**2 лаба метод Якоби и Зейделя**

Рассматривается два итерационных метода – суть заключается в нахождении по приближенному значению величины следующего приближения, которое является более точным.

Условия для применения итерационного метода:

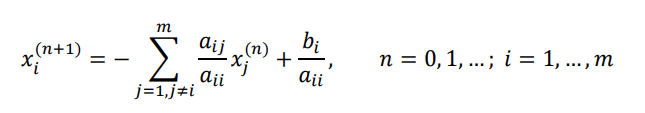
E> 0

Строгое диагональное преобладание.

Метод Якоби

Метод для приведения матрицы к удобному для итерации: из 1-го уравнения матрицы выражаем неизвестное x1 и т.д.

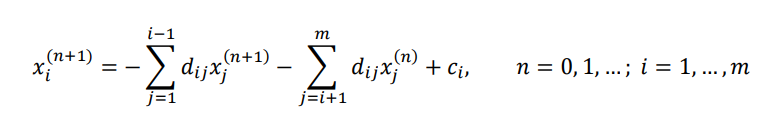
Если все диагональные элементы не равны нулю, то систему можно получить. По формуле



Метод Зейделя

При вычислении очередного n-го приближения к неизвестному, используются уже ранее найденные приближения к неизвестным, а не как в методе Якоби только n-ое приближение.

Мы на каждой итерации используем уже вычисленные значения, при этом и изменяется сама формула



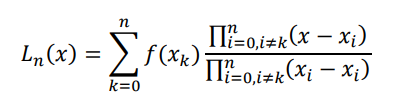
Ошибка округления – с каждым округлением, невязка становиться больше, результат через какое-то количество итераций, не будет соответствовать реальности.

**3 лаба интерполяционный многочлен Лагранжа (Ньютона)**

Интерполяция - – нахождение неизвестных промежуточных значений некоторой функции, по имеющемуся дискретному набору её известных значений, определенным способом.

Интерполяционный полином Лагранжа

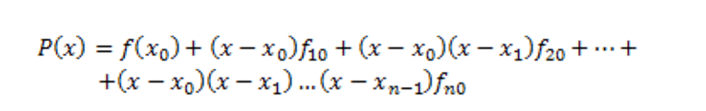
для построения многочлена Лагранжа используют следующую формулу:



xi, i = 0…n – заданные значения аргументов f(xk) = fk – соответствующие этим аргументам значение функции

При изменении количества узлов интерполяции, полином нужно пересчитывать только для Лагранжа, в Ньютоне такого нету).

Интерполяционный многочлен Ньютона является другой формой записи интерполяционного многочлена Лагранжа



**Отклонение —** это разность между полученной функцией и интерполяционной.

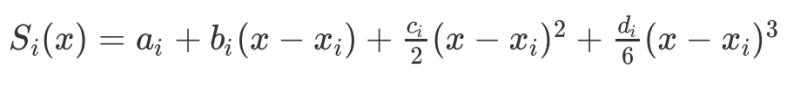
**4 лаба интерполирование сплайнами**

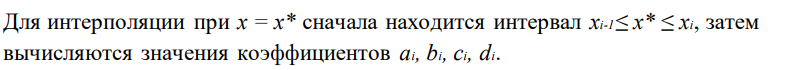
Отличие интерполяции сплайнами от интерполяции через полином Лагранжа заключается в количестве функций. В интерполяции сплайнами за промежутки между узлами интерполяции отвечает свой кубический многочлен. Преимущество такого способа интерполяции заключается в плавности переходов от одного сплайна к другому и непрерывности результирующей функции.

Достигается это благодаря равенству производных.

Равенство производных второго порядка – непрерывность сплайнов в узлах интерполяции (отсутствие изломов);

Равенство производных первого порядка обеспечивает равенство касательных для двух сплайнов в узле интерполяции. Это позволяет проходить по узлам интерполяции по максимально короткому маршруту (с соблюдением всех условий).

