Domain decomposition of level set updates for salt segmentation

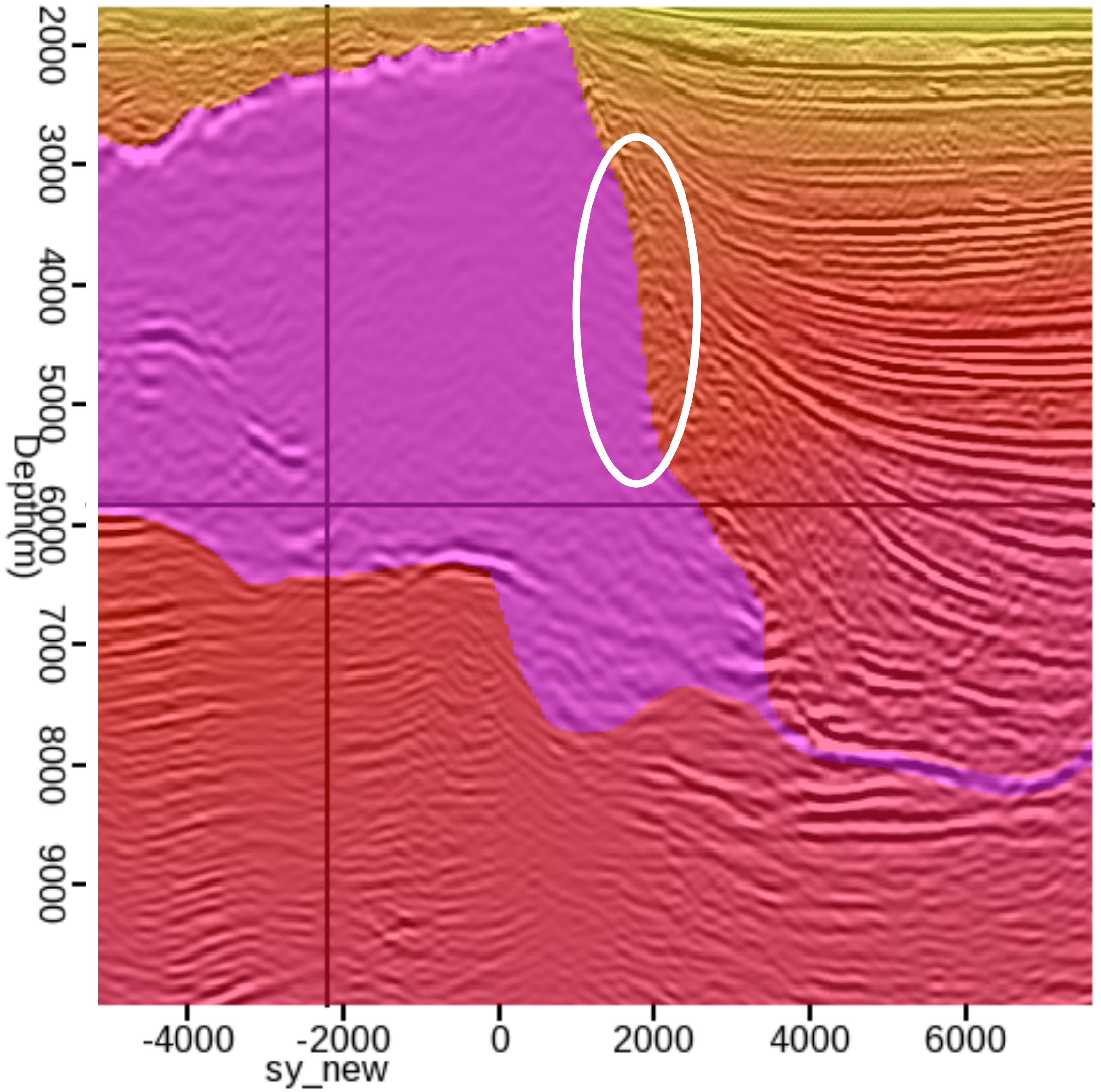
Taylor Dahlke
5/19/2015

SEP 158, pg. 51

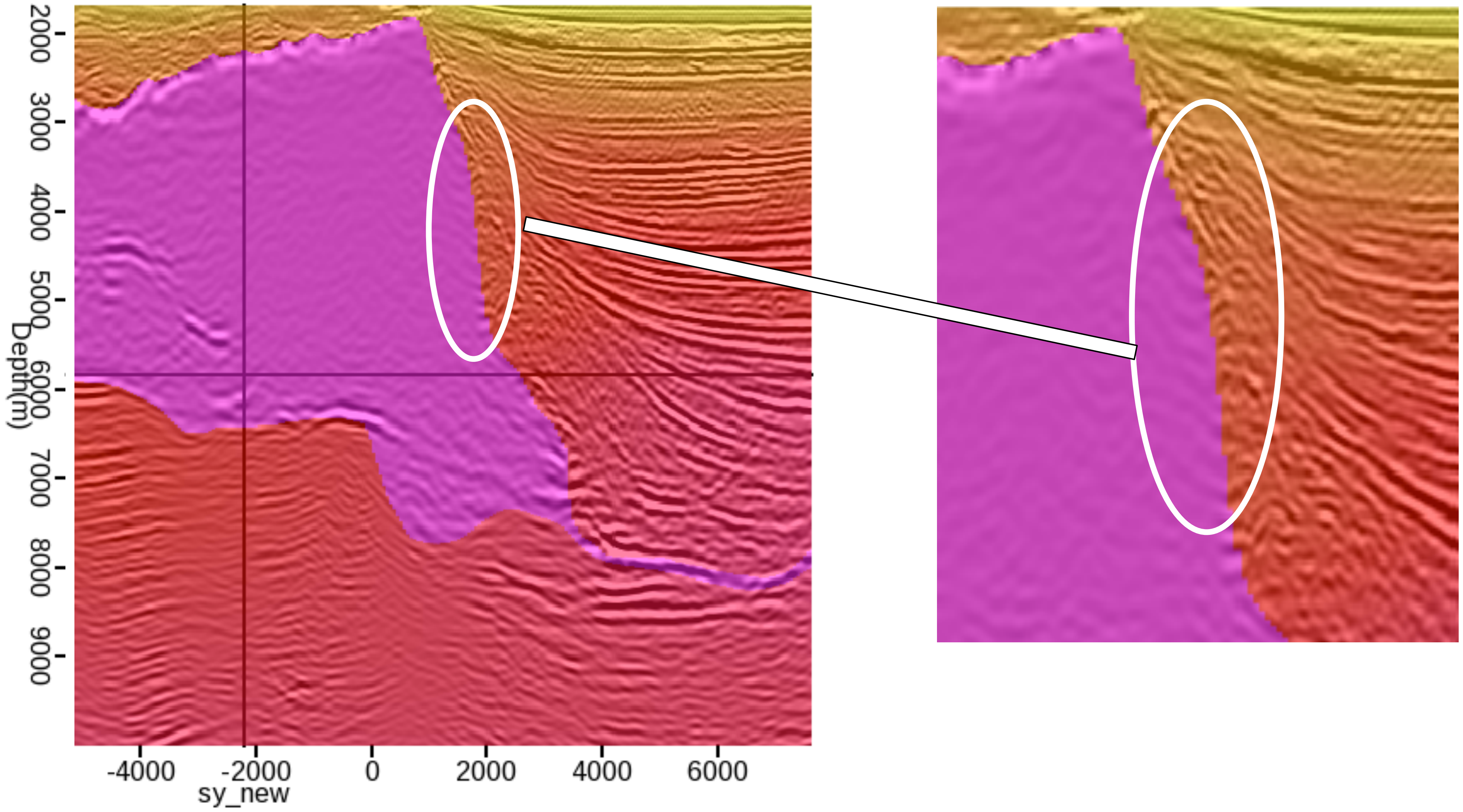
Outline

- What are the problems?
- Why level sets?
- Why domain decomposition?

Motivation

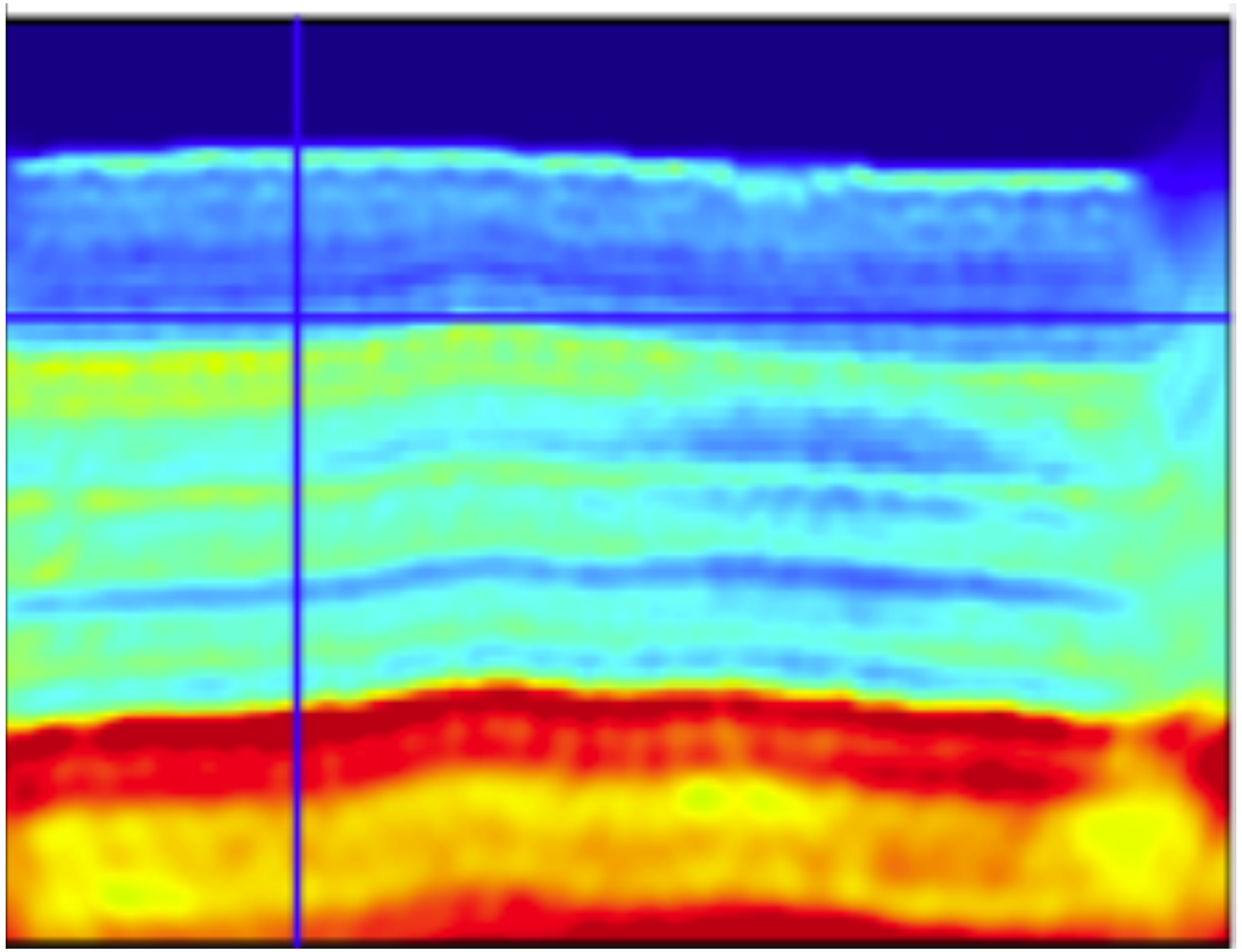

2000
3000
4000
5000
6000
7000
8000
9000
Depth(m)
-4000
-2000
0
2000
4000
6000
sy_new

Motivation



- Is that really where the salt boundary is?
- Can test with trial/error (expensive).
- What about inversion approach?

Motivation

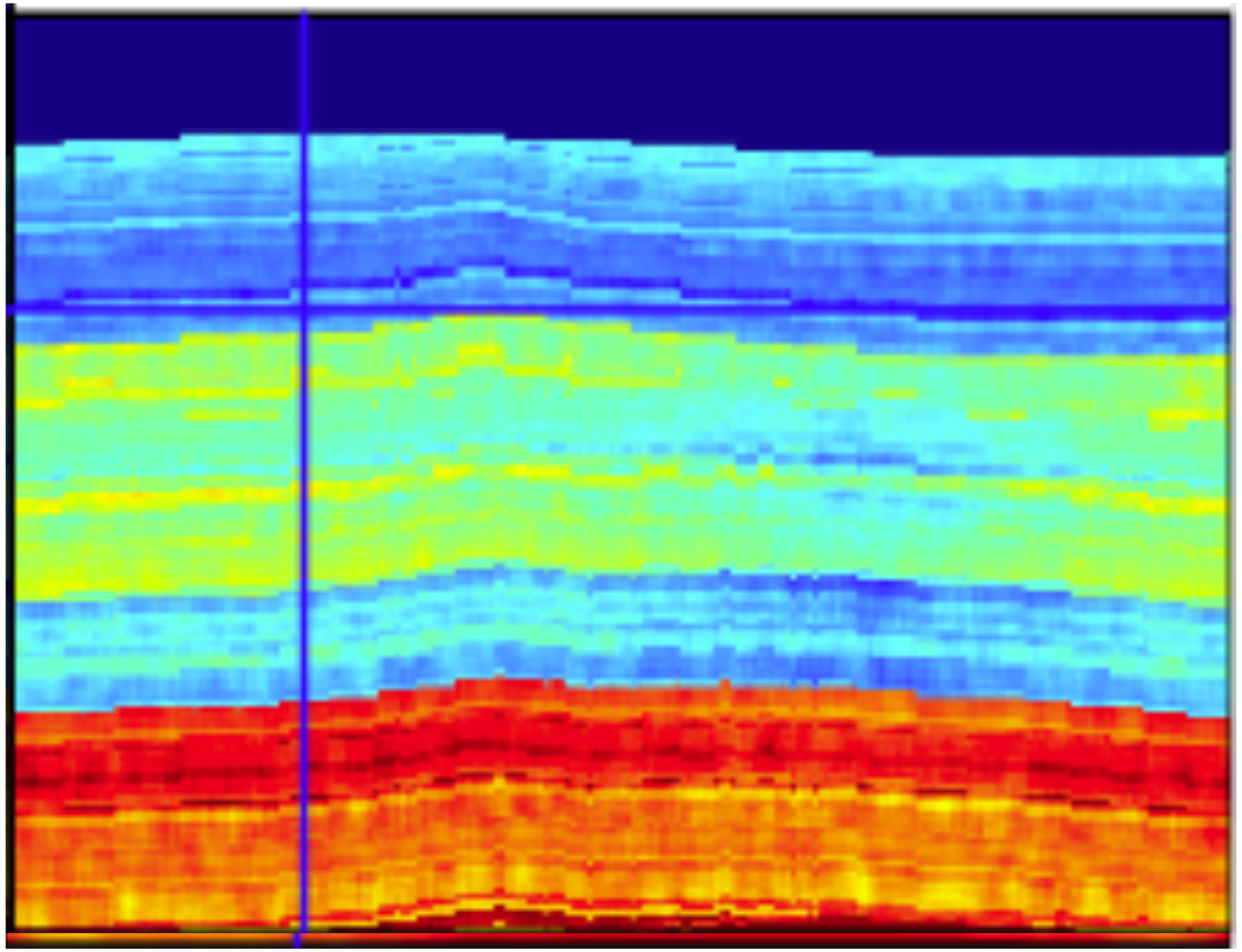


Figure, Xukai Shen 2015

Final FWI result

- Inverting for the velocity field (FWI) can’t give us sharp edges on salt
- What about parameterizing the body and inverting for that?

