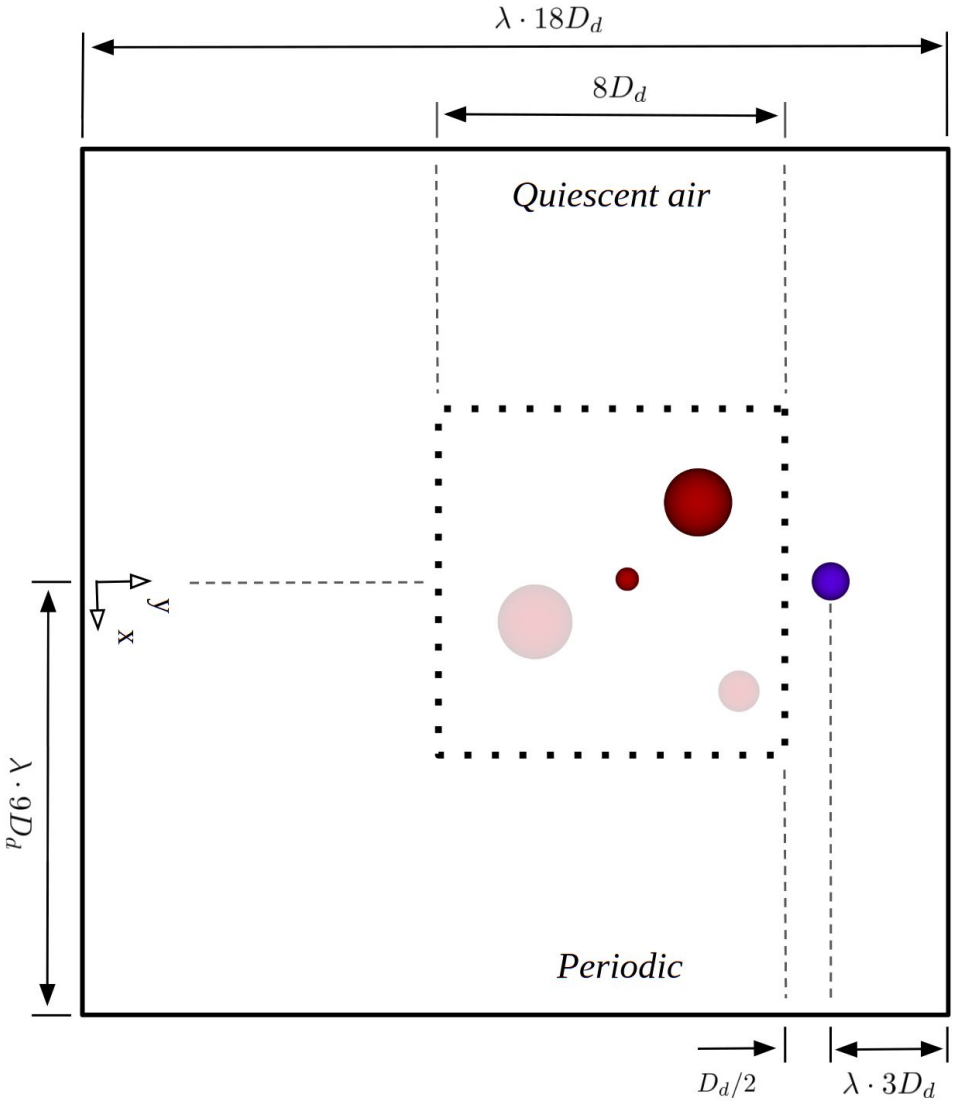
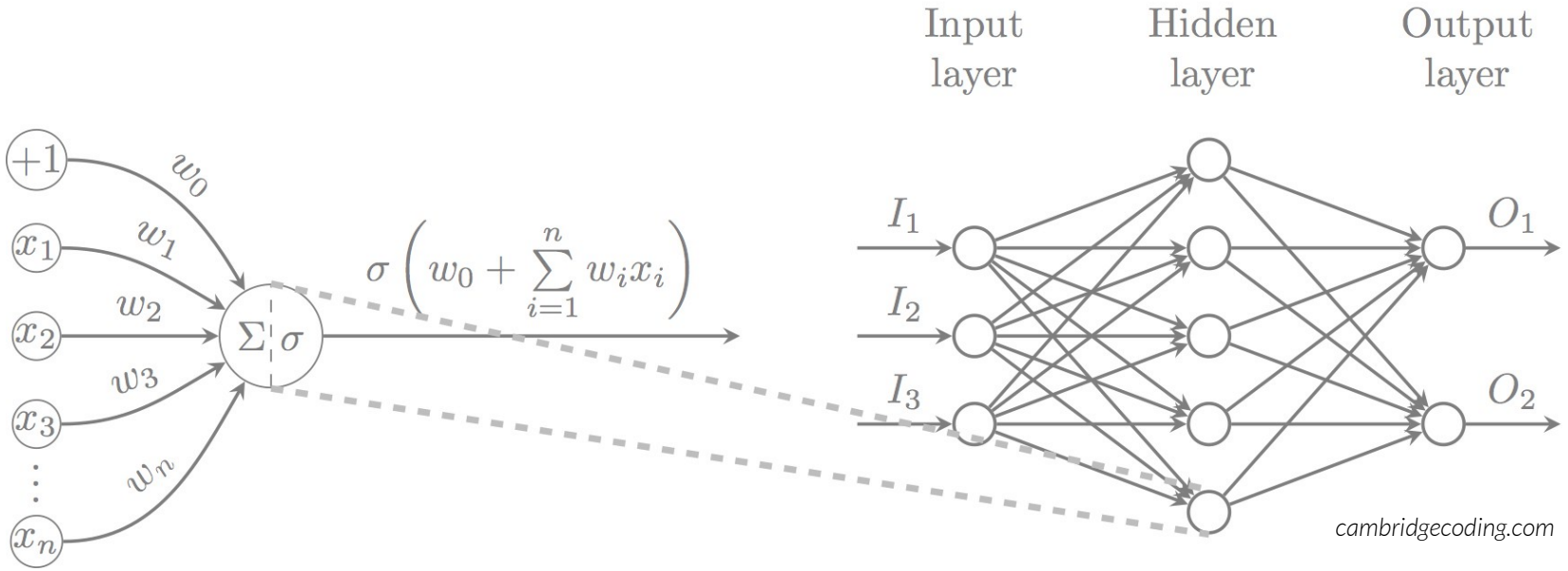
Aglomerado Partículas





