A simple Fluid Solver based on FFT

Sven Klein

DATUM EINTRAGEN NICHT VERGESSEN!!!!!!!!

Ziele

1. Verhalten von Gasen und Flüssigkeiten simulieren
2. schnell
3. stabil
4. interaktiv
5. visuelle Korrektheit wichtiger als physikalische Korrektheit
6. einfach zu implementieren

Naiver-Stokes-Gleichungen

Voraussetzungen

- Dichte und Temperatur des Fluids sind überall fast konstant
- Geschwindigkeit und Druck ist zu einem Zeitpunkt $t = 0$ bekannt

Naiver-Stokes-Gleichungen

$$\nabla \cdot u = 0 \tag{1}$$

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u - \frac{1}{\rho}\nabla p + \upsilon \nabla^2 u + f \tag{2}$$

u : Geschwindigkeitsfeld

p : Druckfeld

ρ : Dichte

f : Kraft

t : Zeit

υ : Viskosität

Die Helmholtz Zerlegung

Ein differenzierbares Vektorfeld w lässt sich stets eindeutig in folgende Form zerlegen

$$w = u + \nabla q$$

wobei u ein Vektorfeld mit $\nabla \cdot u = 0$ und q ein Skalarfeld ist.

Die Helmholtzprojektion

Sei P der Operator, der jedes Vekorfeld w auf seinen divergenzfreien Teil u abbildet. Dann gilt:

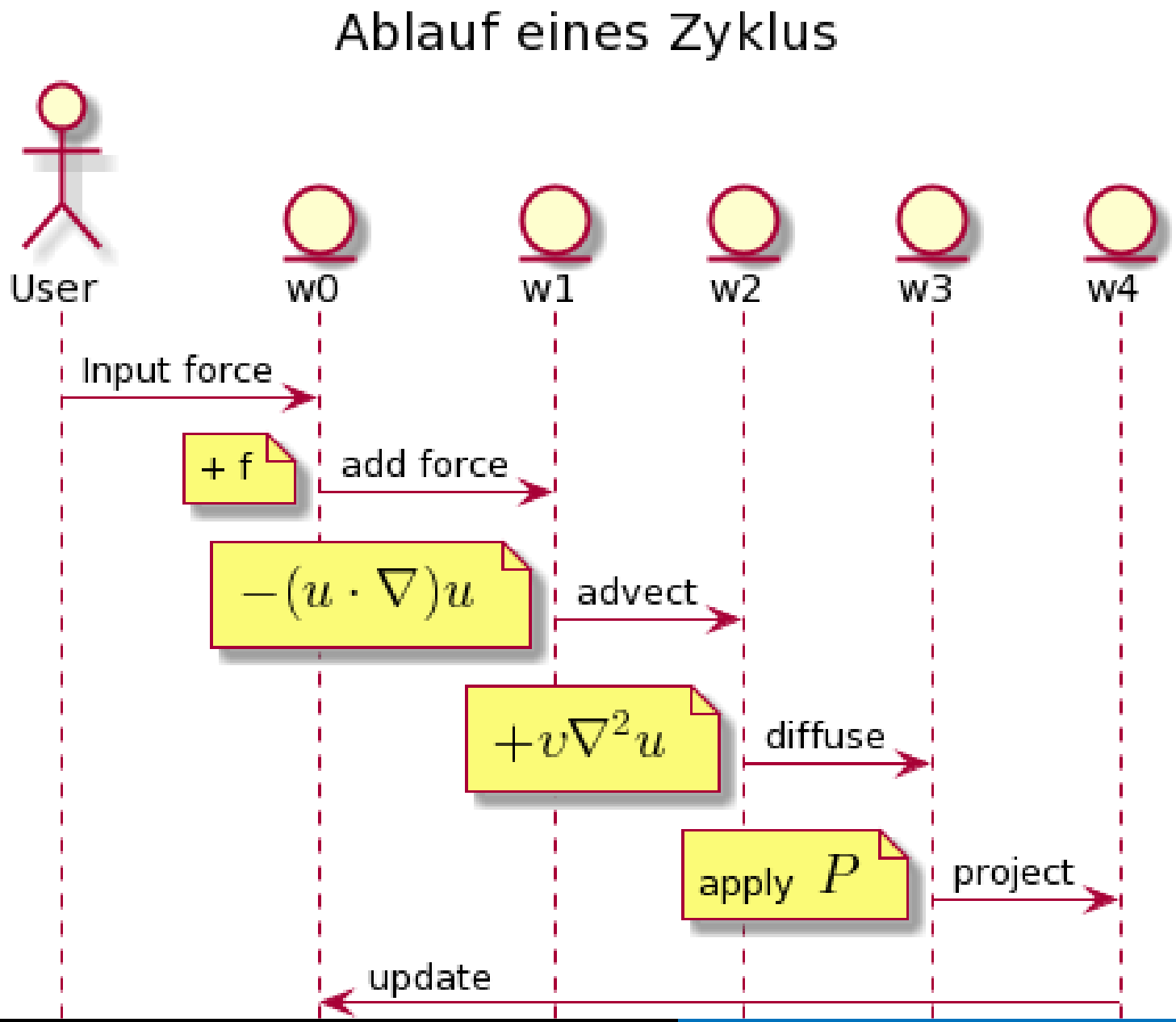
1. P ist linear
2. Für u aus den Navier Stokes Gleichungen gilt $P(u) = u$
3. Für p aus den Navier Stokes Gleichungen gilt $P(\nabla p) = 0$