Motivation

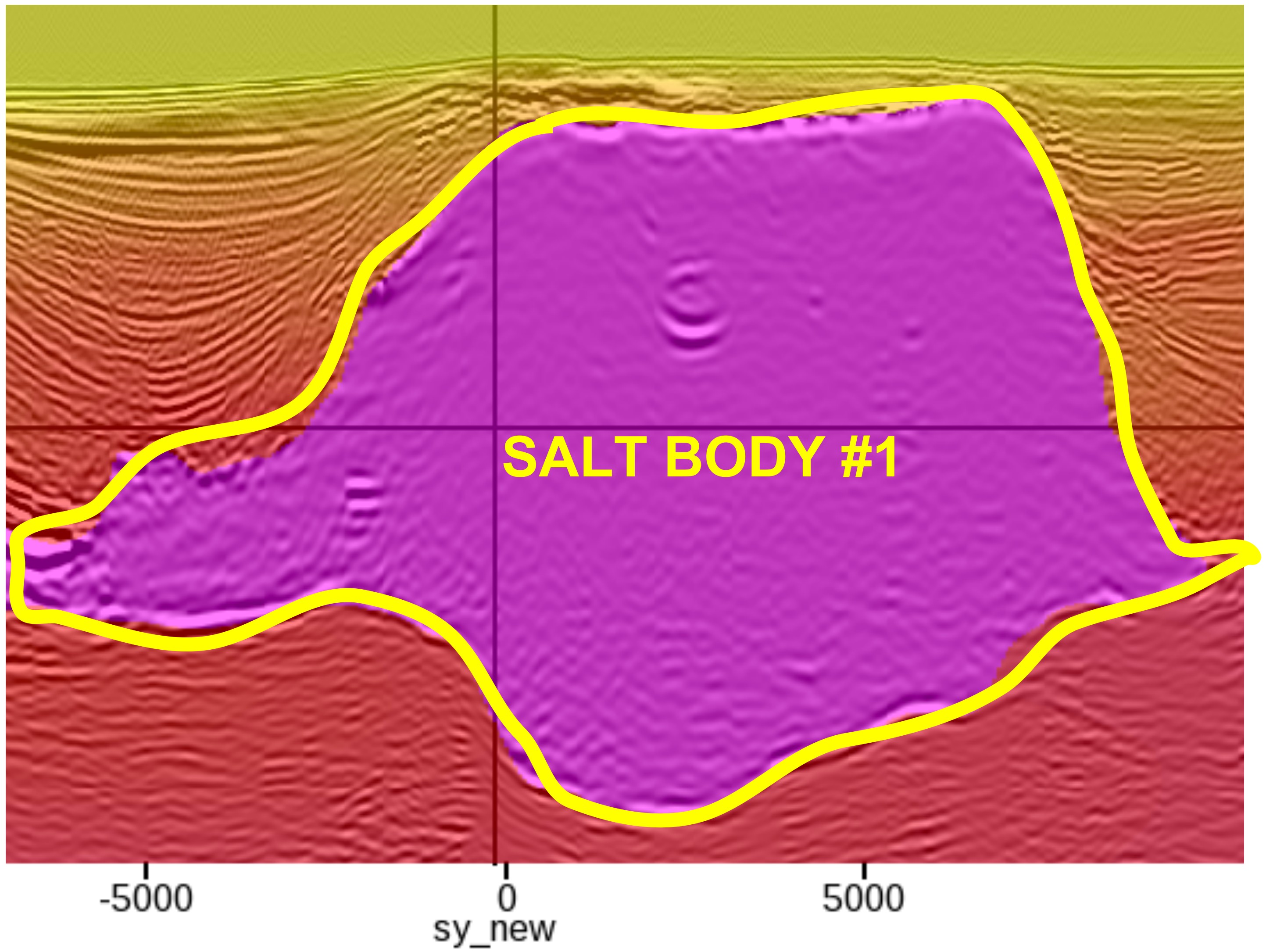


True Model

Figure, Xukai Shen 2015

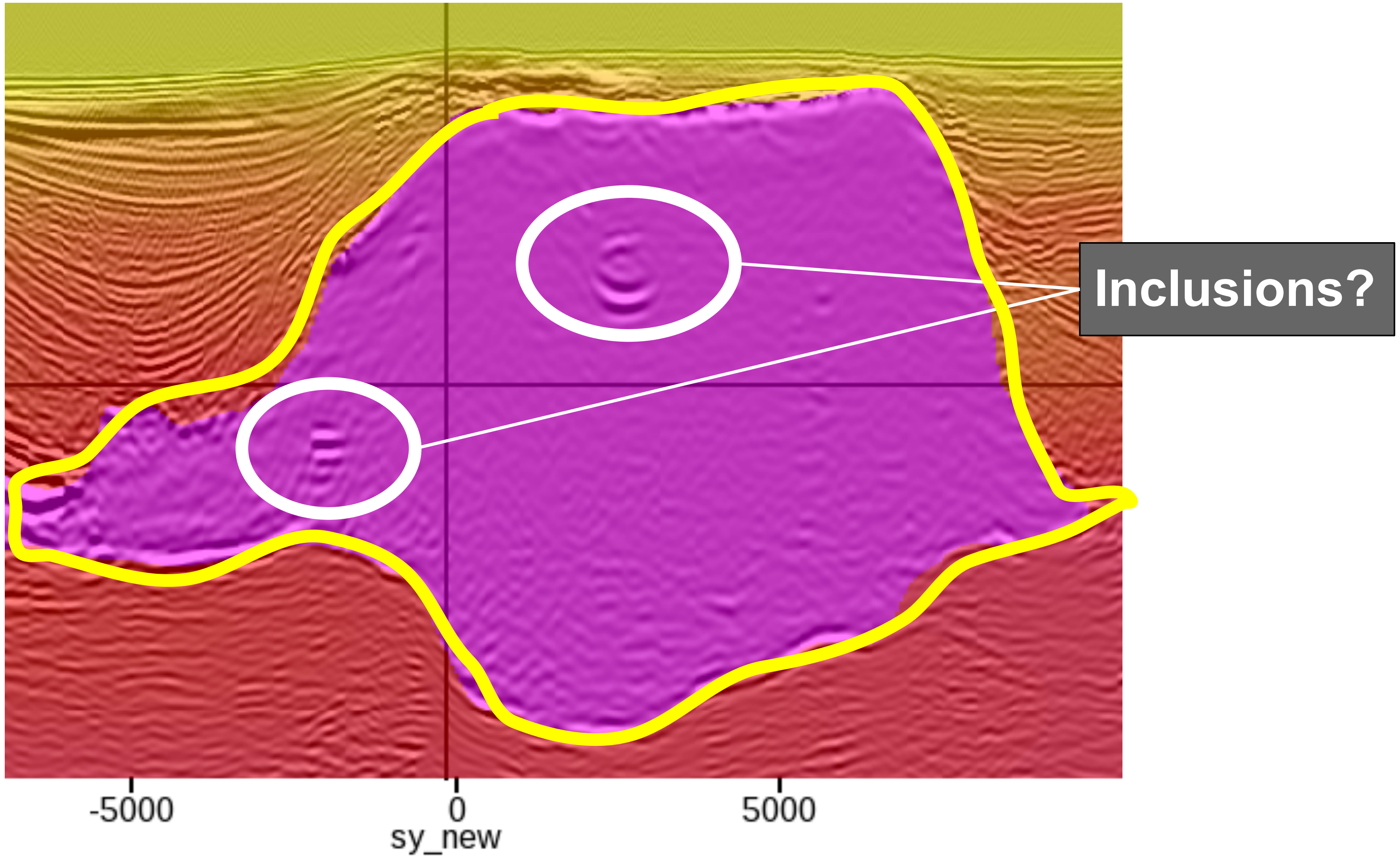
- Inverting for the velocity field (FWI) can’t give us sharp edges on salt
- What about parameterizing the body and inverting for that?

Motivation



- How do we define the salt body parameterization so that we can handle complex topologies?
- Can the topology change as we iteratively invert?

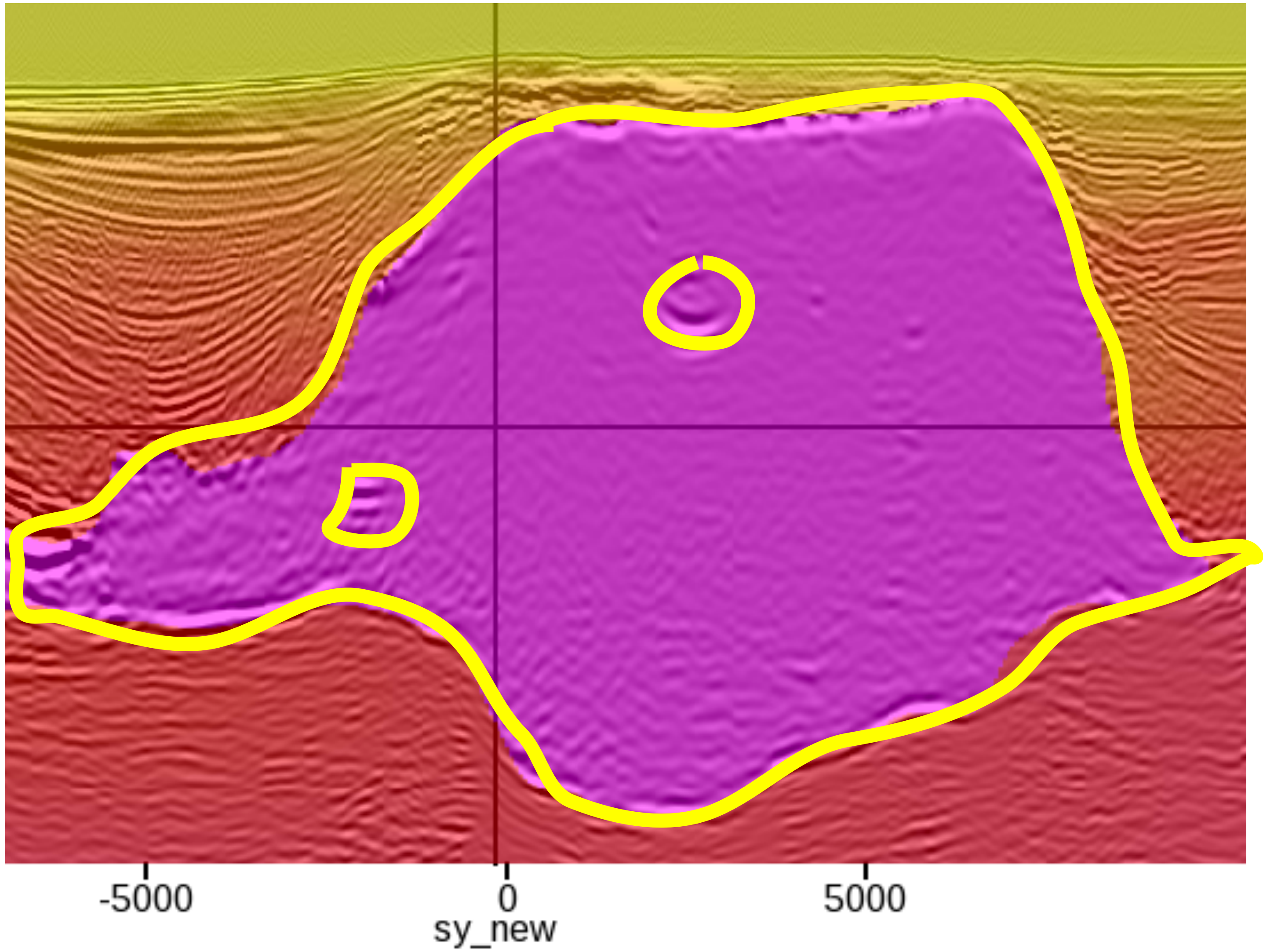
Motivation



- How do we define the salt body parameterization so that we can handle complex topologies?

- Can the topology change as we iteratively invert?

Motivation

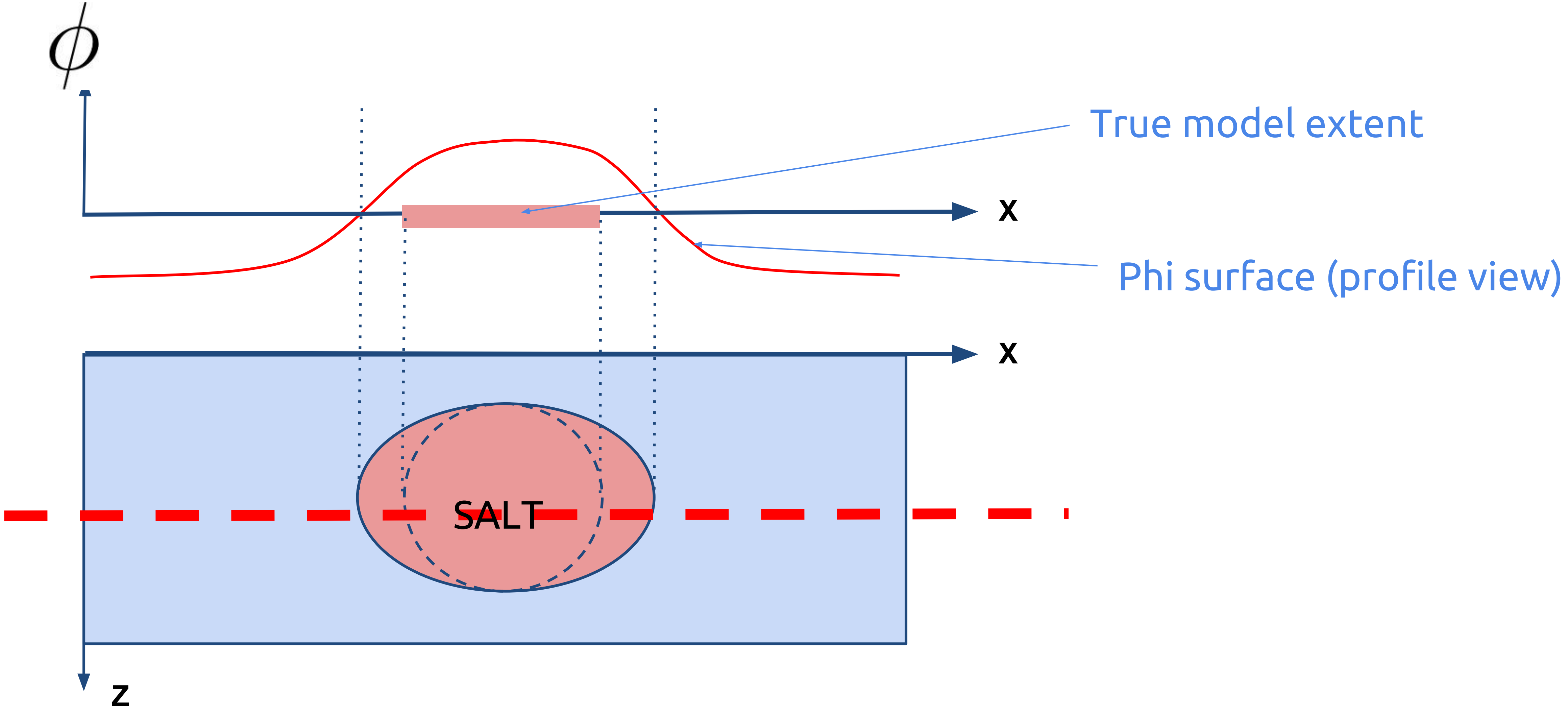


- How do we define the salt body parameterization so that we can handle complex topologies?
- Can the topology change as we iteratively invert?

Outline

- Why level sets?
  - It can delineate sharp boundaries
  - It can handle complex geometries, inclusions, merging, separation of bodies as inversion progresses.
- How does it work?

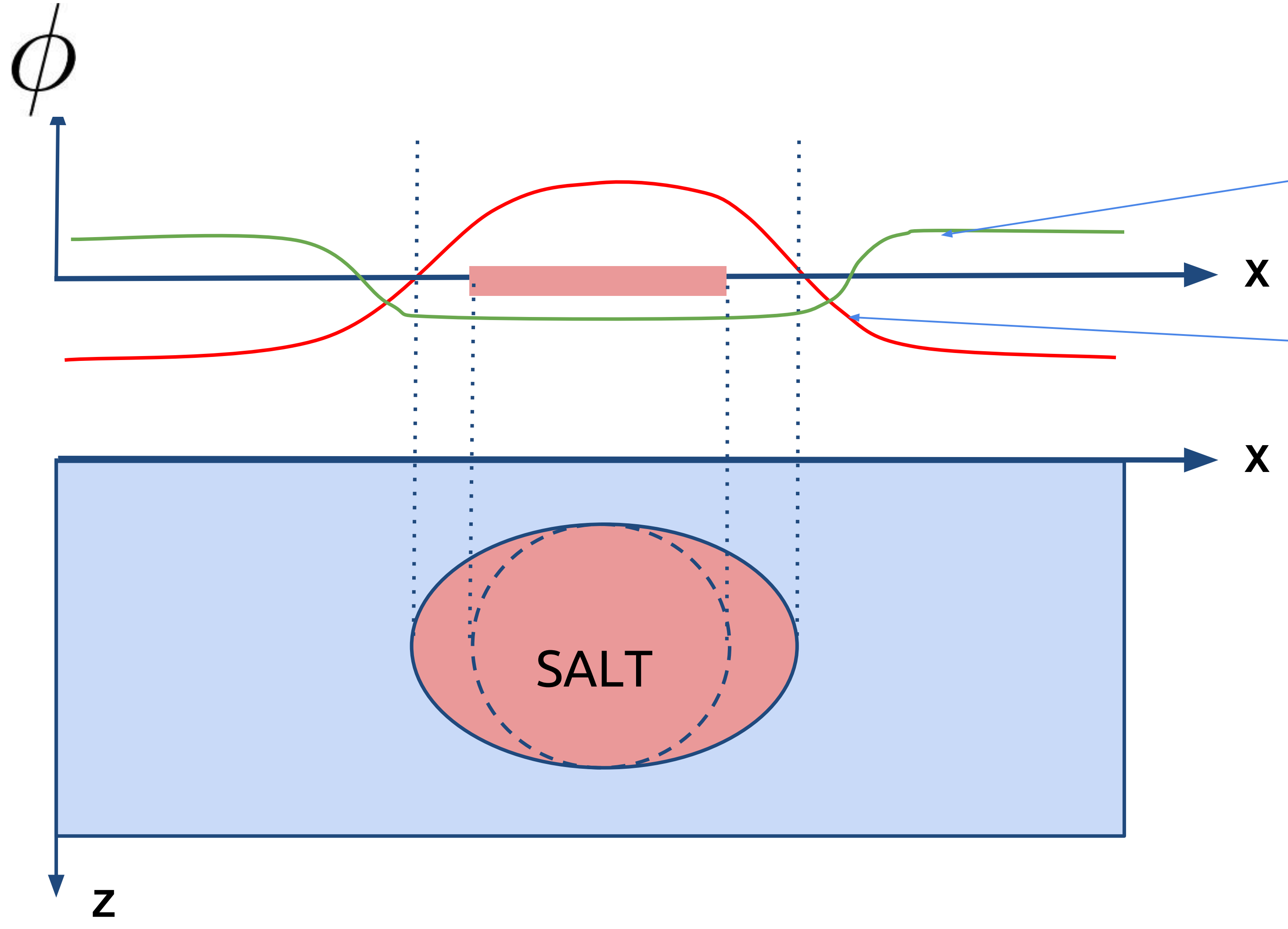
Level set overview



True model extent

Phi surface (profile view)

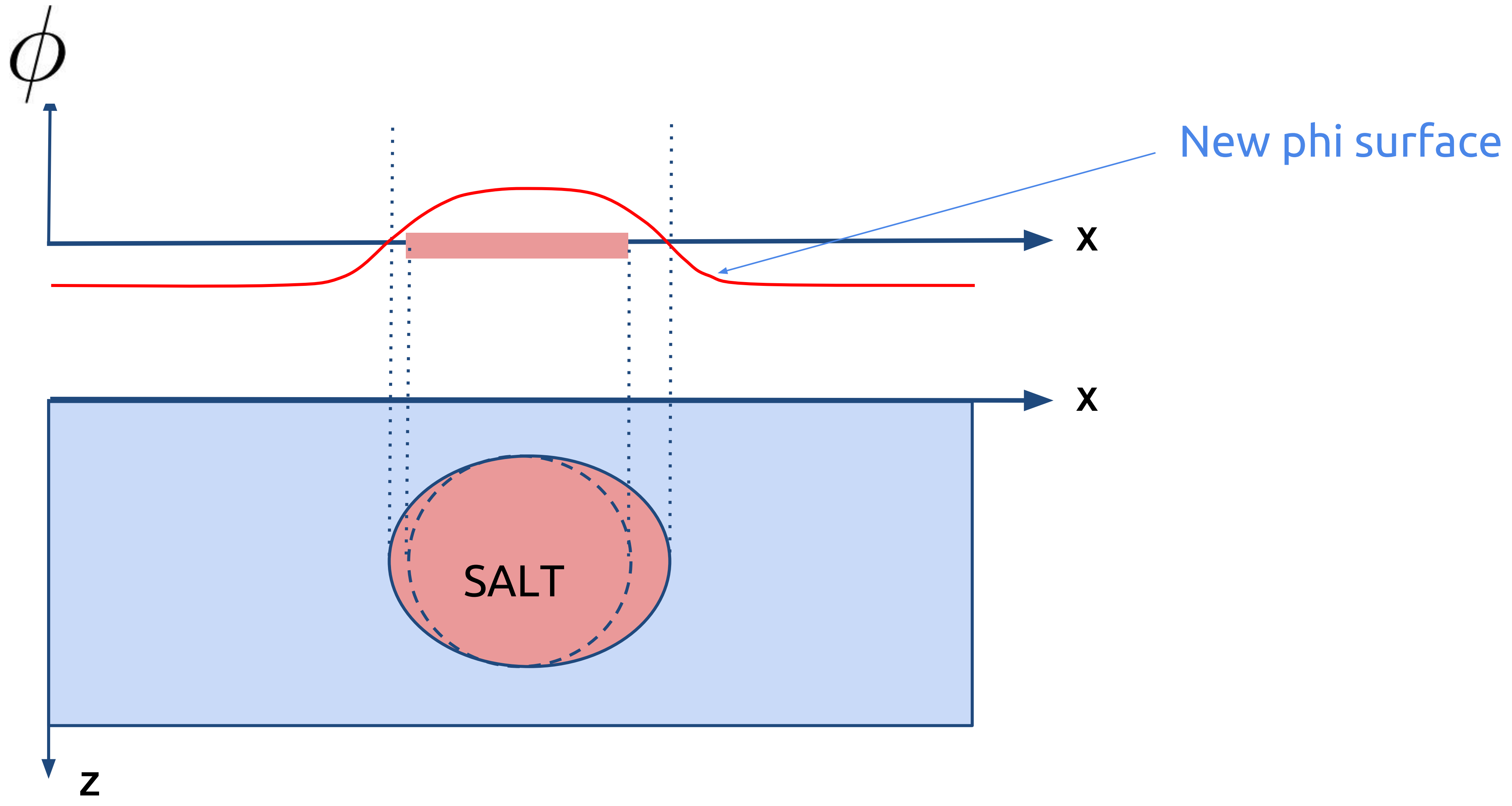
Level set overview



Phi gradient update

Phi surface (profile view)

Level set overview



Basic gradient derivation

$$\phi^{\tau+1} = \phi^{\tau} + \frac{\partial \phi(x_\tau)}{\partial \tau}\delta\tau$$

Basic gradient derivation

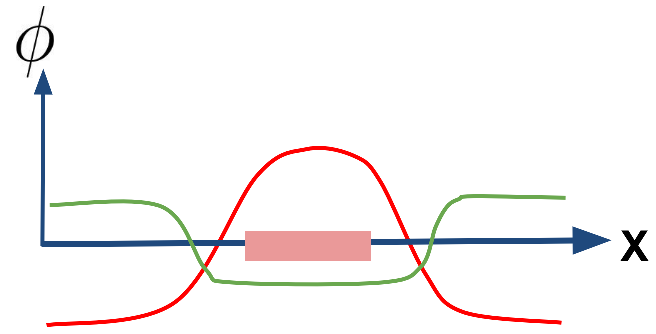
$$\phi^{\tau+1} = \phi^{\tau} + \frac{\partial \phi(x_{\tau})}{\partial \tau}\delta\tau$$

How do we derive this gradient?