Loss function: MSE

Taxa de aprendizado: $1 \cdot 10^{-2}$

Otimizador: Adam

Função de ativação: SELU

Dropout: 50%

Batch size: 1.600 - 8.000

Weight initialization: $\mathcal{N}$

Arquitetura: IL *[6 - 18] VI* + HL *2xVI* + HL *10xVI* + OL *[1 - 2] VD*

Multilayer Perceptron

Re

We

C_{sg}

C_{ls}

A_{ls}

N_p

$D_{p_1 \ldots p_2}$

$X_{p_1 \ldots p_2}$

$Y_{p_1 \ldots p_2}$

	$STD\,(W_a)$	$STD\,(L_a)$
N_p	0,41	0,40
C_{sg}	0,44	0,40
C_{ls}	-0,44	-0,40
A_{ls}	-0,46	-0,42

→ *MAPE (%) DUMMY para variáveis STD foram de 587% e 1.689%*

→ *MAPE (%) MLP para variáveis conjuntas AVG e STD foram de 211% e 853%*

→ Melhora de 25% em relação aos modelos de *machine learning*

→ Melhora de 9% com *data augmentation*

→ Melhora de 51% com *duas variáveis de saída*

→ Melhora de 7% com *18 variáveis de entrada*

	$AVG(Wa)\,[\%]$	$STD(Wa)\,[\%]$	$AVG(La)\,[\%]$	$STD(La)\,[\%]$
MEAN	$-0,9$	$-0,6$	$0,3$	$12,6$
STD	$13,5$	$0,8$	$51,7$	$30,9$

