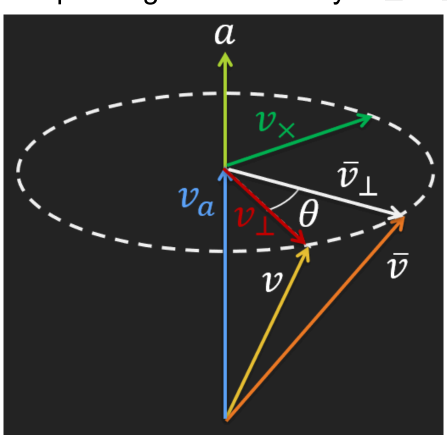
1. **Metaprogramming**. What is a metaprogram (in relation to compilation, templates, and recursion). Write a metaprogram computing A^N, where A, N are integers. Describe the following metaprogram and find a defect in its implementation

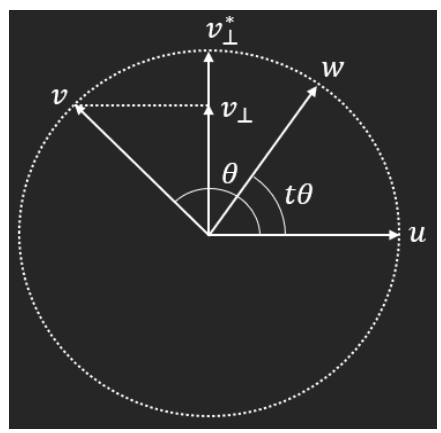
```
template<int N> struct F{
        static inline float result(float* u, float* v) {
                return (*u) * (*v) + F<N-1>::result(u, v);
        }
};

template<> struct F<1> {
        static inline float result(float* u, float* v) {
                return (*u) * (*v);
        }
};
```

2. **Rotations I**. Describe the relation between the rotation matrix and orthonormal coordinate basis vector. Euler angles (source and target frame, line of nodes, conventions, intrinsic vs. extrinsic rotations, gimbal lock). Tait-Bryan angles (relation to Euler angles, conventions).

3. **Rotations II**. Axis angle representation. Explain the Rodrigues' rotation formula (use the picture below left - rotating vector v about a by angle θ). Linear and spherical interpolation of axis angle (issues with LERP; for SLERP use the picture below right - interpolating from u to v by $0 \leq t \leq 1$).





4. **Rotations (Quaternions)**.Definition, Scalar-vector notation. Operations (addition, multiply by scalar, conjugation, length). Quaternions representing rotations (similarity to Rodrigues' rotation formula - relation of angles; inverse rotation, composition of rotations). Linear and spherical interpolation of quaternions (for slerp(q_0, q_1, t) - explain the use of "delta quaternion": $\Delta q q_0 = q_1$).

5. **Particle system**. Particle representation. Newton's equations of motion. Describe the initial value problem of 1st-order ODEs. Describe the interface of an ODE solver:

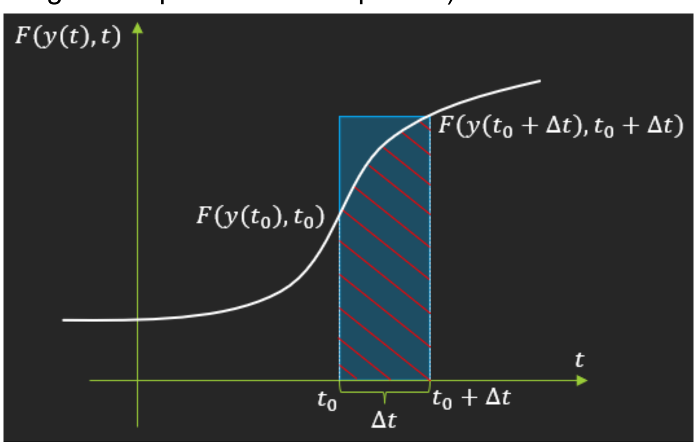
```
using F_y_t = std::function<float(std::vector<float> const&,float)>;
void ODE(
        std::vector<float> const& y0,
        std::vector<F_y_t> const& Fyt,
        float& t,
        float const dt,
        std::vector<float>& y
        );
```

   External force acting on a particle (gravity, viscous drag, simple friction). Collision of a particle with a plane.

