Newton's Second Law / Momentum Equation Derivation

$$m\mathbf{a} = m\mathbf{g} - V\nabla p + V\mu\nabla^2\mathbf{u}$$

divided by m

$$\mathbf{a} = \mathbf{g} - \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2\mathbf{u}$$

simplified kinematic viscosity: *μ/ρ to v*

$$\mathbf{a} = \mathbf{g} - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}$$

Acceleration as a time derivative of velocity to get the velocity equation

$$\mathbf{a} = \frac{D\mathbf{u}}{Dt} \equiv \frac{d}{dt}\left[\mathbf{u}(x(t), y(t), z(t), t)\right]$$

$$= \frac{\partial \mathbf{u}}{\partial x}\cdot\frac{dx}{dt} + \frac{\partial \mathbf{u}}{\partial y}\cdot\frac{dy}{dt} + \frac{\partial \mathbf{u}}{\partial z}\cdot\frac{dz}{dt} + \frac{\partial \mathbf{u}}{\partial t}\cdot\frac{dt}{dt}$$

$$= \left(\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt}\right)\cdot\left(\frac{\partial \mathbf{u}}{\partial x}, \frac{\partial \mathbf{u}}{\partial y}, \frac{\partial \mathbf{u}}{\partial z}\right) + \frac{\partial \mathbf{u}}{\partial t} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}$$

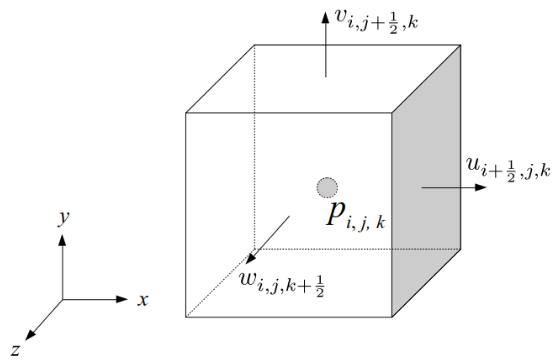
Final Form of Navier-Stokes Derivation

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

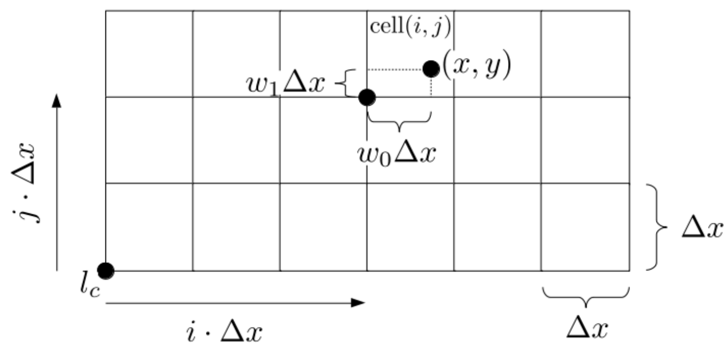
Viscosity assumed zero / Inviscus approximation (gives us the Euler Momentum Equation)

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = \mathbf{g} - \frac{1}{\rho}\nabla p$$

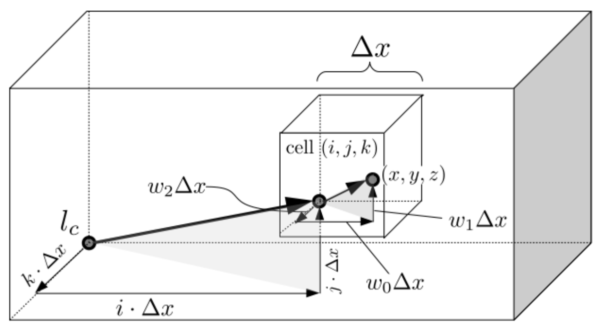
A cell of the grid and one of its stored velocities in each direction at the boundaries and pressure



Justifies the equation used to compute weights computed based on the particle's position in a cell