Basic gradient derivation

$$\phi(x_T) = 0$$

Boundary represented by level set



Osher, S. and J. A. Sethian, 1988; Burger, M., 2003

Basic gradient derivation

$$\phi(x_T) = 0$$

Boundary represented by level set

$$\frac{\partial \phi(x_\tau)}{\partial \tau} + \frac{\partial \phi(x_\tau)}{\partial x_\tau} \frac{\partial x_\tau}{\partial \tau} = 0$$

CHAIN RULE

Osher, S. and J. A. Sethian, 1988; Burger, M., 2003

+

Basic gradient derivation

$$\phi(x_\tau) = 0$$

Boundary represented by level set

$$\frac{\partial \phi(x_\tau)}{\partial \tau} + \frac{\partial \phi(x_\tau)}{\partial x_\tau} \frac{\partial x_\tau}{\partial \tau} = 0$$

CHAIN RULE

$$\frac{\partial \phi(x_\tau)}{\partial \tau} = - \frac{\partial \phi(x_\tau)}{\partial x_\tau} \frac{\partial x_\tau}{\partial \tau}$$

RE-ARRANGE TERMS

Osher, S. and J. A. Sethian, 1988; Burger, M., 2003

+

Basic gradient derivation

$$\phi^{\tau+1} = \phi^{\tau} + \boxed{\frac{\partial \phi(x_\tau)}{\partial \tau}} \delta \tau$$

EVOLUTION UPDATE EQUATION

$$\boxed{\frac{\partial \phi(x_\tau)}{\partial \tau}} = - \frac{\partial \phi(x_\tau)}{\partial x_\tau} \frac{\partial x_\tau}{\partial \tau}$$

RE-ARRANGE TERMS

Osher, S. and J. A. Sethian, 1988; Burger, M., 2003

Basic gradient derivation

$$\min_{m} \| F(m) - d_{\text{obs}} \|$$

FWI OBJECTIVE FUNCTION

$$\frac{\partial \phi(x_\tau)}{\partial \tau} = - \frac{\partial \phi(x_\tau)}{\partial x_\tau} \frac{\partial x_\tau}{\partial \tau}$$

Osher, S. and J. A. Sethian, 1988; Burger, M., 2003

Basic gradient derivation

$$\frac{\partial \phi(x_\tau)}{\partial \tau} = (m_s - m_b) \sum_k \int_0^T \lambda_k(x,z,t) \frac{\partial^2 u_k(x,z,t)}{\partial^2 t} dt \, |\nabla \phi|$$

$$\frac{\partial \phi(x_\tau)}{\partial \tau} = - \frac{\partial \phi(x_\tau)}{\partial x_\tau} \frac{\partial x_\tau}{\partial \tau}$$

Guo, Z. and M. de Hoop, 2013

Basic gradient derivation

$$\frac{\partial \phi(x_\tau)}{\partial \tau} = (m_s - m_b) \sum_k \int_0^T \lambda_k(x,z,t) \frac{\partial^2 u_k(x,z,t)}{\partial^2 t} dt \, |\nabla \phi|$$

SPATIAL GRADIENT OF PHI

$$\frac{\partial \phi(x_\tau)}{\partial \tau} = -\frac{\partial \phi(x_\tau)}{\partial x_\tau} \frac{\partial x_\tau}{\partial \tau}$$

Guo, Z. and M. de Hoop, 2013

Basic gradient derivation

$$\frac{\partial\phi(x_\tau)}{\partial\tau} = \underbrace{(m_s - m_b)\sum_k \int_0^T \lambda_k(x,z,t)\frac{\partial^2 u_k(x,z,t)}{\partial^2 t}dt}_{\text{SCALAR “SPEED” TERM}}|\nabla\phi|$$

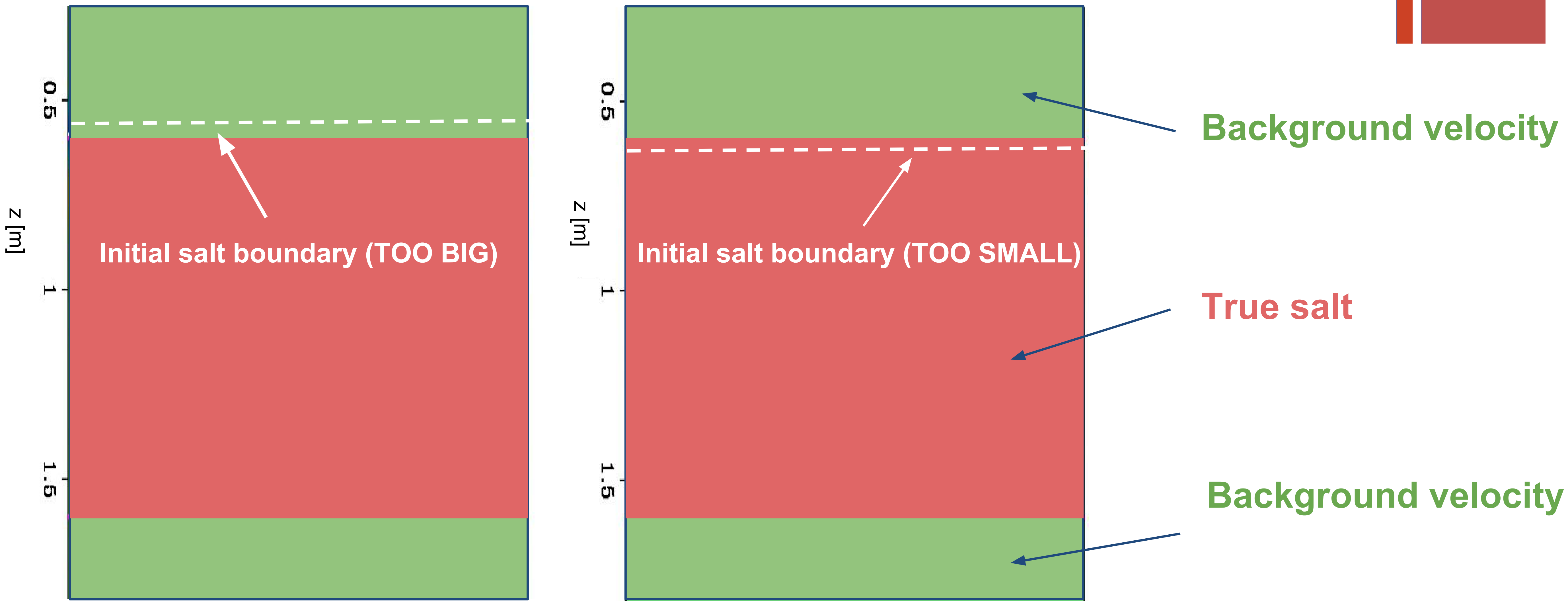
$$\frac{\partial\phi(x_\tau)}{\partial\tau} = -\frac{\partial\phi(x_\tau)}{\partial x_\tau}\boxed{\frac{\partial x_\tau}{\partial\tau}}$$

Basic gradient derivation

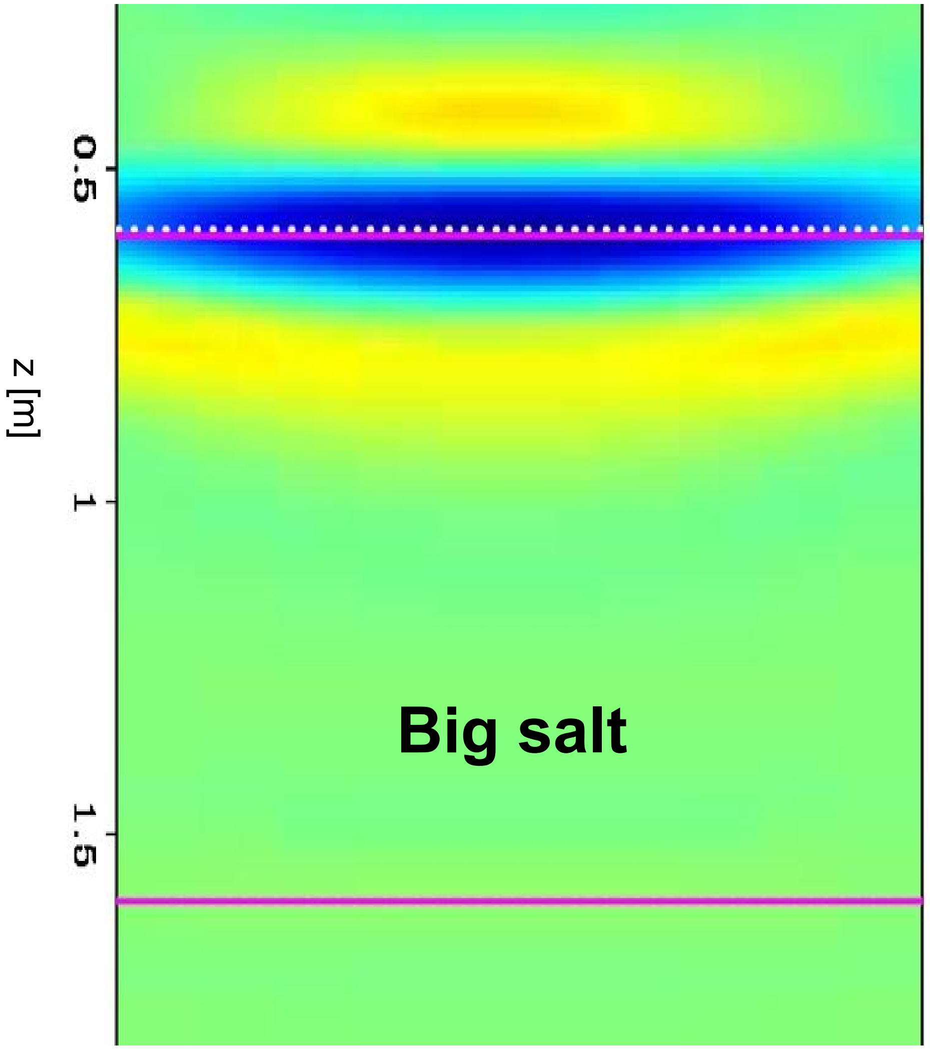
$$\frac{\partial \phi(x_\tau)}{\partial \tau} = (m_s - m_b) \sum_k \int_0^T \lambda_k(x,z,t) \frac{\partial^2 u_k(x,z,t)}{\partial^2 t} dt |\nabla \phi|$$

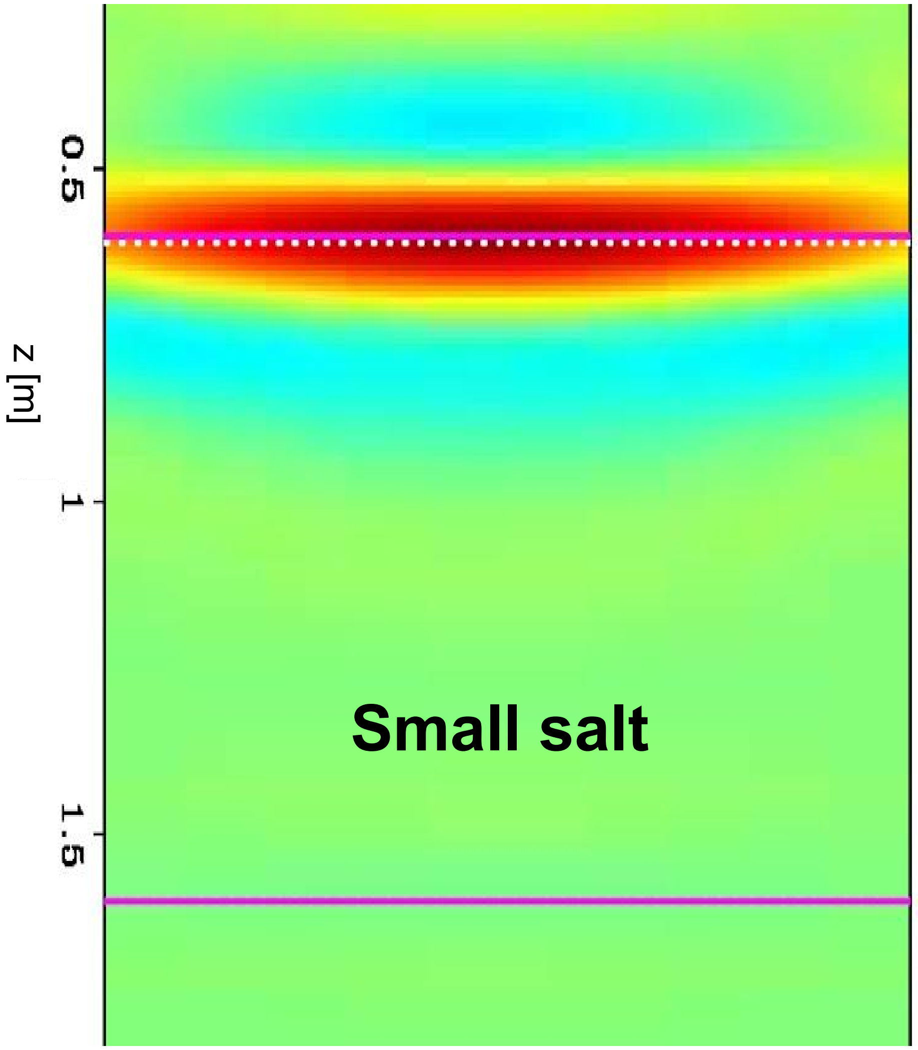
Back-propagated residual (RTM image)

Relationship to RTM image


0.5
1
1.5
z [m]
Initial salt boundary (TOO BIG)
Initial salt boundary (TOO SMALL)
Background velocity
True salt
Background velocity

Relationship to RTM image





Red = Increase velocity

Blue = Decrease velocity

Relationship to RTM image

BOUNDARY GRADIENT

$$\frac{\partial \phi(x_\tau)}{\partial \tau} = (m_s - m_b)\boxed{\sum_k \int_0^T \lambda_k(x,z,t)\frac{\partial^2 u_k(x,z,t)}{\partial^2 t}dt}\,|\nabla \phi|$$

Relationship to RTM image

Positive when salt faster than background

BOUNDARY GRADIENT

$$\frac{\partial \phi(x_\tau)}{\partial \tau} = (m_s - m_b) \sum_k \int_0^T \lambda_k(x,z,t) \frac{\partial^2 u_k(x,z,t)}{\partial^2 t} dt \, |\nabla \phi|$$

Relationship to RTM image

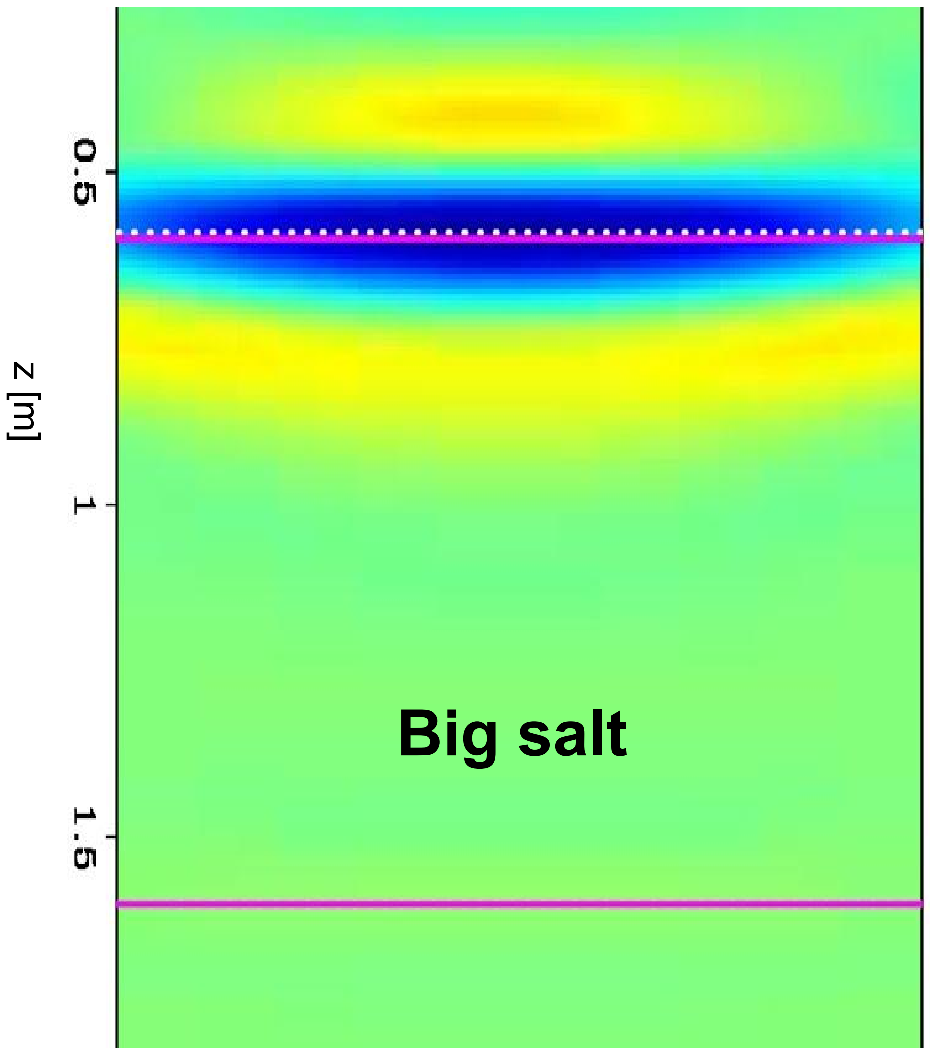
Positive when salt faster than background

