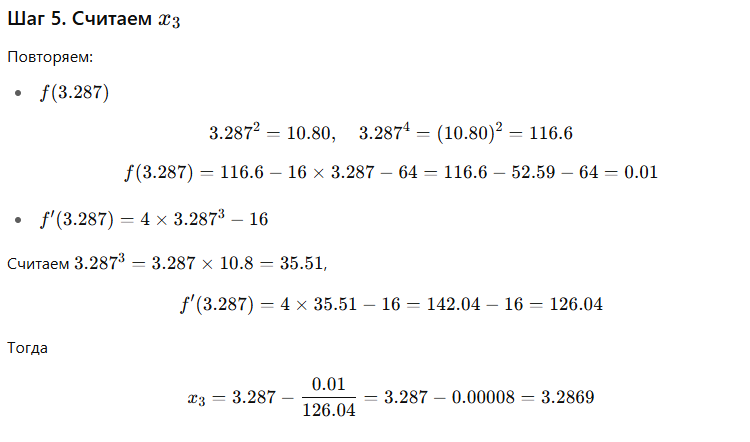
****

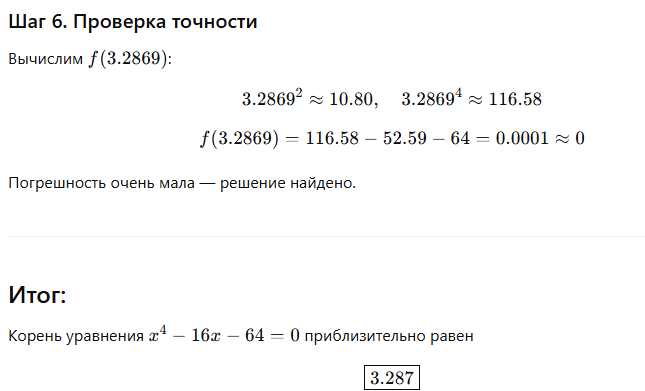
****

**Подставляем**

****

**3.287 – это новый корень (х3), повторяем то же самое**

****

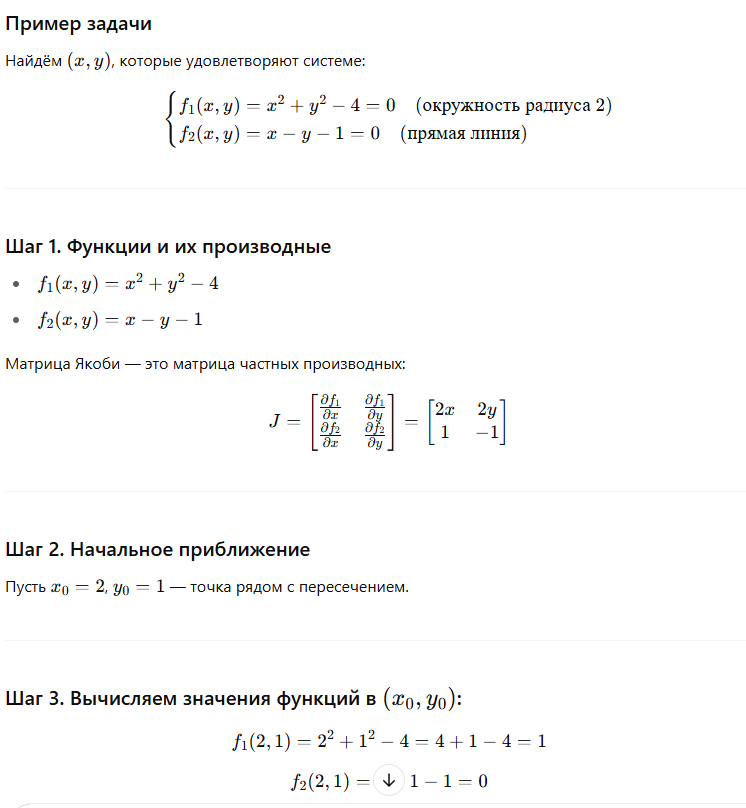
****

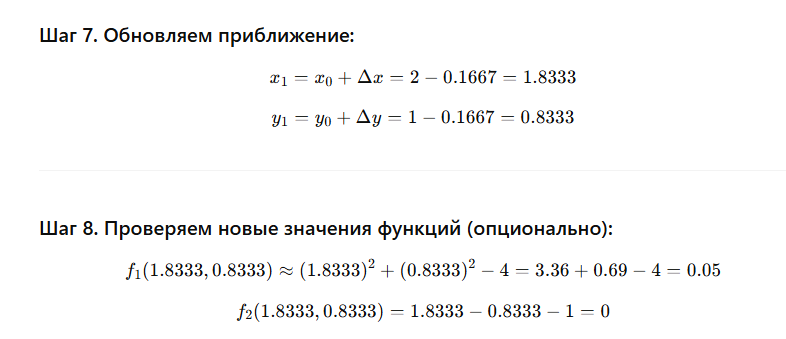
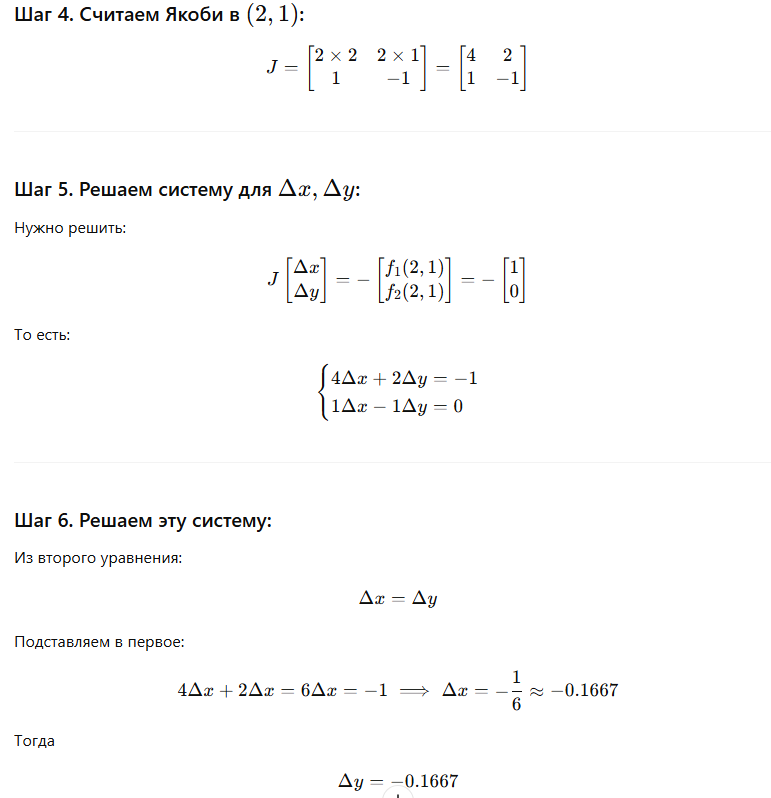
### Метод Ньютона–Рафсона (для систем нелинейных уравнений)

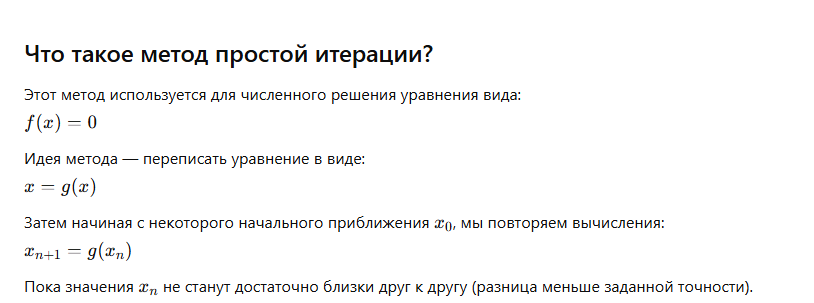
### **Суть:** То же, что метод Ньютона, но производная считается один раз в начале и не пересчитывается.