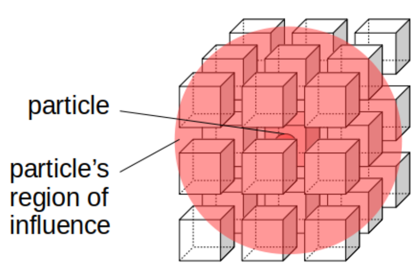


$$(x,y) = l_c + (i \cdot \Delta x, j \cdot \Delta x) + (w_0\Delta x, w_1\Delta x)$$

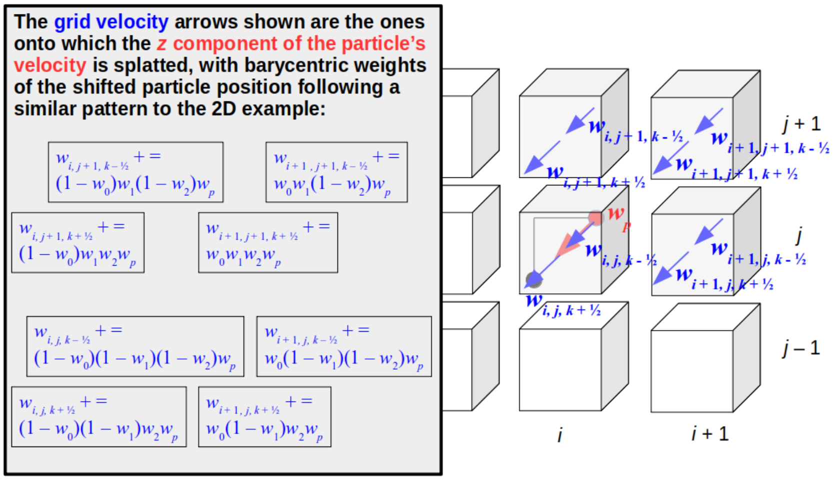


w(0,1,2) are the barycentric weights, lc is the lower corner of the scattered grid, (i,j,k) are the dimensions of the grid and cell respectively, and delta x is the length of a cell.

The cells for which velocities may need to be changed based on weights due to a particle when splatting



Example of Splatting w direction velocities



How splatting is applied to each entry of su sv sw (velgrid) and fu fv fw (fvel) respectively

velgrid(i, j, k) += (1 - w0) · (1 - w1) · (1 - w2) · velp
velgrid(i + 1, j, k) += w0 · (1 - w1) · (1 - w2) · velp
velgrid(i, j + 1, k) += (1 - w0) · w1 · (1 - w2) · velp
velgrid(i + 1, j + 1, k) += w0 · w1 · (1 - w2) · velp
velgrid(i, j, k + 1) += (1 - w0) · (1 - w1) · w2 · velp
velgrid(i + 1, j, k + 1) += w0 · (1 - w1) · w2 · velp
velgrid(i, j + 1, k + 1) += (1 - w0) · w1 · w2 · velp
velgrid(i + 1, j + 1, k + 1) += w0 · w1 · w2 · velp

fvel(i, j, k) += (1 - w0) · (1 - w1) · (1 - w2)
fvel(i + 1, j, k) += w0 · (1 - w1) · (1 - w2)
fvel(i, j + 1, k) += (1 - w0) · w1 · (1 - w2)
fvel(i + 1, j + 1, k) += w0 · w1 · (1 - w2)
fvel(i, j, k + 1) += (1 - w0) · (1 - w1) · w2
fvel(i + 1, j, k + 1) += w0 · (1 - w1) · w2
fvel(i, j + 1, k + 1) += (1 - w0) · w1 · w2
fvel(i + 1, j + 1, k + 1) += w0 · w1 · w2

How advection/trilinear interpolation velocity (u,v,w) is computed for advect and PIC/FLIP

