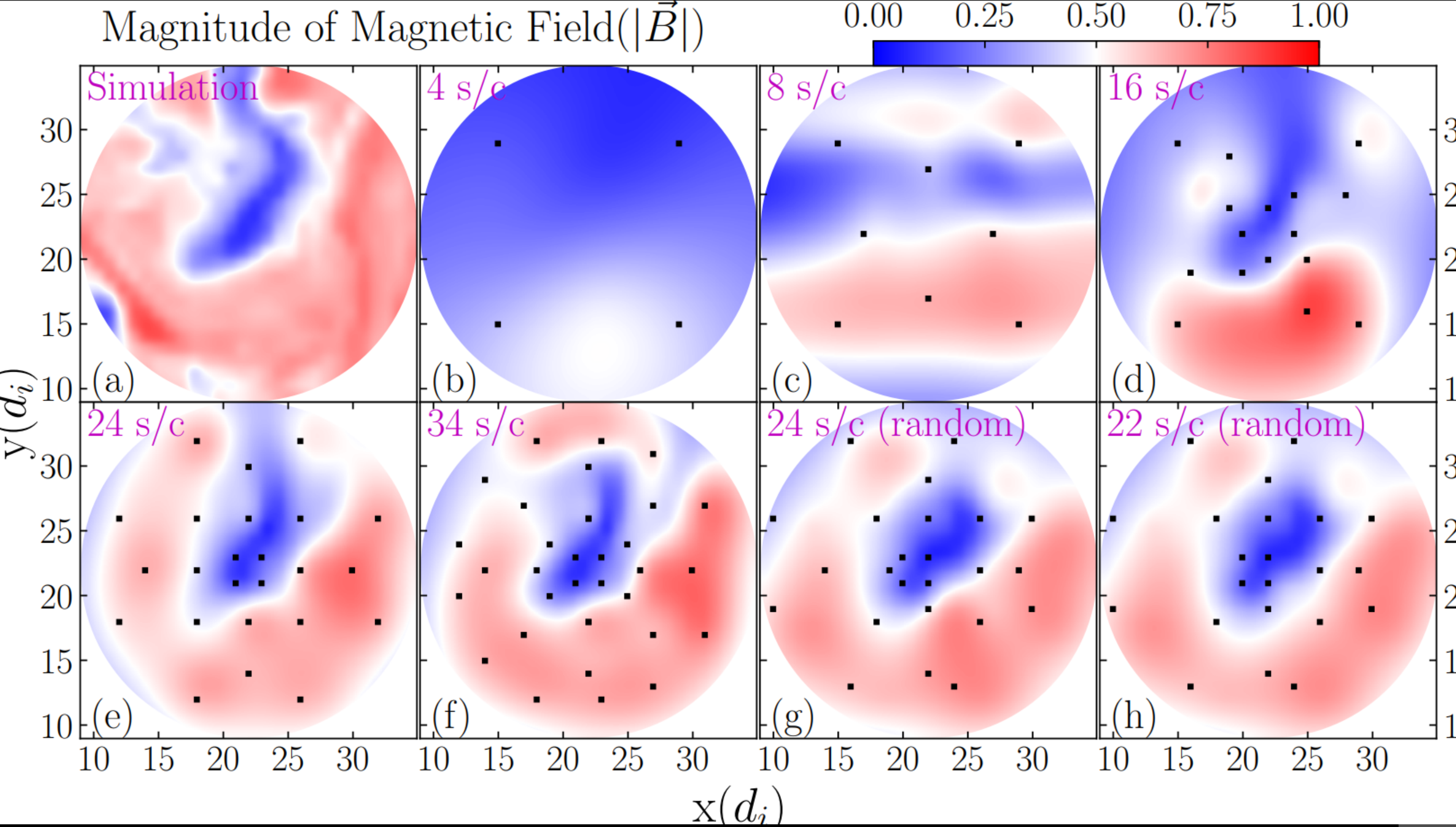
Gaussian Process Regression for Magnetic Field Reconstruction



https://slides.com/qudsi/magnetore

Some relevant scales:

$$d_i(1\mathrm{AU}) \sim 100 \text{ km}$$

$$V_{sw} \sim 500 \text{ km/sec}$$

$$X_{sim}(\text{boxsize}) \sim 40 d_i$$

$$\sim 4 \times 10^3 \text{km}$$

$$d_{spc} \sim [1, 11]\, d_i$$

$$\sim [10^2, 10^3] \text{ km}$$

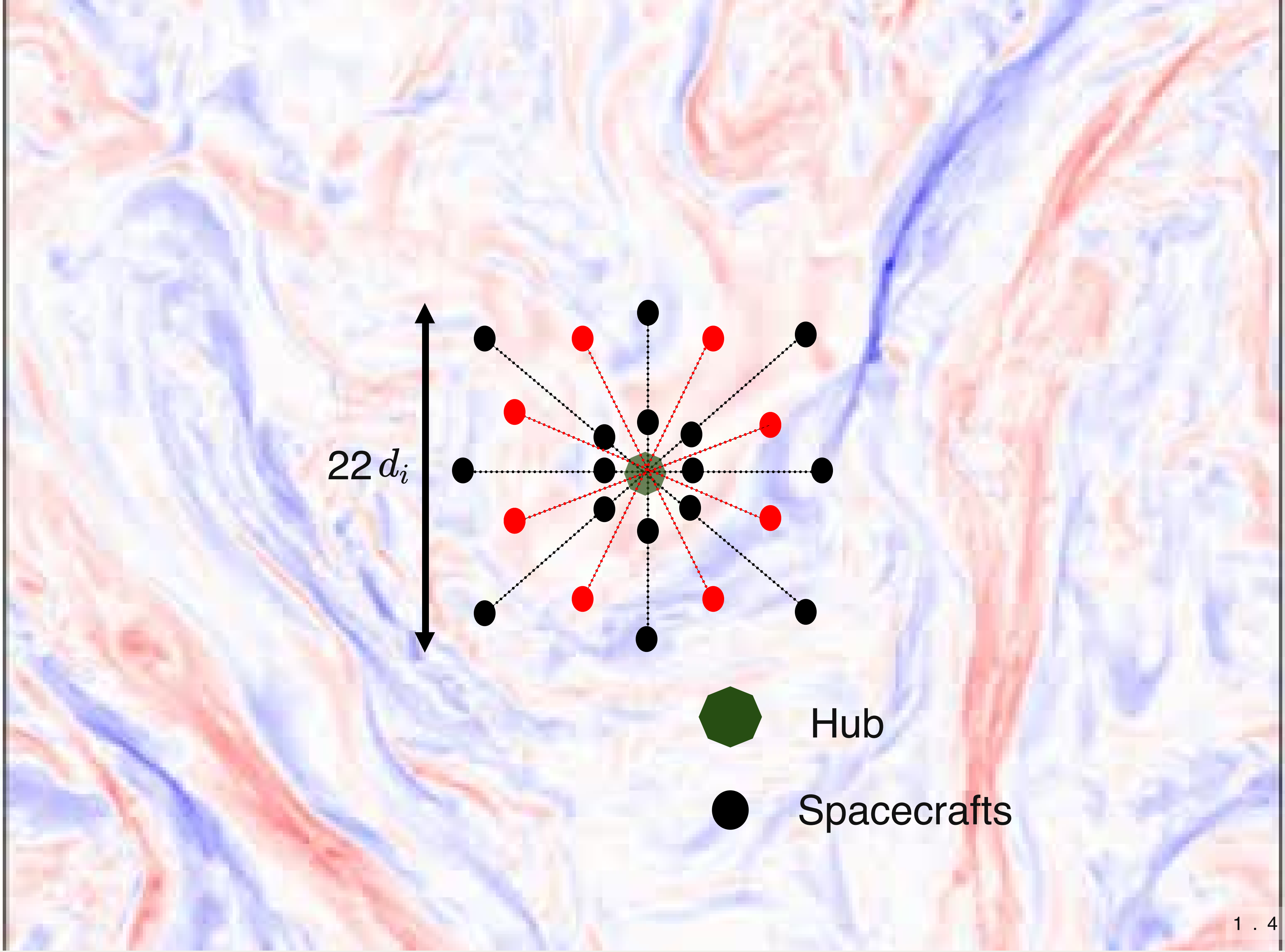
$$f[min, max] \sim V_{sw}/(2 \times d_{spc,max,min})$$