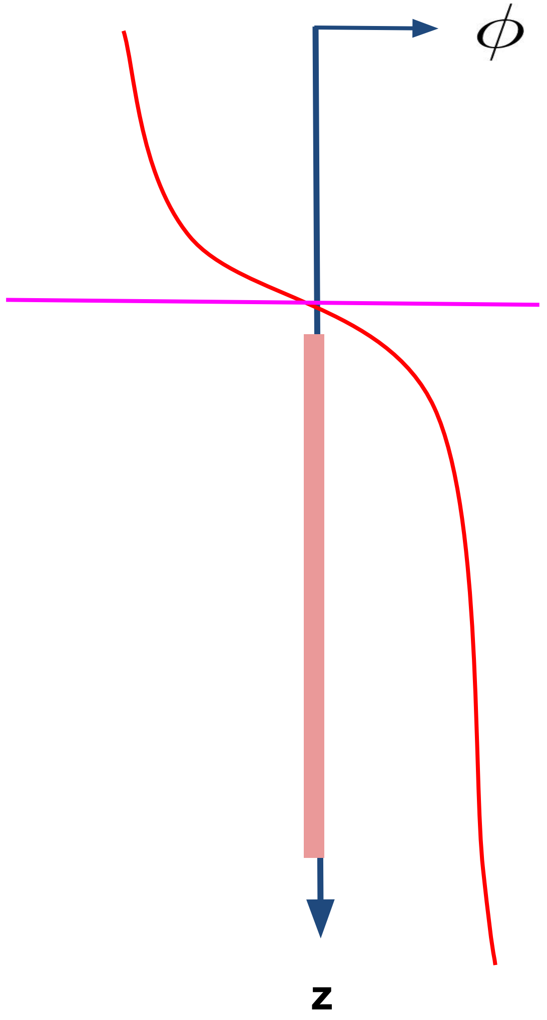
BOUNDARY GRADIENT

$$\frac{\partial \phi(x_\tau)}{\partial \tau} = (m_s - m_b) \sum_k \int_0^T \lambda_k(x, z, t) \frac{\partial^2 u_k(x, z, t)}{\partial^2 t} dt \, |\nabla \phi|$$