**Что делает:** ищет решение сразу для нескольких уравнений с несколькими переменными.

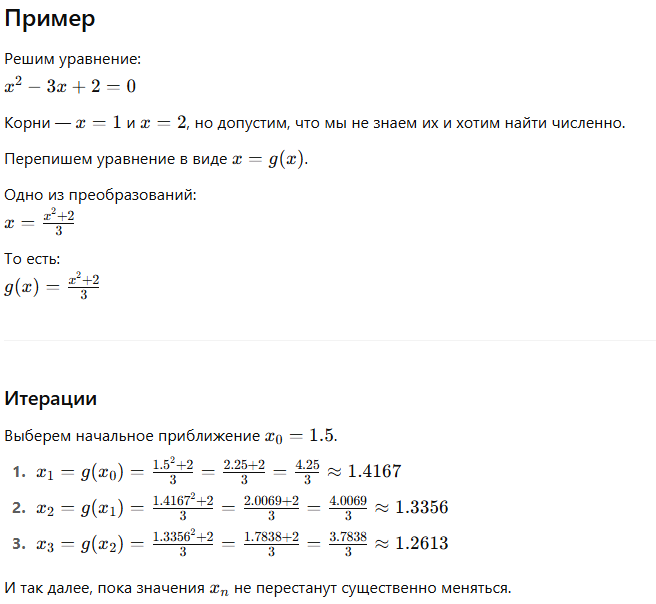
**Как работает:**  
Похож на метод Ньютона, но для системы — одновременно улучшает все переменные, используя матрицы производных (Якобиан).

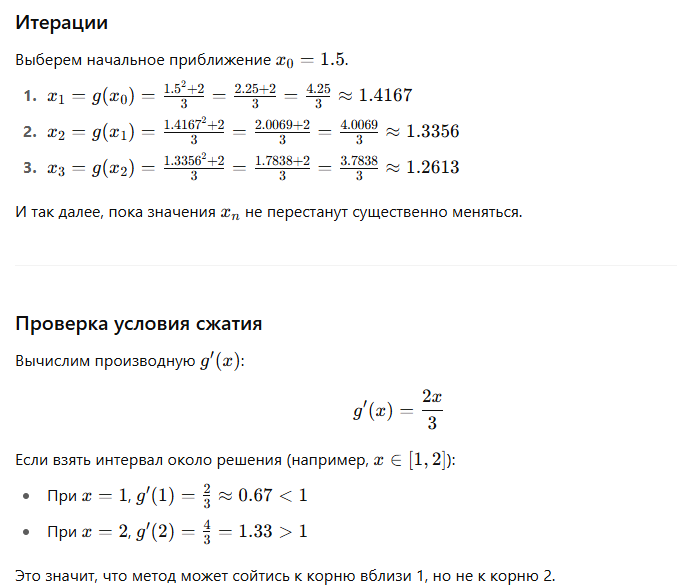




1. **Метод простой итерации  
   **

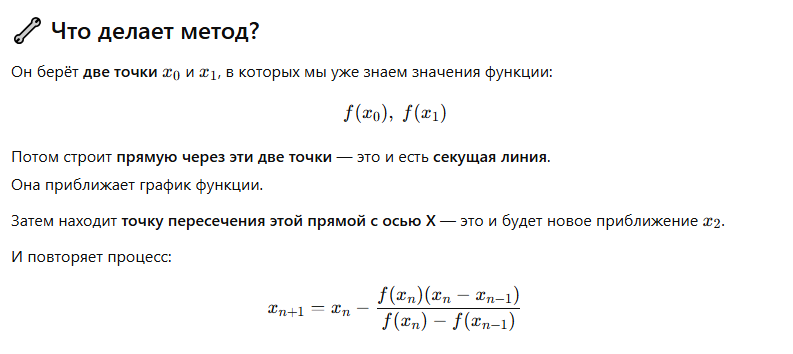
**Приближение мы выбираем там где ближе всего п пересечению гарфика функции**