$$\sim [0.25, 2.5] \text{ Hz}$$

$$\nu(\text{sampling rate}) \sim 40\text{Hz}$$

$$(\text{just because of box size and stuff})$$

$$\text{time to move across the simulation box} \sim 10 \text{ sec}$$



$22\,d_i$
Hub
Spacecrafts

Gaussian Process Regression Interpolation

Gaussian Process Regression:

Gaussian Process Regression:

It is a probabilistic data imputation method

Gaussian Process Regression:

It is a probabilistic data imputation method

It is a probability distribution over possible functions that fit the given finite dataset

Gaussian Process Regression:

It is a probabilistic data imputation method

It is a probability distribution over possible functions that fit the given finite dataset

dataset with finite number of observation is modelled as if it were a multivariate normal distribution

Gaussian Process Regression:

It is a probabilistic data imputation method

It is a probability distribution over possible functions that fit the given finite dataset

dataset with finite number of observation is modelled as if it were a multivariate normal distribution

Mean function

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$

Gaussian Process Regression:

It is a probabilistic data imputation method

It is a probability distribution over possible functions that fit the given finite dataset

dataset with finite number of observation is modelled as if it were a multivariate normal distribution

Mean function

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$

Covariance function

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

Gaussian Process Regression:

It is a probabilistic data imputation method

It is a probability distribution over possible functions that fit the given finite dataset

dataset with finite number of observation is modelled as if it were a multivariate normal distribution

Mean function

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$

Covariance function

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

Gaussian Processes

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right)$$

Kernels

Kernels

$$k(x_1, x_2) = \text{constant\_value} \; \forall \; x_1, x_2$$ ← Constant

Kernels

$$k(x_1, x_2) = \text{constant\_value} \; \forall \; x_1, x_2$$ Constant

$$k(x_i, x_j) = \sigma_0^2 + x_i \cdot x_j$$ Linear

Kernels

$$k(x_1, x_2) = \text{constant\_value} \; \forall \; x_1, x_2$$ Constant

$$k(x_i, x_j) = \sigma_0^2 + x_i \cdot x_j$$ Linear

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right)$$ RBF

Kernels

$$k(x_1, x_2) = \text{constant\_value} \; \forall \; x_1, x_2$$ ← Constant

$$k(x_i, x_j) = \sigma_0^2 + x_i \cdot x_j$$ ← Linear

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right)$$ ← RBF

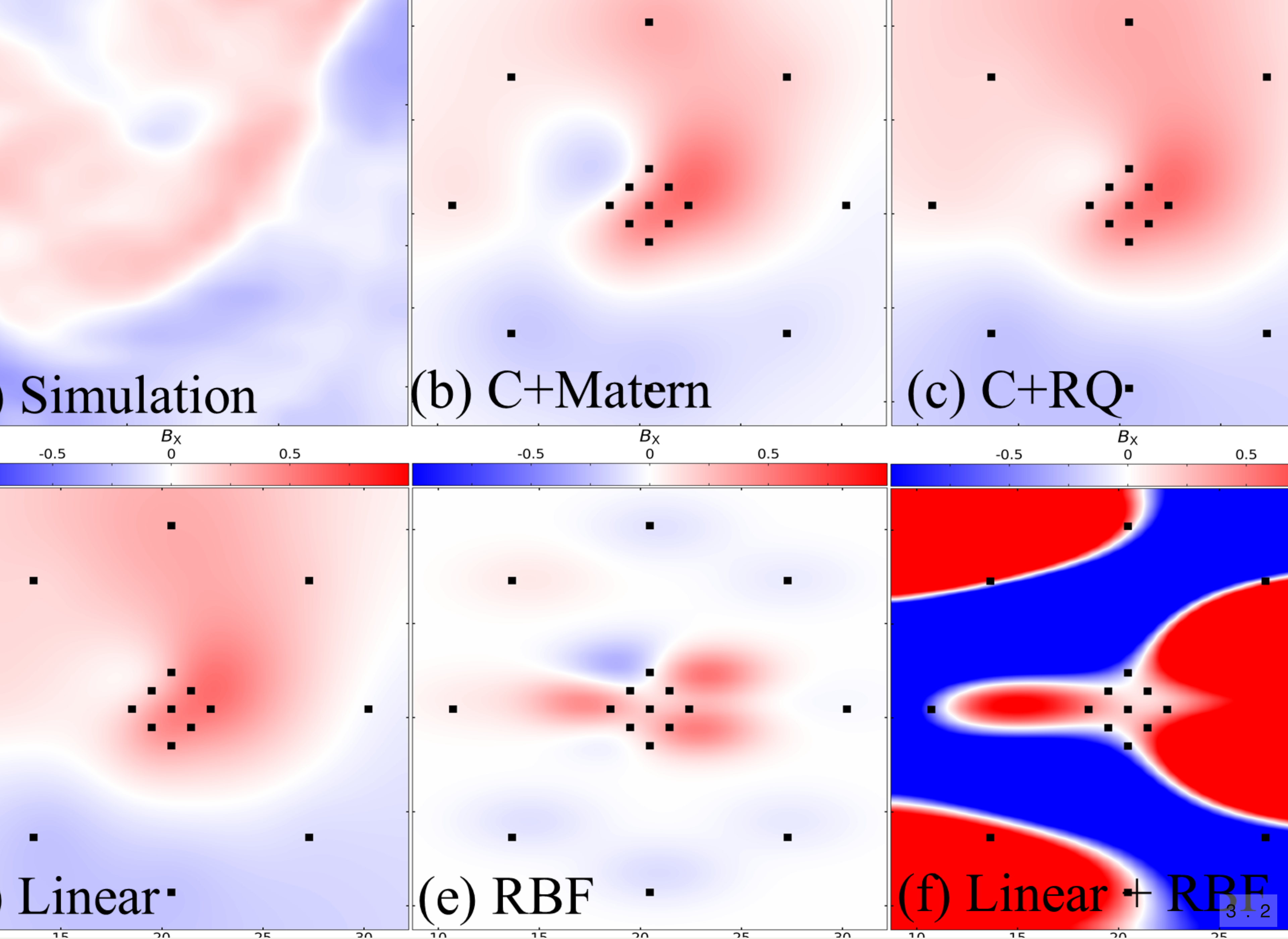
$$k(x_i, x_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right)^{\nu}$$ ← Matern