Always positive, regularization keeps gradient to ~1.0

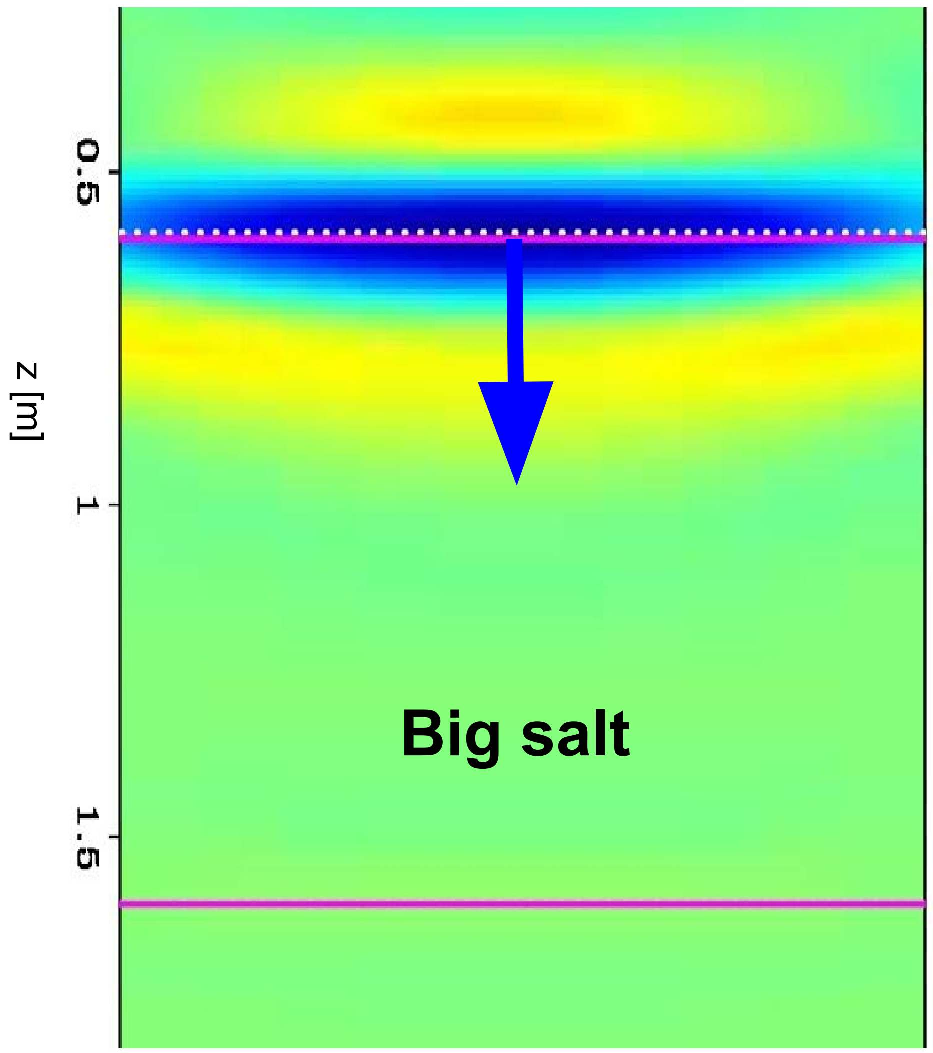
Relationship to RTM image

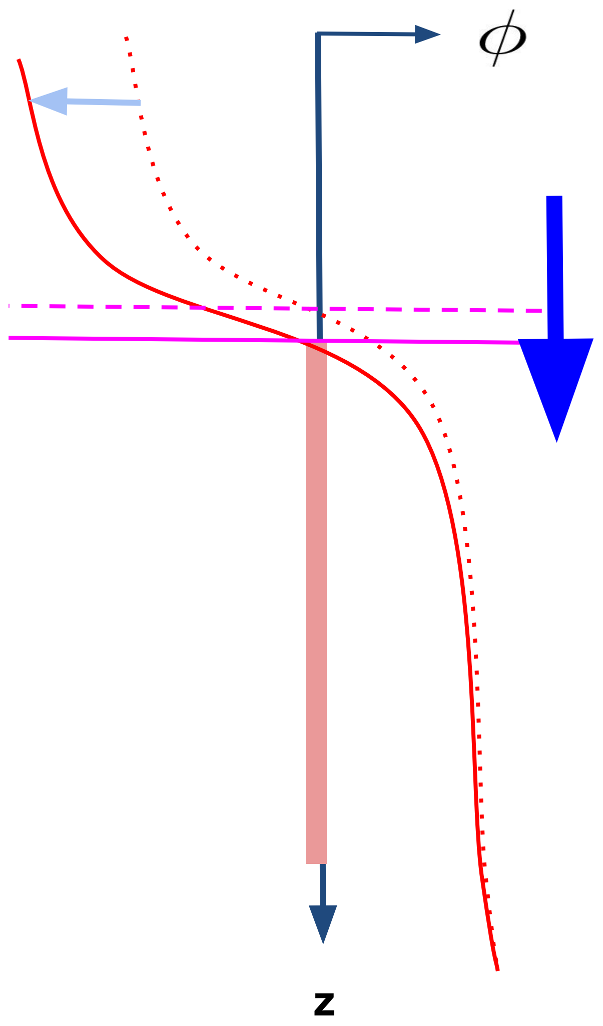




Blue = Decrease in ϕ

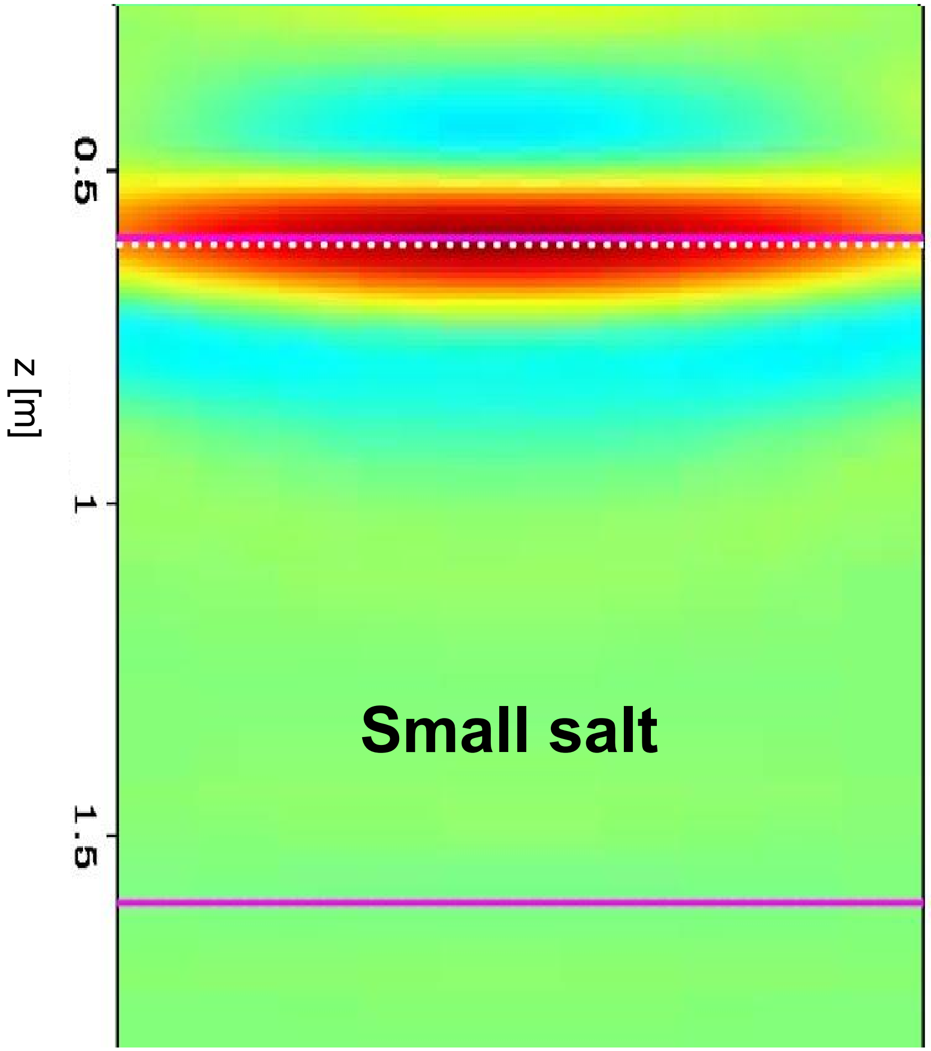
Relationship to RTM image

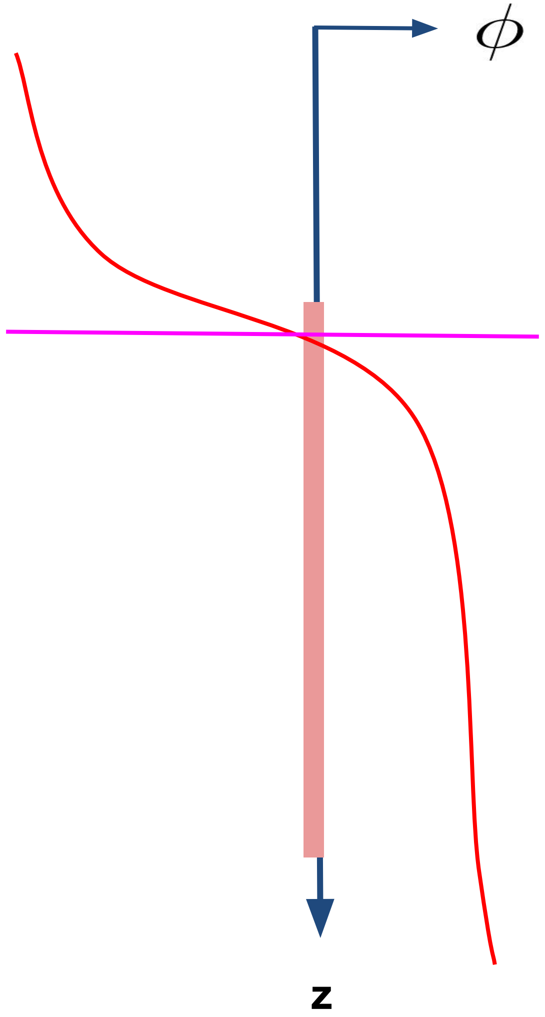




Blue = Inward salt boundary movement

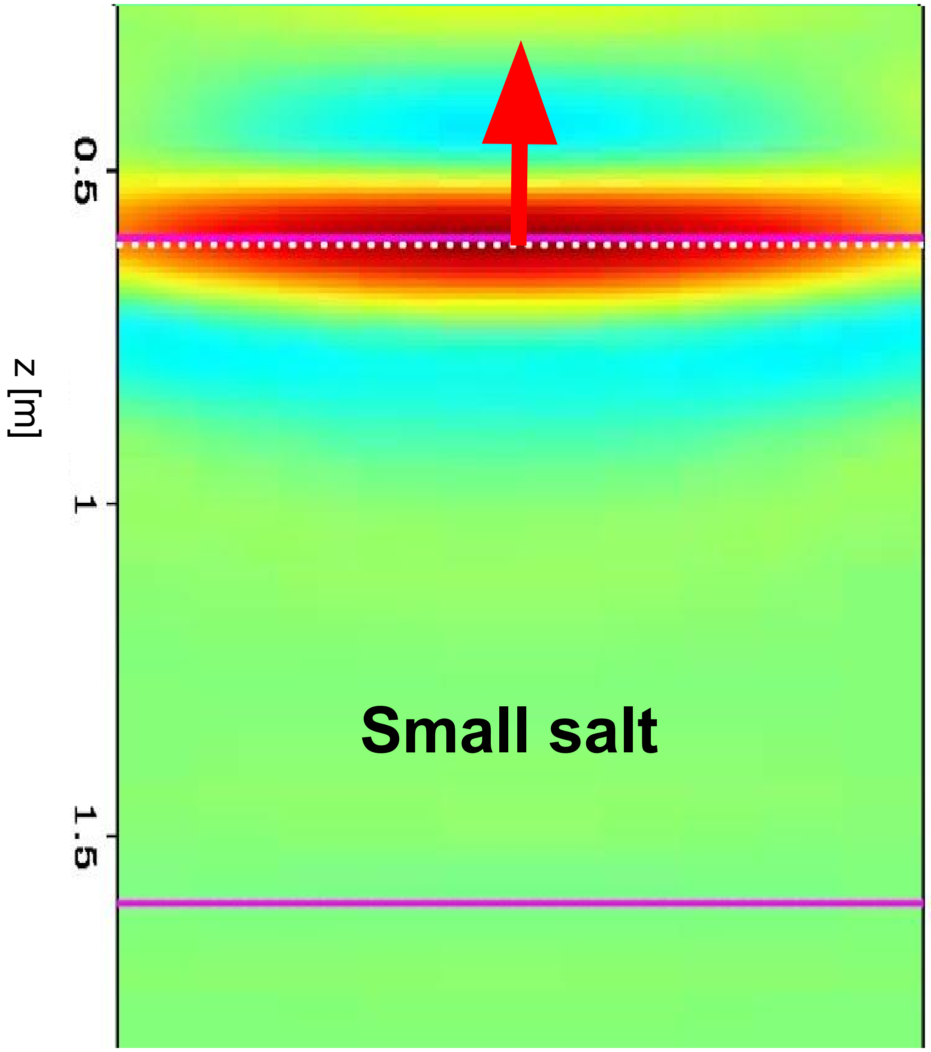
Relationship to RTM image

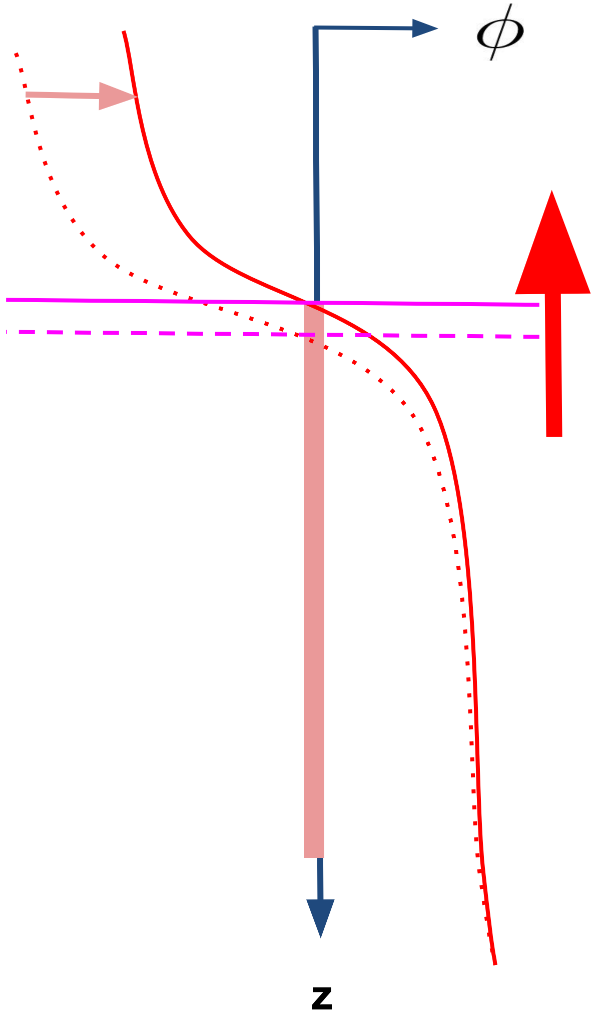




Red = Increase in ϕ

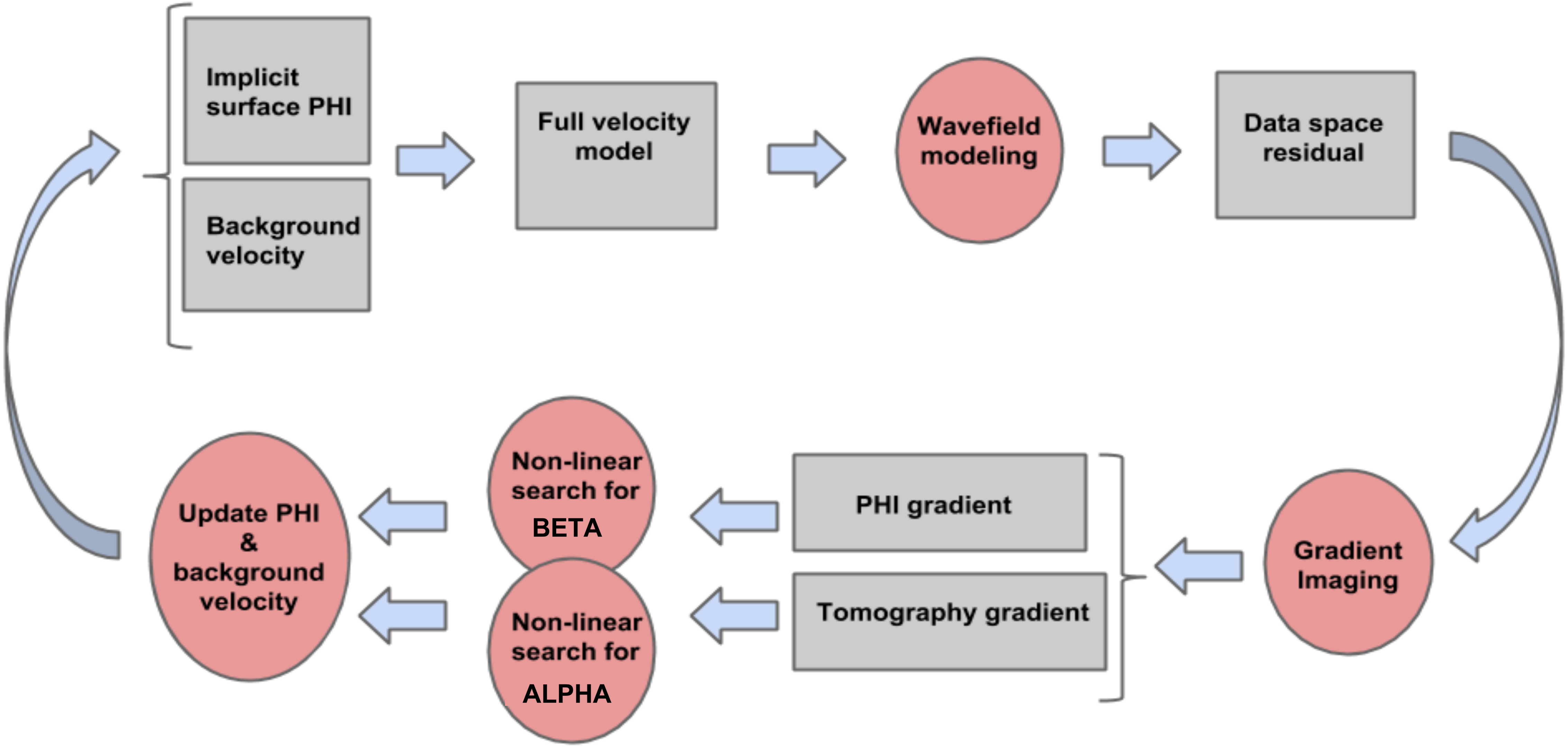
Relationship to RTM image



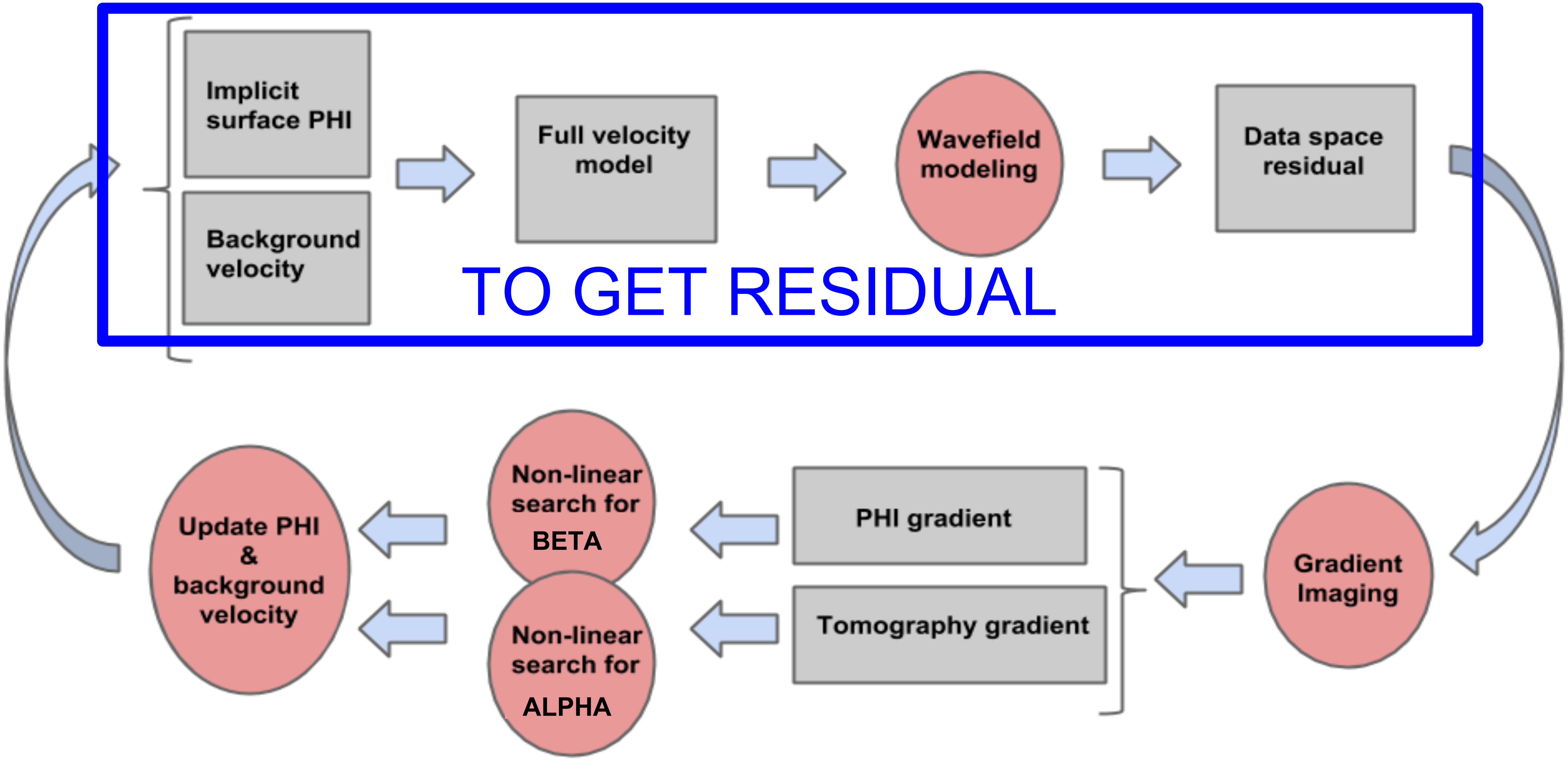


Red = Outward salt boundary movement

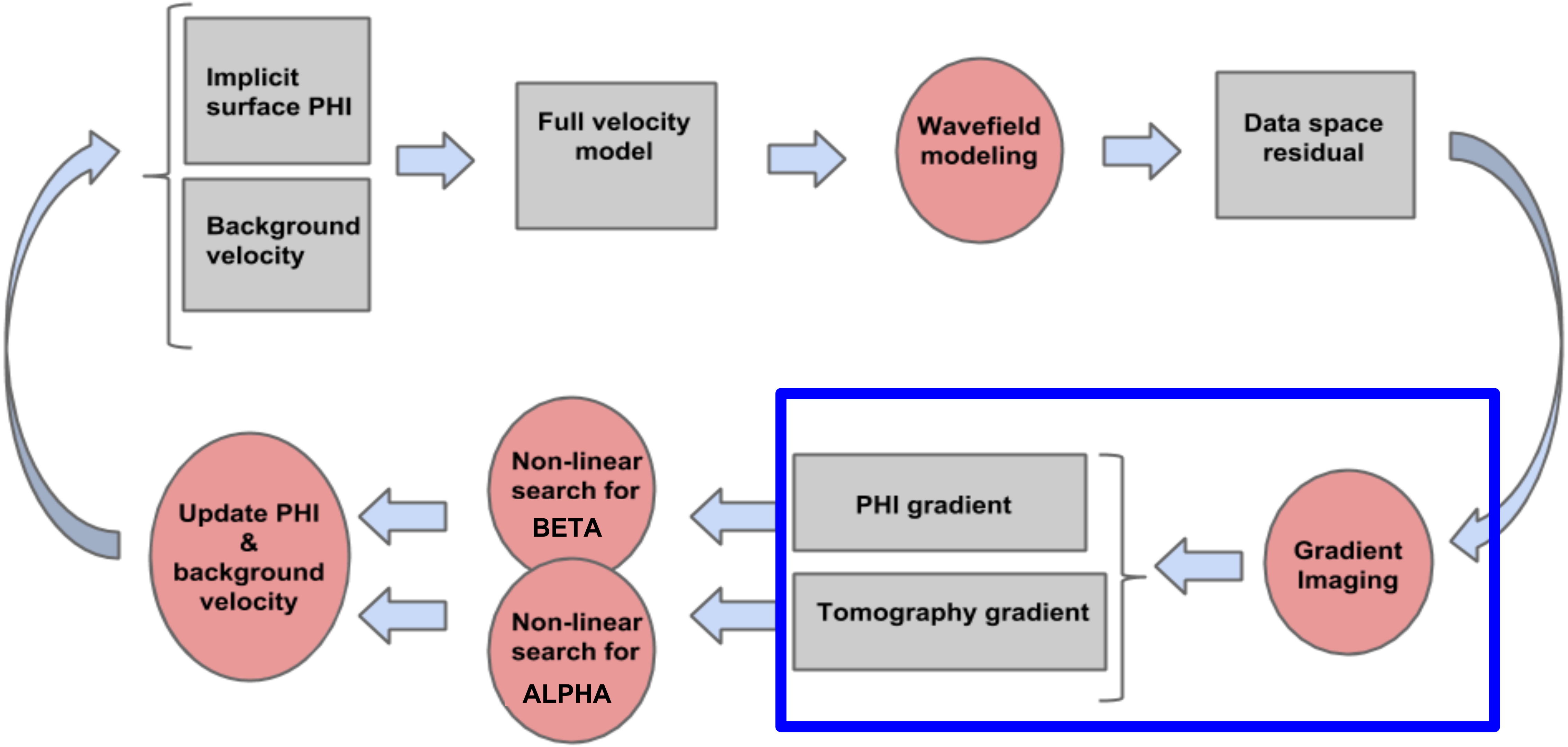
Algorithm workflow



Algorithm workflow



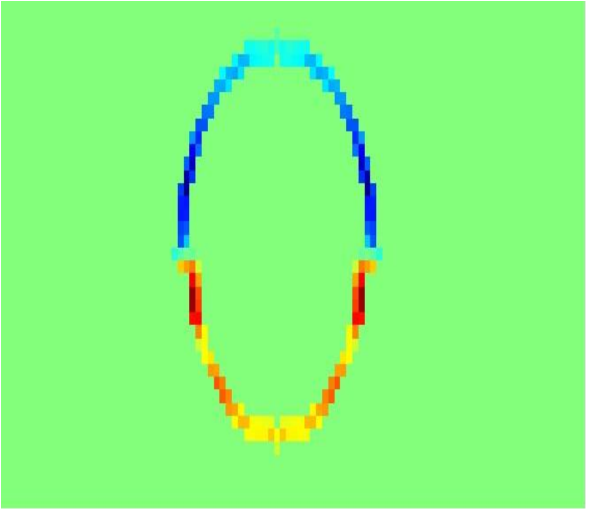
Algorithm workflow



+

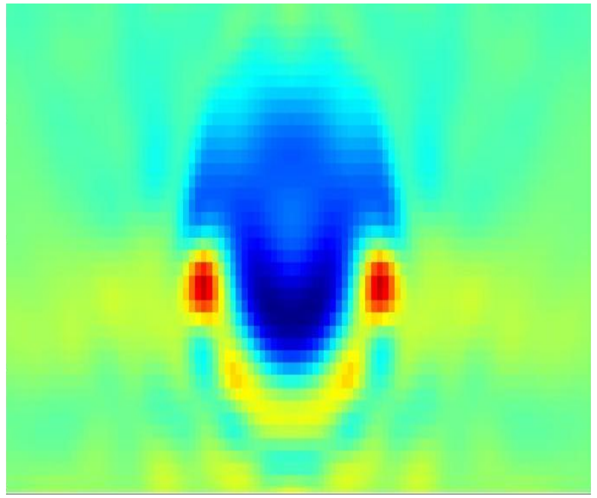
Calculate gradients

Salt Boundary



$$\frac{\partial \phi}{\partial \tau} = (m_s - m_b) \sum_k \int_0^T \lambda_k(x,z,t) \frac{\partial^2 u_k(x,z,t)}{\partial t^2} dt \, |\nabla \phi|$$

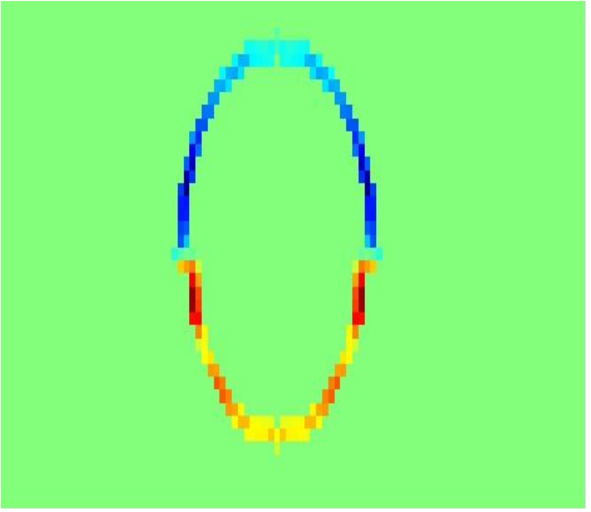
Background velocity



$$\frac{\partial V_{back}}{\partial \tau} = \sum_k \int_0^T \lambda_k(x,z,t) \frac{\partial^2 u_k(x,z,t)}{\partial t^2} dt$$

+ Calculate gradients

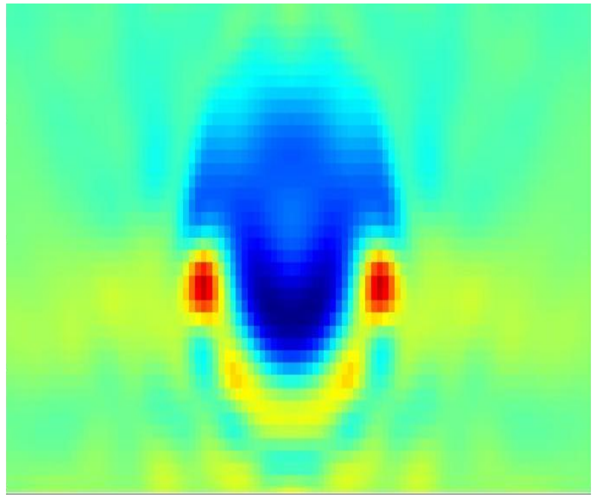
Salt Boundary



$$\frac{\partial \phi}{\partial \tau} = (m_s - m_b) \underbrace{\sum_k \int_0^T \lambda_k(x,z,t) \frac{\partial^2 u_k(x,z,t)}{\partial t^2} dt}\, |\nabla \phi|$$

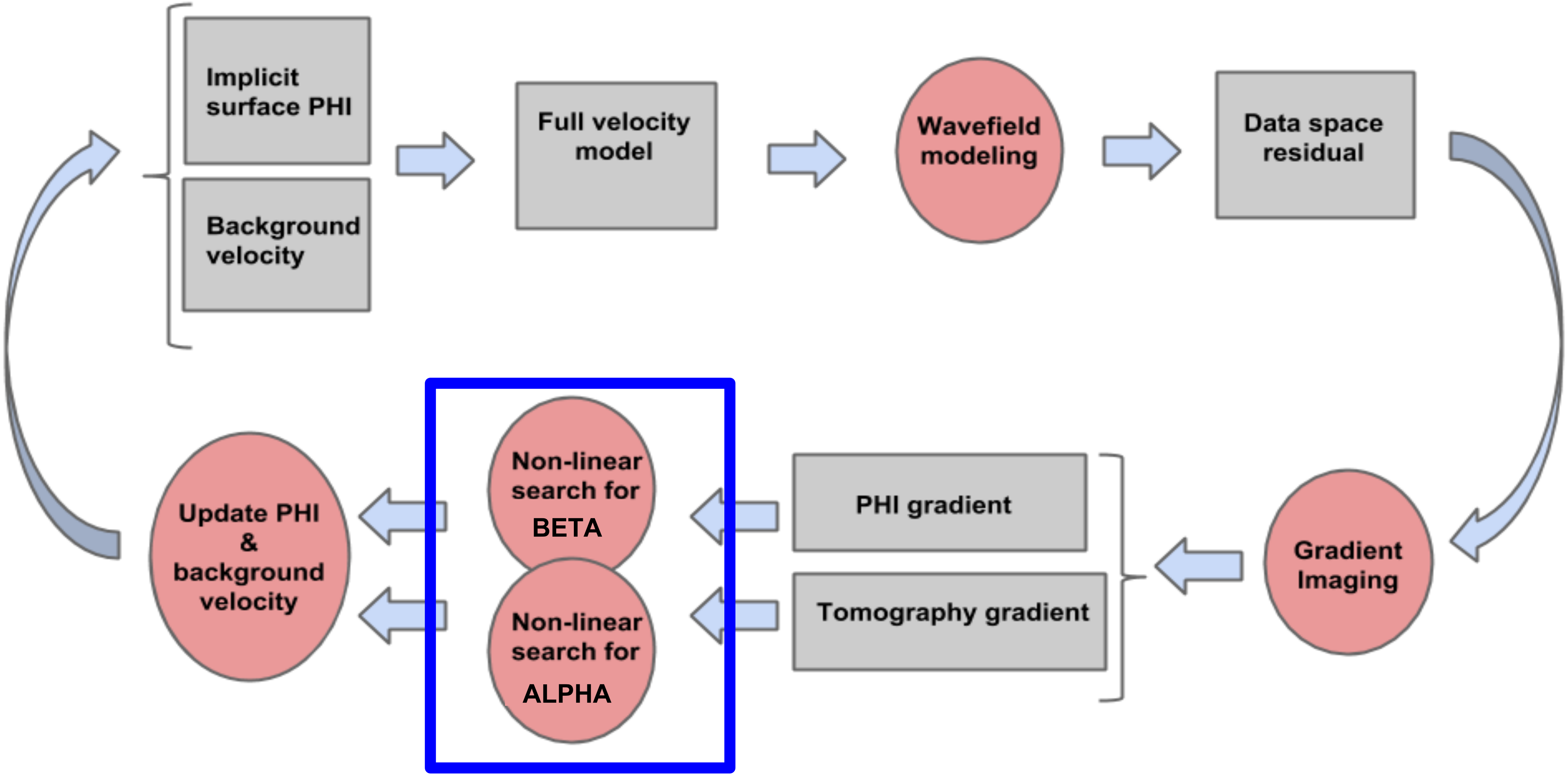
Adjoint linearized Born operator (RTM)

Background velocity



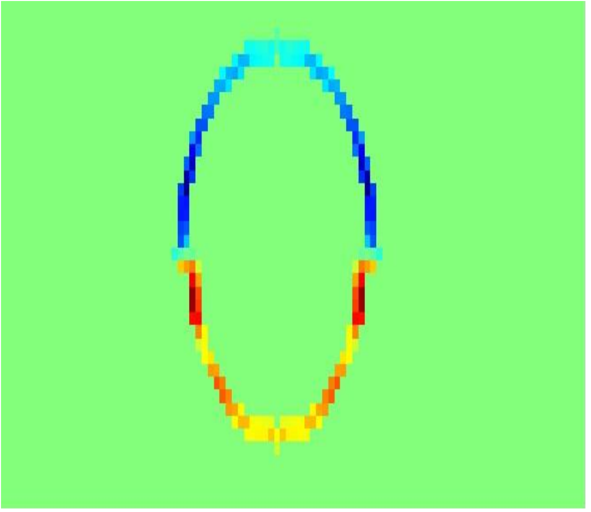
$$\frac{\partial V_{back}}{\partial \tau} = \overbrace{\sum_k \int_0^T \lambda_k(x,z,t) \frac{\partial^2 u_k(x,z,t)}{\partial t^2} dt}$$