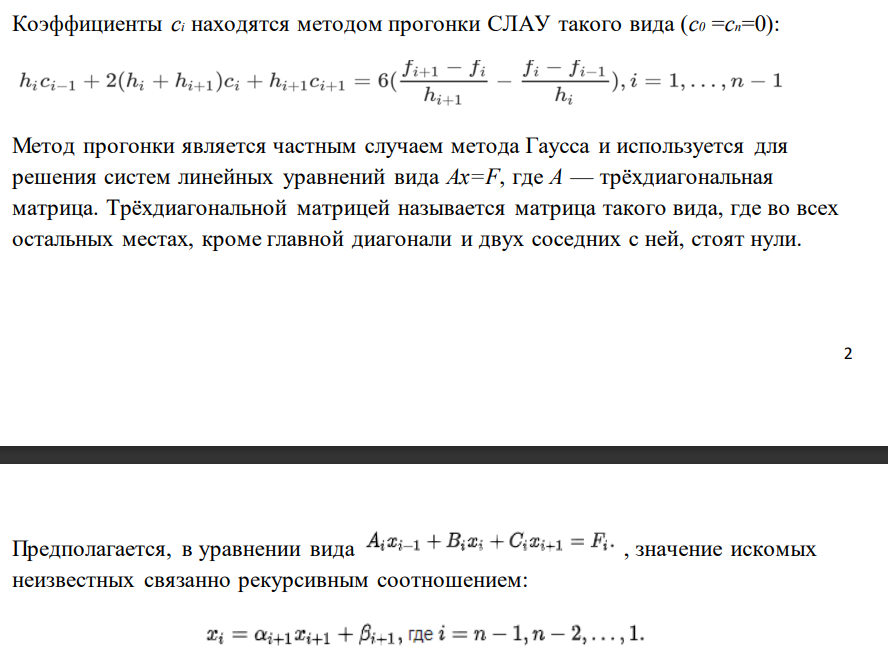




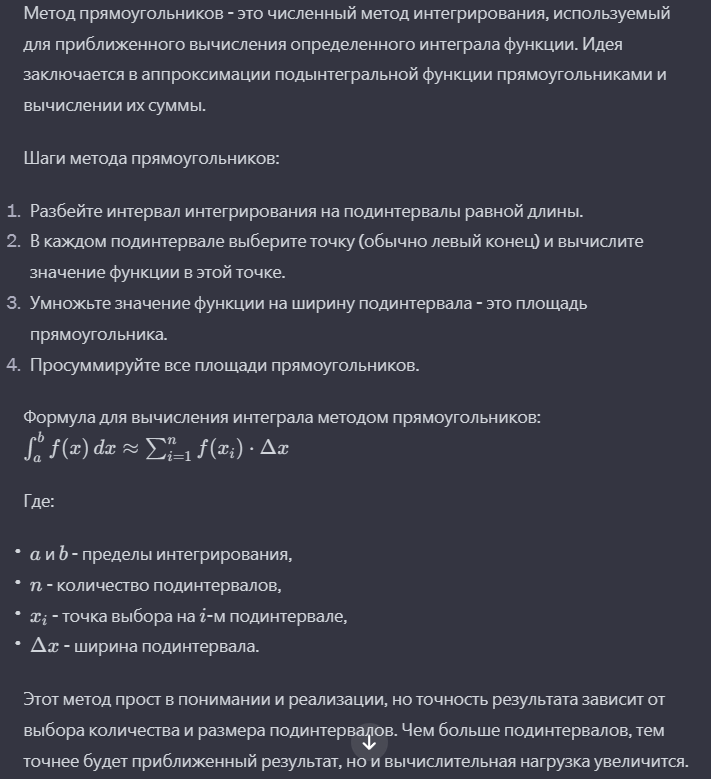
Требования для Сплайна:

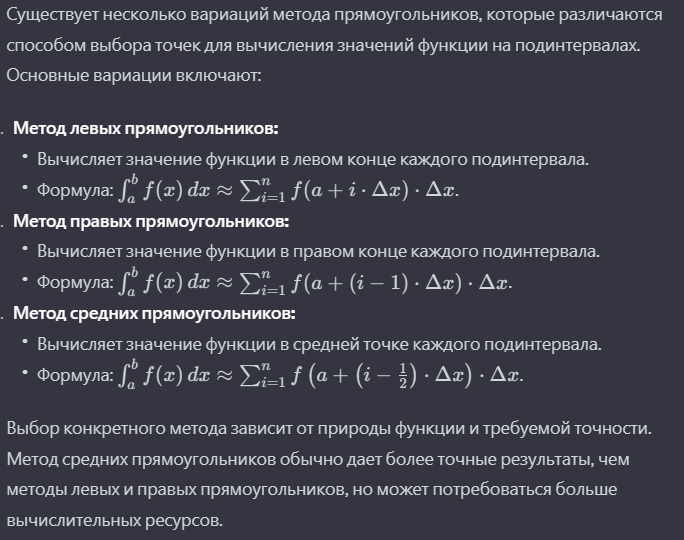
1. Гладкость в стыках, непрерывность - Первая производная и вторая равны
2. Проход через узловые точки
3. Задать условия для граничных точек

