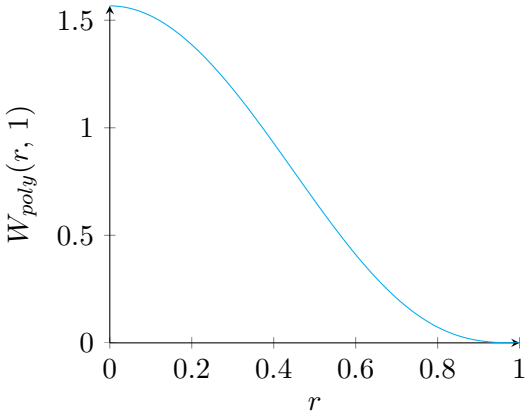
1 Theoretical model

1.1 Smoothing Kernel

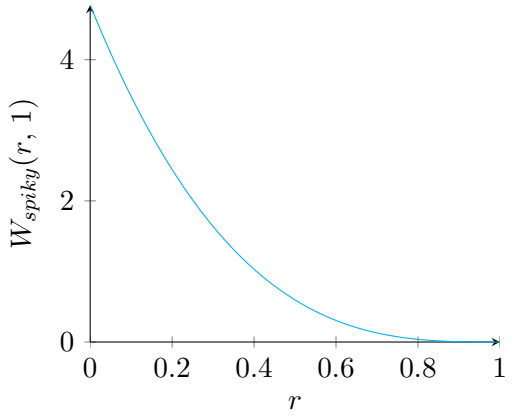
Each SPH particle has a circle of influence determined by the Smoothing Kernel $W(r, h)$ where r is the distance from the sample point to the particle center and h is the smoothing radius. The influence of a particle on a sample point increases as the sample point gets closer to the particle center. The choice of smoothing kernel is crucial as it is used by the **Interpolation Equation** to calculate scalar quantities, such as density. Müller *et al.*[**?**] describe two popular smoothing kernels, each with different properties.

$$W_{\text{poly}}(r, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3 & 0 \leq r \leq h \\ 0 & \text{otherwise,} \end{cases}$$



$W_{\text{poly}}(r, h)$ is a versatile kernel designed to simplify distance computations by squaring the distance term, eliminating the need for square root calculations when using the Pythagorean Theorem. However, Müller *et al.*[**?**] note that when $W_{\text{poly}}(r, h)$ is used for pressure computations, as required for enforcing incompressibility in this project, particles tend to cluster due to the gradient's effect on pressure calculation. The gradient of $W_{\text{poly}}(r, h)$ approaches zero for small distances, resulting in a diminishing repulsion force.

$$W_{\text{spiky}}(r, h) = \frac{15}{\pi h^6} \begin{cases} (h - r)^3 & 0 \leq r \leq h \\ 0 & \text{otherwise,} \end{cases}$$



$W_{\text{spiky}}(r, h)$ is used specifically for pressure computations. Its gradient is high when r is close to 0, generating the required repulsion forces for pressure calculations and therefore making it the superior choice for my simulation. However, the h^6 term is computationally challenging to calculate for every particle, therefore a more appropriate smoothing kernel for our uses would be a variation of $W_{\text{spiky}}(r, h)$ where $W(r, h) = (h - r)^3$ if $0 \leq r \leq h$, 0 otherwise.

For the simulation, the influence value returned will also be divided by the volume of our kernel, this keeps values consistent regardless of our values of h.

1.2 Interpolation Equation

The Interpolation Equation is responsible for calculating a scalar quantity A at a location r by a weighted sum of contributions from all the particles within our simulation. It sits at the heart of this project. This equation will be used to calculate density and viscosity which will indicate the pressure and therefore the net force a particle observes.

$$A_s(r) = \sum_j m_j \frac{A_j}{\rho_j} W(r - r_j, h)$$

A_s is the property we want to calculate,
m_j is the mass of the particle,
A_J is the value of that property of particle with index j,
ρ_j is the local density,
$W(r - r_j, h)$ is the value of the smoothing kernel with the distance between the 2 particles being $r - r_j$. [**?**]

1.3 SPH gradient optimisation

The simulation step updates the position vectors of the particles according to the rate of change of the properties every frame. We can calculate this rate of change by taking the derivative of our smoothing kernel with respect to r:

$$W(r,h) = (h-r)^3$$
$$\frac{\partial W}{\partial r} = -3(h-r)^2$$

1.4 Pressure Forces and Newton's Third Law

After finding the local density at the location for each particle, the local density needs to be compared to a constant target density to get a density error value which is multiplied by some constant to get our Pressure Force. By Newton's second law, $F = ma$, we can find the acceleration of the particle and apply this to the particles within the simulation step. Furthermore, Newton's Third Law of Motion must also be applied in this context, where if a particle exerts a pressure force, it must experience an equal and opposite reaction force.

1.5 Viscosity

1.6 Predicted Position optimisation

An interesting paper by B. Solenthaler and R. Pajarola [**?**] uses the current velocities of particles to determine a predicted position and uses predicted positions in the density and pressure calculations instead of current particle positions. This Predictive-Corrective Incompressible SPH (PCISPH) model allows for greater enforcement on incompressibility whilst having low computational cost per update with a large timestep, which is useful as other methods of enforcing incompressibility rely on smaller timesteps that are computationally heavy.