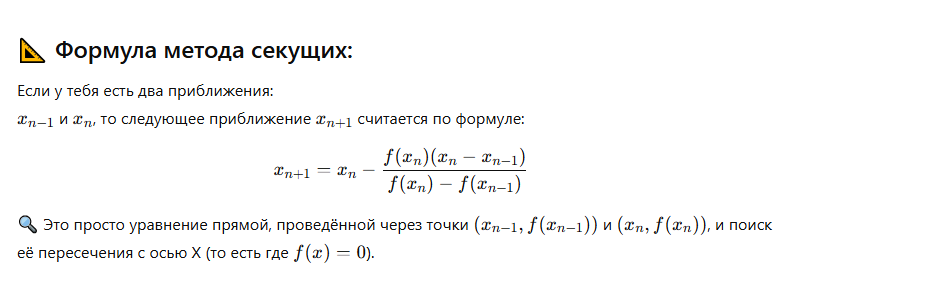
****

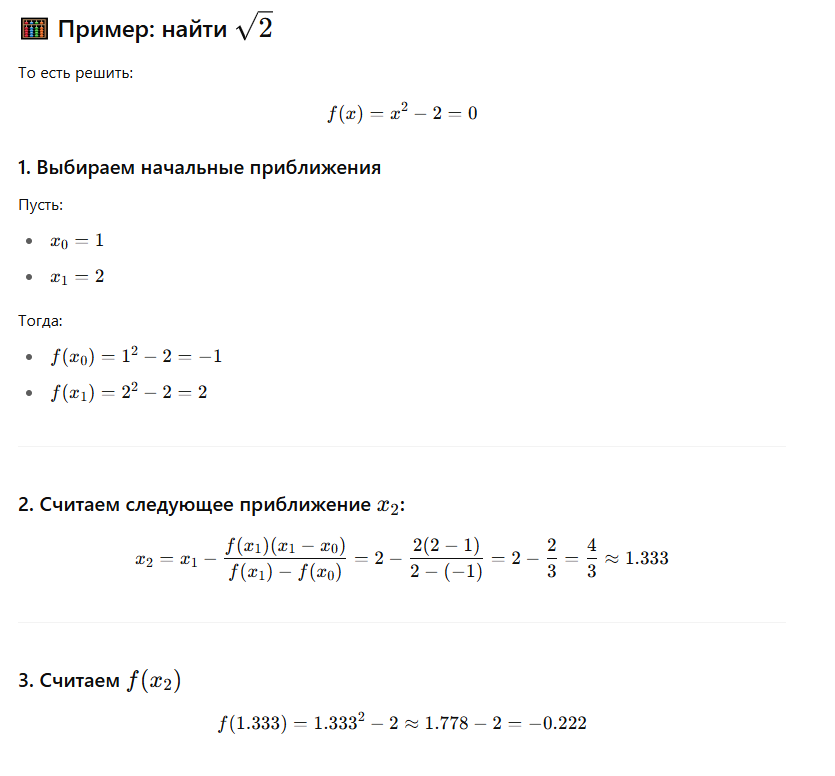
****

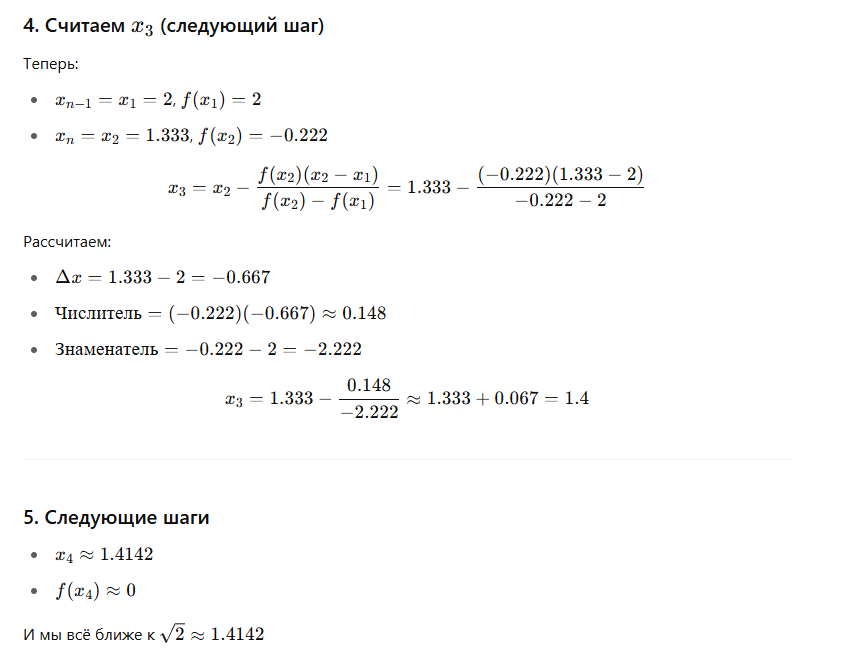
1. **Метод секущих**

**Суть:** Проводим хорду между двумя последними точками, где функция меняет знак.

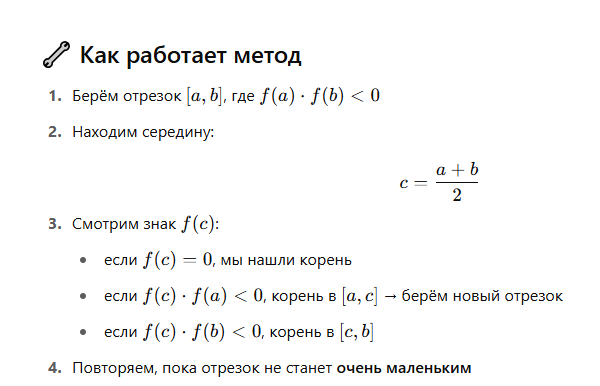
****

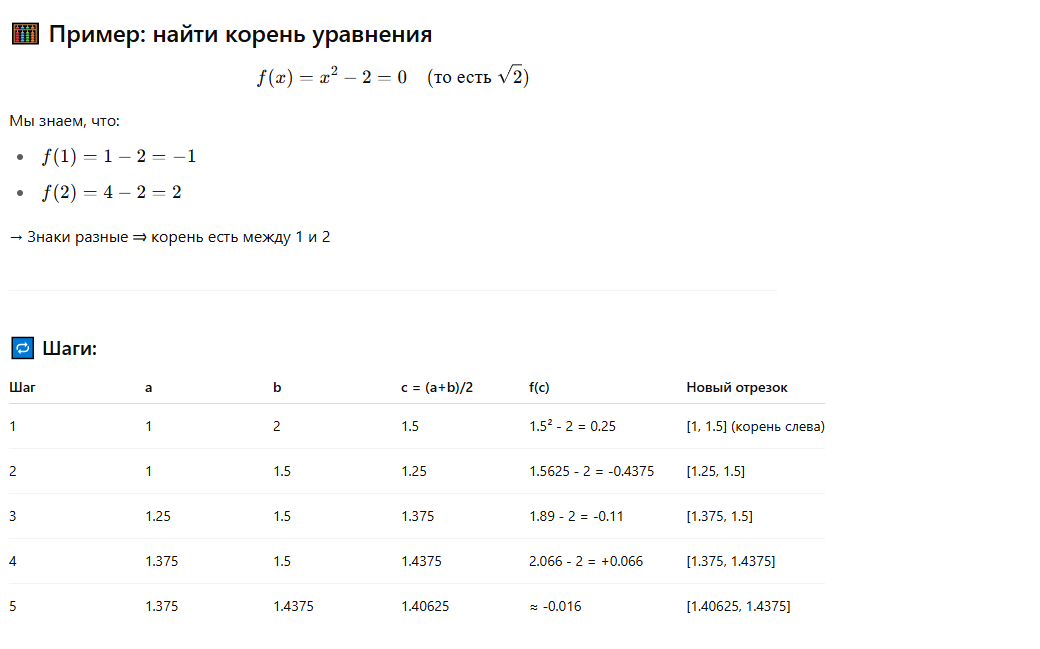
****

****

****

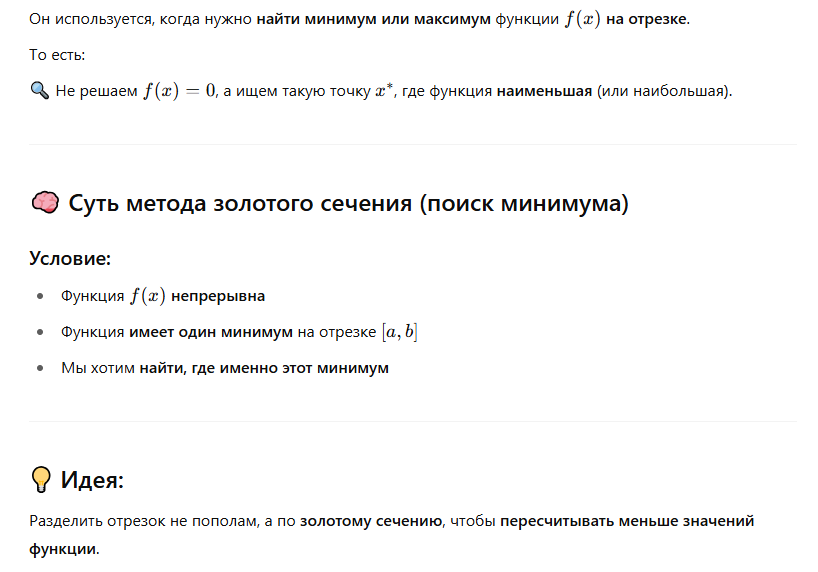
1. **Метод дихтомии(биссекции)**

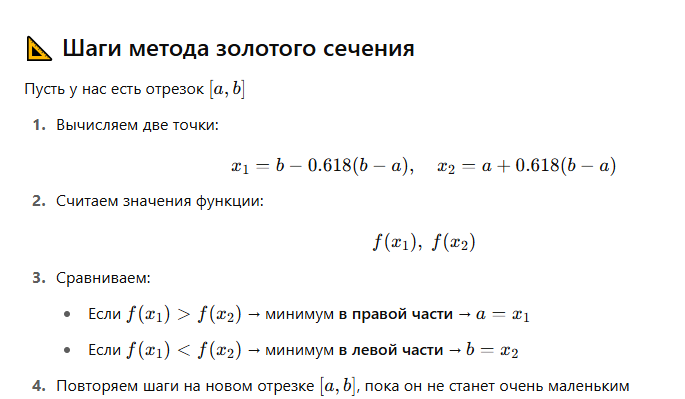
****

****

1. **Метод золотого сечения**

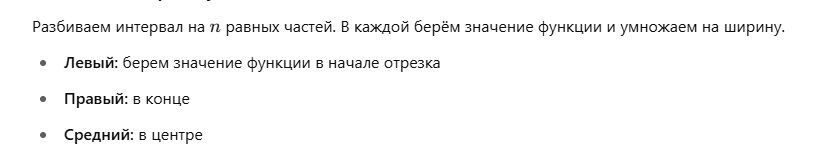
**ЧИСЛО ЗОЛОТОГО СЕЧЕНИЯ = 0.618**

****

****

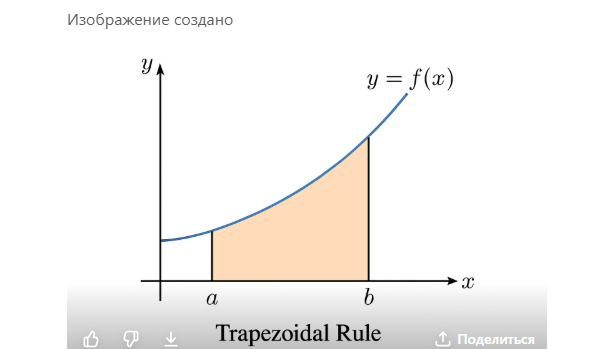