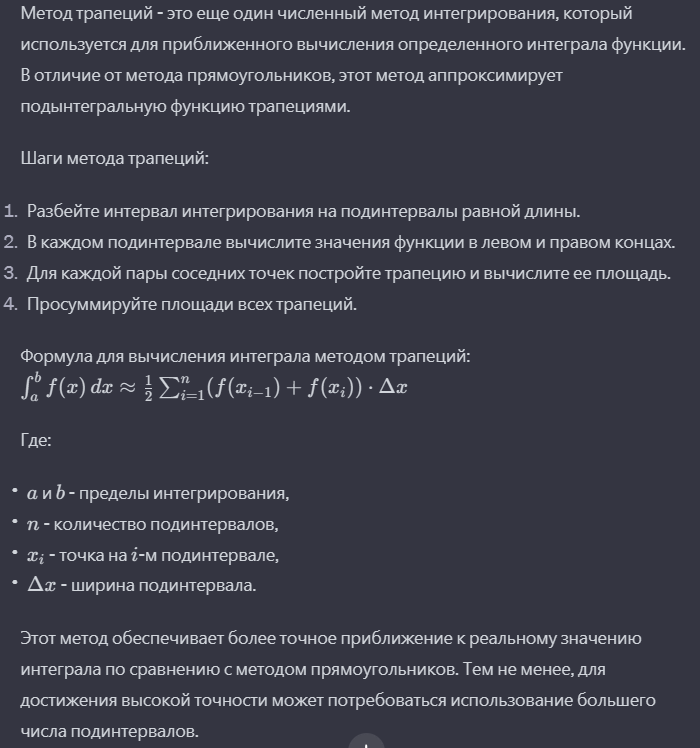
*Метод прогонки*

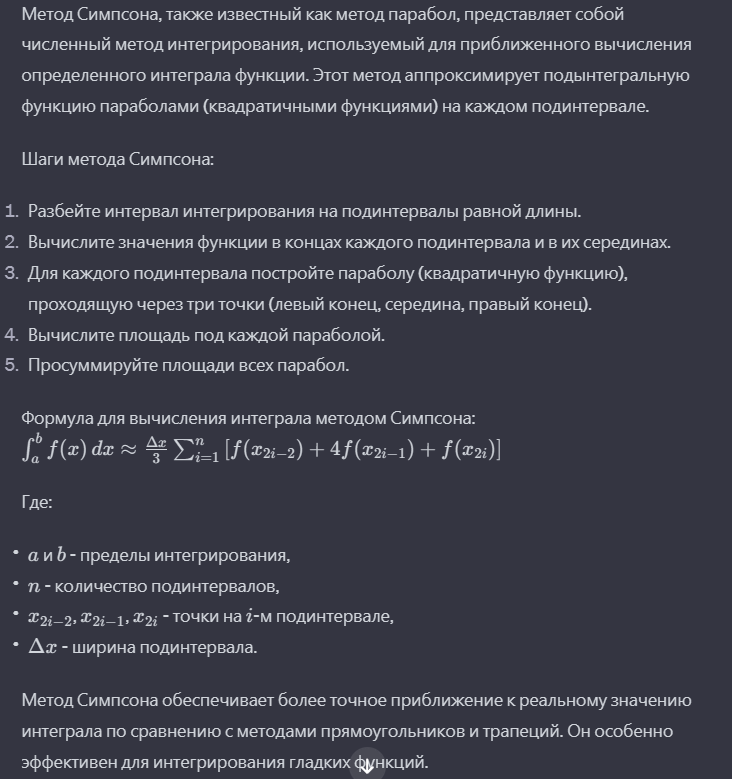


**5 лаба Методы прямоугольников, трапеции и Симпсона**

****

****

****

****

**6 лаба**

**Метод Монте-Карло**

Методом статистических испытаний или методом Монте-Каpло принято называть совокупность приемов, позволяющих получать решения задач при помощи многократных случайных испытаний.