Durch Anwenden von P auf beide Seiten von (2) erhält man somit

$$\frac{\partial u}{\partial t} = P(-(u \cdot \nabla)u + v\nabla^2 u + f)$$

Ablauf

Das Vektorfeld $u(x,t)$ sei zu einem Zeitpunkt t bekannt. Wir wollen nun $u(x,t+\Delta t)$ berechnen. Wir setzen dazu $w_0(x) = u(x,t)$ und approximieren $u(x,t+\Delta t)$ in 4 Schritten.


Ablauf eines Zyklus
User
w0
w1
w2
w3
w4
Input force
+ f
add force
$-(u \cdot \nabla)u$
advect
$+v\nabla^2 u$
diffuse
apply P
project
update

Schritt 1: Kräfte hinzufügen

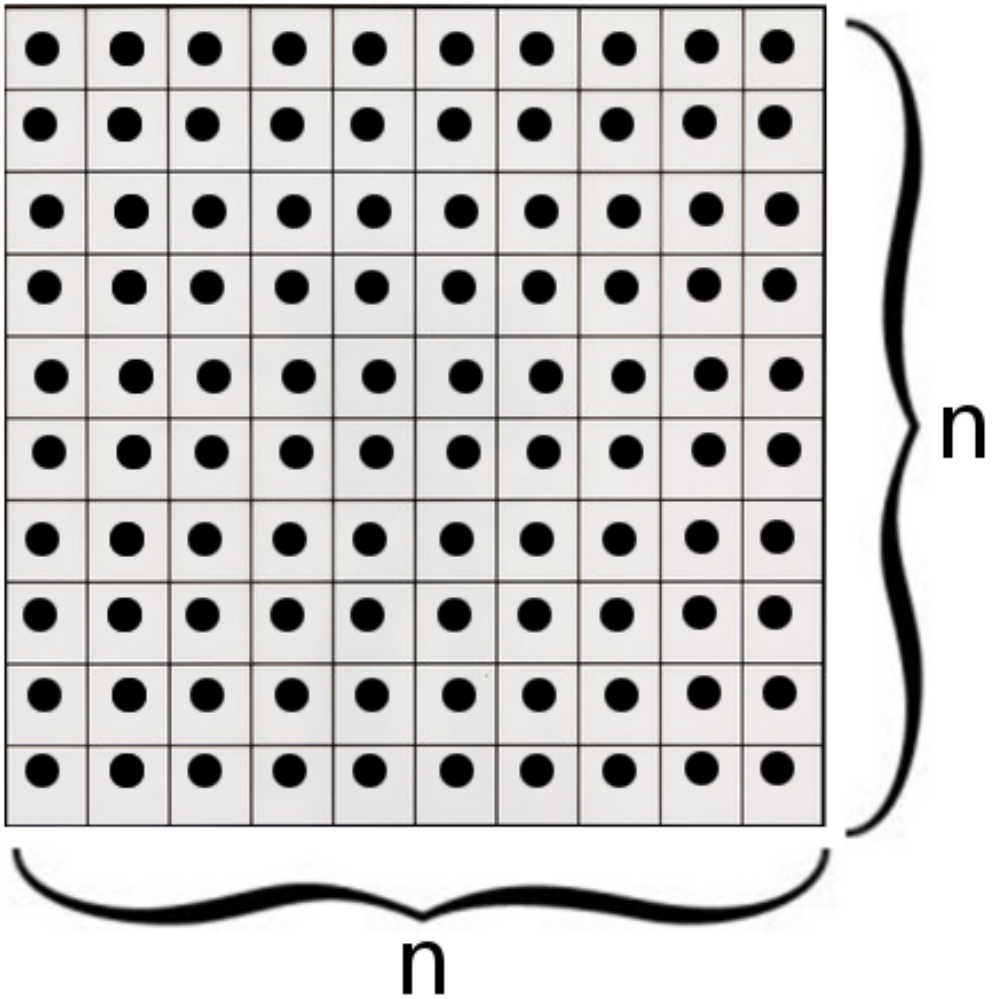
Wir haben also folgende Differentialgleichung:

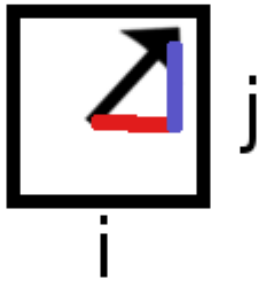
$$\frac{\partial w_0}{\partial t} = f$$

Wenn wir annehmen, dass f sich über den Zeitraum Δt nicht stark verändert, so erhalten wir eine Lösung durch:

$$w_1 = w_0 + \Delta t \cdot f$$

Diskretisierung von u





u[i,j]
v[i,j]

Schritt 1: Code

Variablen

- u: 2d array der Größe n$\times$n welches die x Komponente des Geschwindigkeitsfeldes disrketisiert.
- v: 2d array der Größe n$\times$n welches die y Komponente des Geschwindigkeitsfeldes disrketisiert.
- uF: Kraft in x Richtung
- vF: Kraft in y Richtung
- visc: Viscosität
- dt: Länge des zu simulierenden Zeitabschnitts

Code

```
u+=dt*uF
v+=dt*vF
```

Schritt 2: Advektion

Definition

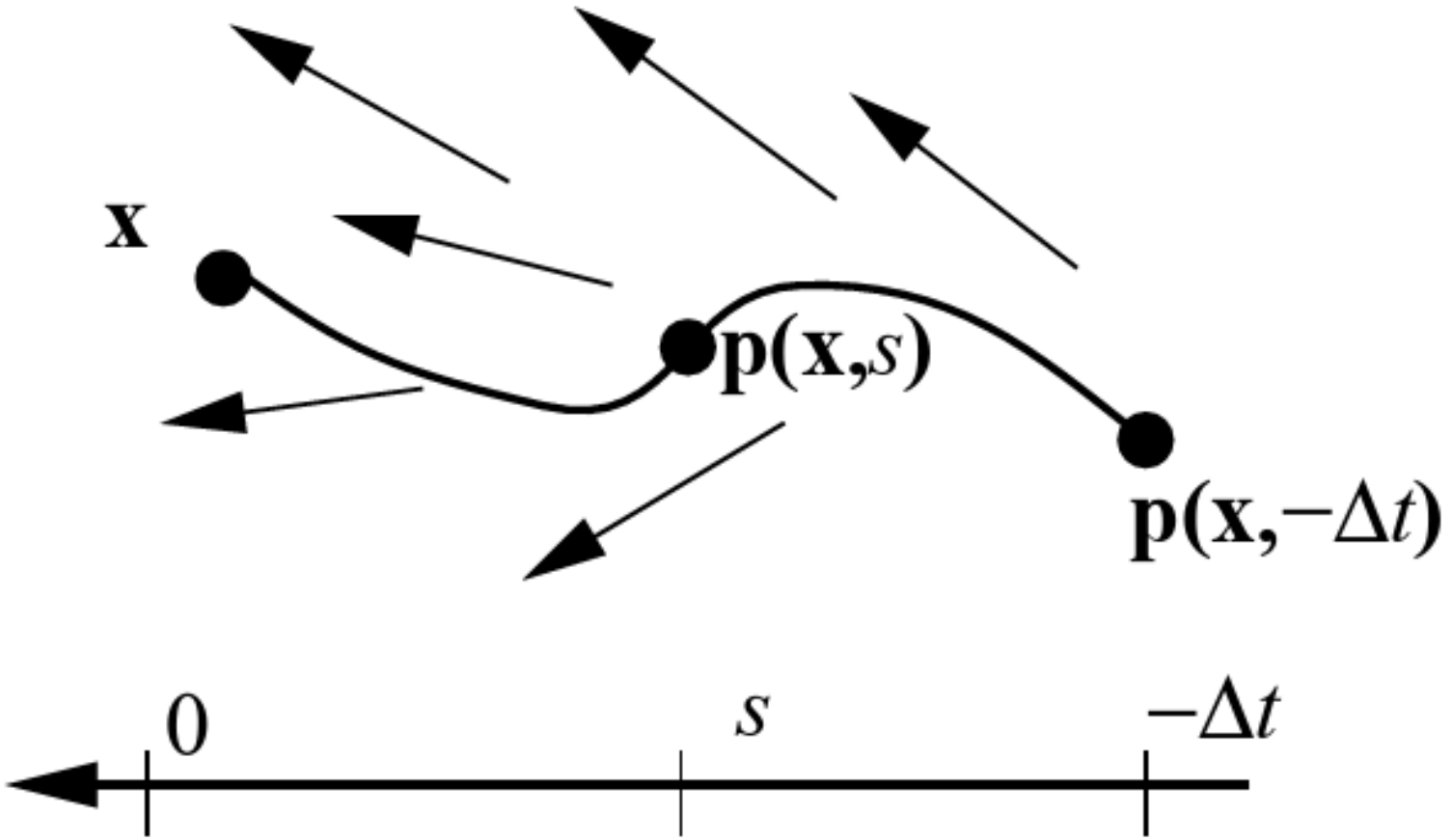
Ein Fluid hat die Eigenschaft, dass es Substanzen von einem Ort zum anderen transportieren kann. Da ein Fluid aus Teilchen besteht werden diese Teilchen ebenfalls transportiert. Dies führt dazu, dass die Geschwindigkeit des Fluids ebenfalls transportiert wird.

Intuitiver Ansatz

In jedem Zeitintervall Δt werden die Teilchen durch die Geschwindigkeit des Fluids bewegt. Um die Geschwindigkeit am Punkt x zum Zeitpunkt $t + \Delta t$ zu ermittlern müssen wir also nur den Weg $p(x, s)$ der Teilchen durch w_1 über den Zeitraum Δt zurückverfolgen. Wir erhalten somit:

$$w_2 = w_1(p(x, -\Delta t))$$

Schritt 2: Advektion



Methode der Charakteristiken

Sei $a(x,t)$ eine beliebige Komponente von $u(x,t)$. Dann haben wir also folgende Differentialgleichung:

$$\frac{\partial a(x,t)}{\partial t} = -v(x) \cdot \nabla a(x,t) \text{ und } a(x,0) = a_0(x)$$