**Численное интегрирование**

**1.Метод прямоугольников**

****

**2 Метод трапеции**

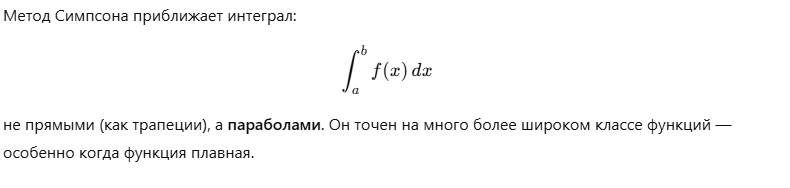
**Суть метода трапеций** — приближённое вычисление определённого интеграла путём замены кривой графика функции на отрезки прямых, образующих трапеции.

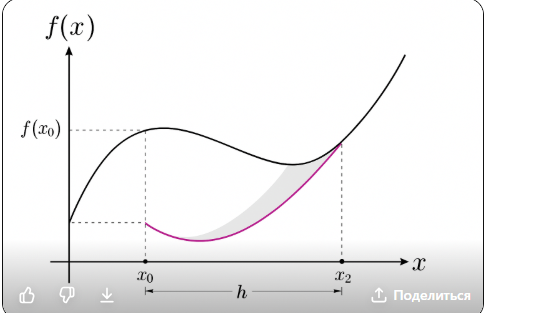
****

**ЕСЛИ ПРОСТЫМ ЯЗЫКОМ, мы под функцией строим**

**Трапеции, считаем их площадь и складываем их**

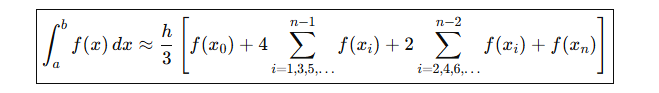
**3 Метод симпсона (Параболы)**

****

****

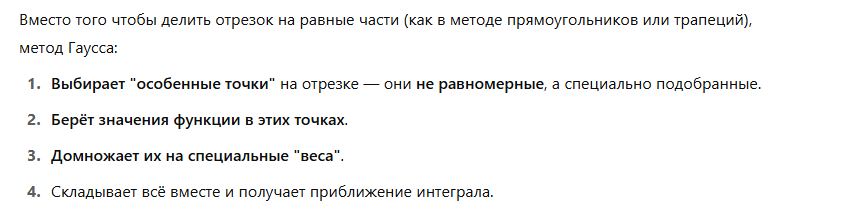
Делим ось х на четное количество. Потом берем первые три точки (х0, х1, х2)

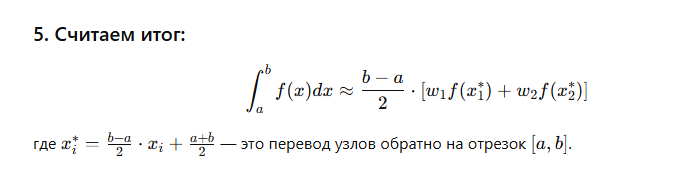
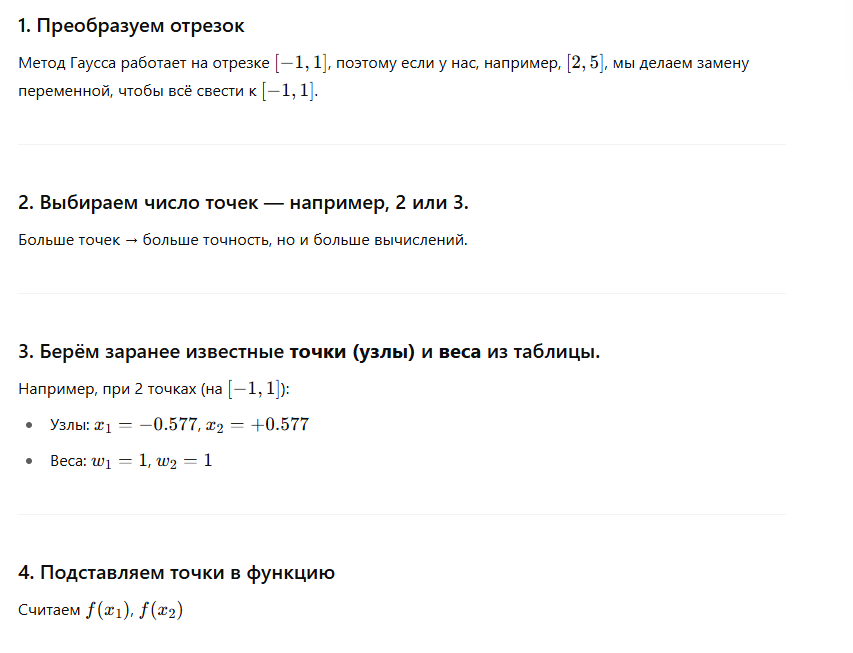
И строим по ним параболу по формуле