**Вычисление интегралов методом Монте-Карло**

Графический метод



Где K – количество точек, лежащих под кривой.

Точки должны быть сгенерированы с помощью датчика равномерно распределенных случайных чисел

**Центр тяжести плоской фигуры**

Основная концепция метода:

1. Генератором случайных равномерных чисел, аналогично методу Монте-Карло, создать точки на определённой площади, в которую входит фигура.
2. Посчитать количество точек входящих в эту фигуру и общее количество точек.
3. Для нахождения Х необходимо найти сумму всех Х у точек внутри фигуры и разделить на количество этих точек.
4. Для нахождения координаты Y необходимо сделать тоже самое, что и для X.

В сам алгоритм мы передаем заданную функцию, верхний и нижний пределы интегрирования, количество точек и шаг интегрирования.

**7 лаба**

**Общее описание алгоритма**

Метод Ньютона решения систем нелинейных уравнений является обобщением метода Ньютона решения нелинейных уравнений, который основан на идее замены нелинейного уравнения линейным. Пусть *F*(*x*): R1→R1 - дифференцируемая функция и необходимо решить уравнение *F*(*x*)=0.

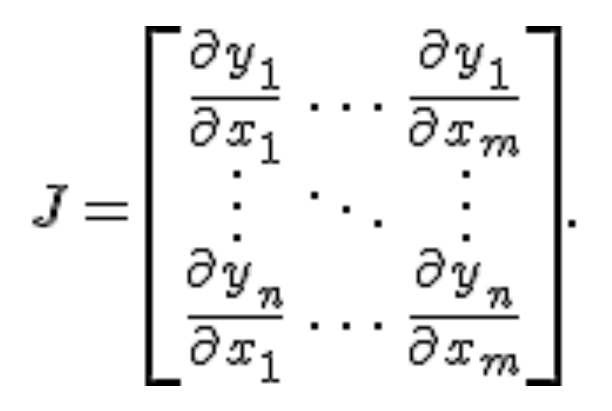
Взяв некоторое *x*0 в качестве начального приближения решения, мы можем построить линейную аппроксимацию

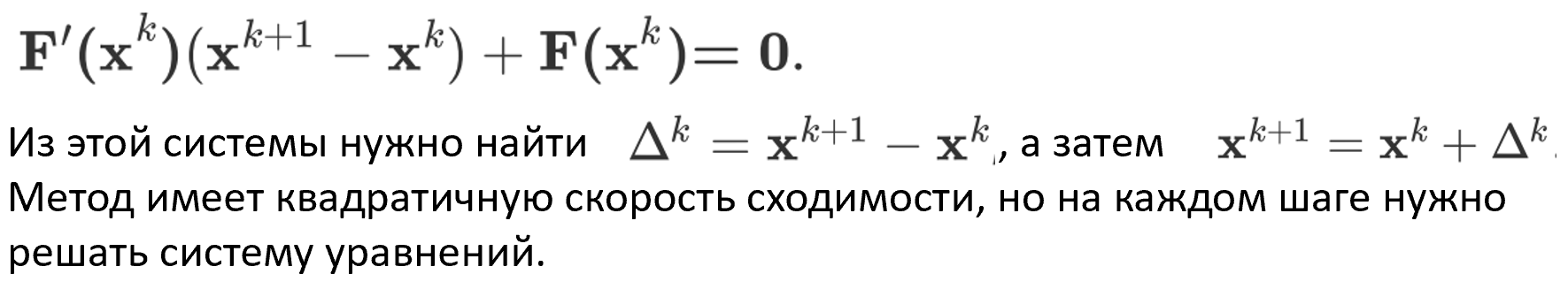
*F*(*x*) в окрестности *x*0: *F*(*x*0+*h*) ≈*F*(*x*0) +*F*′(*x*0)*h* и решить получающееся линейное уравнение *F*(*x*0) +*F*′(*x*0)*h*=0.

Таким образом получаем итеративный метод: *xk*+1=*xk*−*F*′(*xk*)−1*F*(*xk*), *k*=0, 1, ….

#### **Вычисление матрицы Якоби**

Пусть задана система nфункций y_1(x_1, x_2, \dots x_m) \dots y_n(x_1, x_2, \dots x_m)от mпеременных. **Матрицей Якоби** или **якобианом** данной системы функций называется матрица, составленная из частных производных этих функций по всем переменным.