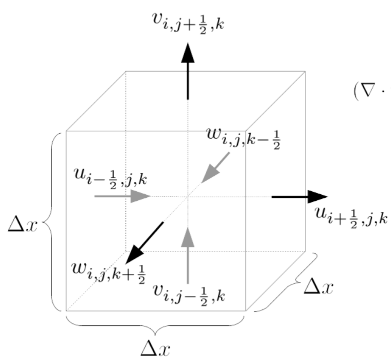
u =
(1 - w0) · (1 - w1) · (1 - w2) · ui, j, k
+ (1 - w0) · (1 - w1) · w2 · ui, j, k + 1
+ (1 - w0) · w1 · (1 - w2) · ui, j + 1, k
+ (1 - w0) · w1 · w2 · ui, j + 1, k + 1
+ w0 · (1 - w1) · (1 - w2) · ui + 1, j, k
+ w0 · (1 - w1) · w2 · ui + 1, j, k + 1
+ w0 · w1 · (1 - w2) · ui + 1, j + 1, k
+ w0 · w1 · w2 · ui + 1, j + 1, k + 1,

v =
(1 - w0) · (1 - w1) · (1 - w2) · vi, j, k
+ (1 - w0) · (1 - w1) · w2 · vi, j, k + 1
+ (1 - w0) · w1 · (1 - w2) · vi, j + 1, k
+ (1 - w0) · w1 · w2 · vi, j + 1, k + 1
+ w0 · (1 - w1) · (1 - w2) · vi + 1, j, k
+ w0 · (1 - w1) · w2 · vi + 1, j, k + 1
+ w0 · w1 · (1 - w2) · vi + 1, j + 1, k

+ w0 · w1 · w2 · vi + 1, j + 1, k + 1,

w =
(1 - w0) · (1 - w1) · (1 - w2) · wi, j, k
+ (1 - w0) · (1 - w1) · w2 · wi, j, k + 1
+ (1 - w0) · w1 · (1 - w2) · wi, j + 1, k
+ (1 - w0) · w1 · w2 · wi, j + 1, k + 1
+ w0 · (1 - w1) · (1 - w2) · wi + 1, j, k
+ w0 · (1 - w1) · w2 · wi + 1, j, k + 1
+ w0 · w1 · (1 - w2) · wi + 1, j + 1, k
+ w0 · w1 · w2 · wi + 1, j + 1, k + 1

Divergence derivation



Discrete Divergence of
Velocity for Cell (i, j, k):

$$(\nabla \cdot \mathbf{u})_{i,j,k} = \frac{u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{v_{i,j+\frac{1}{2},k} - v_{i,j-\frac{1}{2},k}}{\Delta x} + \frac{w_{i,j,k+\frac{1}{2}} - w_{i,j,k-\frac{1}{2}}}{\Delta x}$$

.

We want the discrete divergence of the velocity to be zero at the next time step:

$$0 = (\nabla \cdot \mathbf{u}^{\text{next}})_{i,j,k} = \frac{\overset{1}{u^{\text{next}}_{i+\frac{1}{2},j,k}} - \overset{2}{u^{\text{next}}_{i-\frac{1}{2},j,k}}}{\Delta x} + \frac{\overset{3}{v^{\text{next}}_{i,j+\frac{1}{2},k}} - \overset{4}{v^{\text{next}}_{i,j-\frac{1}{2},k}}}{\Delta x} + \frac{\overset{5}{w^{\text{next}}_{i,j,k+\frac{1}{2}}} - \overset{6}{w^{\text{next}}_{i,j,k-\frac{1}{2}}}}{\Delta x}$$

1 $u^{\text{next}}_{i+½,j,k} = u_{i+½,j,k} - (\Delta t / \rho)(p_{i+1,j,k} - p_{i,j,k}) / \Delta x$

2 $u^{\text{next}}_{i-½,j,k} = u_{i-½,j,k} - (\Delta t / \rho)(p_{i,j,k} - p_{i-1,j,k}) / \Delta x$