Algorithm workflow



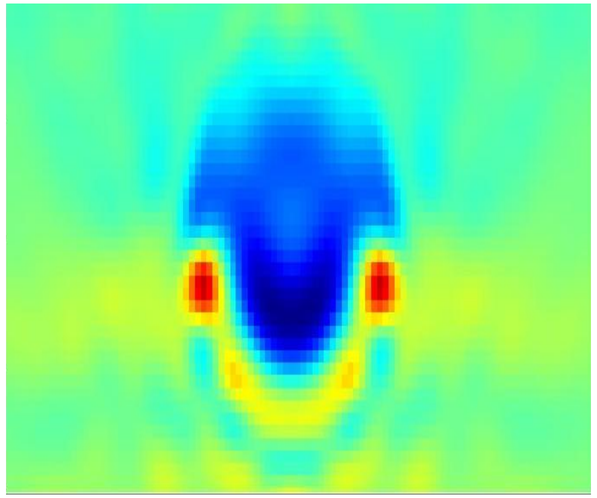
+ Non-linear search for gamma

Salt Boundary β



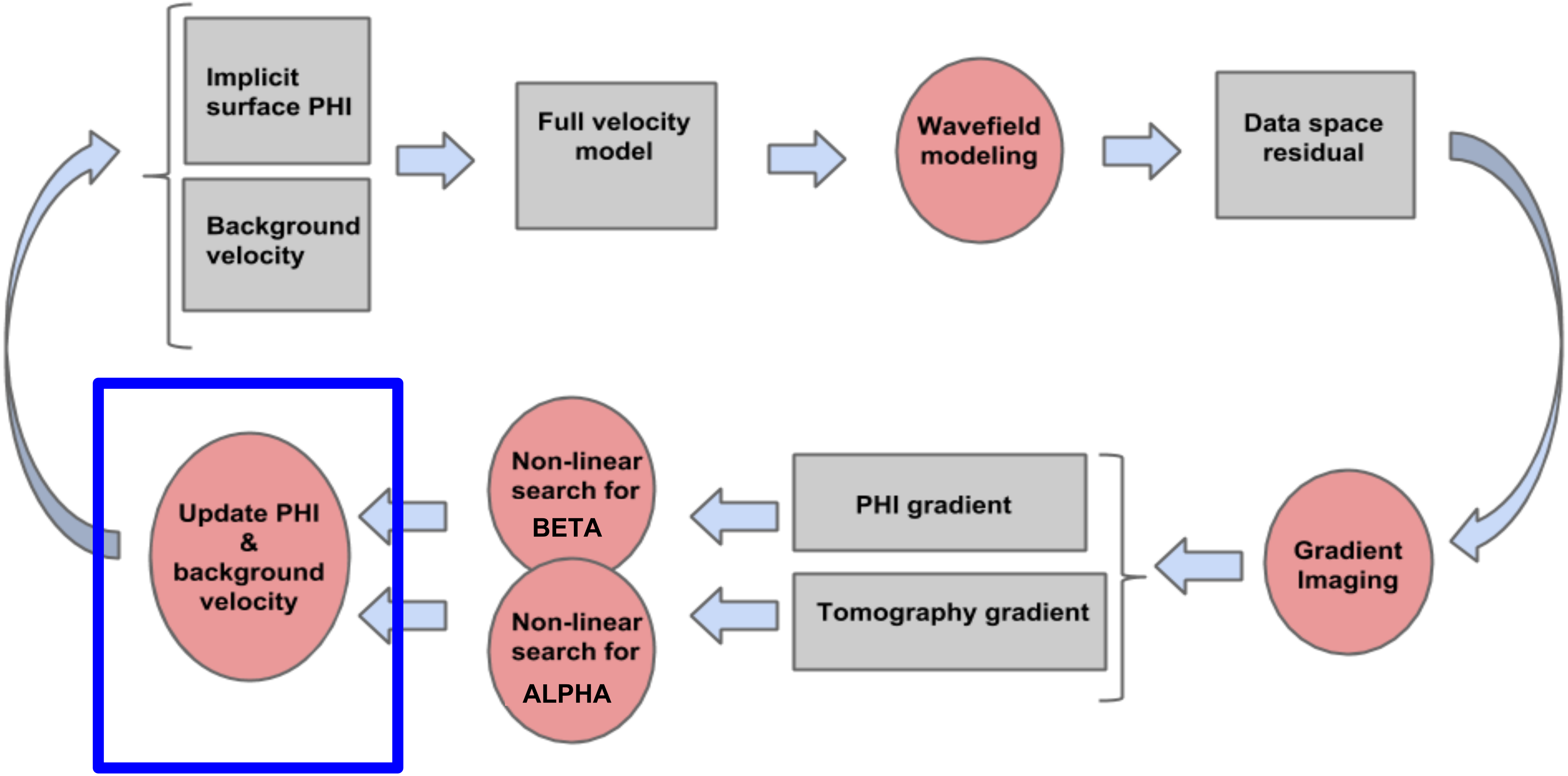
$$\min_{\beta} \|F(m(\beta)) - d\|$$

Background velocity α



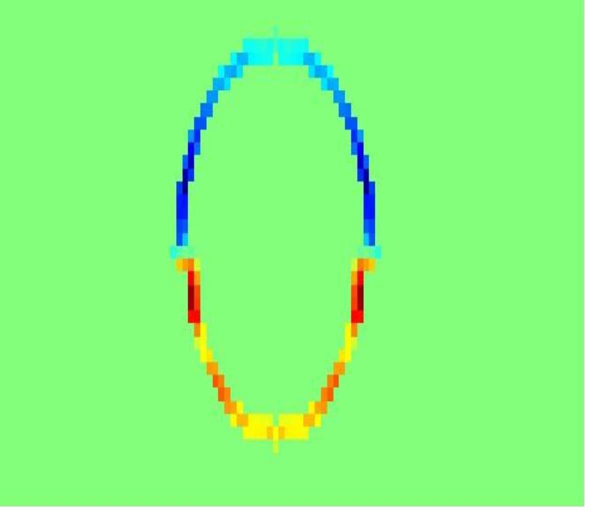
$$\min_{\alpha} \|F(m(\alpha)) - d\|$$

Algorithm workflow

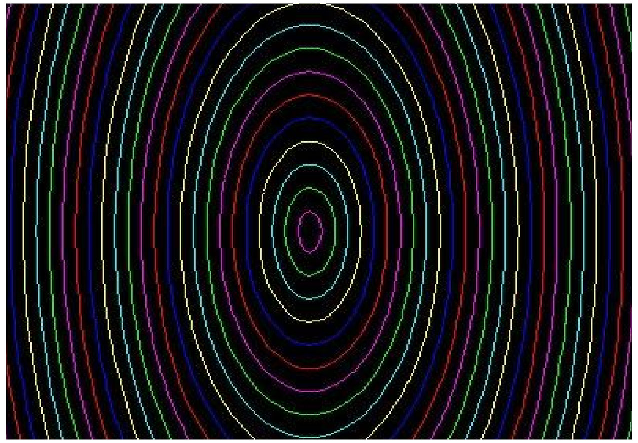


Apply scaling and update fields

Salt Boundary β

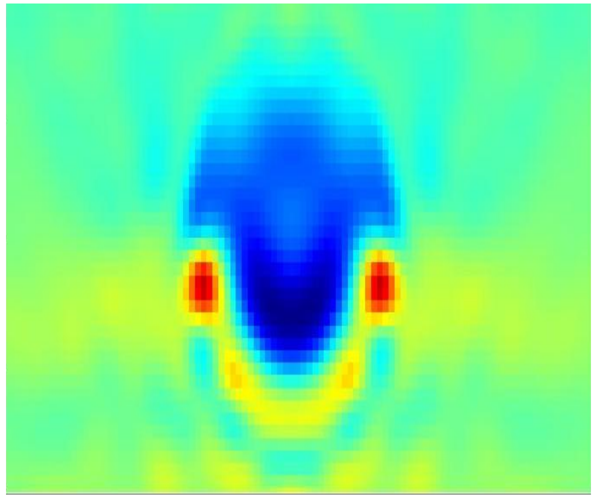


+

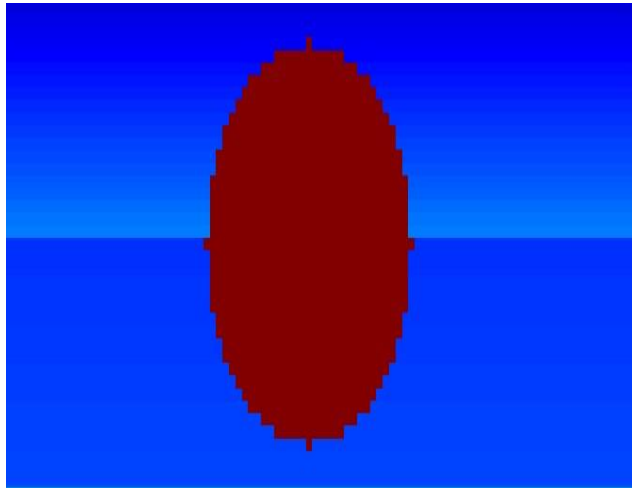


Implicit surface phi

Background velocity α



+



Background velocity model

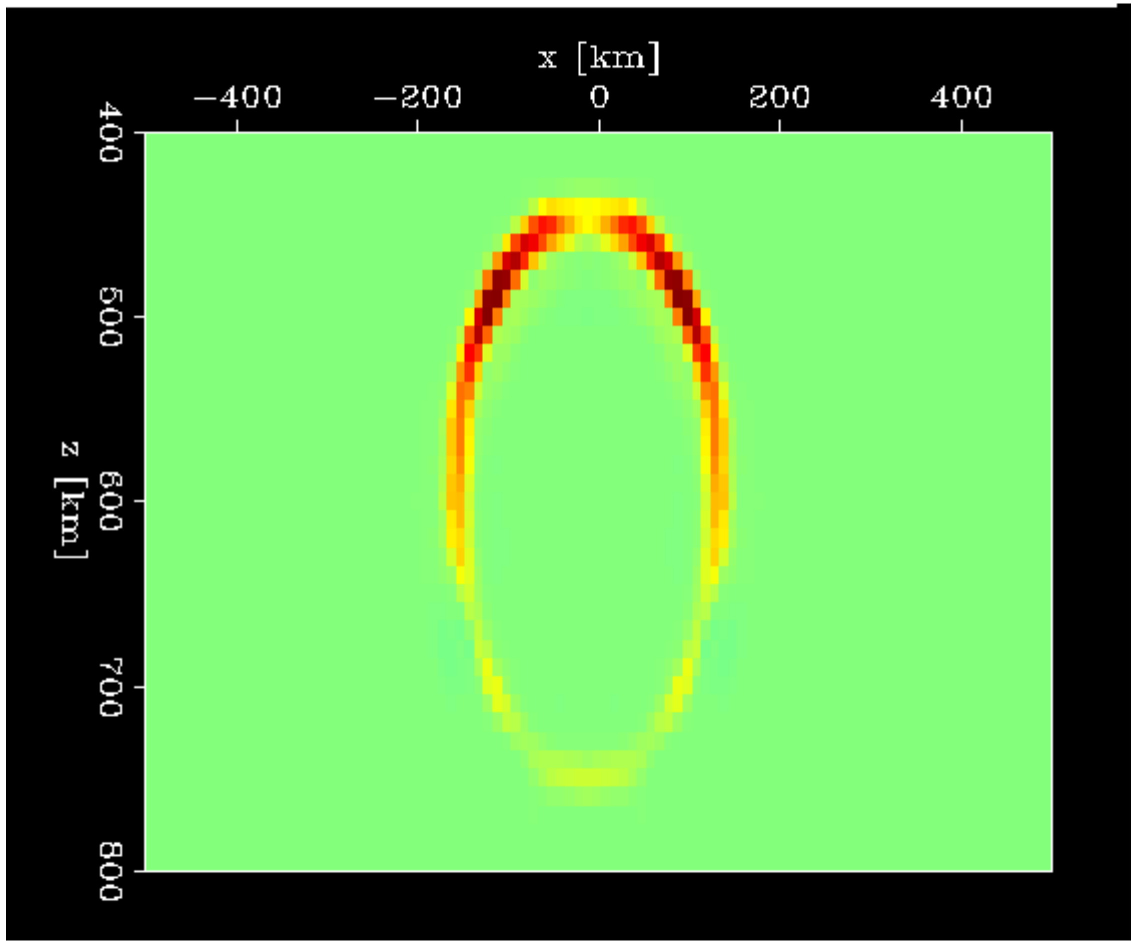
+

Outline

- What do you mean by domain decomposition?
  - How does it work?
  - Demonstration of method on various models

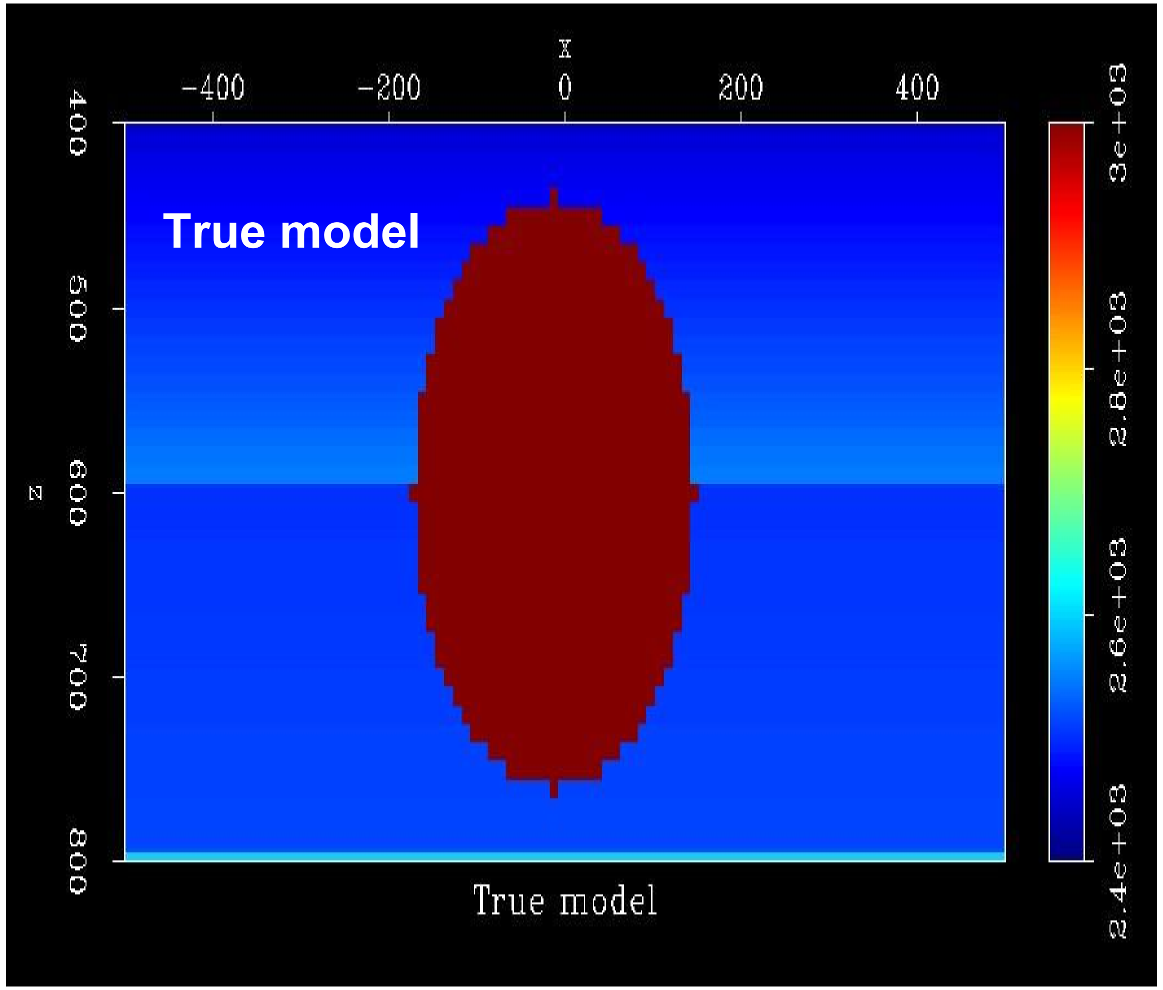
Why domain decomposition?

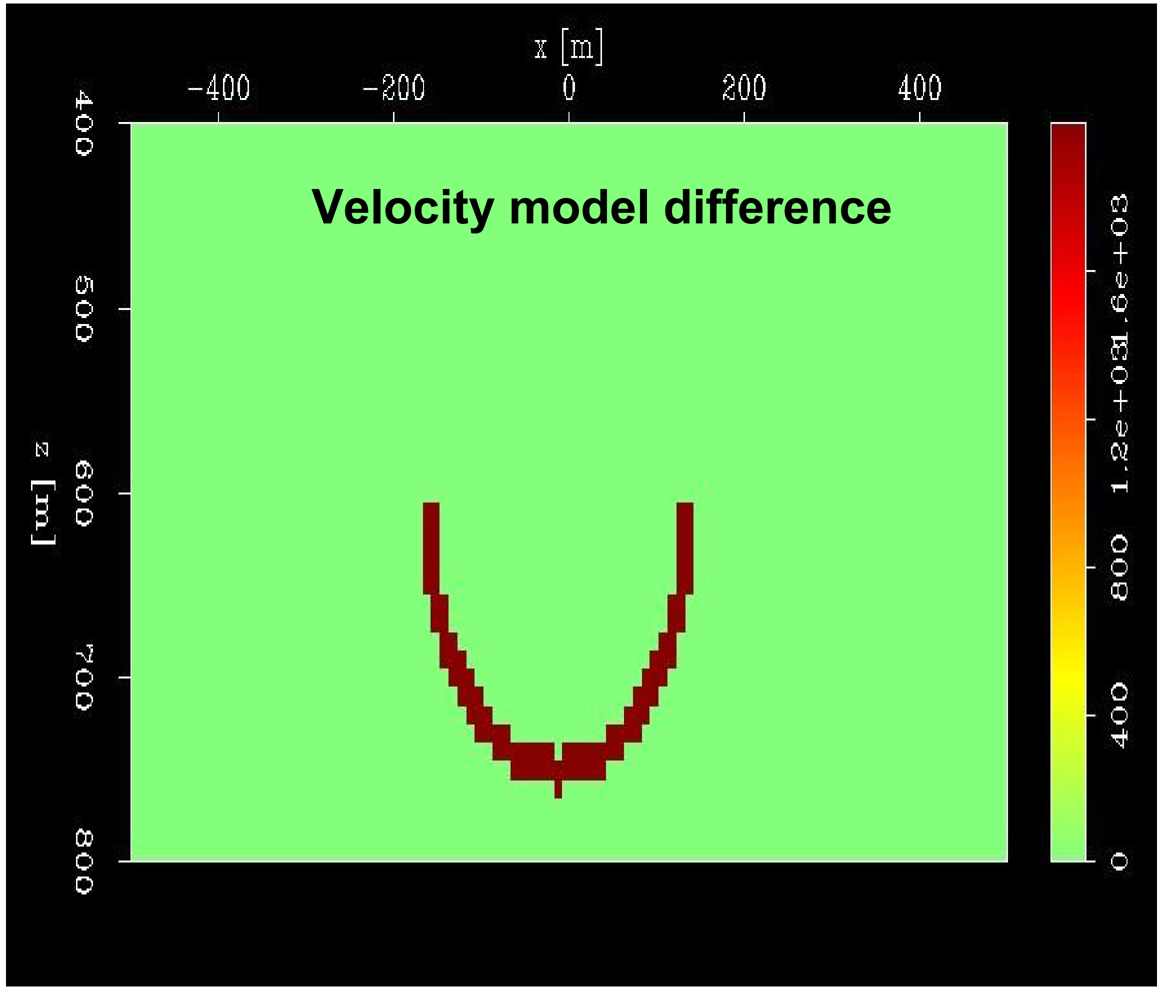
- Top of salt (TOS) dominates the salt updating
- Shows up as strongest reflector
- Non-linear step search prefers to correct TOS
- Leaves BOS under-corrected