6. **Differential equations I**. Describe the initial value problem of 1st-order ODEs. Numerical solution (implicit vs explicit method, Taylor theorem). Forward Euler method (accuracy, stability issue). Solve this example (what equations should be used?):

   Let's consider a particle $\mathscr{P}(t)=(\boldsymbol{x},\boldsymbol{v},\boldsymbol{F},m)$, where m=0.1kg, in a homogenous gravity field with $\boldsymbol{g}=(0,0,-10)^\top\ \mathrm{m\cdot s^{-2}}$. At time t=1 we have $\boldsymbol{x}=(1,-1,5)^\top$ m, $\boldsymbol{v}=(1,0,0)^\top\ \mathrm{m\cdot s^{-1}}$. Using forward Euler's method with Δt=0.5s compute $\mathscr{P}(2)$.

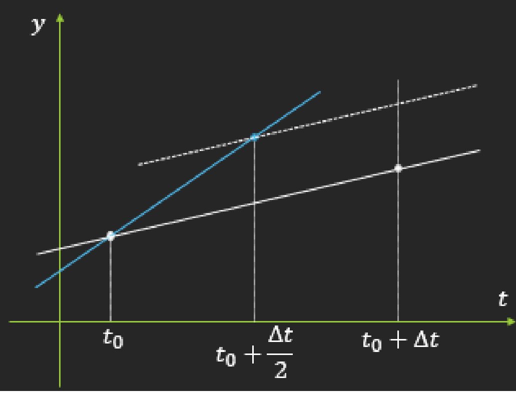
7. **Differential equations II**. Explain the backward Euler method (write down the fundamental theorem of the calculus, infer the formula of the method, explaining the integral computation of this picture):



   Solve this example: Let us consider IVP $\dot{y}=(3-4y)/2t, y(1)=-4$. Compute $y(2)$ by backward Euler's method.

8. **Differential equations III**. Explain the principal functionality of the midpoint method (use this picture):

$$y(t_0 + \Delta t) = y(t_0) + \Delta t F\left(y(t_0) + \frac{\Delta t}{2} F(y(t_0), t_0), t_0 + \frac{\Delta t}{2}\right)$$



Explain the principle of the implicit midpoint method (how to compute the integral?).
Runge-Kutta methods (explain the idea of the integral computation):

$$\int_{t_0}^{t_0+\Delta t} F(y(t), t)dt \approx \Delta t \sum_{i=1}^{n} b_i F(y(t_0 + c_i \Delta t), t_0 + c_i \Delta t)$$

Solve this example:

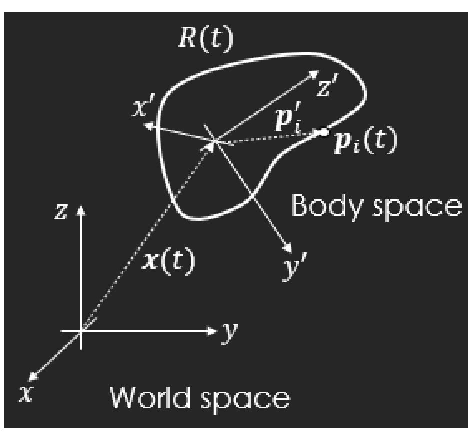
Runge-Kutta method of order 1 has the form:

$k_1 = F(y(t_0), t_0)$

$y(t_0 + \Delta t) = y(t_0) + \Delta t b_1 k_1$

What value to choose for b_1 (what polynomial to use for the comparison)?