3 $v^{\text{next}}_{i,j+½,k} = v_{i,j+½,k} - (\Delta t / \rho)(p_{i,j+1,k} - p_{i,j,k}) / \Delta x$

4 $v^{\text{next}}_{i,j-½,k} = v_{i,j-½,k} - (\Delta t / \rho)(p_{i,j,k} - p_{i,j-1,k}) / \Delta x$

5 $w^{\text{next}}_{i,j,k+½} = w_{i,j,k+½} - (\Delta t / \rho)(p_{i,j,k+1} - p_{i,j,k}) / \Delta x$

6 $w^{\text{next}}_{i,j,k-½} = w_{i,j,k-½} - (\Delta t / \rho)(p_{i,j,k} - p_{i,j,k-1}) / \Delta x$

So, substituting for all “next” velocities in the zero-divergence equation,

$$0 = (\nabla \cdot \mathbf{u}^{\text{next}})_{i,j,k} = \frac{u^{\text{next}}_{i+\frac{1}{2},j,k} - u^{\text{next}}_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{v^{\text{next}}_{i,j+\frac{1}{2},k} - v^{\text{next}}_{i,j-\frac{1}{2},k}}{\Delta x} + \frac{w^{\text{next}}_{i,j,k+\frac{1}{2}} - w^{\text{next}}_{i,j,k-\frac{1}{2}}}{\Delta x}$$

$$= \frac{1}{\Delta x}\left[u^{\text{next}}_{i+\frac{1}{2},j,k} - u^{\text{next}}_{i-\frac{1}{2},j,k} + v^{\text{next}}_{i,j+\frac{1}{2},k} - v^{\text{next}}_{i,j-\frac{1}{2},k} + w^{\text{next}}_{i,j,k+\frac{1}{2}} - w^{\text{next}}_{i,j,k-\frac{1}{2}}\right]$$

$$\begin{aligned}
= \frac{1}{\Delta x}\Bigg[&\left(u_{i+\frac{1}{2},j,k} - \frac{\Delta t}{\rho}\frac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x}\right) - \left(u_{i-\frac{1}{2},j,k} - \frac{\Delta t}{\rho}\frac{p_{i,j,k} - p_{i-1,j,k}}{\Delta x}\right) \\
&+ \left(v_{i,j+\frac{1}{2},k} - \frac{\Delta t}{\rho}\frac{p_{i,j+1,k} - p_{i,j,k}}{\Delta x}\right) - \left(v_{i,j-\frac{1}{2},k} - \frac{\Delta t}{\rho}\frac{p_{i,j,k} - p_{i,j-1,k}}{\Delta x}\right) \\
&+ \left(w_{i,j,k+\frac{1}{2}} - \frac{\Delta t}{\rho}\frac{p_{i,j,k+1} - p_{i,j,k}}{\Delta x}\right) - \left(w_{i,j,k-\frac{1}{2}} - \frac{\Delta t}{\rho}\frac{p_{i,j,k} - p_{i,j,k-1}}{\Delta x}\right)\Bigg]
\end{aligned}$$

Divergence
of velocity

$$(\nabla \cdot \mathbf{u})_{i,j,k} \xrightarrow{\text{discretization}}$$

$$= \frac{u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{v_{i,j+\frac{1}{2},k} - v_{i,j-\frac{1}{2},k}}{\Delta x} + \frac{w_{i,j,k+\frac{1}{2}} - w_{i,j,k-\frac{1}{2}}}{\Delta x}$$

$$+\frac{\Delta t}{\rho}\left(\frac{6p_{i,j,k} - p_{i+1,j,k} - p_{i,j+1,k} - p_{i,j,k+1} - p_{i-1,j,k} - p_{i,j-1,k} - p_{i,j,k-1}}{(\Delta x)^2}\right)$$

