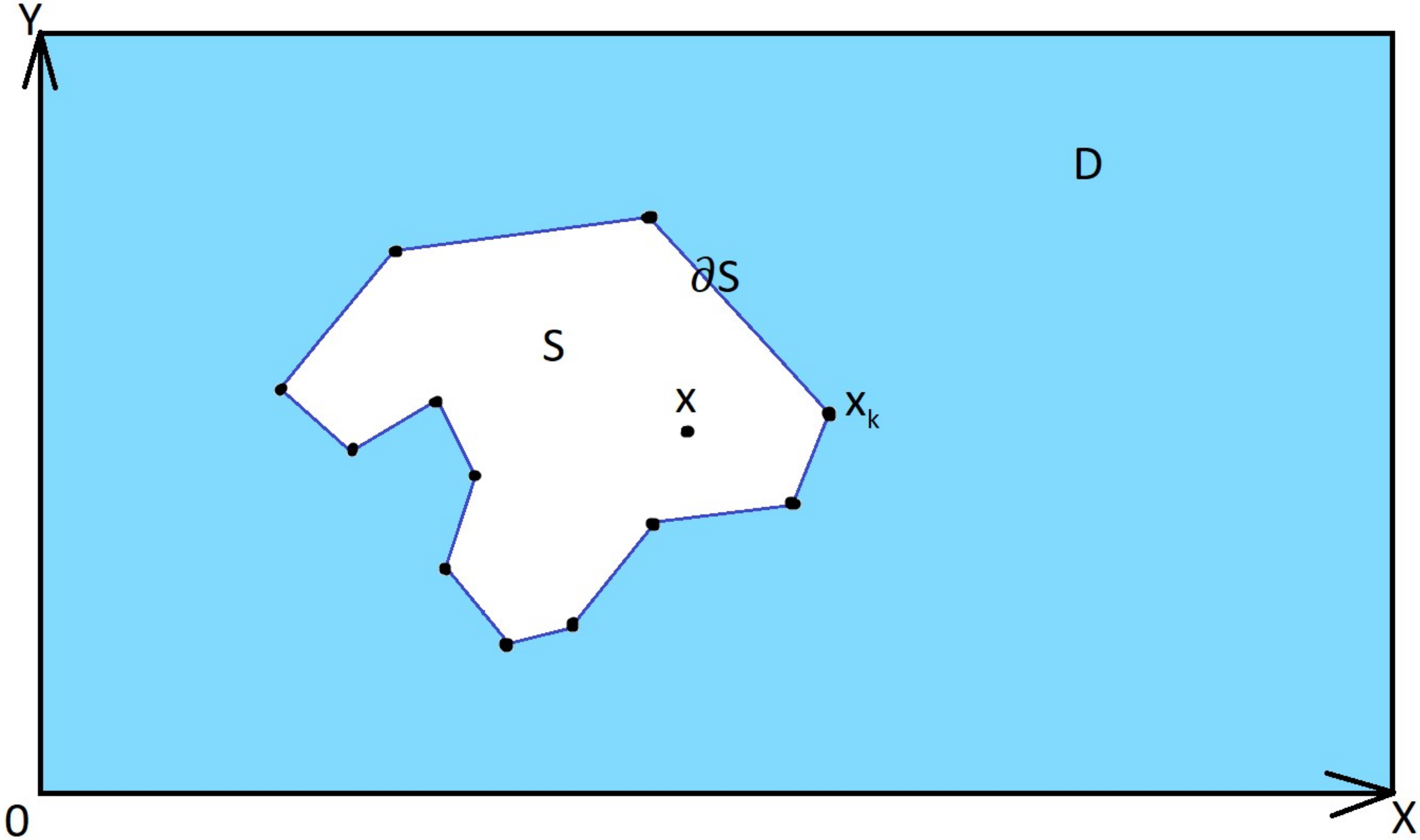
Численное решение уравнения переноса

Студент 421 группы
Сенченок Григорий Антонович.
Научный руководитель:
д.ф.-м.н., профессор
Меньшов Игорь Станиславович.

Введение

- В данной работе рассматривается проблема численного расчета движения твердого тела в сплошной среде.
- Твердое тело задано как совокупность точек, лежащих на его границе.
- Сплошная среда задается векторным полем скоростей в любой точке.