Für p gilt weiterhin:

$$\frac{d}{dt} p(x_0) = v(p(x_0,t)) \text{ und } p(x_0,0) = x_0$$

Wir definieren nun $\bar{a}(x_0,t) = a(p(x_0,t),t)$. Wir berechnen nun die Ableitung mit Hilfe der Kettenregel:

$$\frac{d\bar{a}}{dt} = \frac{\partial a}{\partial t} + v \cdot \nabla a = 0$$

Daraus folgt nun also:

$$a(p(x_0,t),t) = \bar{a}(x0,t) = \bar{a}(x0,0) = a_0(x_0)$$

Schritt 2: Code

```
u0 = u.copy()
v0 = v.copy()
for x in range(n):
  for y in range(n):
    x0 = x-n*dt*u0[x,y]; y0 = y-n*dt*v0[x,y]
    i0 = int(np.floor(x0)); j0 = int(np.floor(y0))
    s = x0-i0; t = y0-j0
    i0 %= n; j0 %= n
    i1 = (i0+1)%n; j1 = (j0+1)%n
    u[x,y] = (1-s)*((1-t)*u0[i0,j0]+t*u0[i0,j1])+\
        s*((1-t)*u0[i1,j0]+t*u0[i1,j1])
    v[x,y] = (1-s)*((1-t)*v0[i0,j0]+t*v0[i0,j1])+\
        s*((1-t)*v0[i1,j0]+t*v0[i1,j1])
```

Die Fourier Transformation

Wir betrachten zunächst den eindimensionalen Fall, den wir in Numerik II bereits kennengelernt haben. Sei dazu $f \in L^1(\mathbb{R})$

Fourier Transformation
$$\hat{f}(k) = \int_{-\infty}^{+\infty} e^{-ikx} f(x)\,dx$$

Inverse
$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{f}(k) e^{+ikt}\,dk$$

Diskrete Fourier Transformation
$$\hat{f}([x_0, \ldots, x_{n-1}]) = [\hat{y}_0, \ldots, \hat{y}_{n-1}]$$
$$\hat{y}_k = \sum_{j=0}^{n-1} e^{-\frac{2\pi i}{n} \cdot jk} x_j$$

Inverse
$$x_k = \frac{1}{n} \sum_{j=0}^{n-1} e^{\frac{2\pi i}{n} jk} \hat{y}_j$$

Hansen, *Fourier transforms : principles and applications*

Die Fourier Transformation

Die Fourier Transformation lässt sich auf beliebig viele Dimensionen verallgemeinern, indem man die eindimensionale Fourier Transformation nacheinander auf jede Dimension anwendet. So ergibt sich zum Beispiel im zweidimensionalen für $f \in L^2(\mathbb{R})$:

Fourier Transformtation
$$\hat{f}(k) = \int_{\mathbb{R}^2} e^{-i<k,x>} f(x)\,dx$$

DFT
$$\hat{f}([x_{0,0}, , \ldots, x_{0,m-1}, x_{1,0}, \ldots, x_{n-1,m-1}]) = [\hat{y}_{0,0}, , \ldots, \hat{y}_{0,m-1}, \hat{y}_{1,0}, \ldots, \hat{y}_{n-1,m-1}])$$

$$\hat{y}_{s,t} = \sum_{k=0}^{n-1} \sum_{l=0}^{m-1} e^{-\frac{2\pi i}{n} \cdot ks} e^{-\frac{2\pi i}{n} \cdot lt} x_{k,l}$$

Hansen, *Fourier transforms : principles and applications*

Die Ableitung im Fourier Raum

Satz

Sei $f \in L^1(\mathbb{R})$. Dann gilt:

$$(\hat{\nabla} f)(k) = ik\hat{f}(k)$$

Beweis

Wir zeigen dies nur für den eindimensionalen Fall.

$$(\hat{\nabla} f)(k) = \int_{-\infty}^{+\infty} f'(x) e^{-ikx} dx$$

$$=^{\text{Partielle Integration}} \left[f(t) e^{-ikt} \right]_{-\infty}^{+\infty} - (-ik) \int_{-\infty}^{+\infty} f(x) e^{-ikx} dx$$

$$= ik\hat{f}(k)$$

Die letzte Gleichheit gilt, da wegen $f \in L^1(\mathbb{R})$ gilt $f(t) \to 0$ für $t \to \pm\infty$.