В векторной форме: 

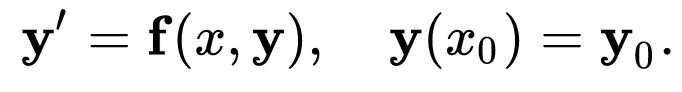
Якобиан вычисляется каждую итерацию, но в модифицированном лишь раз в несколько итераций, что снижает сходимость с квадратичной до линейной, но дает уменьшение времени вычислений за счет отсутствия перерасчета Якобиана на каждом шаге.

Ход работы:

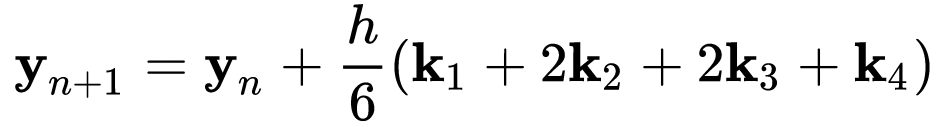
1. Задаем начальное приближение x.
2. Задаем начальное приближение y.
3. Задаем точность вычислений ε.
4. Рассчитываем матрицу Якоби.
5. Находим дельты для текущей итерации, перемножая матрицу Якоби на исходную систему, записанную в виде матрицы.
6. Отнимаем от начальных приближений x и y соответствующие им дельты.
7. Если максимальная по модулю текущая дельта <= предыдущей макс. по модулю дельте, то проверяем условие прерывания цикла, иначе сходимость не достигается, и мы прекращаем вычисления.
8. Если максимальная по модулю текущая дельта <= ε, то выходим из цикла (точность достигнута) и получаем результат, иначе продолжаем итерационный процесс.

**8 лаба**

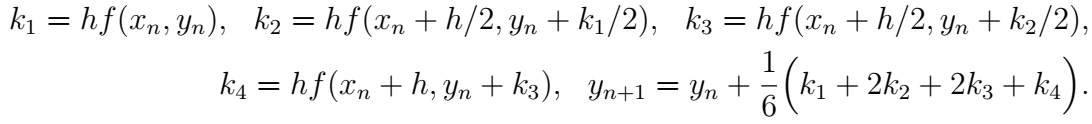
**Метод Рунге-Кутты** используют для решения [задачи Коши](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%9A%D0%BE%D1%88%D0%B8) для [обыкновенных дифференциальных уравнений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8B%D0%BA%D0%BD%D0%BE%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B4%D0%B8%D1%84%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F) и их систем, он обладает точностью метода *Ο*4(*h*).

Задача Коши для системы обыкновенных дифференциальных уравнений первого порядка

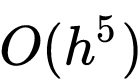
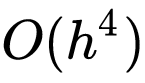
Приближенное значение в последующих точках вычисляется по итерационной формуле:



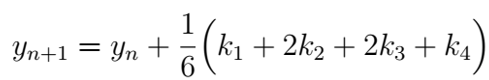
Вычисление нового значения проходит в четыре стадии:



где **h h** — величина шага сетки по **x**xxx.

Этот метод имеет четвёртый порядок точности. Это значит, что ошибка на одном шаге имеет порядок O ( h 5 ) , а суммарная ошибка на конечном интервале интегрирования имеет порядок O ( h 4 ) .

Ход решения:

1. Задаем промежуток [a, b]
2. Задаем начальное вхождение f(t), где t = 1, 2, …, j.
3. Задаем соответствующие начальным вхождениям функции.
4. Задаем точность или количество шагов.
5. При вводе количества шагов используем постоянный шаг, а при вводе точности автоматический шаг.
6. Вычисляем шаг, как , где h – шаг, n – количество шагов (итераций).
7. Находим для всех функций коэффициенты k1, k2, k3 и k4.
8. Находим dy для всех функций, берем максимальный dy.
9. Находим текущее приближение с использованием предыдущего: 

И используем его для вычисления следующего приближения.

1. Повторяем, пока не дойдем до введенного количества шагов.
2. При использовании автоматического шага производим вычисления методом Рунге-Кутты (каждый раз передаем количество шагов и действуем, как описано выше для постоянного шага).
3. Если максимальное последнее вычисленное отклонение dy > ε, то считаем, что алгоритм завершил работу на текущем шаге, иначе умножаем шаг на 2 и повторяем пункт 10.
4. Получаем суммарную погрешность и изменения dy для соответствующих функций.