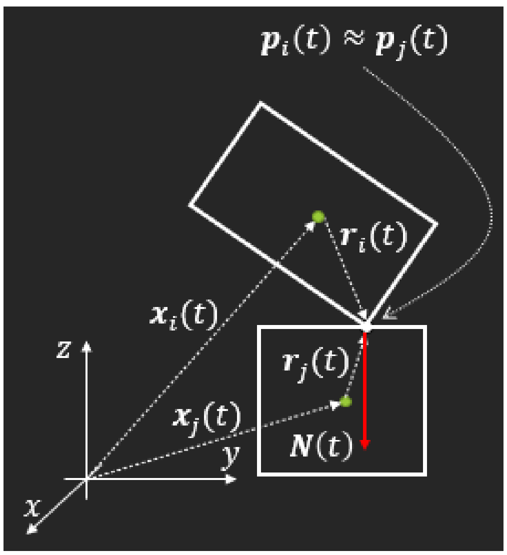
9. **Unconstrained rigid body motion I**. Concept of particles (explain the picture below and write equations for computing $p_i(t)$ from p_i' and vice versa)



Linear and angular velocity (relation between $\dot{x}(t)$ and $v(t)$ and $\dot{R}(t)$ and $\omega(t)$). The velocity of a particle ($v_i(t) = \dot{p}_i(t) = ...$). Mass center in the body and world space. Force and torque (acting on a particle; total force and torque; how to get torque given a force).

10. **Unconstrained rigid body motion II**. Linear and angular momentum (on a particle and total on body, relations: $P(t)$ and $v(t)$, $\dot{P}(t)$ and $F(t)$, $L(t)$ and $\omega(t)$, $\dot{L}(t)$ and $\tau(t)$), Inertial tensor (in world space and related important issue, solution of the issue and relation $I(t)$ and I' via $R(t)$). Newton-Euler equations of motion.

11. **Constrained rigid body motion I**. Collision constraint (describe the constraint using the picture below and formulate the constraint in terms of particle positions; how to express the constraint in terms of particle velocities?)

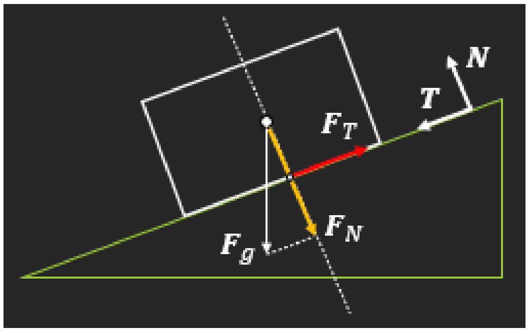


A collision constraint can be expressed in the following general form. Explain it:

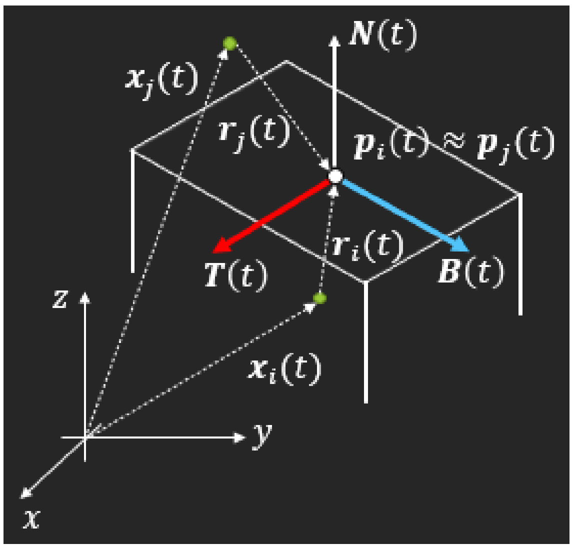
$$\begin{pmatrix} -\boldsymbol{N} \\ -(\boldsymbol{r}_i \times \boldsymbol{N}) \\ \boldsymbol{N} \\ \boldsymbol{r}_j \times \boldsymbol{N} \end{pmatrix}^{\top} \begin{pmatrix} \boldsymbol{v}_i \\ \boldsymbol{\omega}_i \\ \boldsymbol{v}_j \\ \boldsymbol{\omega}_j \end{pmatrix}$$

Collision constraint force and torque (write down the equation for the force acting on the bodies i and j; use the vector in the picture above to express the forces and torques).