Введение



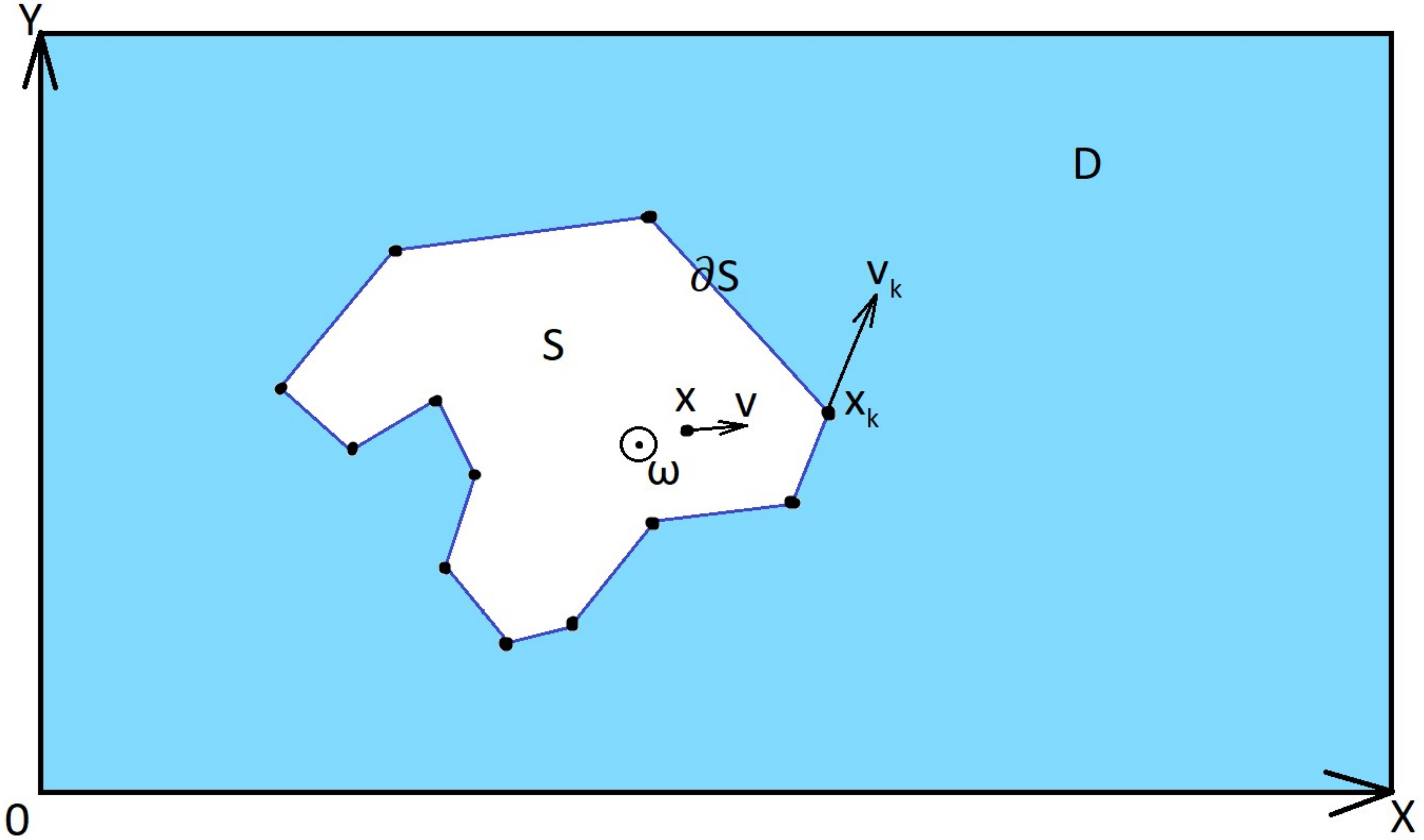
Постановка задачи

1. Нахождение векторного поля скоростей твердого тела
2. Нахождение характеристической функции твердого тела и жидкости в рассматриваемой области на каждом временном шаге

1. Поле скоростей твердого тела

- Твердое тело представляет собой совокупность точек, расстояния, между положениями которых не изменяются.
- Начальные условия задачи движения твердого тела:
  1. Положение всех точек в начальный момент времени
  2. Скорость центра масс
  3. Угловая скорость

1. Поле скоростей твердого тела



1. Поле скоростей твердого тела

Необходимо найти:

1. Поле скоростей в области D, индуцированное твердым телом при движении
2. Траекторию всех точек твердого тела для сравнения результатов решения уравнения переноса

Для нахождения положения и скоростей точек твердого тела в любой момент времени используется формула Эйлера:

$$\frac{d\vec{x}_k}{dt} = \vec{v}(t) - [(\vec{x}_k - \vec{x}) \times \vec{\omega}(t)], \qquad \vec{x}_k|_{t=0} = \vec{x}_k(0)$$

2. Характеристическая функция твердого тела

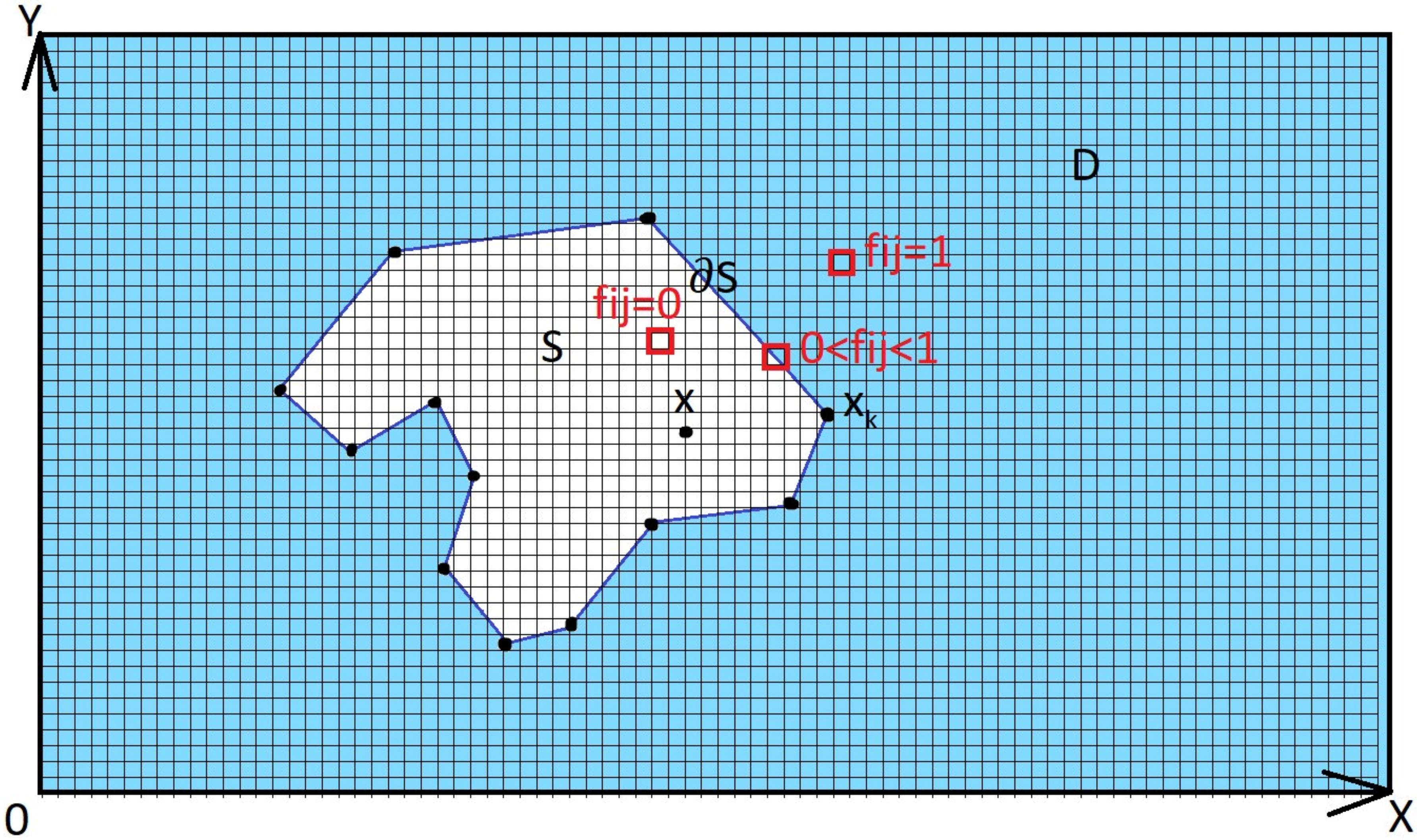
- Характеристическая функция твердого тела представляет собой функцию – индикатор жидкости:

$$f(x,t) = \begin{cases} 0, & \text{если в точке } x \text{ в момент времени } t \text{ нет жидкости} \\ 1, & \text{если в точке } x \text{ в момент времени } t \text{ есть жидкость} \end{cases}$$

- Уравнение переноса — дифференциальное уравнение в частных производных, описывающее изменение скалярной величины в пространстве и времени:

$$\frac{\partial f}{\partial t} + \nabla \cdot (uf) - f\nabla \cdot u = 0$$

2. Характеристическая функция твердого тела



2. Характеристическая функция твердого тела

- Начальные условия для уравнения переноса:
  1. Поле скоростей u
  2. Начальное положение точек границы твердого тела
- Необходимо найти:

Характеристическую функцию f в области D на отрезке времени [0; T]

Постановка задачи

- Таким образом, задачу можно представить в виде системы дифференциальных уравнений:

$$
\begin{cases}
\dfrac{\partial f}{\partial t} + u\nabla \cdot f = 0 \\
\dfrac{d\vec{x}}{dt} = \vec{v}(t) \\
\vec{v}_k(t) = \vec{v}(t) + [\vec{\omega}(t) \times \vec{r}_k(t)]
\end{cases}
$$

- С начальными условиями:

$$
\begin{cases}
f|_{t=0} = f(x,0) \\
x|_{t=0} = \vec{x}(0) \\
\vec{x}_k|_{t=0} = \vec{x}_k(0)
\end{cases}
$$

Точное решение: Virtual Motion

- Формула Эйлера:

$$\vec{v}_k(t) = \vec{v}(t) + [\vec{\omega}(t) \times \vec{r}_k(t)]$$

- При подстановке в нее rk и vk и интегрировании по времени дает:
  - Формулу для расчета положения центра масс:

$$\vec{x} = \vec{x}(0) + \int_0^t \vec{v}(t)dt$$

  - Уравнение для расчета положения k-ой точки твердого тела:

$$\vec{x}_k(t) - \int_0^t [\vec{x}_k(\tau) \times -\vec{\omega}(\tau)]d\tau = \int_0^t (\vec{v}(\tau) + [\vec{x}(\tau) \times \vec{\omega}(\tau)])d\tau + \vec{x}_k(0)$$

Дискретизация

- Необходимо найти решение полученных интегральных уравнений на отрезке времени T=[0; t]
- Даный отрезок разбивается на stepN подотрезков [t(i-1); t(i)], i=1..stepN. Длина каждого отрезка h=t(i)-t(i-1)

Положение центра масс

- Интеграл $\vec{x} = \vec{x}(0) + \int_0^t \vec{v}(t)dt$ был рассчитан методом квадратур. В качестве квадратурной формулы была использована составная формула трапеции.

```
VectorArray getXc(VectorArray v, double tMax, VectorXd xc0) {
    VectorArray xc = integrateVector(v, 0, tMax);
    for (int i = 0; i < xc.size(); i++)
        xc(i) = xc0 + xc(i);
    return xc;
}
```

Положение k-ой точки твердого тела

- Функция в правой части уравнения

$$\vec{f}(t) = \int_{0}^{t} (\vec{v}(\tau) + [\vec{x}(\tau) \times \vec{\omega}(\tau)])d\tau + \vec{x}_k(0)$$

была рассчитана методом квадрату с использованием составной квадратурной формулы трапеции.

```
VectorArray getRightFuncNoMargin(VectorArray v, Vector3Array omega, VectorArray xc, double maxT) {
    VectorArray integrand(v.size());
    for (int i = 0; i < v.size(); i++) {
        VectorXd xcTau = xc(i);
        Vector3d xcTau3(xcTau(0), xcTau(1), dimN < 3 ? 0 : xcTau(2));
        Vector3d crossProd = xcTau3.cross(omega(i));
        VectorXd crossProdX(dimN);
        for (int i = 0; i < dimN; i++) crossProdX(i) = crossProd(i);
        integrand(i) = v(i) + crossProdX;
    }
    return integrateVector(integrand, 0, maxT);
}
```

- Далее интегралы в уравнении заменяются квадратурными формулами, и в результате мы приходим к следующей системе линейных алгебраических уравнений:

$$\vec{x}_k^i - \left[\vec{x}_k^i \times -hA_i^{(stepN+1)}\overrightarrow{\omega_i}\right] = \overrightarrow{f_\imath} + h\sum_{j=0}^{i-1} A_j^{(stepN+1)}\left[\vec{x}_k^j \times \overline{-\omega_j}\right]$$