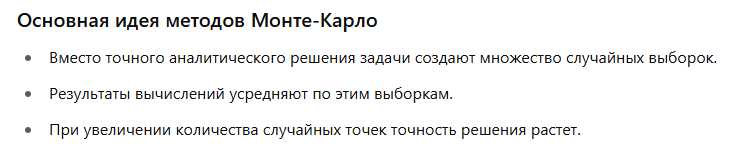
В конце эти параболы складываем

**4 Квадратные формулы гауса**

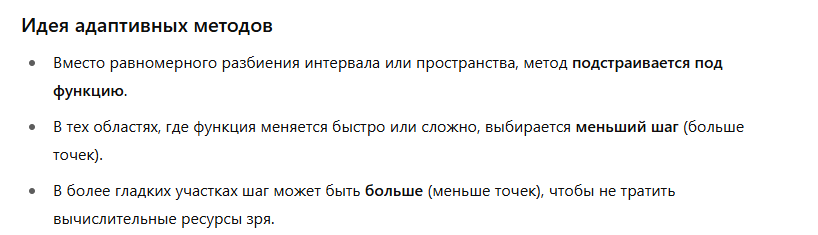
****

****

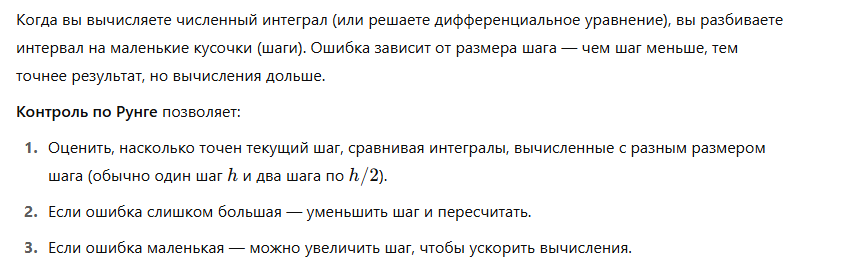
**5.Методы Монте карло**

****

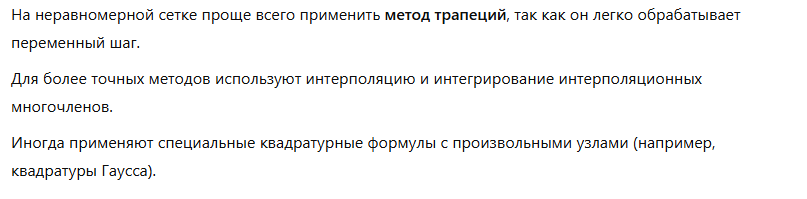
**6.Адаптивные методы**

****

**Контроль по рунге**

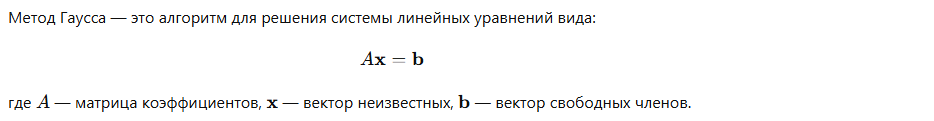
****

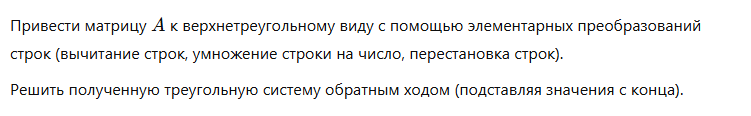
**7.Интегрирование на неравномерной сетке**

****

**Решение СЛАУ**

**1.Метод гауса**

****

****

То есть мы берем матрицу, оставляем первую строку, считаем коэфицент (нулевой элемент второй строки делим на первую)

И вычитаем из второй строки первую, умноженную на коэфицент

И так для каждой

Потом у нас получается система, что если решать снизу вверх, то переменные сами заполняются

**2.Метод гауса жордана.**

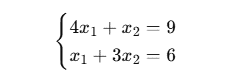
То же самое что и обычный гаус но мы интегрируем по столбцу, ищем наибольший элемепнт в столбце и переносим наверх

Потом делим всю строку (сверху на пивот), остальные строки обнуляем (строка – первый\_элемент\_в\_строке \* строку)

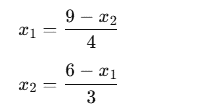
В итоге получится система диагонального вида с единицами

**3.Метода гауса зейделя.**

Берем систему уравнений



Приводим ее к такому виду, нам надо вынести чтобы одна переменная была выражена через остальные переменные и свободный член

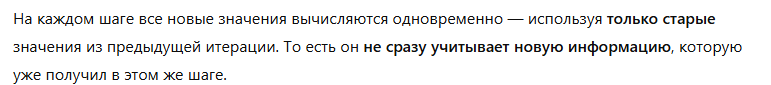


И начинаем считать с минимильного приблежения например х = 0, дальше берем и подставляем ответы в эту же формулу.

Чтобы проверить, мы полученные данные всставляем в исходные формулы чтобы получился ответ

**4 Метод простой итерации.**

То же самое что и метод гауса-зейделя, но

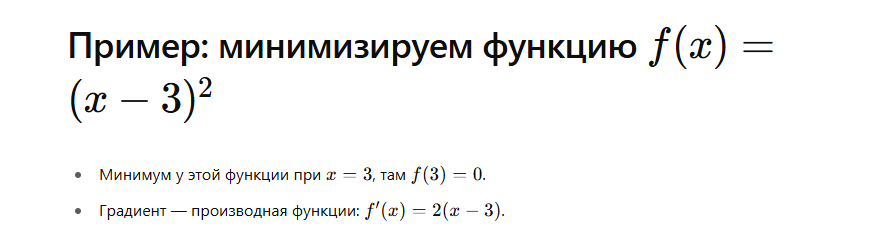


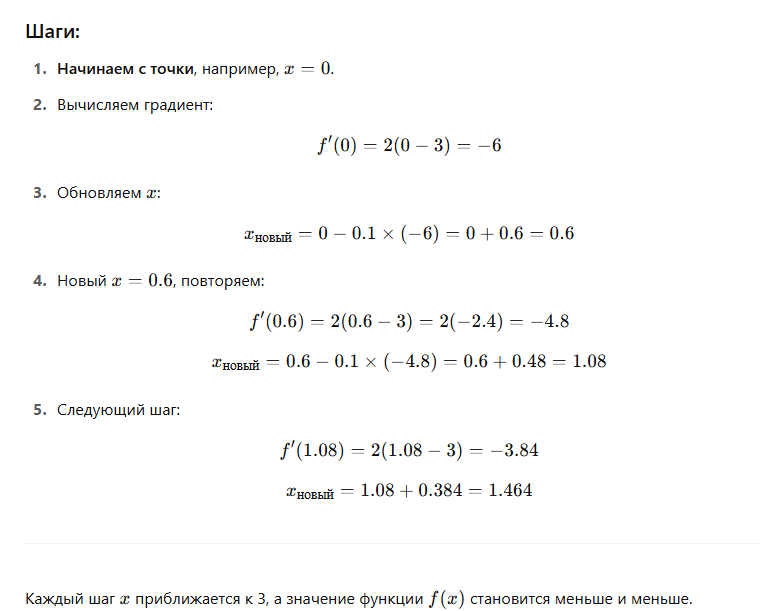
Поэтому он дольше сходится по сравнению с методом гауса-зейделя

**5.Градиентый метод**

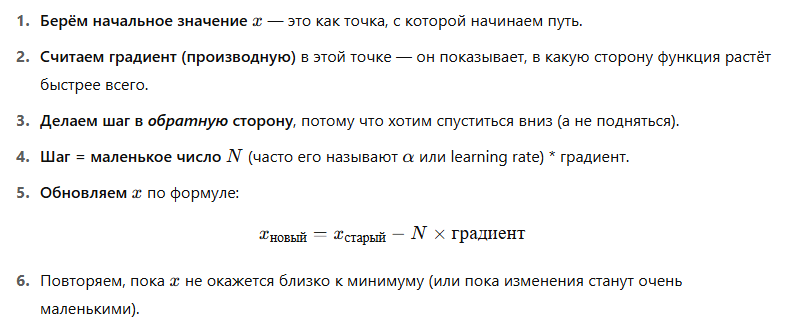
Это способ найти минимум функции (то есть такое значение, где функция принимает самое маленькое число).

Например, представь, что ты стоишь на горе в тумане и хочешь спуститься вниз к самой низкой точке, но ты не видишь далеко вокруг — только сразу под ногами. Градиентный метод — это как шаги вниз по склону, делая маленькие шаги в направлении, где земля спускается быстрее всего.