$$K_\nu\left(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right)$$

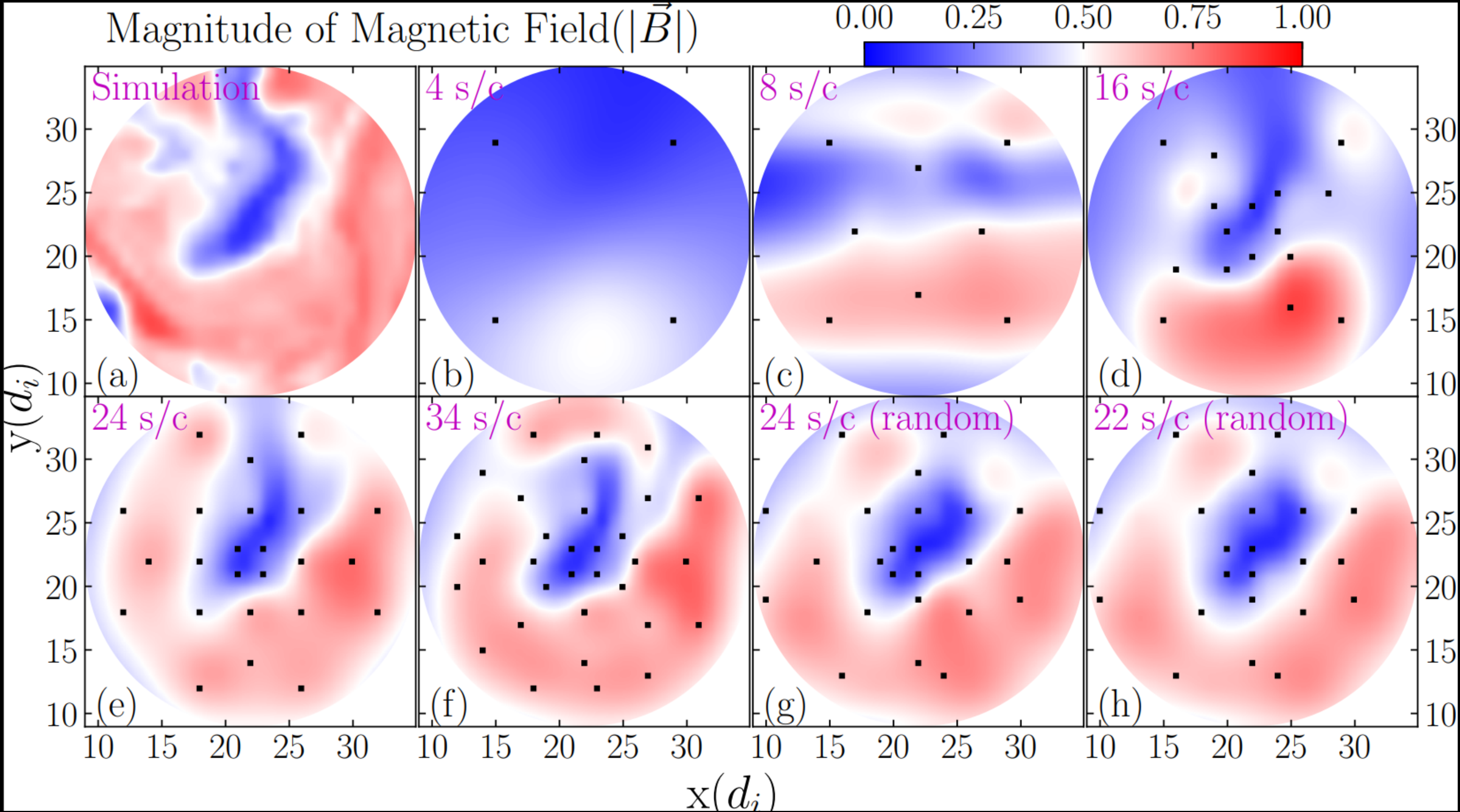
https://scikit-learn.org/stable/modules/gaussian_process.html

```python

# Define the plane an ddirection of the motion of spacecrafts
pln = 'xy'
drn = 'z'
ck_len = 5
mat_len_scl = [2,2,6]
mat_nu = 5/2
sigma_0 = 0
#Define the kernel you want to use and the associated required parameters
#kernel = CK(1, (1e-2, 1e2)) * RBF([2,2,2], (1e-2, 1e2))
#kernel = CK(ck_len, (1e-2, 1e2)) * Matern(length_scale=mat_len_scl, nu=mat_nu) #The relevant one
#kernel = CK(ck_len, (1e-2, 1e2)) + CK(ck_len, (1e-2, 1e2)) *\
#            Matern(length_scale=mat_len_scl, nu=mat_nu)
#kernel = CK(5, (1e-2, 1e2)) * Matern(length_scale=2, nu=3/2)
#kernel = CK(1.0, (1e-2, 1e2)) * RQ(length_scale=1, length_scale_bounds=(1e-2, 1e2))
#kernel = RBF([2,2,2], (1e-3, 1e3))
#kernel = DP( sigma_0=1.0, sigma_0_bounds=(1e-02, 100.0)) * RBF([2,2,2], (1e-3, 1e3))
kernel = DP(sigma_0=sigma_0, sigma_0_bounds=(1e-02, 100.0)) + CK(ck_len, (1e-2, 1e2))\
            * Matern(length_scale=mat_len_scl, nu=mat_nu)

# Define the gaussian processor regressor, keep the number of optimizers low, preferably within 100,
# thoug it is fine if you want to go for a higher number. However, if you do go for a higher number,
# please do so with ample caution

print('Running Gaussian processes')
n_restarts_optimizer = 18
gp = GaussianProcessRegressor(kernel=kernel, n_restarts_optimizer=n_restarts_optimizer)

#Get the model based on the defined kernel and test dataset
gp.fit(X, y)

print('Fit model created')

y_pred, MSE = gp.predict(x1x2x3, return_std=True)
Zp = np.reshape(y_pred,(indx_max - indx_min + 1, indy_max - indy_min + 1,
            indz_max - indz_min + 1))
MSE = np.reshape(MSE,(indx_max - indx_min + 1, indy_max - indy_min + 1,
            indz_max - indz_min + 1))
print('Done running Gaussian Processes successfully \n')
print('Saving GP data to a custom file name')
```