Обозначим:

$$\vec{x}_k^i = \vec{x}$$

$$-hA_i^{(stepN+1)}\overrightarrow{\omega_i} = \vec{b}$$

$$\overrightarrow{f_\imath} + h\sum_{j=0}^{i-1} A_j^{(stepN+1)}\left[\vec{x}_k^j \times \overline{-\omega_j}\right] = \vec{c}$$

И в результате:

$$\vec{x} + \left[\vec{b} \times \vec{x}\right] = \vec{c}$$

- Операция векторного произведения заменяется на произведение кососимметрической матрицы на вектор, и после выражения x получим:

$$\vec{x} = \left(E + [\vec{b}]_{\times}\right)^{-1} \vec{c}$$

- Таким образом, мы получаем алгоритм последовательного расчета значений вектора xki для i=0..stepN.

```
VectorArray getXk(VectorArray rightFunc, Vector3Array omega, VectorArray xc, double maxT)
```

В качестве квадратурной формулы была использована комбинация составной формулы Симпсона и правила трех восьмых.

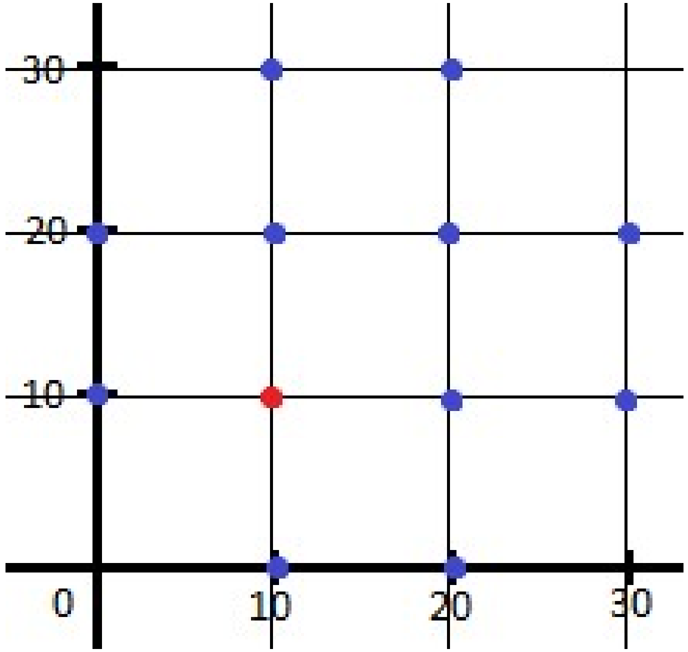
Результаты вычислений

- В результате была реализована программа для вычисления положений k-ой точки твердого тела в моменты времени ti, i=0..stepN. Результаты вычислений записываются в файл “output.txt”.
- Для более наглядной визуализации результатов расчет была реализована программа на языке программирования Python, которая создает анимацию движения твердого тела по предрассчитанным данным о положениях точек.

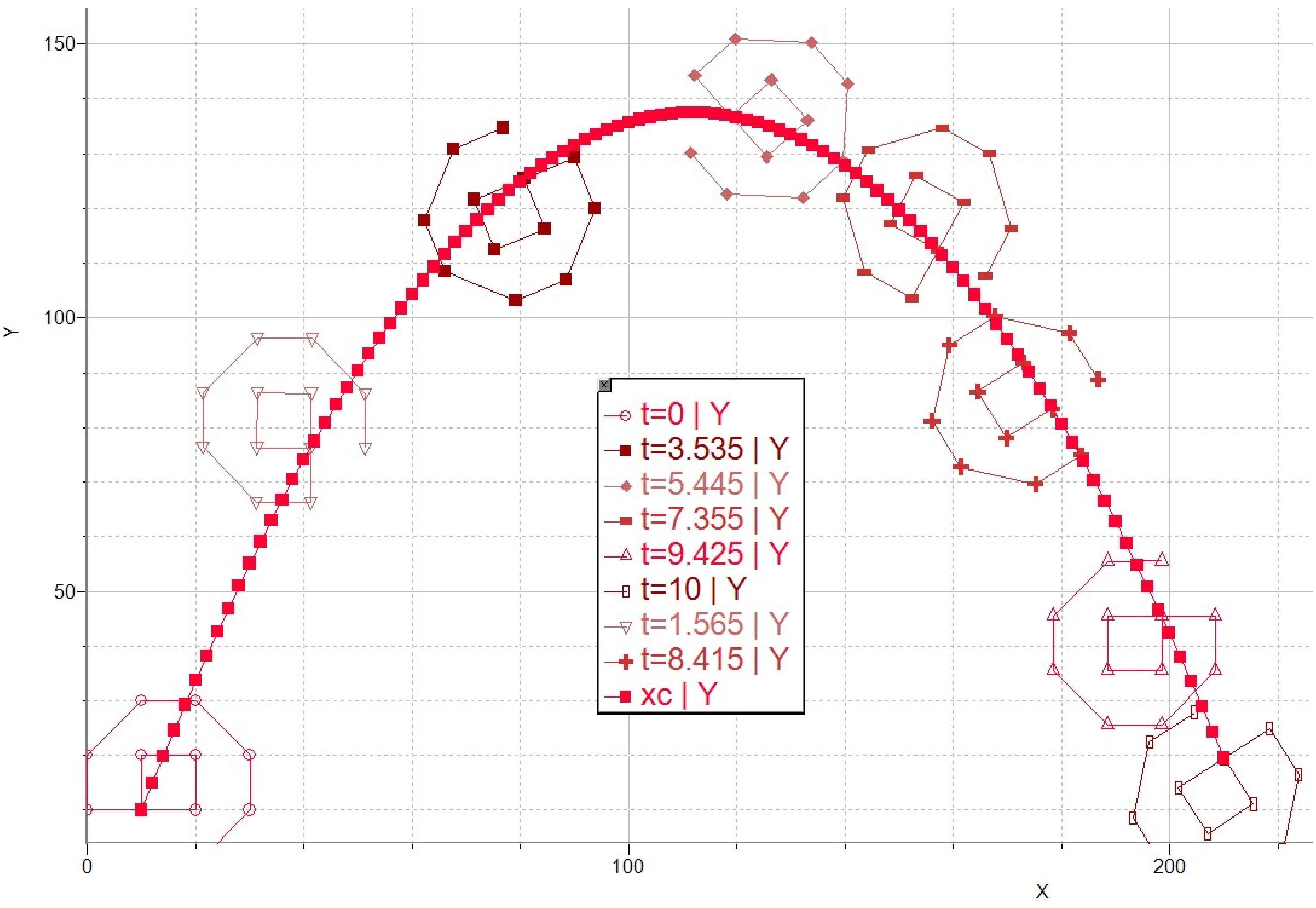
Пример расчета

```
function<VectorXd(double)> v = [=](double t)->VectorXd {
        VectorXd v(n);
        v(0) = 1;
        v(1) = 50 - 9.81 * t;
        return v;
    };
```