$$-(\nabla^2 p)_{i,j,k} \xrightarrow{\text{discretization}}$$

Laplacian
of pressure

$$-\frac{\Delta t}{\rho}\left(\frac{-6p_{i,j,k} + p_{i+1,j,k} + p_{i,j+1,k} + p_{i,j,k+1} + p_{i-1,j,k} + p_{i,j-1,k} + p_{i,j,k-1}}{(\Delta x)^2}\right)$$

$$= -\left(\frac{u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{v_{i,j+\frac{1}{2},k} - v_{i,j-\frac{1}{2},k}}{\Delta x} + \frac{w_{i,j,k+\frac{1}{2}} - w_{i,j,k-\frac{1}{2}}}{\Delta x}\right)$$

central difference approximation of:

$$-\frac{\Delta t}{\rho}(\nabla^2 p)_{i,j,k} = -(\nabla \cdot \mathbf{u})_{i,j,k}$$

$$6p_{i,j,k} - p_{i+1,j,k} - p_{i,j+1,k} - p_{i,j,k+1} - p_{i-1,j,k} - p_{i,j-1,k} - p_{i,j,k-1}$$

$$= \frac{\rho(\Delta x)^2}{\Delta t} \cdot -\left(\frac{u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{v_{i,j+\frac{1}{2},k} - v_{i,j-\frac{1}{2},k}}{\Delta x} + \frac{w_{i,j,k+\frac{1}{2}} - w_{i,j,k-\frac{1}{2}}}{\Delta x}\right)$$