→ Efeito da malha

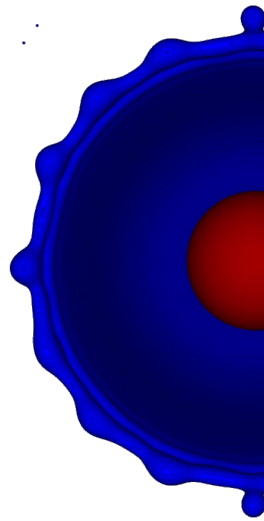
Anexo

Método Numérico

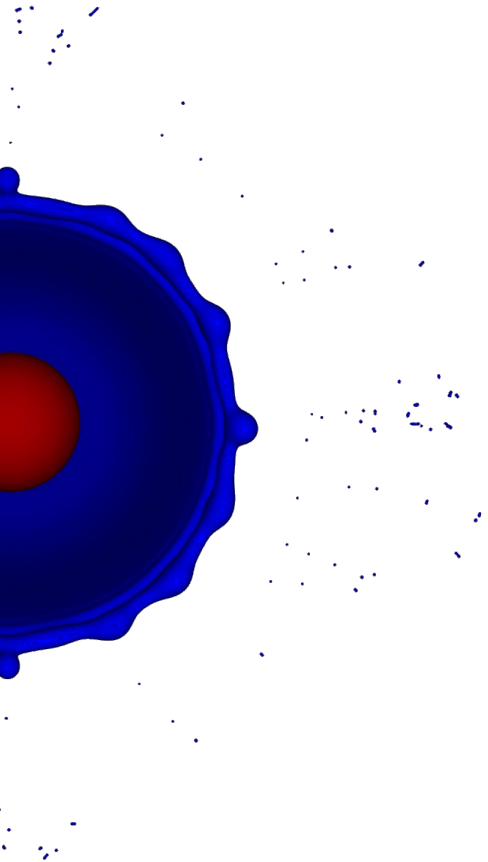
```
event vof (i++) {

  vector q1 = q, q2[];
    foreach()
      foreach_dimension() {
        double u = q.x[]/rho(f[]);
        q1.x[] = f[]*rho1*u;
        q2.x[] = (1. - f[])*rho2*u;
      }

  f.tracers = (scalar *){q1,q2};
  vof_advection ({f}, i);

}
```

```
foreach_dimension() {
  q2.x.inverse = true;
  q1.x.gradient = q2.x.gradient = minmod 2;
}
```

```
foreach()
  foreach_dimension()
    q.x[] = q1.x[] + q2.x[];
```



http://basilisk.fr/src/momentum.h



```
event pressure (i++) {

  foreach()
    foreach_dimension()
      u.x[] = (q.x[] + dt*g.x[])/rho[];
      mgu = viscosity (u, mu, rho, dt);

  foreach_face()
    uf.x[] = alpha.x[]*(q.x[] + q.x[-1])/2. + dt*a.x[];

  mgp = poisson (p, rhs, alpha, tolerance = TOLERANCE/sq(dt));
  ...

}
```

$\rho a - \nabla p$

$1/\rho$

```
foreach()
  foreach_dimension()
    q.x[] = u.x[]*rho[] - dt*g.x[];
```

```
double div = 0.;
foreach_dimension()
  div += uf.x[1] - uf.x[];
  rhs[] += div/(dt*Delta);
```

$a - \frac{\nabla p}{\rho}$

```
foreach()
  foreach_dimension() {
    g.x[] = rho[]*(gf.x[] + gf.x[1])/2.;
    q.x[] += dt*g.x[];
  }
```

http://basilisk.fr/src/all-mach.h