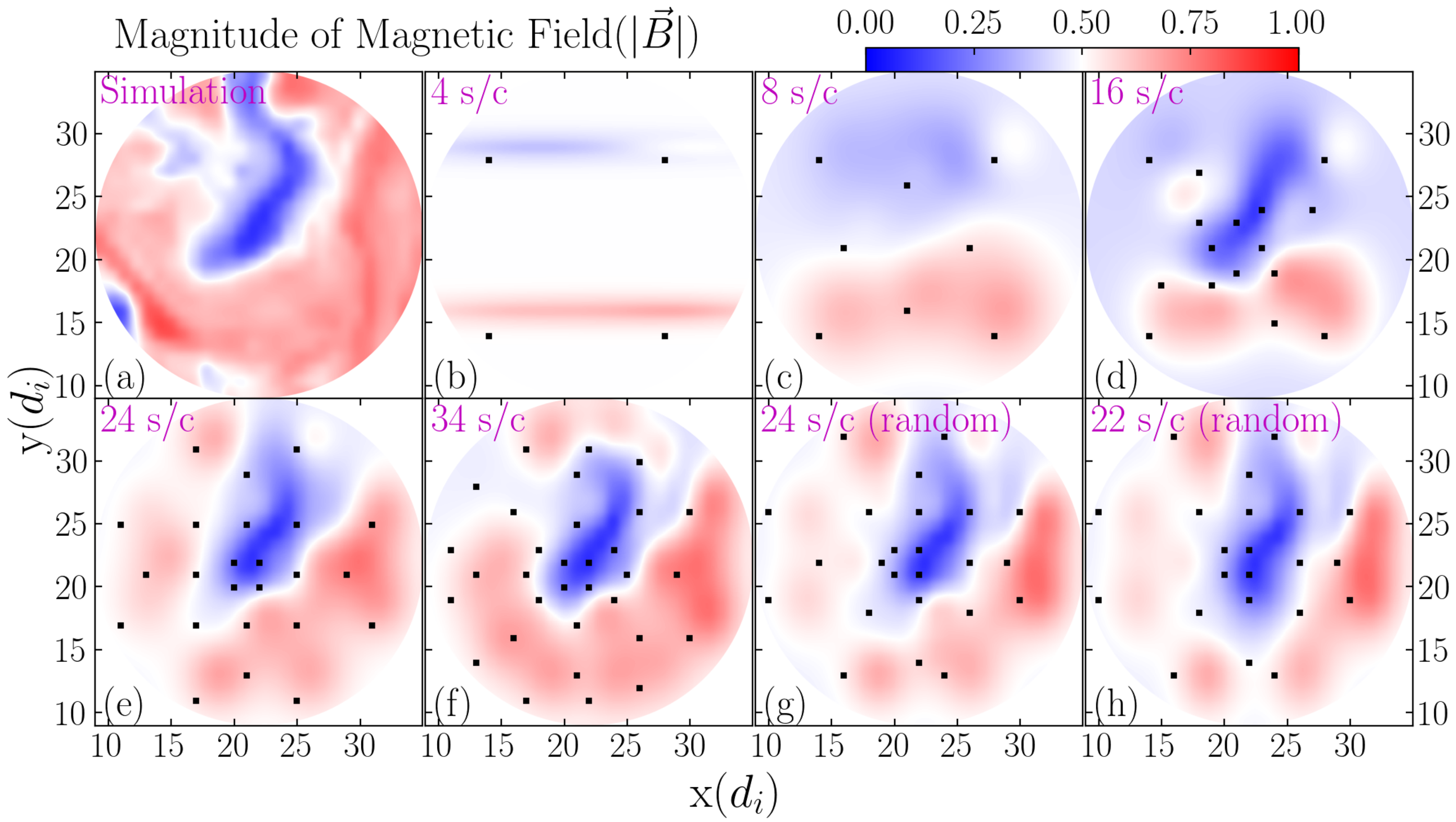
Results



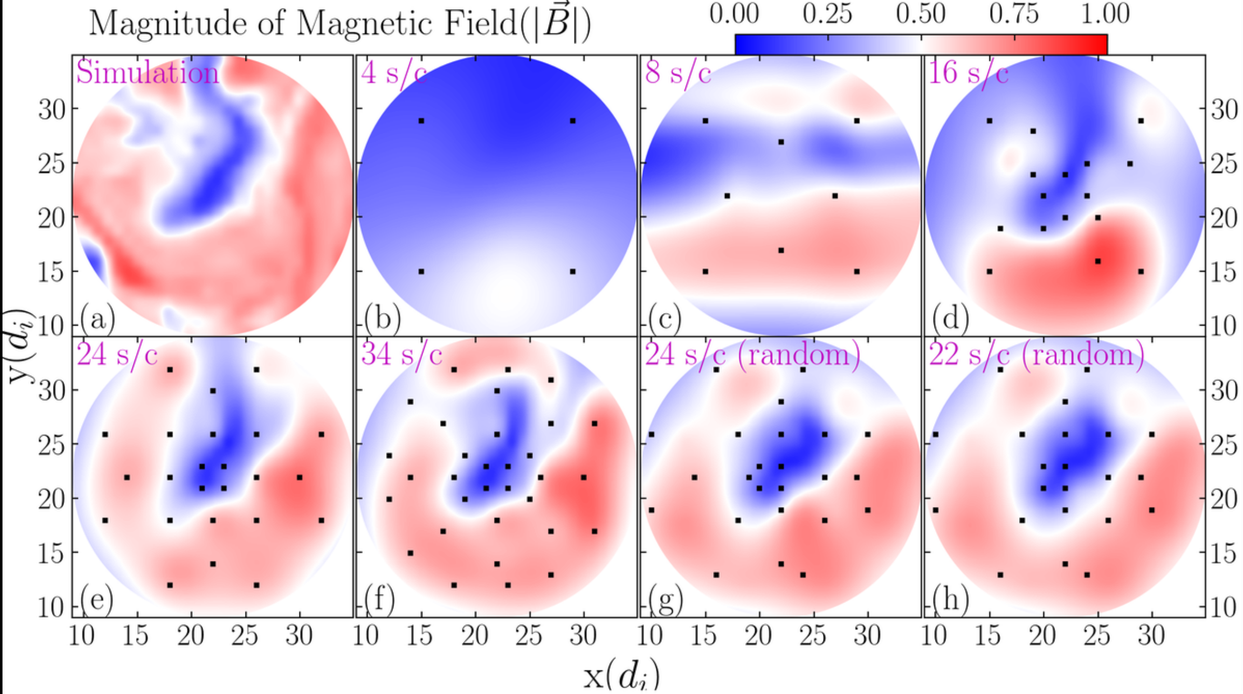
) Simulation
(b) C+Matern
(c) C+RQ
B_X
-0.5
0
0.5
B_X
-0.5
0
0.5
B_X
-0.5
0
0.5
) Linear
(e) RBF
(f) Linear + RBF
3 . 2