Example of salt boundary update gradient

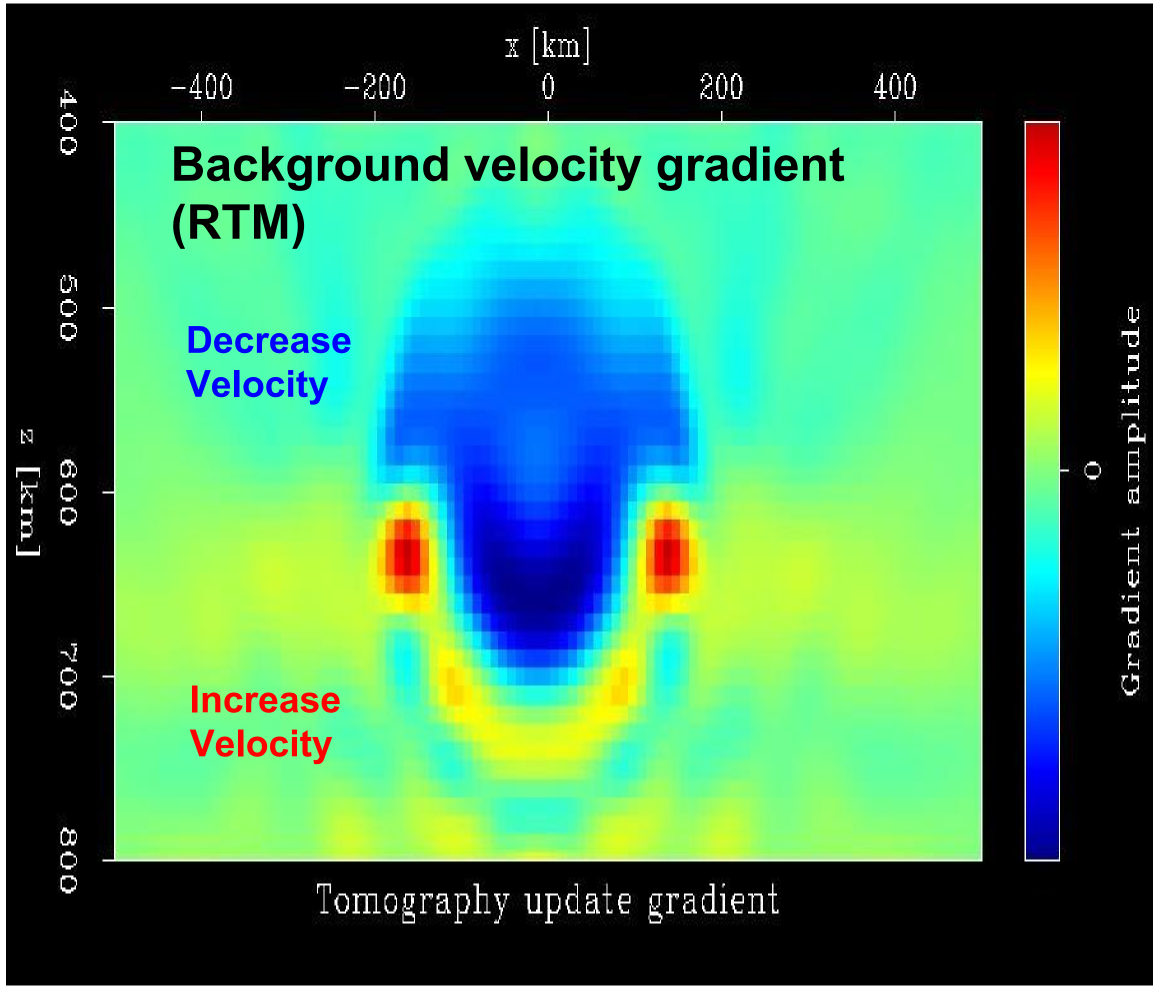
Salt update example

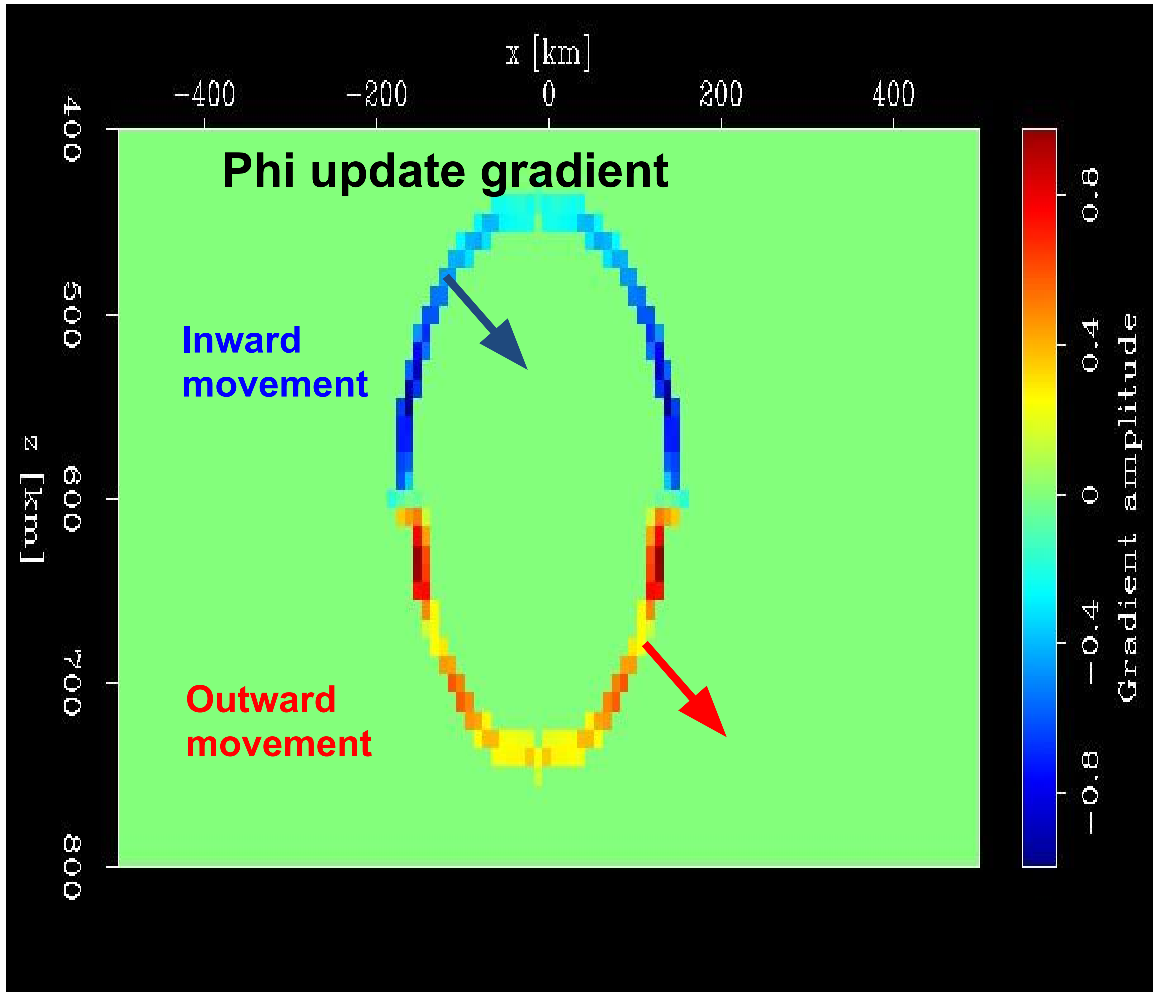




Perfect top of salt (TOS) model

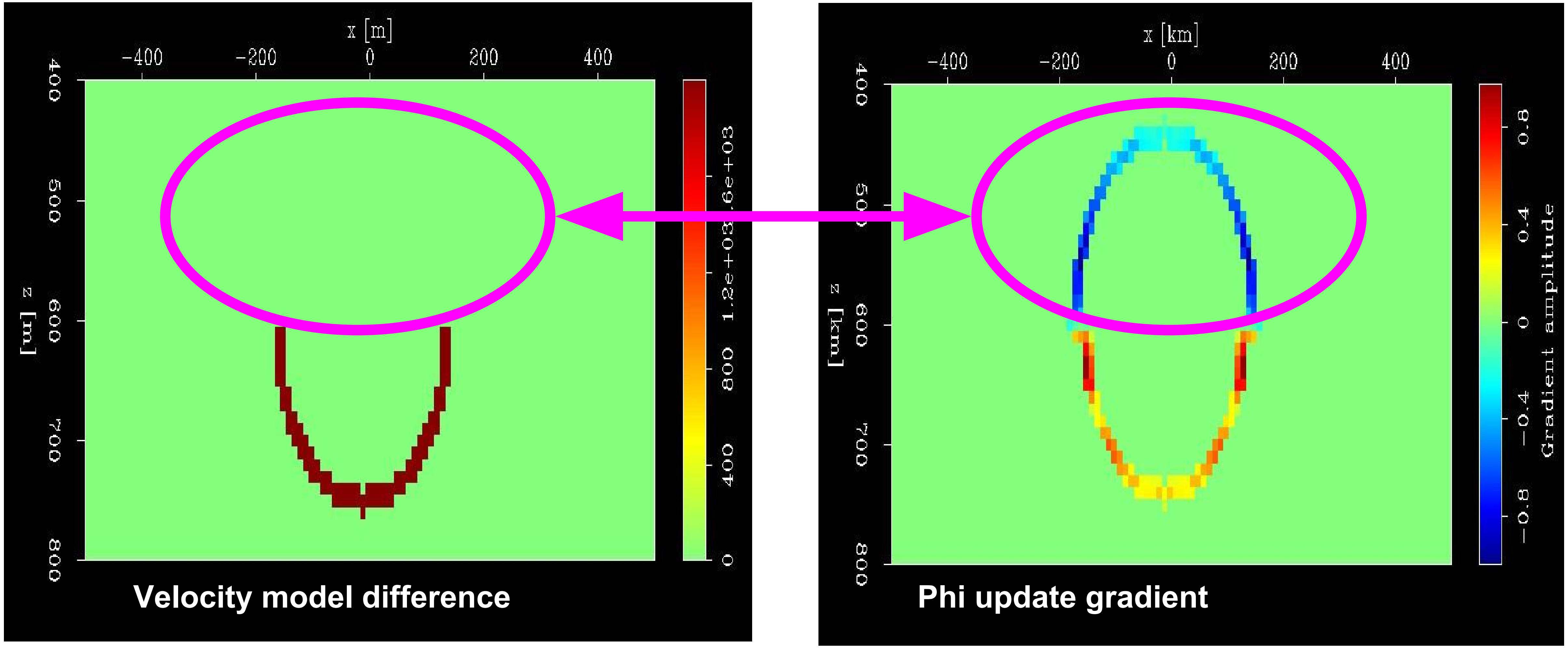
Salt update example





Gradients at first iteration for model with perfect TOS

Salt update example



Velocity model difference

Phi update gradient

No update necessary!

+

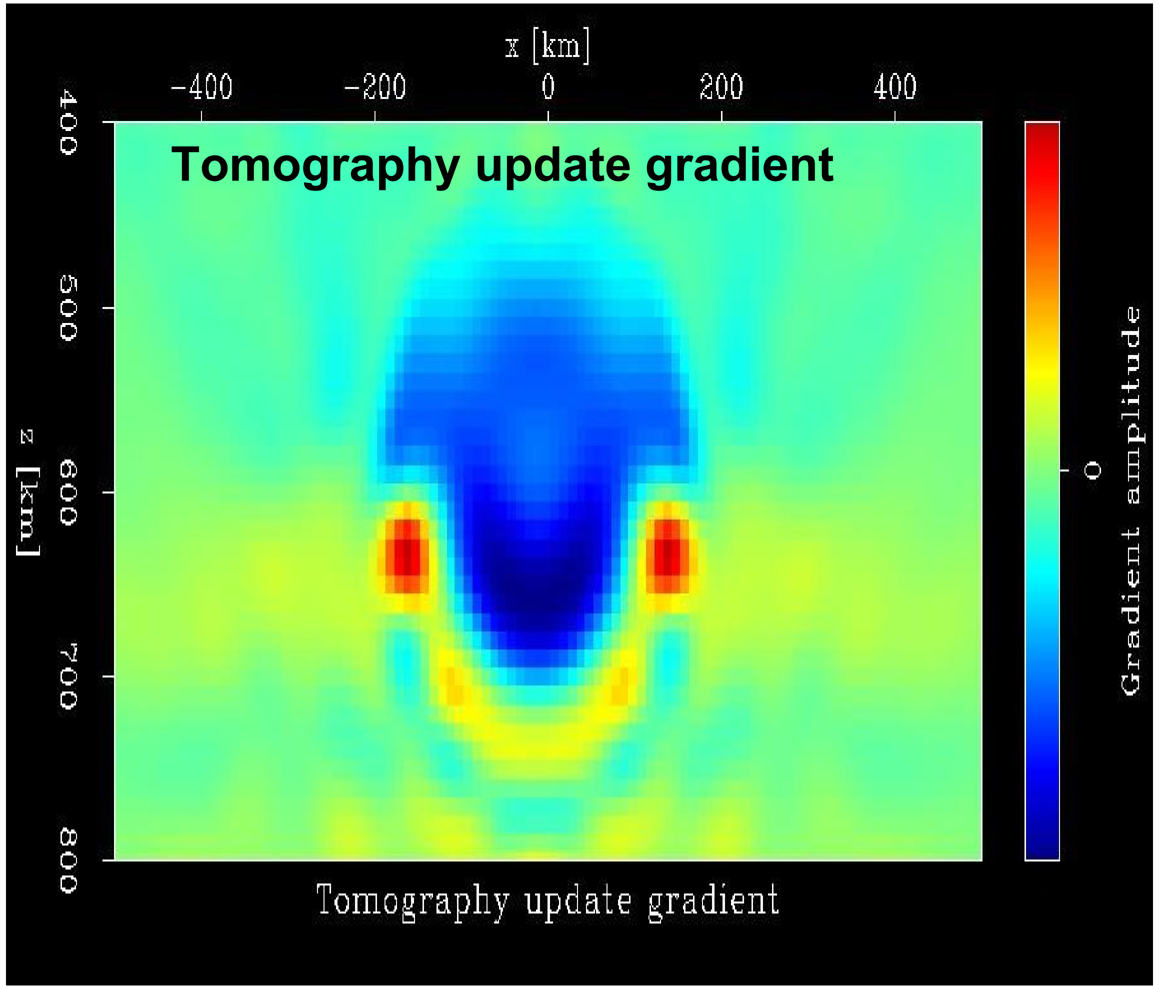
Fundamental problem

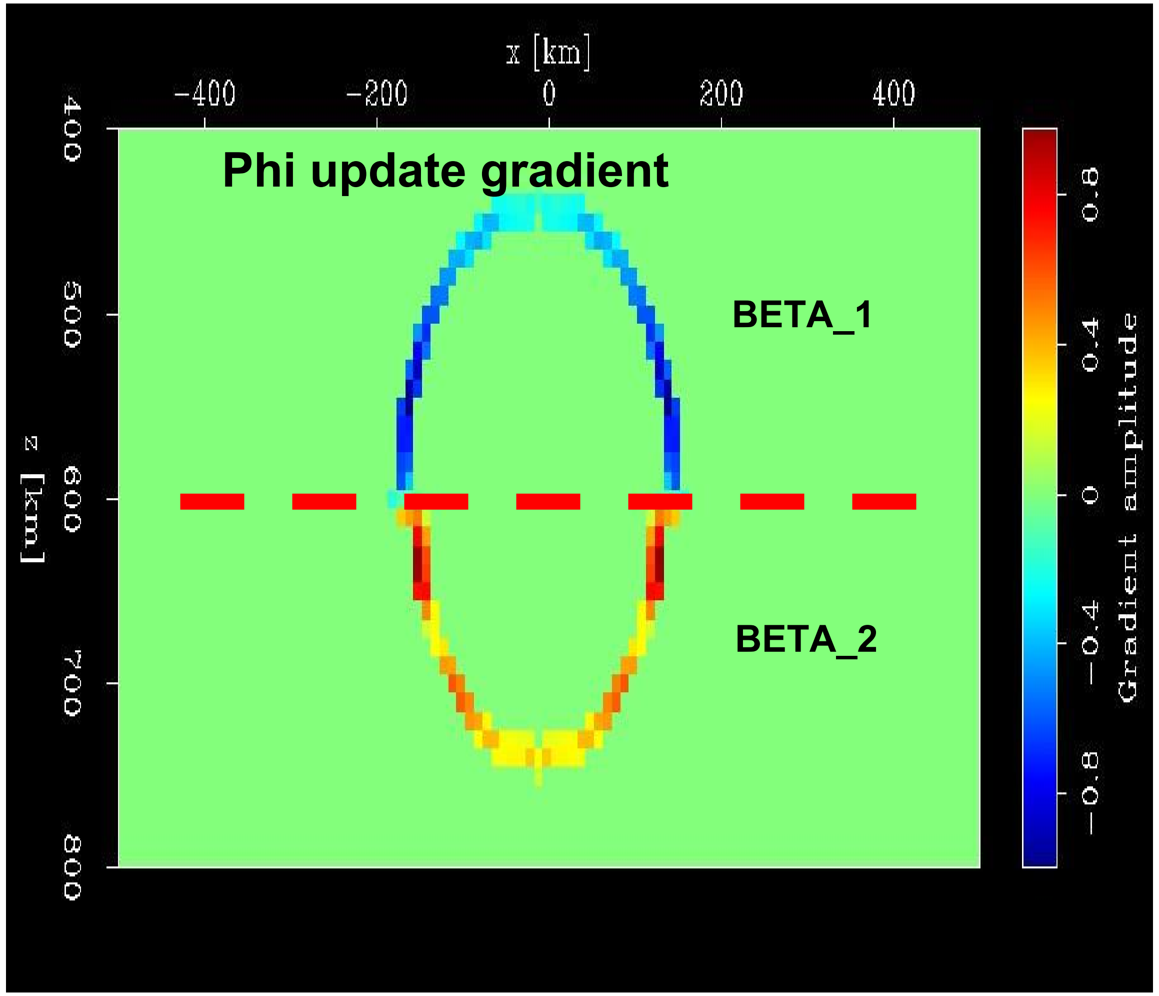
- Background velocity gradient will have ‘wrong’ update at salt boundary for perfect TOS model.