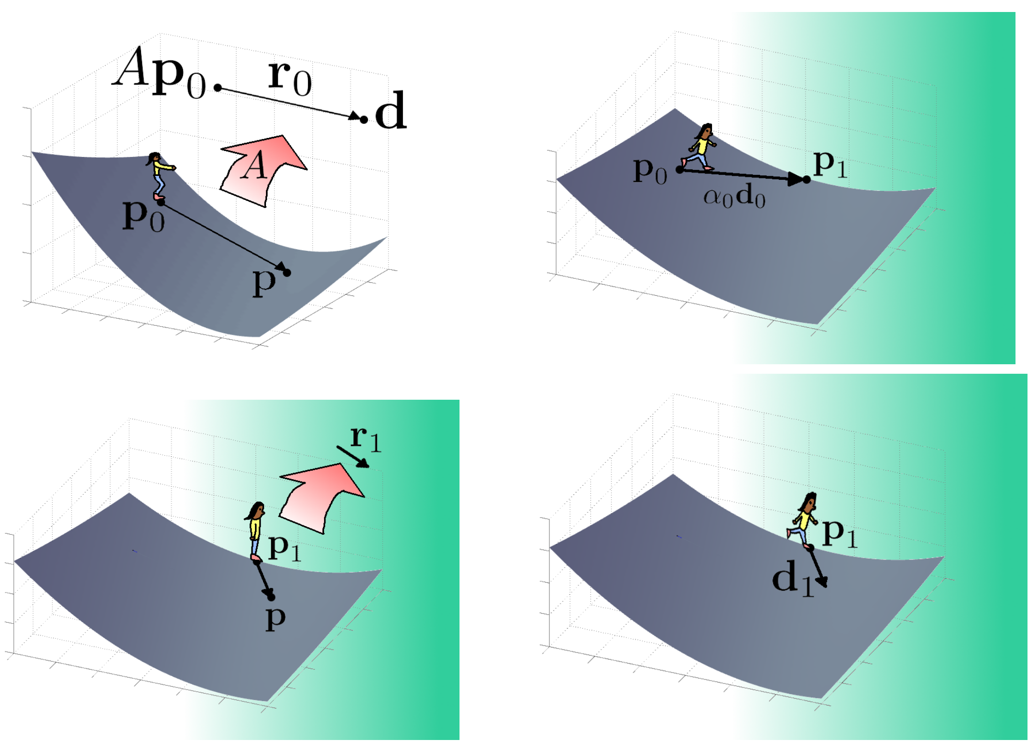
Ap = d simplification

$$A\mathbf{p} = \mathbf{d}$$

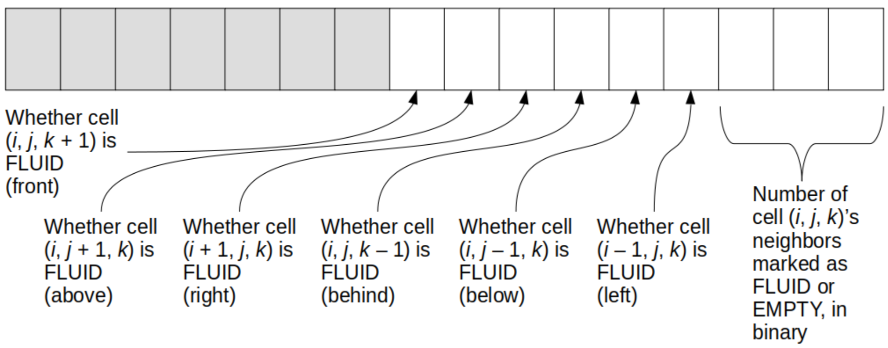
$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdots & 0 & 0 & -1 & -1 & -1 & 6 & -1 & -1 & -1 & 0 & 0 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \vdots \\ p_{i,j,k-1} \\ p_{i,j-1,k} \\ p_{i-1,j,k} \\ p_{i,j,k} \\ p_{i+1,j,k} \\ p_{i,j+1,k} \\ p_{i,j,k+1} \\ \vdots \end{bmatrix} = \frac{\rho(\Delta x)^2}{\Delta t} \begin{bmatrix} \vdots \\ -(\nabla \cdot \mathbf{u})_{i,j,k-1} \\ -(\nabla \cdot \mathbf{u})_{i,j-1,k} \\ -(\nabla \cdot \mathbf{u})_{i-1,j,k} \\ -(\nabla \cdot \mathbf{u})_{i,j,k} \\ -(\nabla \cdot \mathbf{u})_{i+1,j,k} \\ -(\nabla \cdot \mathbf{u})_{i,j+1,k} \\ -(\nabla \cdot \mathbf{u})_{i,j,k+1} \\ \vdots \end{bmatrix}$$

Conjugate Gradient Algorithm

Start with a guess P0 that can be = 0
Determine how far from the bottom of the "bowl" and solution p we are with residual vector, r0 = d - Ap0
A step size can be found from it α0 = r0 · r0 / (d0 · Ad0)
And then a new guess p1 = p0 + α0d0
Then again r1 = r0 - α0Ad0
β1 = (r1 · r1) / (r0 · r0)
d1 = r1 + β1d0
and so on
Set some arbitrary number of steps to determine when done or stop if threshold 10e-6*r is reached



How bits are set in each entry of neighbors used to compute pressure neighbors(i, j, k) is a 16-bit unsigned integer:



Implementation of pressure solver

Start with $p = 0$.
Start with $r = d$, i.e., the starting residual should equal the starting step direction.
Compute the squared magnitude of the residual: $\sigma = r \cdot r$.
Set the threshold for how close we want to get to the ideal solution, tolerance = 10-6 · σ.
Repeat until 1000 steps are taken or $\sigma \leq$ tolerance:
Set the matrix-mapped step direction to be $q = Ad$.
Set the step size, $\alpha = \sigma / d \cdot q$.
Move to the new solution guess, $p \mathrel{+}= \alpha d$.
Compute the new residual, $r \mathrel{-}= \alpha q$.
Save a copy of the old residuals magnitude, $\sigma old = \sigma$.
Compute the squared magnitude of the new residual, $\sigma = r \cdot r$.
Compute how small the new residual is relative to the old one, $\beta = \sigma / \sigma old$.
Set the new step direction based on the old step direction: $d = r + \beta d$.

Source:

https://unusualinsights.github.io/fluid_tutorial/