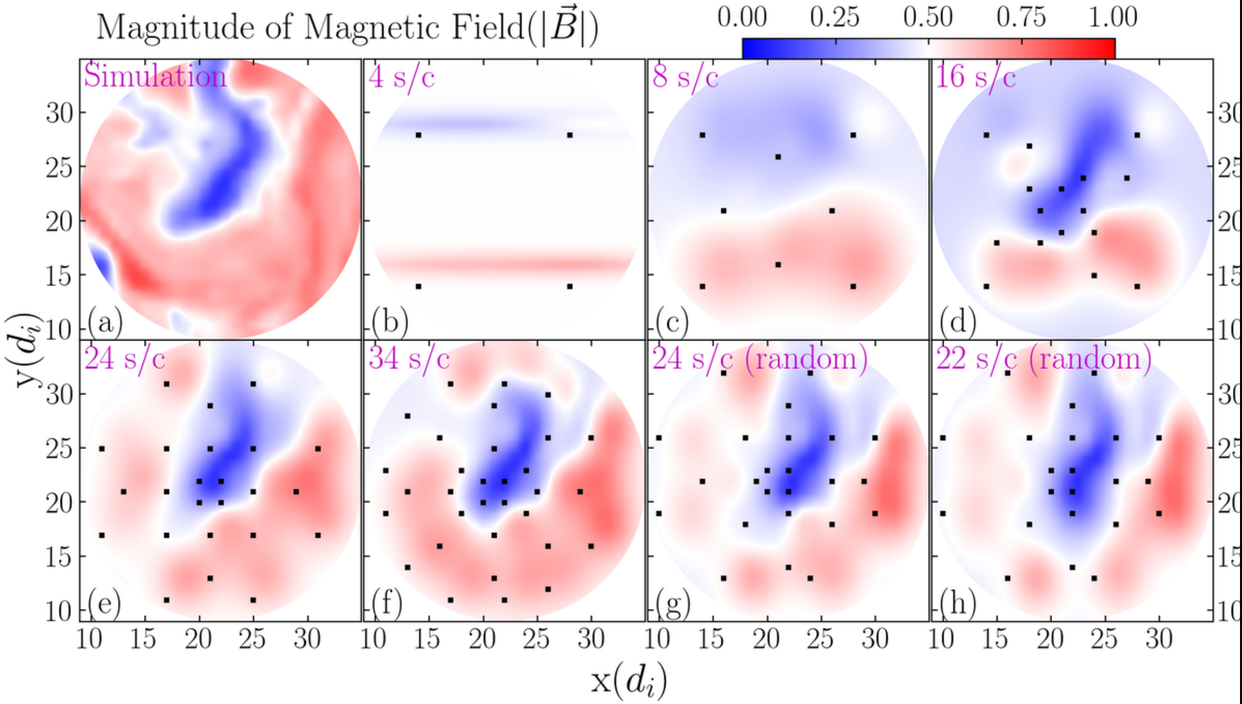
One component at a time



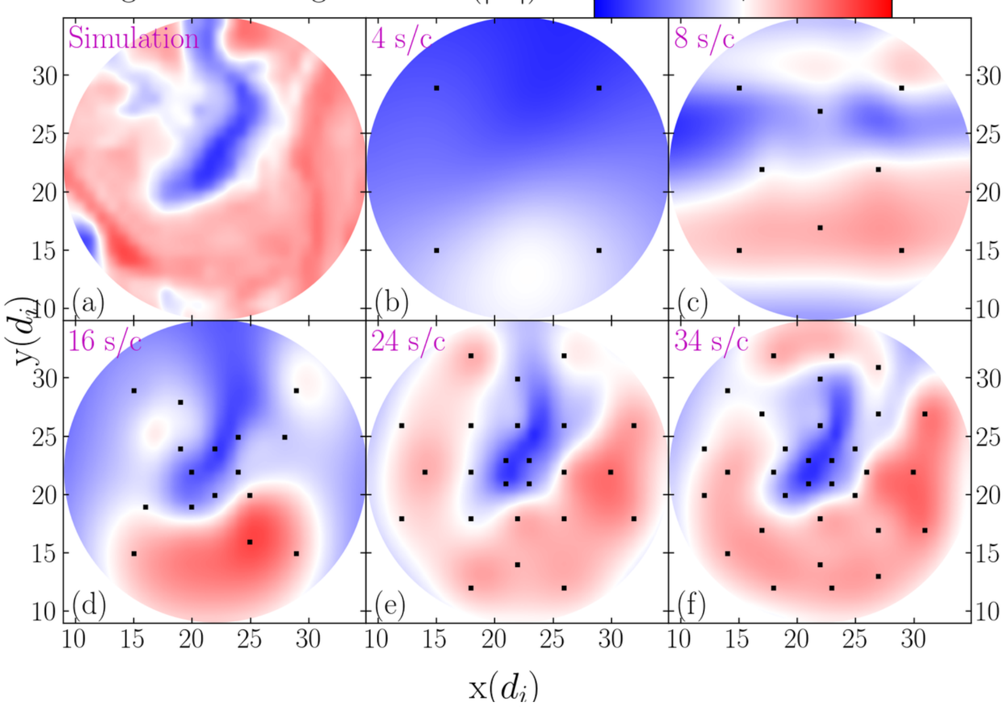
All components at a time

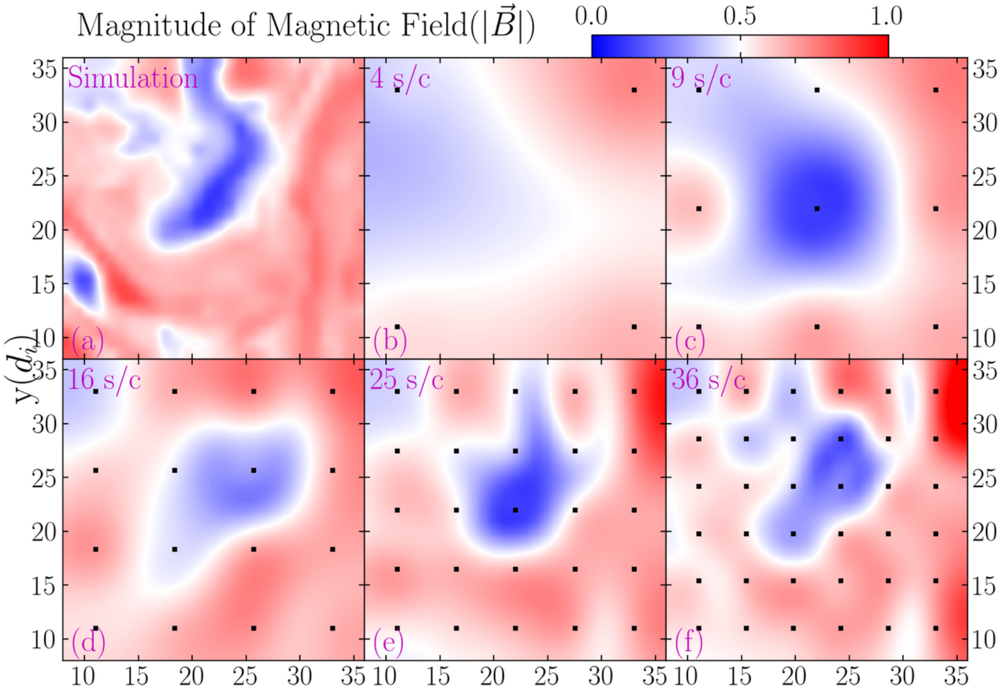


Single component input

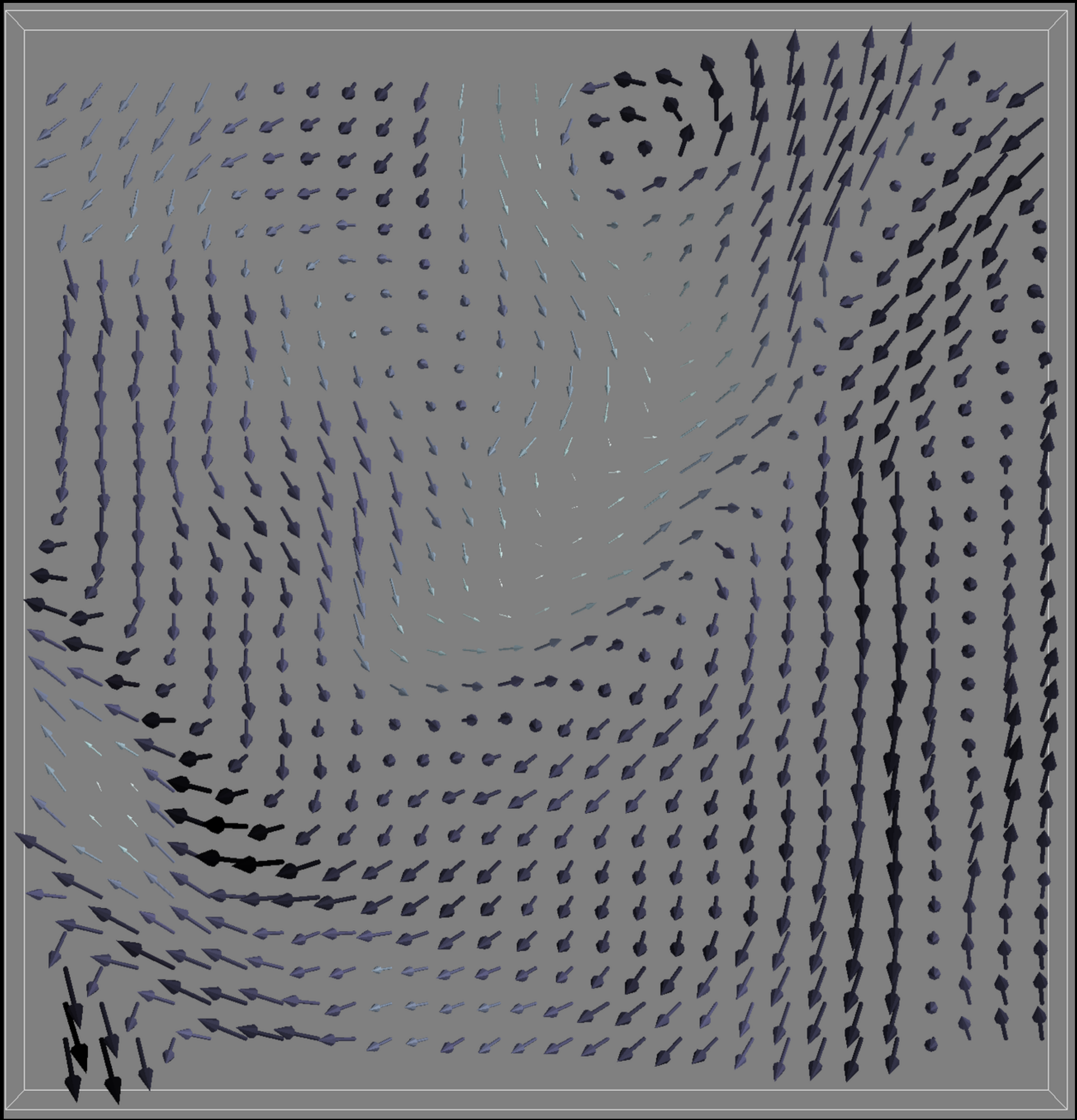


Vector input