12. **Constrained rigid body motion II**. Friction (explain the Coulomb law using this picture:



static vs dynamic friction). Write down friction constraints (in terms of velocities; use this picture for reference):

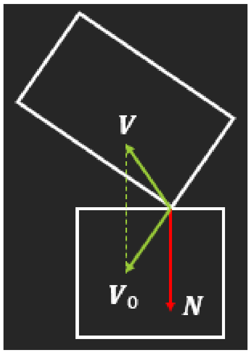


Explain the friction constraints:

$$
\begin{aligned}
J_T \boldsymbol{V} = 0 &\;\leftrightarrow\; -\mu\lambda_N < \lambda_T < \mu\lambda_N \\
J_T \boldsymbol{V} > 0 &\;\leftrightarrow\; \lambda_T = -\mu\lambda_N \\
J_T \boldsymbol{V} < 0 &\;\leftrightarrow\; \lambda_T = +\mu\lambda_N
\end{aligned}
$$

$$
\begin{aligned}
J_B \boldsymbol{V} = 0 &\;\leftrightarrow\; -\mu\lambda_N < \lambda_B < \mu\lambda_N \\
J_B \boldsymbol{V} > 0 &\;\leftrightarrow\; \lambda_B = -\mu\lambda_N \\
J_B \boldsymbol{V} < 0 &\;\leftrightarrow\; \lambda_B = +\mu\lambda_N
\end{aligned}
$$

13. **Constrained rigid body motion III**. Constraint bias (explain for “bouncing” collision $\dot{C}(p_i, p_j) = N \cdot V \geq -\beta(N \cdot V_0)$) using this picture:



Explain these constraints (their type and method for their solving):

$$\begin{array}{lll} J_N V \geq 0 & J_D V = 0 & J_T V = 0 \;\leftrightarrow\; -\mu\lambda_N < \lambda_T < \mu\lambda_N \\ \lambda_N \geq 0 & -\infty < \lambda_D < \infty & J_T V > 0 \;\leftrightarrow\; \lambda_T = -\mu\lambda_N \\ \lambda_N \cdot J_N V = 0 & & J_T V < 0 \;\leftrightarrow\; \lambda_T = \mu\lambda_N \end{array}$$

Explaining transformation of constraints to the unified form:

$$\begin{array}{l} J_D V = 0 \\ -\infty < \lambda_D < \infty \end{array} \;\Rightarrow\; \begin{array}{l} J_D V = 0 \;\leftrightarrow\; -\infty < \lambda_D < \infty \\ J_D V > 0 \;\leftrightarrow\; \lambda_D = -\infty \\ J_D V < 0 \;\leftrightarrow\; \lambda_D = \infty \end{array}$$

$$\begin{array}{l} J_N V \geq 0 \\ \lambda_N \geq 0 \\ \lambda_N \cdot J_N V = 0 \end{array} \;\Rightarrow\; \begin{array}{l} J_N V = 0 \;\leftrightarrow\; 0 < \lambda_N < \infty \\ J_N V > 0 \;\leftrightarrow\; \lambda_N = 0 \\ J_N V < 0 \;\leftrightarrow\; \lambda_N = \infty \end{array}$$

14. **Constrained rigid body motion IV**. Explain the constraint system (meaning of variables (matrices, vectors), what are the unknowns):

$$JM^{-1}J^{\top}\lambda = \frac{1}{\Delta t}\gamma - J\left(\frac{1}{\Delta t}V_0 + M^{-1}F^{ext}\right)$$