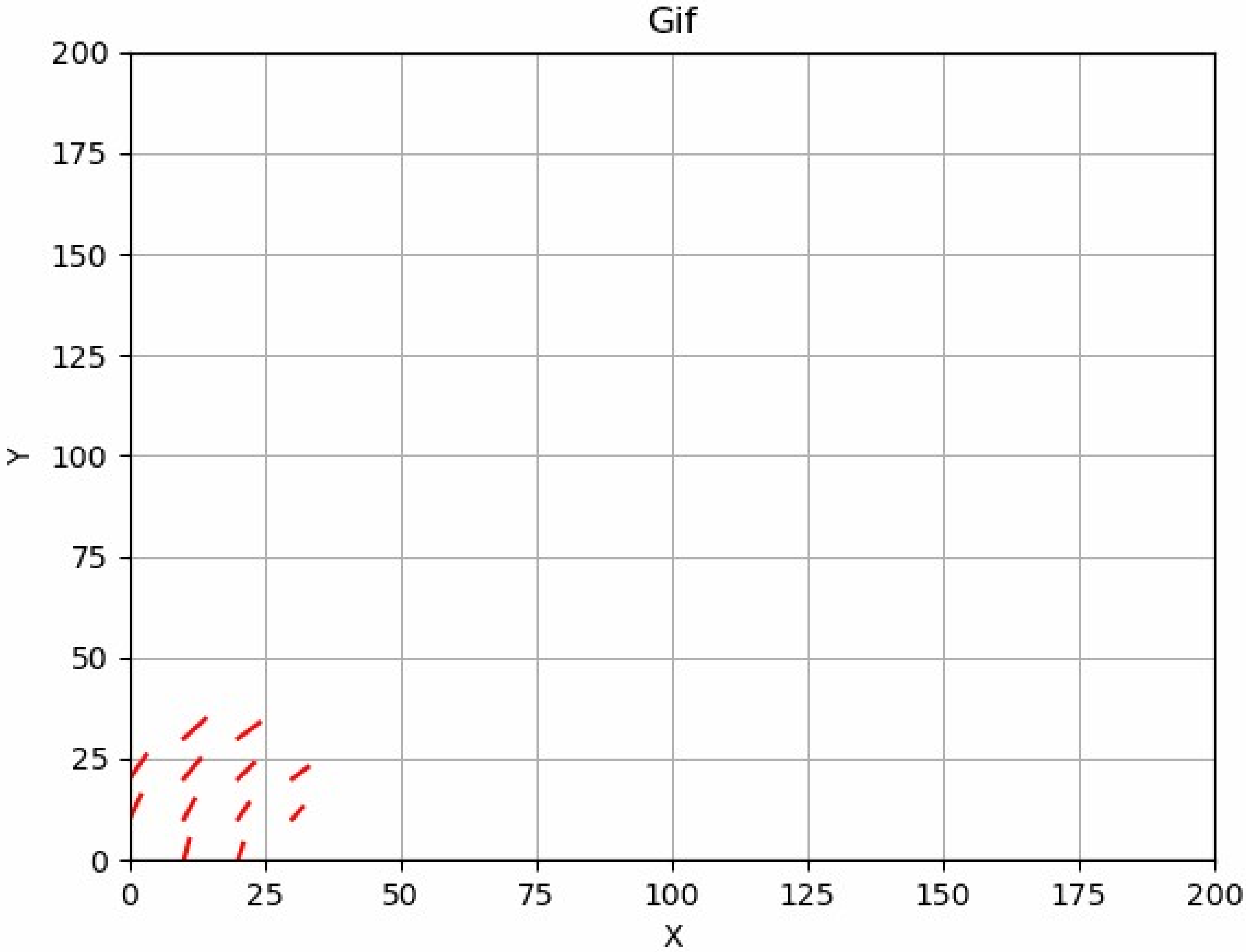
```
function<Vector3d(double)> omega = [=](double t)->Vector3d {
    return Vector3d(0, 0, 1);
};
```




```
int stepsN = 100;
double h = 0.1;
```



150
100
50
Y
0
100
200
X
t=0 | Y
t=3.535 | Y
t=5.445 | Y
t=7.355 | Y
t=9.425 | Y
t=10 | Y
t=1.565 | Y
t=8.415 | Y
xc | Y



Gif
200
175
150
125
100
75
50
25
0
Y
0
25
50
75
100
125
150
175
200
X

Погрешность вычислиней

- Ошибка:

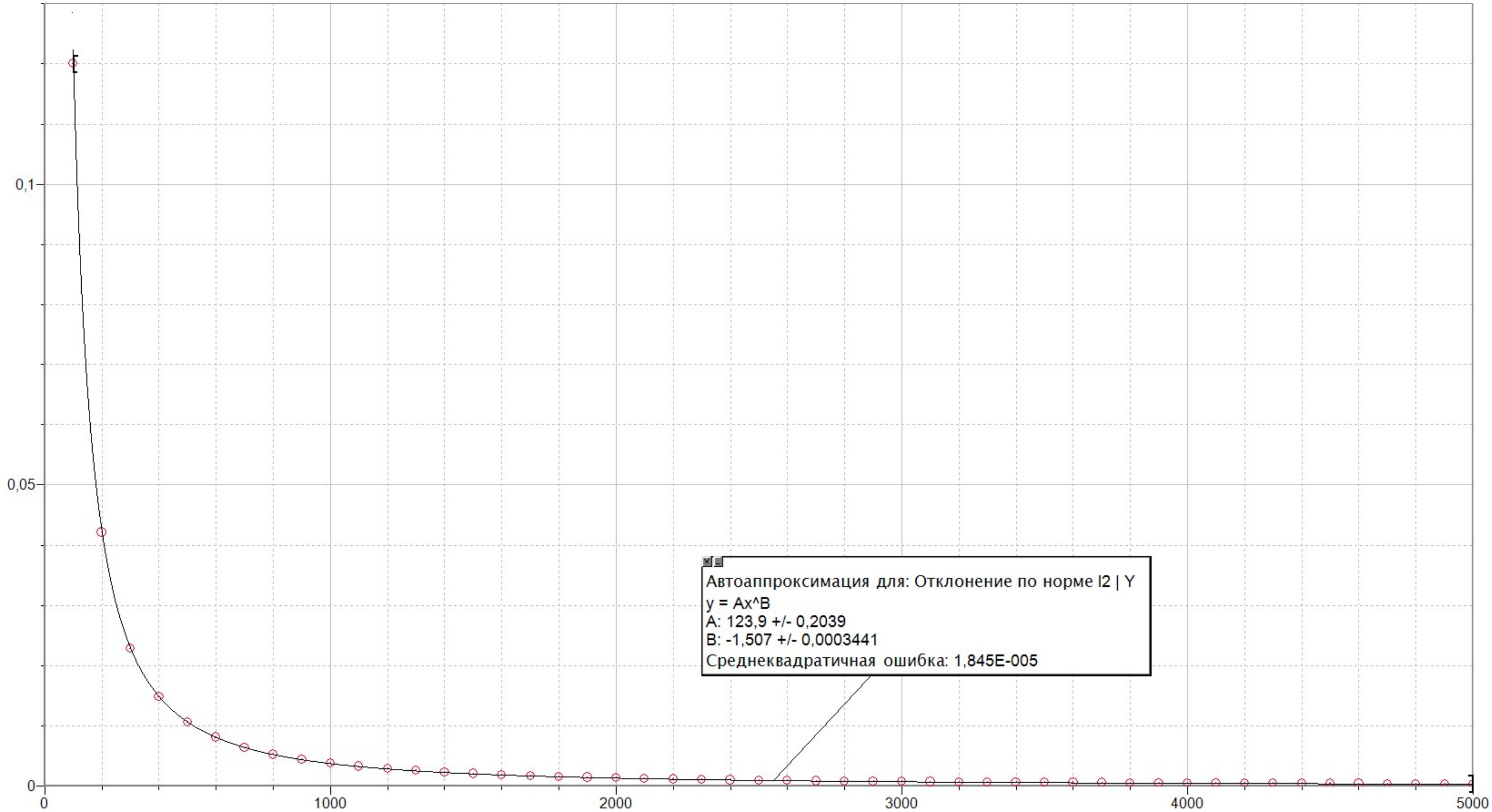
$$error_k = \sqrt{\sum_{i=0}^{stepN} \left\| \overrightarrow{x_k^i} - \overrightarrow{xreal_k^i} \right\|^2}$$

  4000 шагов на отрезке времени 10с (длина временного шага 0.0025)

- Для расчета траектории центра масс:
  error0=2.47756e-10.

- Для расчета траектории 3-ей точки из примера:
  error3= 0.000469316

Сходимость



Автоаппроксимация для: Отклонение по норме l2 | Y
y = Ax^B
A: 123,9 +/- 0,2039
B: -1,507 +/- 0,0003441
Среднеквадратичная ошибка: 1,845E-005
0,1
0,05
0
1000
2000
3000
4000
5000

Схема THINC

- Для численного решения уравнения переноса была рассмотрена схема THINC (tangent of hyperbola for interface capturing: гиперболический тангенс для отслеживания поверхности)

$$\frac{\partial f}{\partial t} + \nabla \cdot (uf) - f\nabla \cdot u = 0$$

где u - векторное поле скоростей

f - переносимая скалярная величина

$$f(x,t) = \begin{cases} 0, & x \in S(t) \\ 1, & x \notin S(t) \end{cases}$$

- В одномерном случае, для соленоидального поля скоростей:

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = 0$$

Дискретизация

- Отрезок [X1; X2], на котором рассматривается данное уравнение, разбивается на cellCount последовательных подотрезков, длиной ΔXi каждый. ΔXi =X(i+1/2)-X(i-1/2). Положения X(i+1/2), X(i-1/2) являются узлами данной сетки.
- Δt – длина шага по времени

- среднее значение функции f на i-ом отрезке ΔXi на n-ом временном шаге:

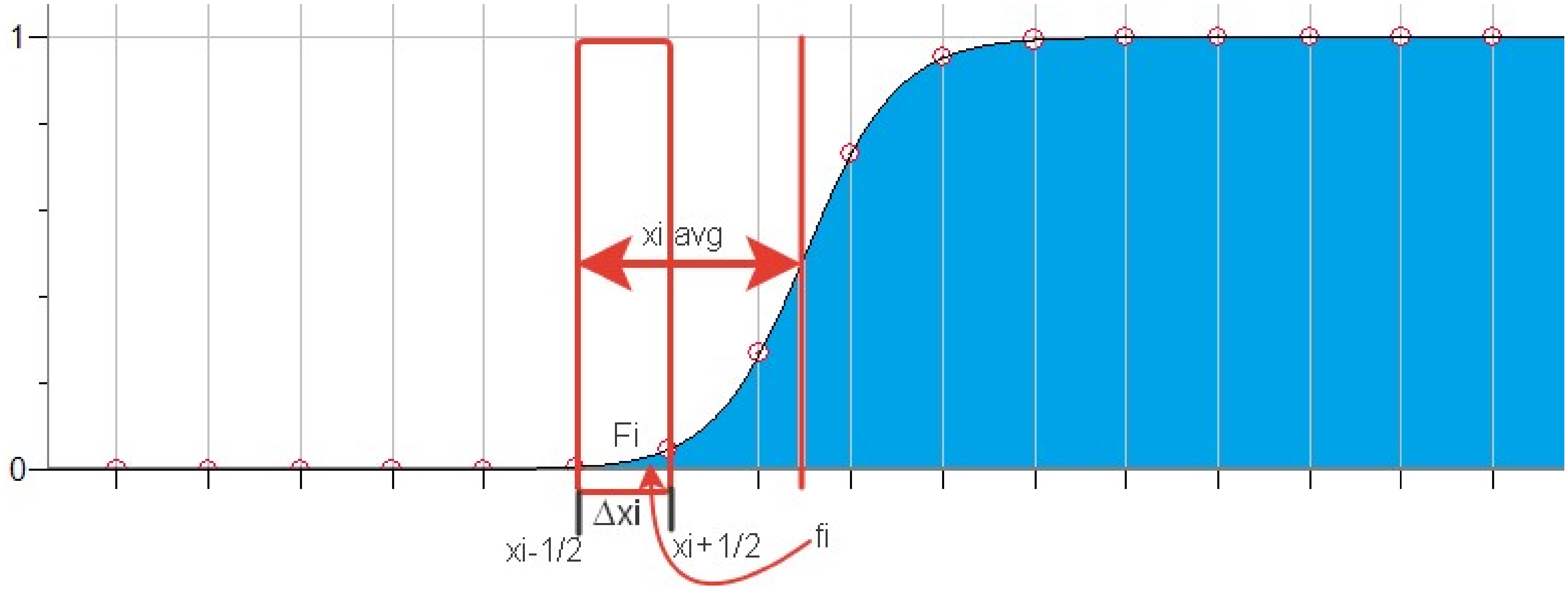
$$\bar{f}_i^n = \frac{1}{\Delta x_i} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} f(x, t^n) dx$$

- Аппроксимация f(i)n:

$$F_i(x) = \frac{1}{2}(1 + \gamma_i \tanh(\beta(\frac{x - x_{i-\frac{1}{2}}}{\Delta x_i} - \tilde{x}_i)))$$

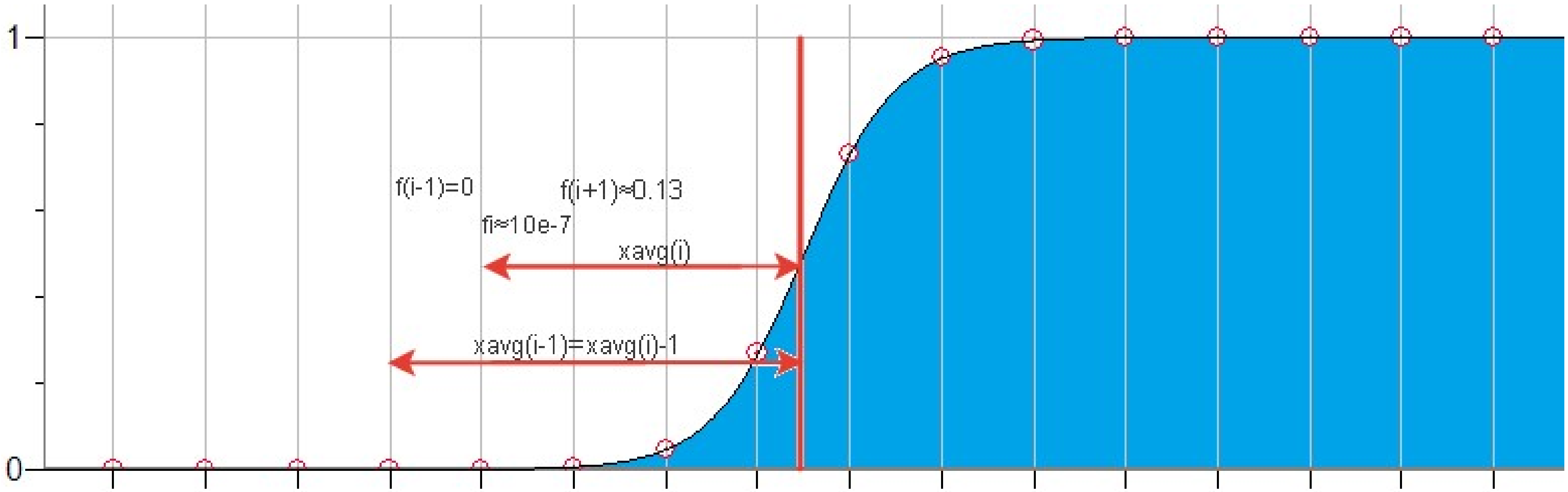
$$\gamma_i = sgn(\bar{f}_{i+1}^n - \bar{f}_{i-1}^n)$$

- Параметр $\tilde{x}_i$ – относительное расстояние до середины прыжка f от левой границы отрезка X(i-1/2).



$$\bar{f}_i^n = \frac{1}{\Delta x_i} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} F_i(x)dx$$

$$\tilde{x}_i = \frac{1}{2\beta} log\left(\frac{\exp\left(\frac{\beta}{\gamma_i}\left(1+\gamma_i-2\bar{f}_i^n\right)\right)}{1-\exp\left(\frac{\beta}{\gamma_i}\left(1-\gamma_i-2\bar{f}_i^n\right)\right)}\right)$$