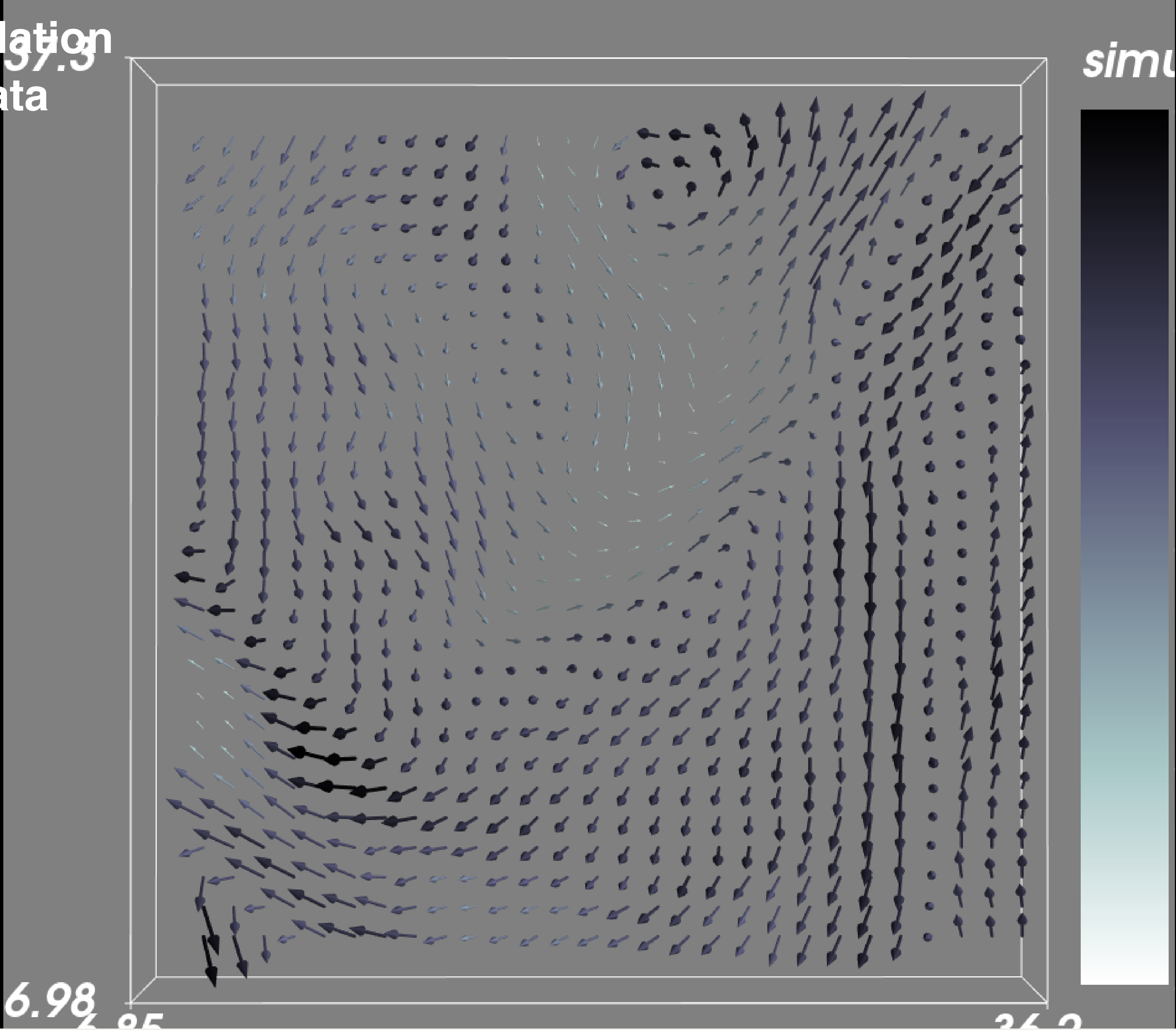

Simulation
4 s/c
8 s/c
16 s/c
24 s/c
34 s/c
(a)
(b)
(c)
(d)
(e)
(f)
$y(d_i)$
$x(d_i)$


Magnitude of Magnetic Field($|\vec{B}|$)
0.0
0.5
1.0
Simulation
4 s/c
9 s/c
16 s/c
25 s/c
36 s/c
(a)
(b)
(c)
(d)
(e)
(f)
$y(d_i)$