What kind of the algorithm is this:

$\boldsymbol{\lambda} = \boldsymbol{\lambda}^0$

for $iter = 1, \ldots, N$ **do** // N is max number of iterations

  for $i = 1, \ldots, s$ **do** // s is the number of equations in $\boldsymbol{JB\lambda} = \boldsymbol{\eta}$

    $c = 0$

    for $j = 1, \ldots, s$ **do**

      $c = c + (\boldsymbol{JB})_{i,j}\boldsymbol{\lambda}_j$

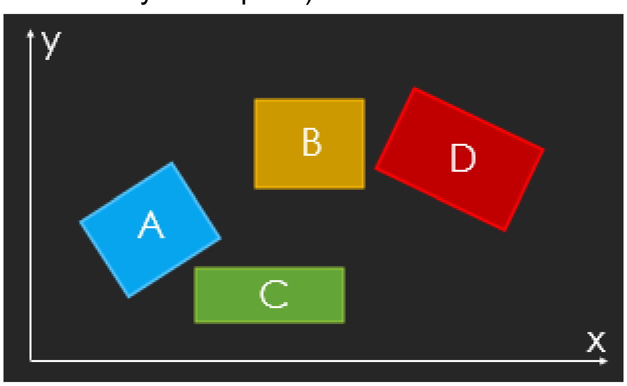
    $\Delta\lambda = (\boldsymbol{\eta}_i - c)/(\boldsymbol{JB})_{i,i}$

    $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_i + \Delta\lambda$

How can we extend this algorithm so that it can solve the constraint system:

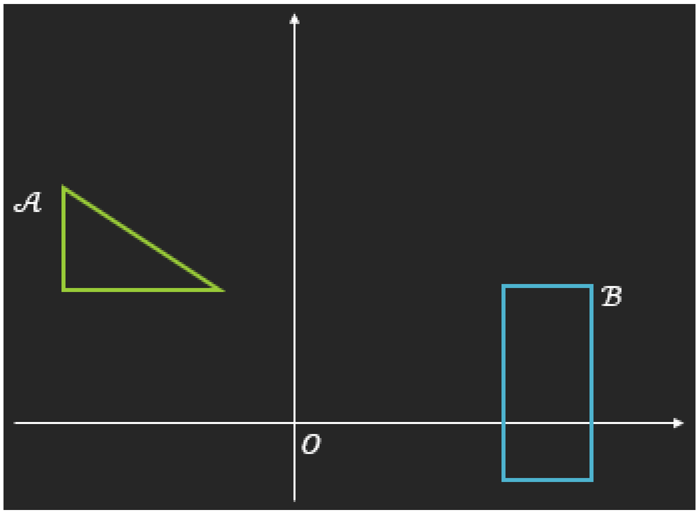
$$\boldsymbol{w} = \boldsymbol{JB\lambda} - \boldsymbol{\eta}$$
$$\boldsymbol{w}_i = 0 \quad \leftrightarrow \quad \lambda_i^-(\boldsymbol{\lambda}) < \boldsymbol{\lambda}_i < \lambda_i^+(\boldsymbol{\lambda})$$
$$\boldsymbol{w}_i > 0 \quad \leftrightarrow \quad \boldsymbol{\lambda}_i = \lambda_i^-(\boldsymbol{\lambda})$$
$$\boldsymbol{w}_i < 0 \quad \leftrightarrow \quad \boldsymbol{\lambda}_i = \lambda_i^+(\boldsymbol{\lambda})$$

15. **Collision detection I**. Broad and narrow phase (difference between phases, input, and output of the phases). Explain the Sweep-and-prune algorithm (use this picture for functionality description):