Hansen, *Fourier transforms : principles and applications*

FFT und die Wellenzahl k

Wir werden zur Berechnung der DFT den FFT Algorithmus nutzen den wir ebenfalls bereits kennengelernt haben. Um die Wellenzahl *k* zu jedem Eintrag zu erhalten können wir die Funktion np.fft.fftfreq nutzen.

```
u = fft2(u)
v = fft2(v)
freq = np.fft.fftfreq(n,1/n)
kx = np.zeros([n,n])
ky = np.zeros([n,n])
for i in range(n):
        for j in range(n):
                kx[i,j] = 2*np.pi*freq[i]
                ky[i,j] = 2*np.pi*freq[j]
```

WICHTIG!

Damit wir die Funktion in den Fourier Raum transformieren können müssen wir periodische Randbedingungen annehmen.

Schritt 3: Diffusion

Wir haben also die Gleichung:

$$\frac{\partial w_2}{\partial t} = \upsilon \nabla^2 w_2$$

Diese lösen wir mit Hilfe einer impliziten Methode:

$$(I - \upsilon \Delta t \nabla^2) w_3(x) = w_2(x)$$

Führen wir vorher noch eine Fourier Transformation durch so vereinfacht sich die Gleichung zu:

$$\hat{w}_3(k) = \frac{\hat{w}_2(k)}{1 + \upsilon \Delta t \kappa^2} \text{ wobei } \kappa = |k|$$

Schritt 3: Code

```
for i in range(n):
    for j in range(n):
        x = kx[i,j]
        y = ky[i,j]
        kappa = x*x+y*y
        f = 1/(1+visc*dt*kappa)
        u[i,j] *= f
        v[i,j] *= f
```

Schritt 4: Projektion

Sei $w = u + \nabla q$ mit $\nabla \cdot u = 0$. Durch Multiplizieren beider Seiten mit ∇ erhält man folgende Gleichung:

$$\nabla \cdot w = \nabla^2 q$$

Löst man die Gleichung nach q auf so erhält man Pw folgendermaßen:

$$u = Pw = w - \nabla q$$

Führen wir vorher eine Fourier Transformation durch so ergibt sich schließlich:

$$\hat{P}\hat{w}(k) = \hat{w}(k) - \frac{1}{\kappa^2}(k \cdot \hat{w}(k))k$$

Schritt 4: Code

```
for i in range(n):
  for j in range(n):
    x = kx[i,j]
    y = ky[i,j]
    kappa = x*x+y*y
    if kappa==0.0:
      continue
    u[i,j] = ( (1-x*x/kappa)*u[i,j]-x*y/kappa *v[i,j] )
    v[i,j] = ( -y*x/kappa *u[i,j] + (1-y*y/kappa)*v[i,j] )
```

Simulieren einer Substanz die sich durch das Fluid bewegt

Sei a eine skalare Größe welche sich durch das Fluid beschreibt. Dann wird das Verhalten von a durch folgendene Differentialgleichung beschrieben:

$$\frac{\partial a}{\partial t} = -u \cdot \Delta a + \kappa_a \Delta^2 - \alpha_a a + S_a$$

κ_a : Diffusionskonstante

α_a : Dissipationskonstante

S_a : Quellterm

Die Gleichung ähnelt der Navier-Stokes-Gleichung. Wir haben wieder einen Advektions-, einen Diffusions- und einen „Kraft"-Term. Mit diesen können wir analog verfahren.

Dissipation

Herleitung

Wir haben also die Differentialgleichung:

$$\frac{\partial a}{\partial t} = -\alpha_a a$$

Diese lösen wir folgendermaßen:

$$(1 + \Delta t \alpha_a) a(x, t + \Delta t) = a(x, t)$$

Code

```
s/= (1+dt*dissRate)
```

Quellen

- Herleitung des Fluid Solvers: Stam, „Stable Fluids“
- Code: Stam, „A Simple Fluid Solver Based on the FFT“

https://github.com/svkle100/FluidSolver