simulation data

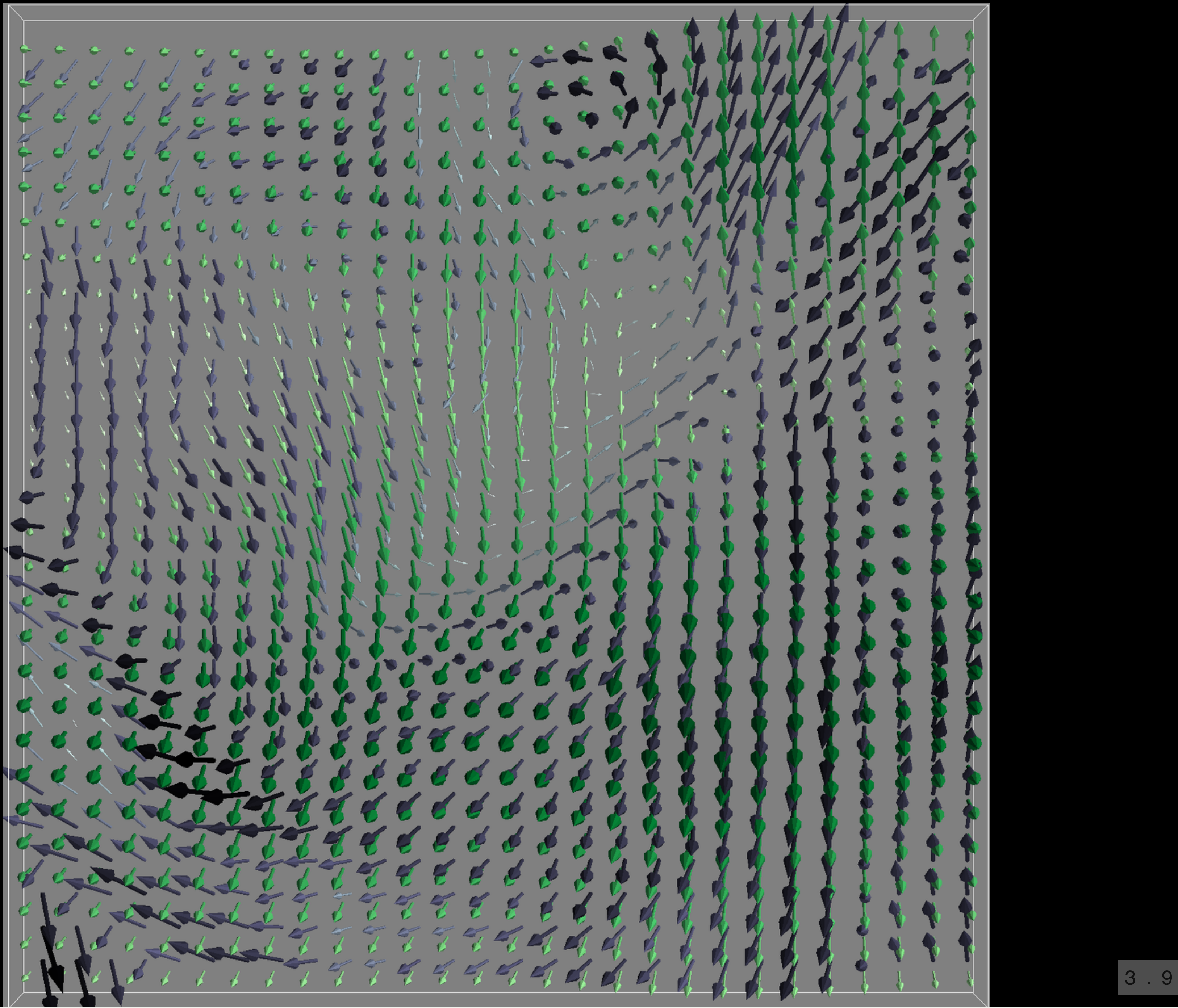


simulation data

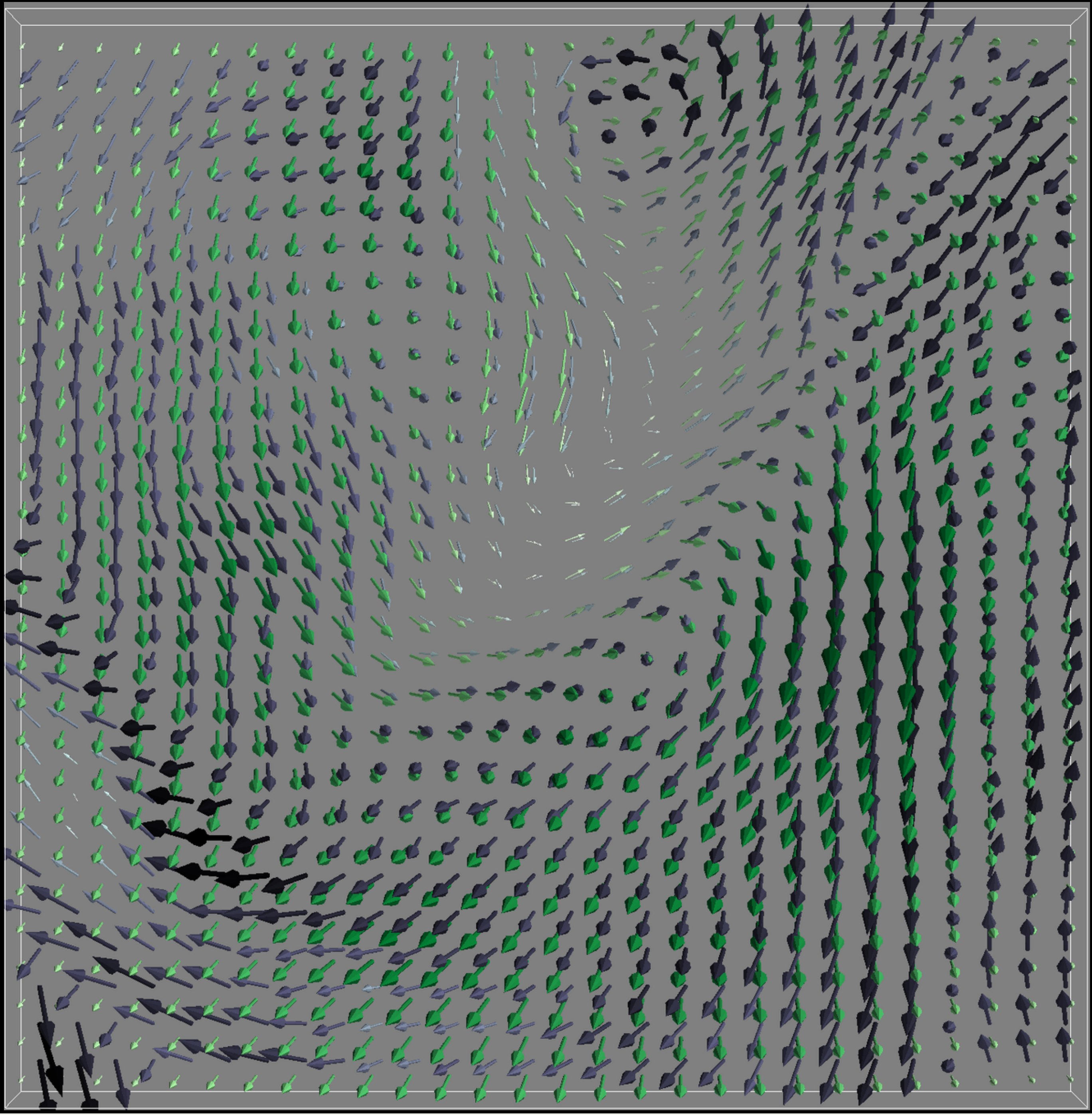


3 . 8

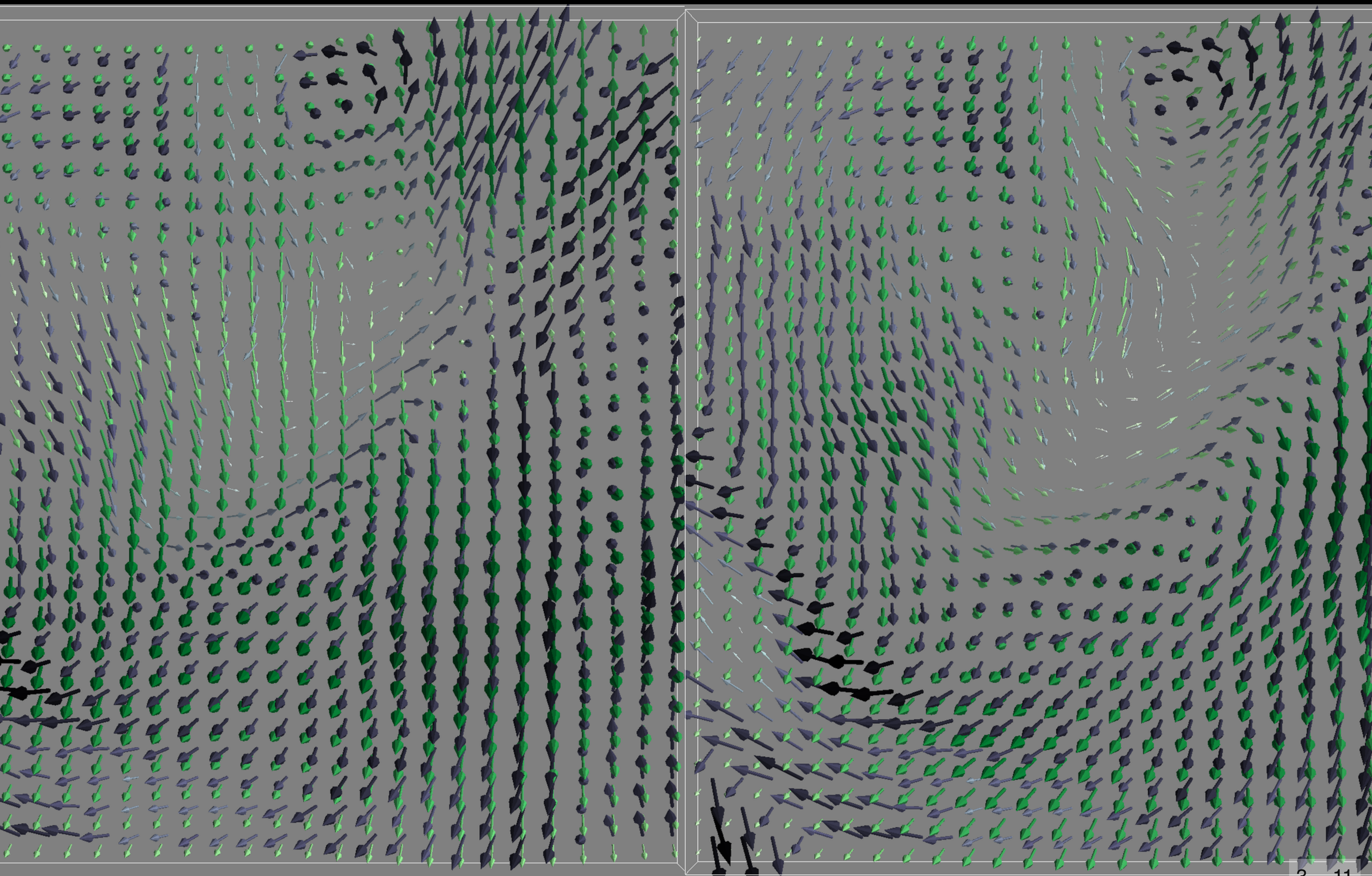
8 spc

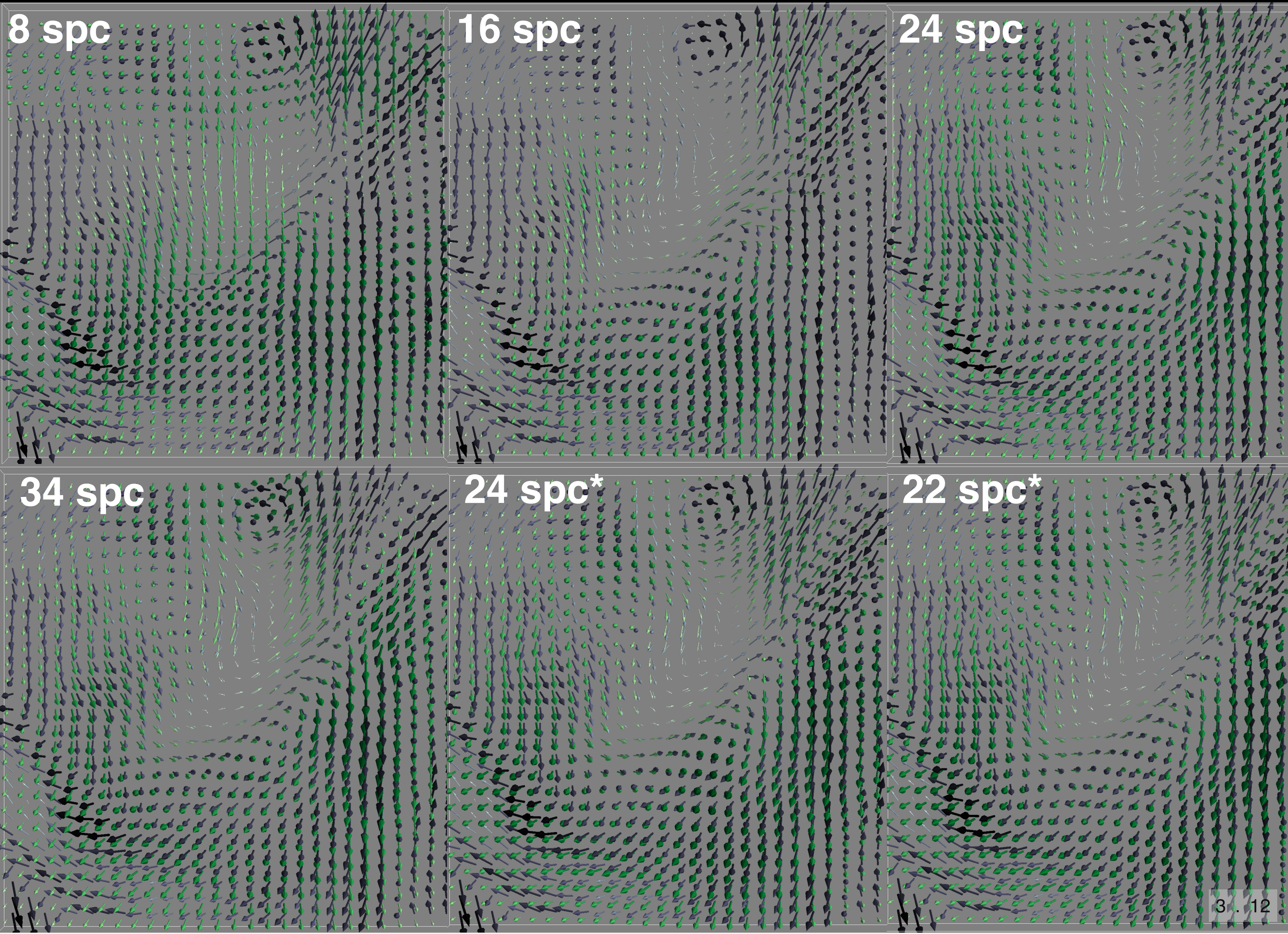


24 spc



8 spc **24 spc**




8 spc
16 spc
24 spc
34 spc
24 spc*
22 spc*
3 . 12

Moving Forward

What's next?

Find ways to compare the reconstructed images for different styles

What's next?

Find ways to compare the reconstructed images for different styles

Explore the impact of relative spacecraft separation and arrangement

What's next?

Find ways to compare the reconstructed images for different styles

Explore the impact of relative spacecraft separation and arrangement

Incorporate the zero divergence condition on magnetic field

What's next?

Find ways to compare the reconstructed images for different styles

Explore the impact of relative spacecraft separation and arrangement

Incorporate the zero divergence condition on magnetic field

Explore alternative algorithms/methods

What's next?

- Find ways to compare the reconstructed images for different styles
- Explore the impact of relative spacecraft separation and arrangement
- Incorporate the zero divergence condition on magnetic field
- Explore alternative algorithms/methods
  - Deep Gaussian Processes
  - Neural Network as GP

Thank You!