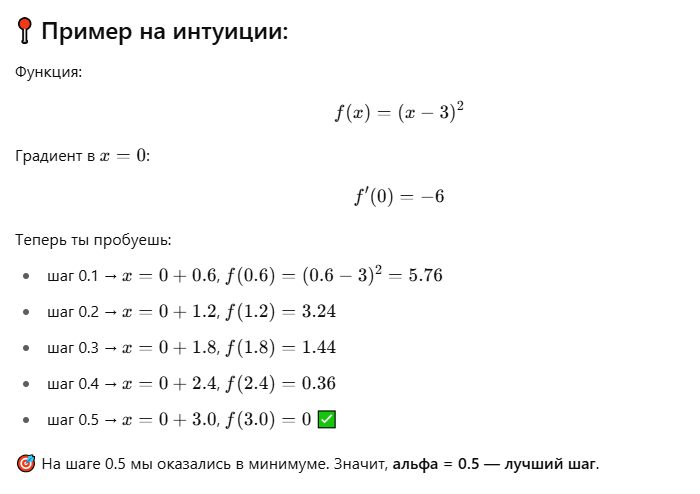


То есть если простым языком мы берем **N = маленький шаг**, считаем производную (Градиент) и **вычитаем из текущего х (Х- N \* градиент), после обновляем х**

****

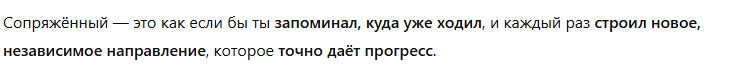
**6.Наизкорейший спуск**

Тот же самый метод, но мы оптимизируем минимальный шаг

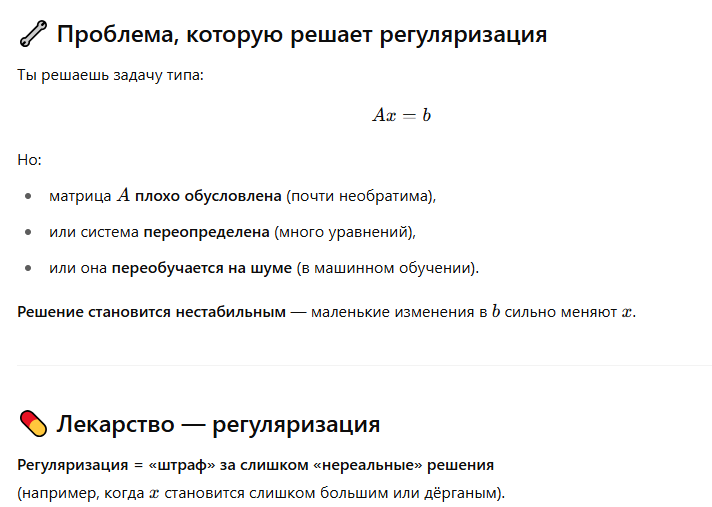


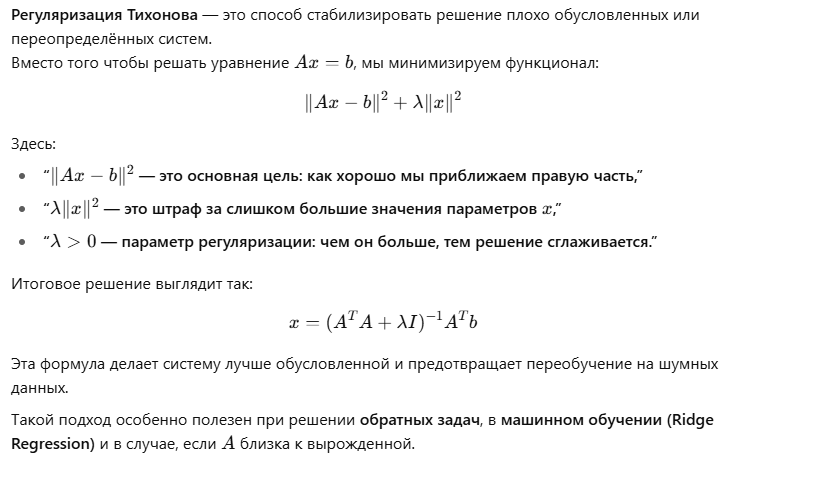
**7.Метод сопряженных направлений**

Это тоже самое что и градиентный с наизкорешим, только



**8.Регуляризация(Метод тихонова)**



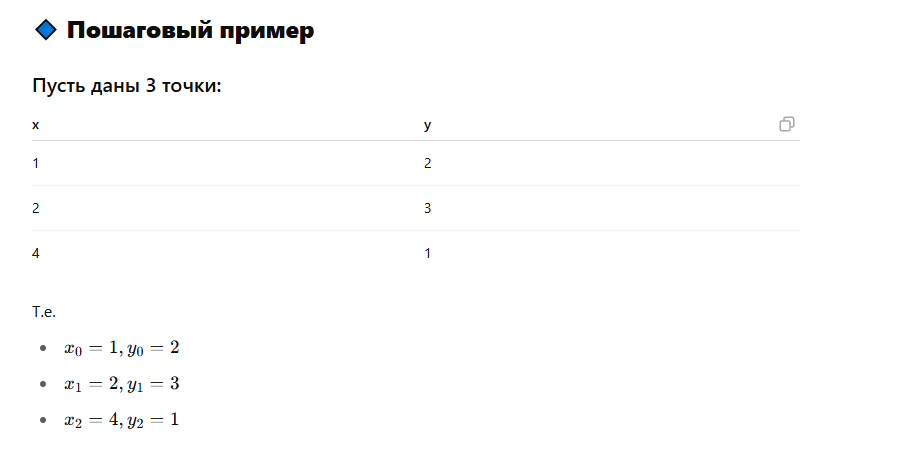


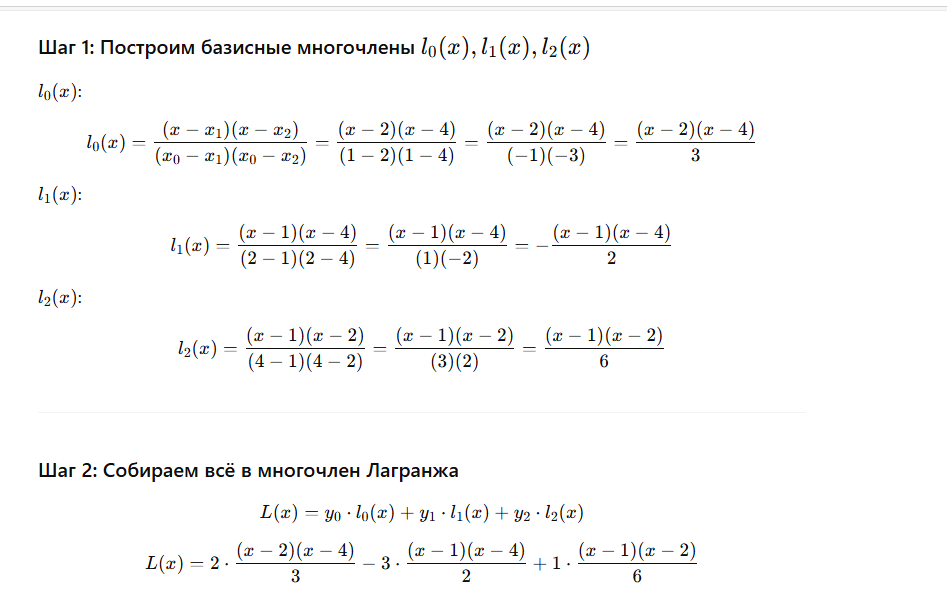
**Интерполяция, аппроксимация, приближение**

**1.Интерполяция многочленом Лагранжа**

Это метод если мы имеем точки функции, и нам надо исходя из них построить уравнение

Метод построения одного многочлена, который **точно проходит через все заданные точки**.