Расчет $\tilde{x}_i$ был реализован в функции на C++:

```
double getXiavg(double beta, double gamma, double fi)
```

После вычисления значений $\bar{f}_i^n$ $\forall i = 1..cellCount$ на n-ом временном шаге, совершается переход к $n+1$ шагу: рассчитываются значения $\overline{fR}_{i-\frac{1}{2}}^{n+1}$ справа от левой границы $x_{i-\frac{1}{2}}$ отрезка Δx_i и $\overline{fL}_{i+\frac{1}{2}}^{n+1}$ слева от правой границы $x_{i+\frac{1}{2}}$ отрезка Δx_i по следующим формулам [4]:

$$\overline{fR}_i^{n+1} = min\bar{f}_i^n + \frac{1}{2}\overline{\Delta f_i}^n(1 + \gamma_i \tanh(\beta(\frac{u_{i-\frac{1}{2}}^- t}{\Delta x_i} - \tilde{x}_i)))$$

$$\overline{fL}_i^{n+1} = min\bar{f}_i^n + \frac{1}{2}\overline{\Delta f_i}^n(1 + \gamma_i \tanh(\beta(1 - \frac{u_{i+\frac{1}{2}}^+ t}{\Delta x_i} - \tilde{x}_i)))$$

Где

$$min\bar{f}_i^n = \min(\overline{fL}_{i-\frac{1}{2}}^n, \overline{fR}_{i+\frac{1}{2}}^n)$$

$$max\bar{f}_i^n = \max(\overline{fL}_{i-\frac{1}{2}}^n, \overline{fR}_{i+\frac{1}{2}}^n)$$

$$\overline{\Delta f_i}^n = max\bar{f}_i^n - min\bar{f}_i^n$$

$$u_{i-\frac{1}{2}}^- = \min(u_{i-\frac{1}{2}}, 0)$$

$$u_{i+\frac{1}{2}}^+ = \max(u_{i+\frac{1}{2}}, 0)$$

Пример расчета

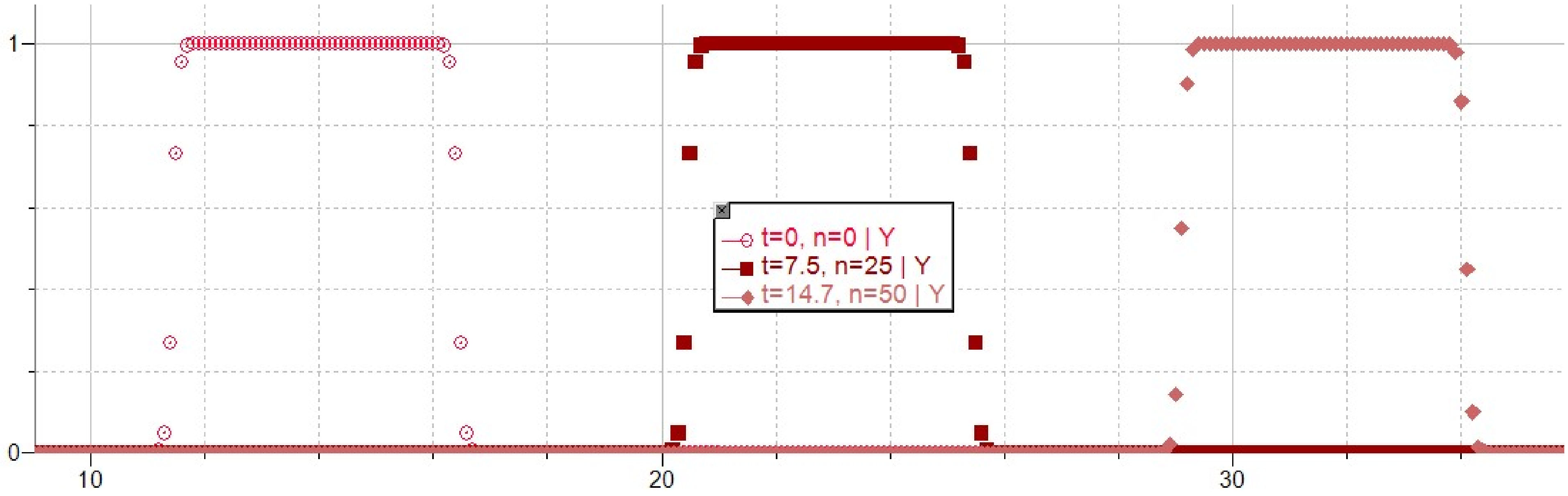
Начальные условия:

Функция f:

$$f(x,0)=\begin{cases}1, x\in[11.5;16.5]\\0, x\notin[11.5;16.5]\end{cases}$$

```
int cellCount = 400;
int stepN = 50;
double h = 0.1;
double timeStep = 0.3;
double u = 1.2; // векторное поле статическое
```

В результате расчета были получены значения $\bar{f}_i^n\ \forall i = 1..cellCount$ для моментов времени $t = j * timeStep, \forall j = 0..stepN$. Некоторые положения при движении:



Заключение

- Таким образом, были исследованы методы численного решения уравнения Эйлера и уравнения переноса. Был реализован программный алгоритм расчета положения твердого тела при движении в двух и трехмерном случае, а также программный алгоритм расчета движения поверхности жидкости в одномерном случае.
- Численное решение уравнения Эйлера позволяет с высокой точностью рассчитать положение любой точки твердого тела в заданный момент времени.
- Метод VOF с использование схемы THINC позволяет рассчитать изменение характеристической функции, описывающей твердое тело в жидкости в пространстве и времени. Полученная реализация одномерного алгоритма необходима для дальнейшей реализации многомерного расчета движения жидкости и твердого тела в пространстве с использованием результатов вычислений поля скоростей твердого тела.