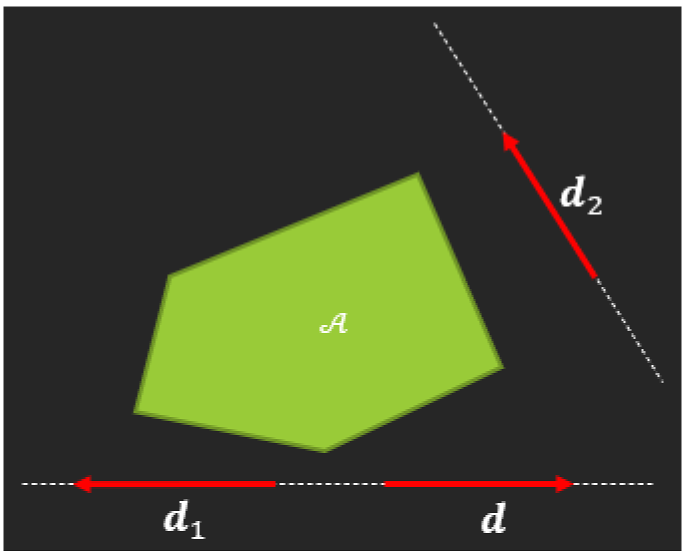


Describe the weakness of the algorithm. How can we deal with it?

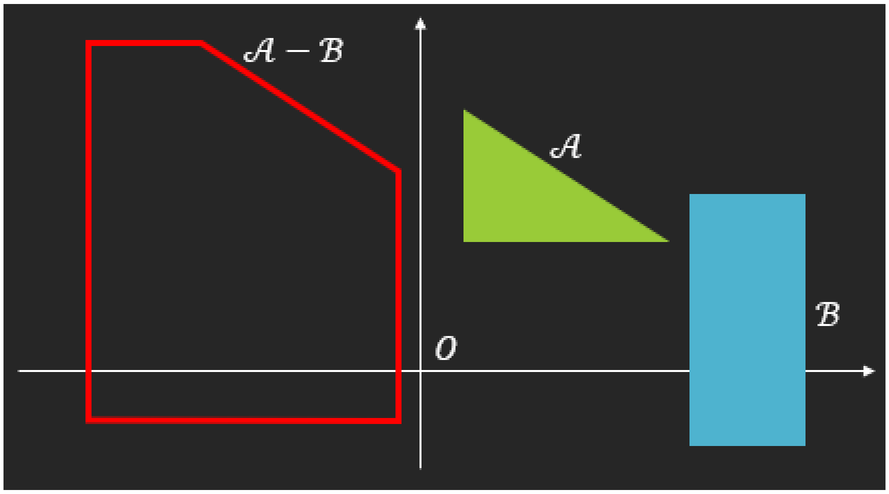
16. **Collision detection II**. Given two shapes $\mathscr{A}$ and $\mathscr{B}$ define the Minkowski sum $\mathscr{A}+\mathscr{B}$ and difference $\mathscr{A}-\mathscr{B}$. Compute $\mathscr{A}+\mathscr{B}$ for objects in this picture:



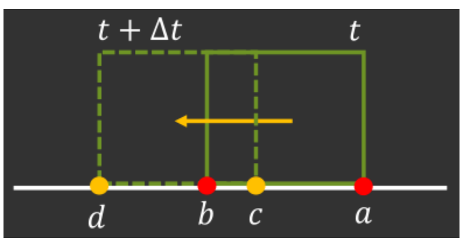
Define a support function for a shape $\mathcal{A}$ (what is the input and the output of the function; demonstrate the computation using following picture):

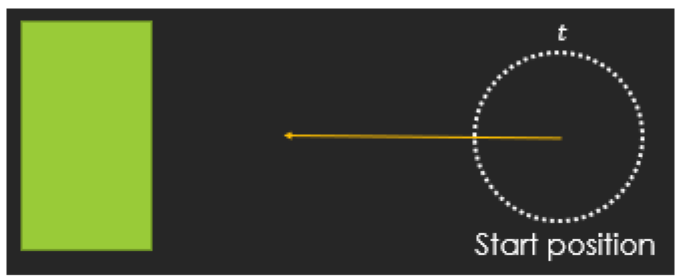


17. **Collision detection III**. Explain the GJK algorithm on the intuitive level using following example:



18. **Collision detection IV**. Explain terms "penetration" and "tunneling". Collision caching (based on distance in world/body space, based on geometrical properties (collision ID) - use the picture below-left; explain the use of the cache in the game loop). Computation of the collision time (explain and demonstrate using picture below-right):





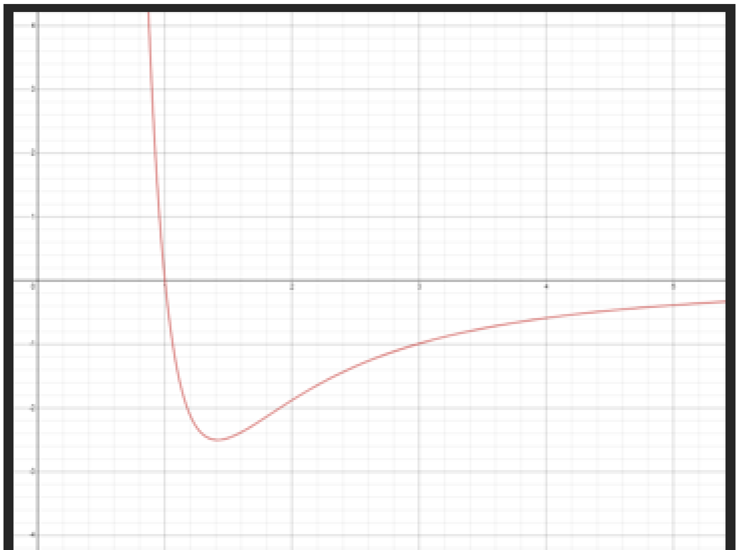
19. **Fluid simulation I**. Describe the principle of Euler's approach. Describe Navier-Stokes equations (variables, operators, and parts of the equations (their purpose)):

$$\frac{\partial \boldsymbol{u}}{\partial t} = -(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} + \boldsymbol{g}$$

$$\nabla \cdot \boldsymbol{u} = 0$$

20. **Fluid simulation II**. Discretization and numerical solution of Navier-Stokes equations (grid, boundary conditions, finite difference - write down how to compute $\nabla p_{i,j,k}$, the method of splitting (principle and usage)). Helmholtz-Hodge decomposition (explain how a vector field **w** is decomposed - into what fields and what are their properties).

21. **Fluid simulation III**. Lagrange approach to fluid simulation (particle properties, equations of motion, Lenard-Jones forces $F_{i,j}$ - explain their purpose and describe their properties on this plot):



$|\boldsymbol{F}_{i,j}|$ for $k_1 = k_2 = 10, m = 4, n = 2.$

22. **Fluid simulation IV**. Smoothed particle hydrodynamics (Lagrange vs. Euler approach, smoothing kernel W(x)). What parts of the Navier-Stokes equations can be ignored?

$$\frac{\partial \boldsymbol{u}}{\partial t} = -(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} + \boldsymbol{g}$$

$$\nabla \cdot \boldsymbol{u} = 0$$

How do we use Newton's equations of motion? What is the computed unknown, and how do we get the pressure?

23. **Fluid simulation V**. Height-field surface approximation (grid and function $h(x,y,t)$), explain the equation:

$$\frac{\partial^2 h}{\partial t^2} = v^2 \nabla^2 h$$

What substitution do we use to decompose the equation into two? Apply the forward Euler method to solve the system of the equations numerically; how to approximate $\nabla^2 h$).

24. **Low-level Engine Systems.** State low-level sub-systems of typical game engines, described selected one in more detail.

25. **Game Loops.** Define what the game loop is - from the developer's point of view, not the designer's PoV. State examples of typical loops in the game engine.