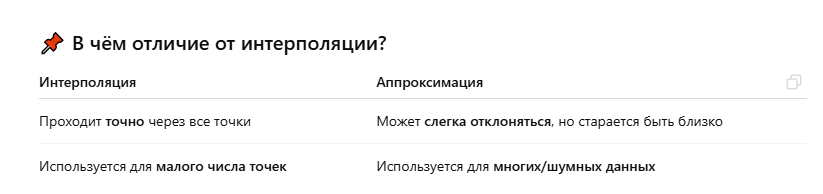




После этого всего мы упрощаем, то есть грубо говоря мы берем точки, загоняем их в уравнение l, в уравнение L, и сокращаем

**2.Полиномиальная аппроксимация**

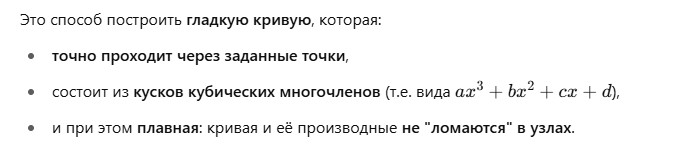
Простым языком мы пытаемся подобрать многочлен функции по точкам, но он не обязательно должен проходить через них

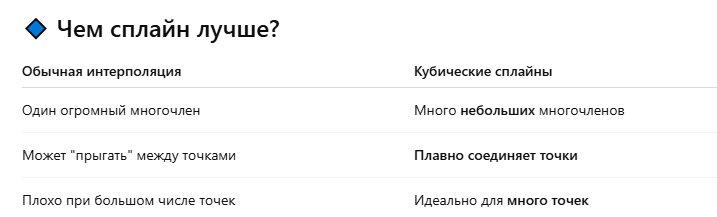


**3.Кубические сплайны (классические)**

Кубические сплайны — это когда ты **соединяешь точки не прямыми, а гладкими кусочками кривых**, которые ведут себя красиво.

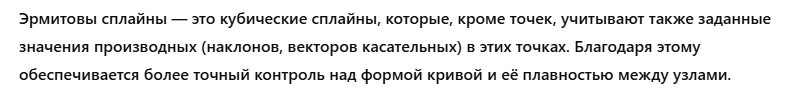
И строишь не один большой многочлен а маленькие многочленики

****

****

**4.Эрмитовы сплайны**

Это как кубические сплайны, но тут мы еще учитываем вектор наклона



**5.Кривая Безье**

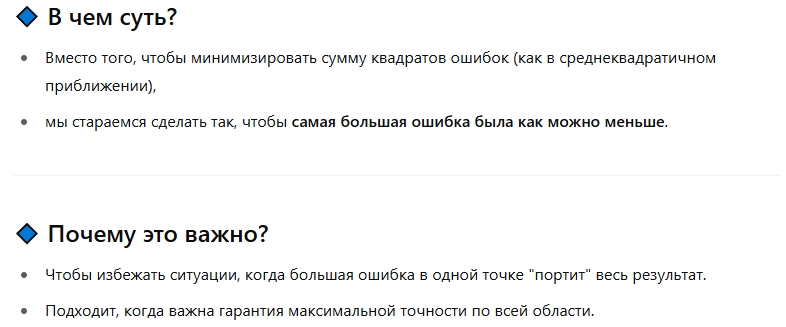
Кривая Безье обязательно проходит через начальную и конечную точки. Контрольные точки не лежат на кривой, но "тянут" её, задавая форму и направление. Чем больше контрольных точек, тем сложнее и точнее форма кривой.

**6.Среднеквадратичное приближение**

Это способ **найти функцию (например, многочлен), которая максимально хорошо "подходит" под набор данных**, минимизируя сумму квадратов отклонений между реальными точками и значениями функции

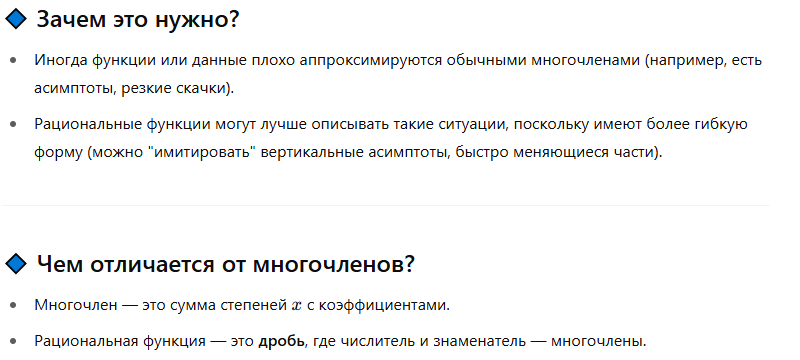
**7.Равномерное приближение**

Это способ подобрать функцию, которая **минимизирует максимальное отклонение** от заданных точек..



**8.Рациональное приближение**

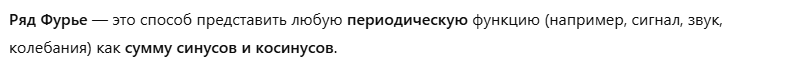
Рациональное приближение — метод приближения функции с помощью дроби из двух многочленов, что позволяет лучше описывать функции с резкими изменениями и особенностями, чем обычные многочлены.

****

**9.Метод наименьших квадратов**

Это способ найти **лучшее приближение данных с помощью функции (чаще всего — прямой или многочлена)**,  
так, чтобы сумма квадратов отклонений между фактическими данными и функцией была минимальна.

**Ряды и преобразование Фурье**

****

**1.Численное разложение в ряд фурье**

**Численное разложение в ряд Фурье** — это способ представить периодическую функцию, заданную в виде численных значений, как сумму синусов и косинусов разной частоты.  
Мы приближённо вычисляем коэффициенты ряда Фурье с помощью сумм, а не интегралов, и таким образом получаем приближённое выражение функции через гармоники.  
Это применяется, когда у нас нет точной формулы, а есть только таблица значений, например, при обработке сигналов или численном моделировании.

**2.Интегралы Фурье**

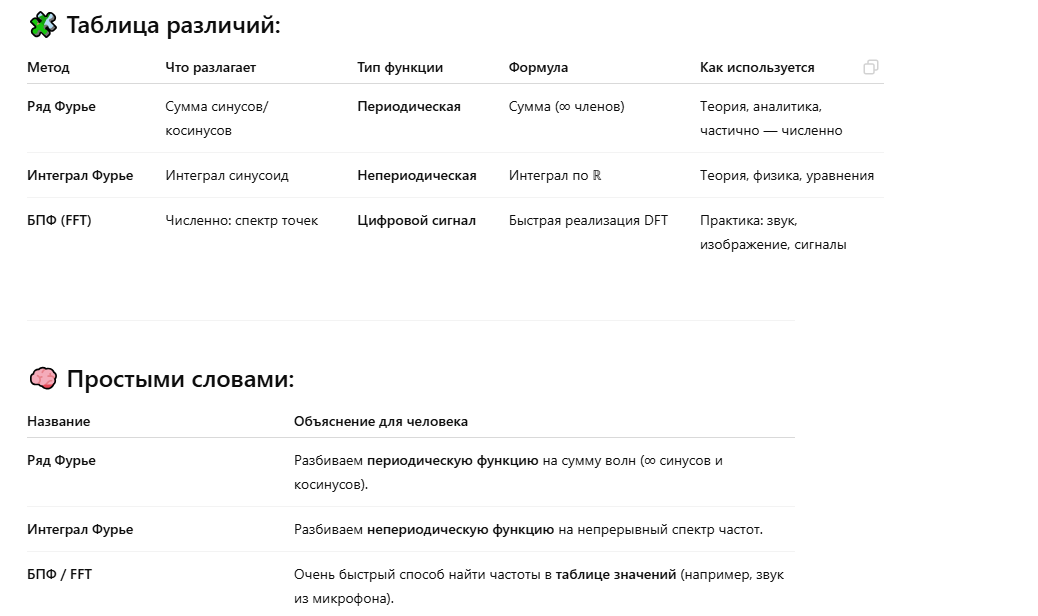
**Интеграл Фурье** — это обобщение ряда Фурье **для непериодических функций**.

Если ряд Фурье работает для функций, которые **повторяются** (периодические),  
то **интеграл Фурье** позволяет разложить функцию, которая определена на всей прямой

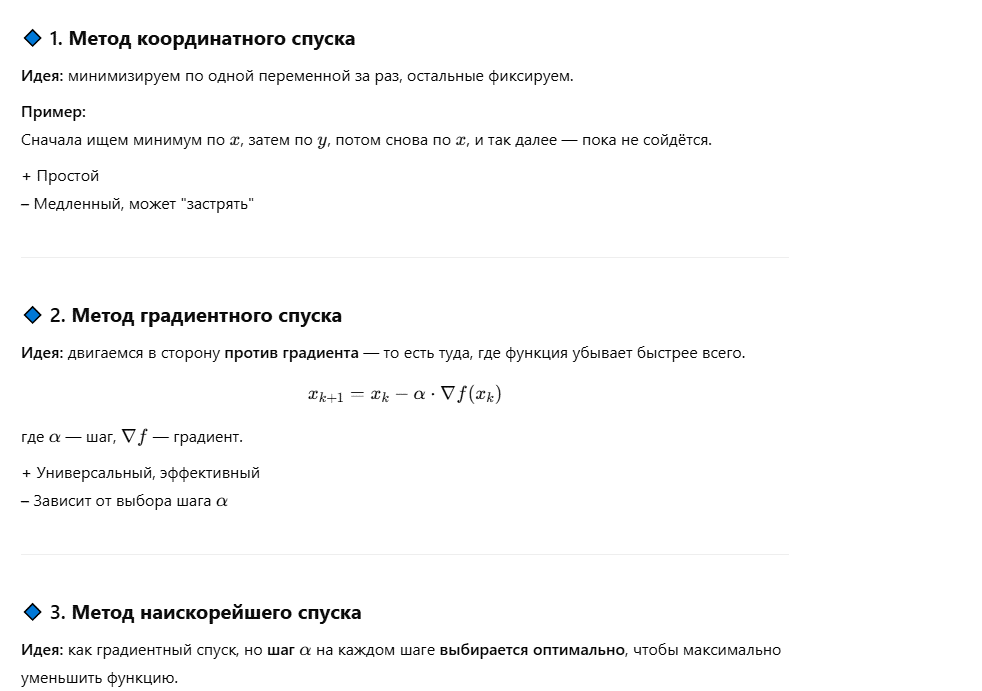
**3.Быстрое преобразование в ряд фурье**

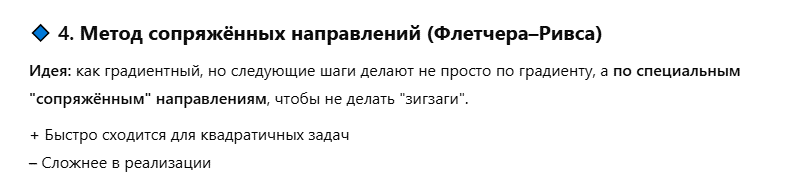
**Быстрое преобразование Фурье (БПФ / FFT)** — это **быстрый способ** вычислить **дискретное преобразование Фурье (DFT)**.

Оно используется, когда функция задана **в виде конечного числа точек** (например, 1024 отсчёта сигнала).

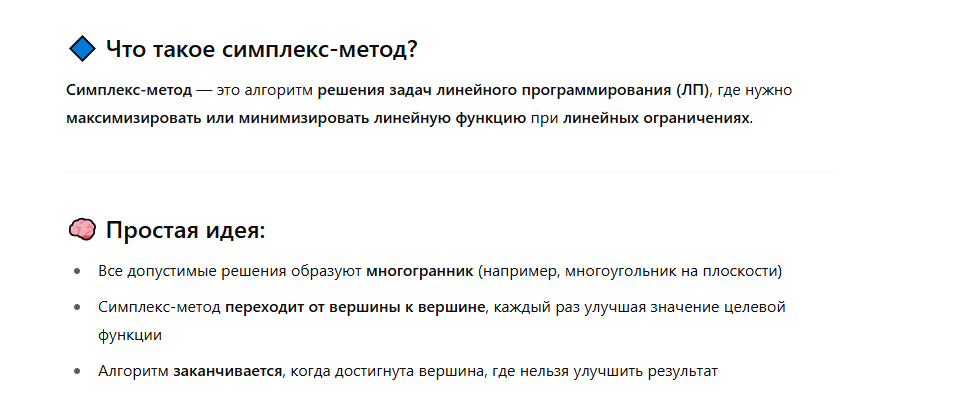
****

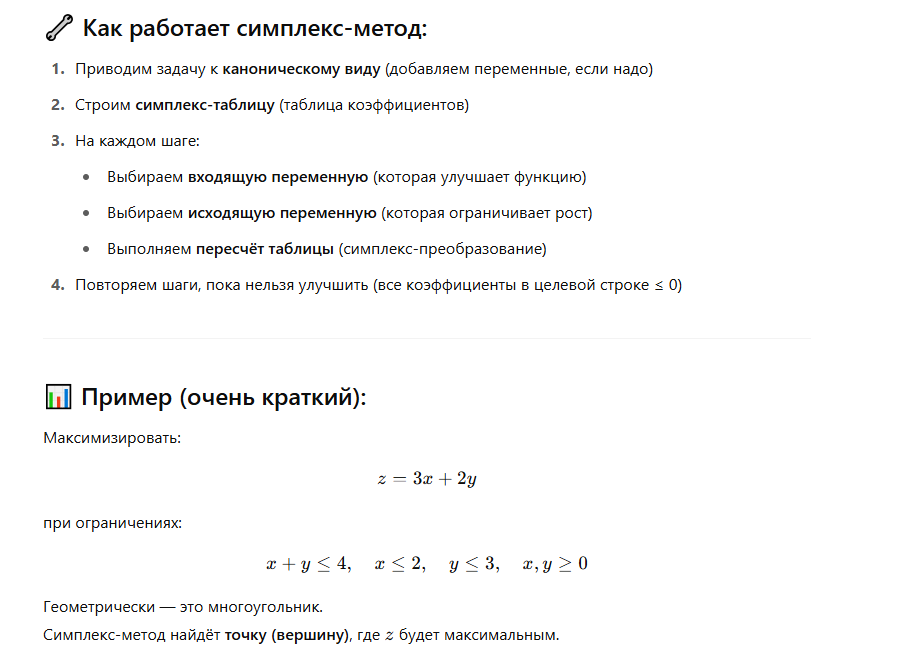
**Оптимизация и функционалы**



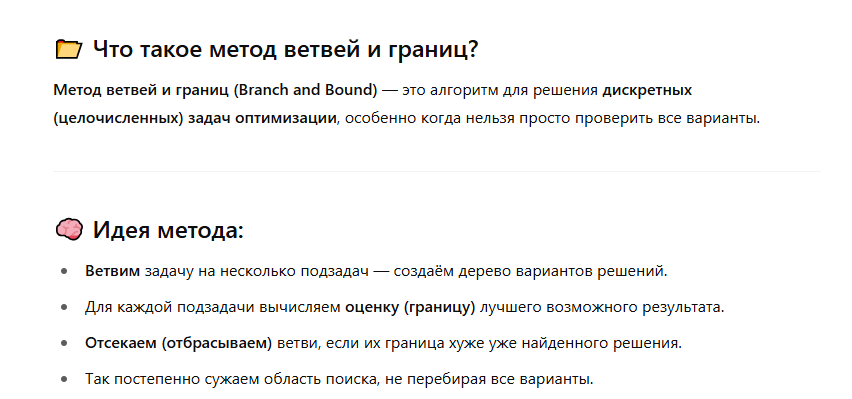


**Симплекс метод**

****

****

**Ветви и границы**

****