- This is because RTM imaging cannot discern between velocity and reflector position errors, so it tries to correct both.

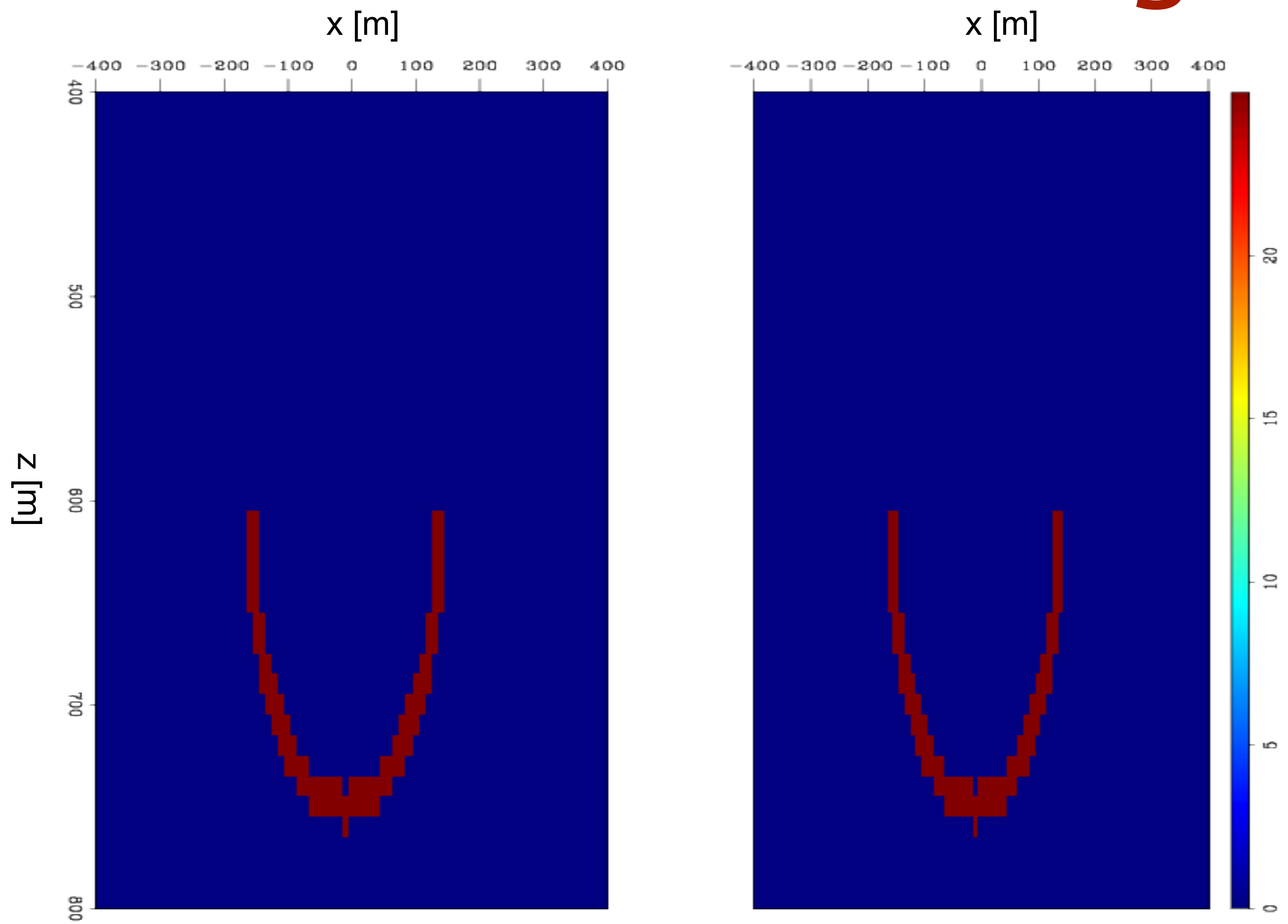
- Boundary update gradient based on RTM image, so it inherits “wrong” update.

What if we split the top/bottom?





Does it ever converge?

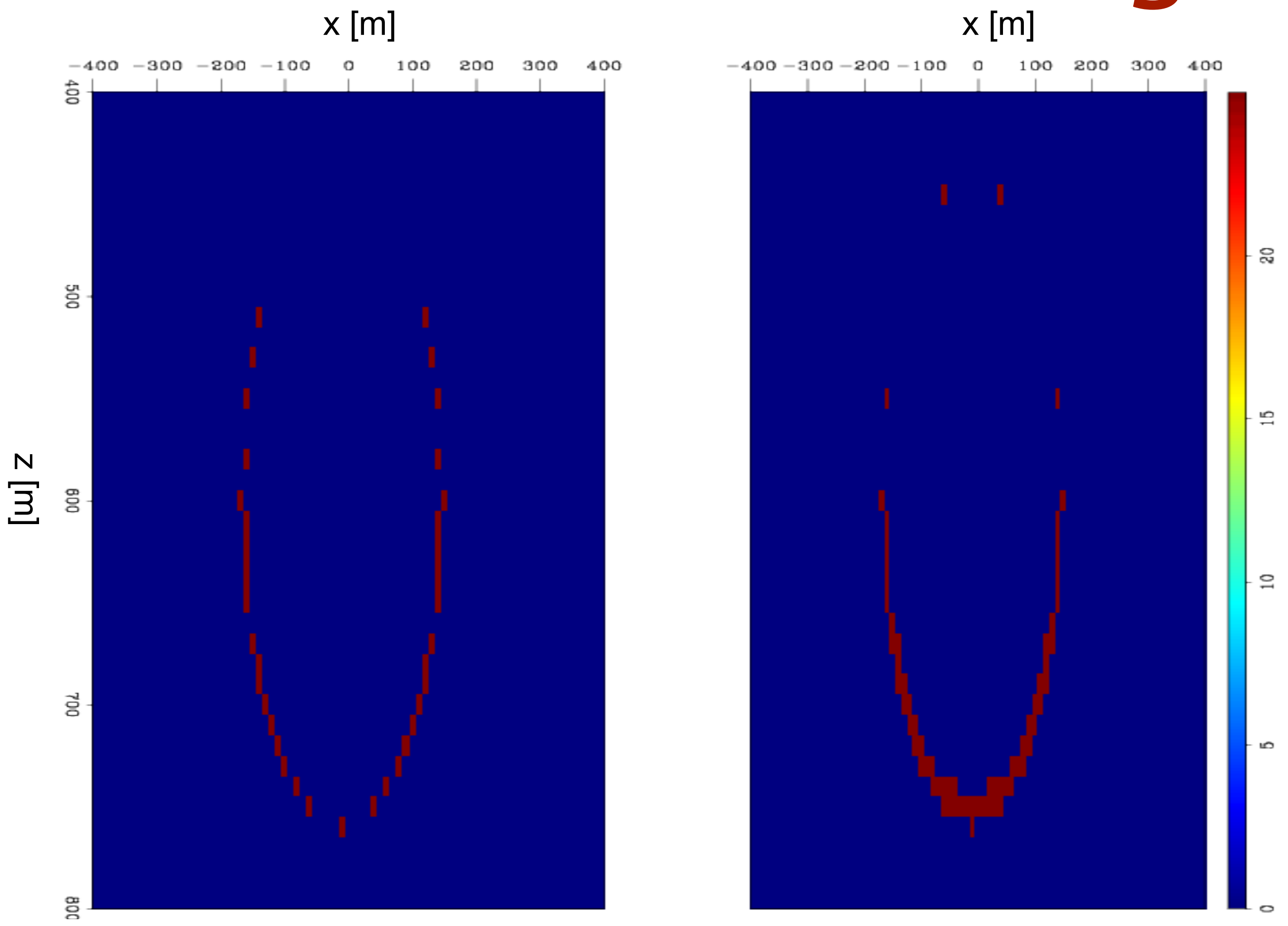


Split algorithm

General algorithm

iteration = 0

Does it ever converge?

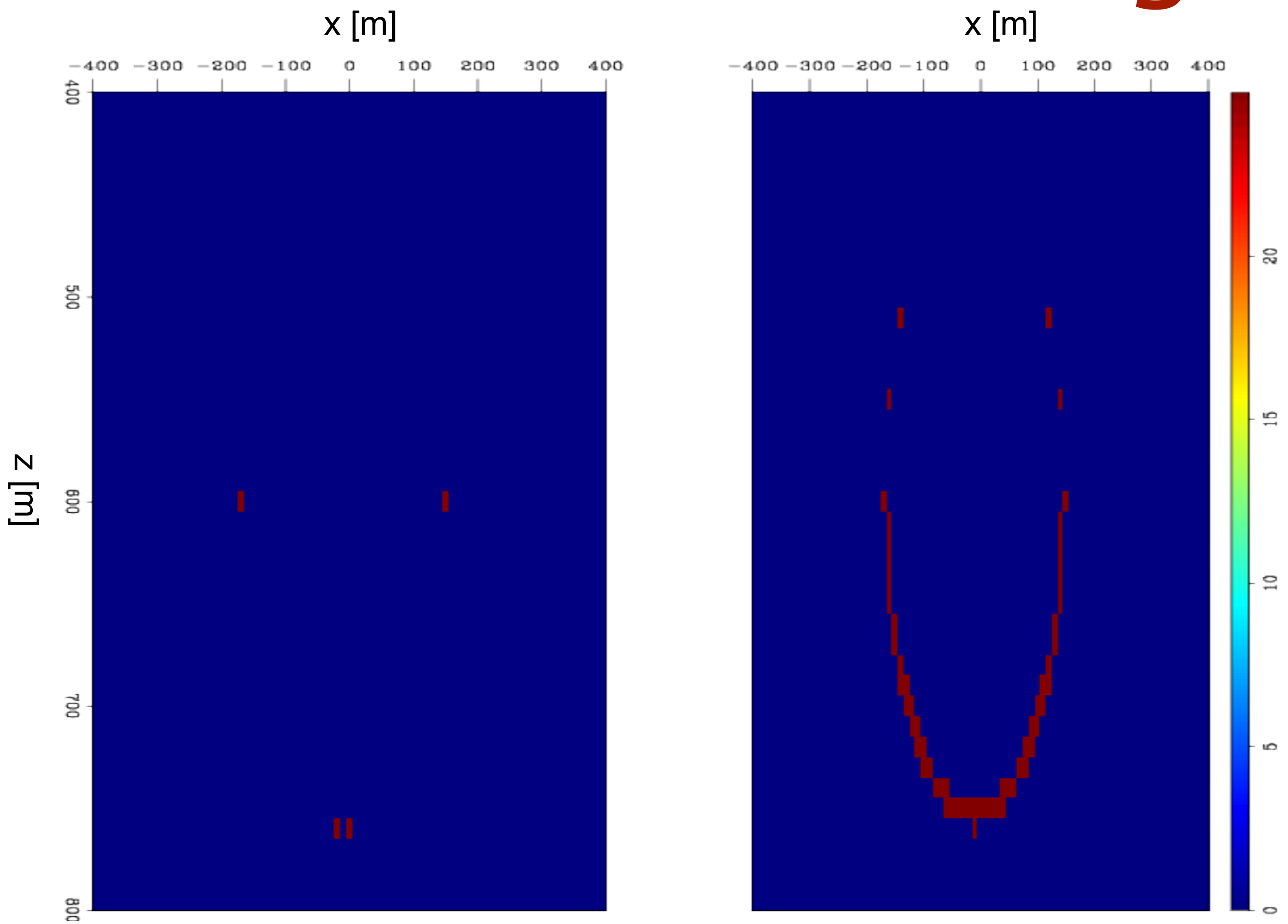


Split algorithm

General algorithm

iteration = 5

Does it ever converge?

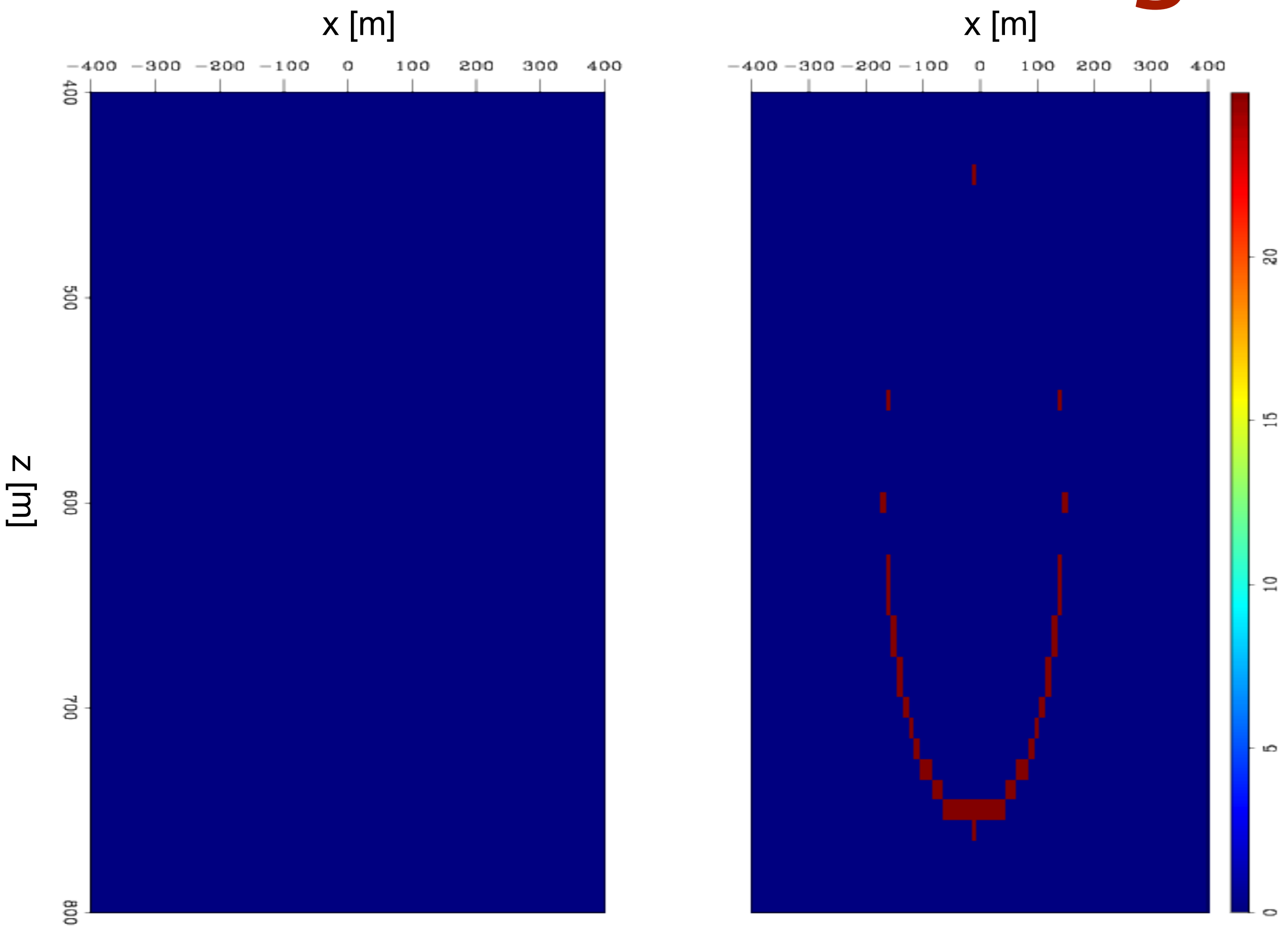


Split algorithm

General algorithm

iteration = 15

Does it ever converge?

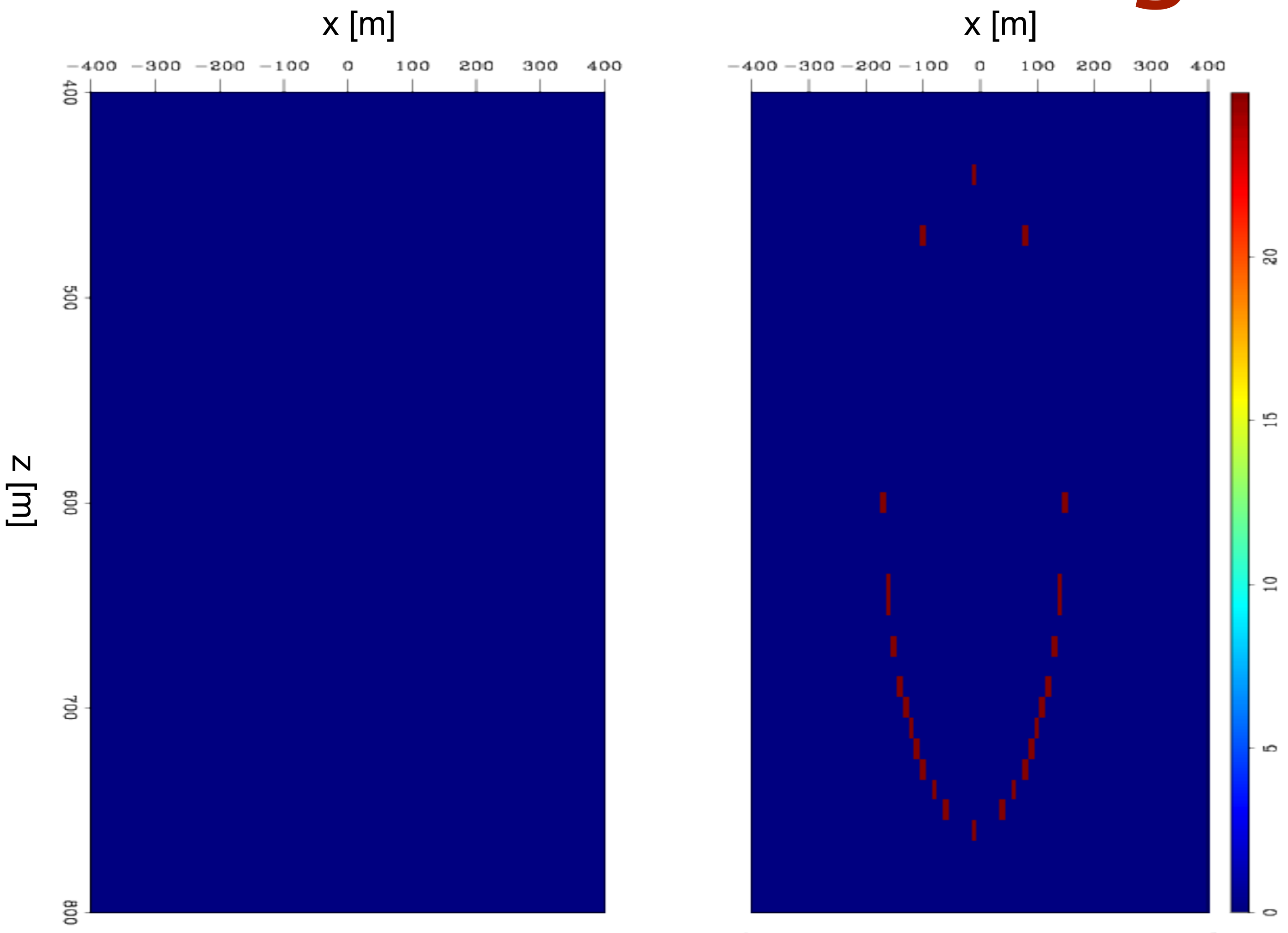


Split algorithm

General algorithm

iteration = 25

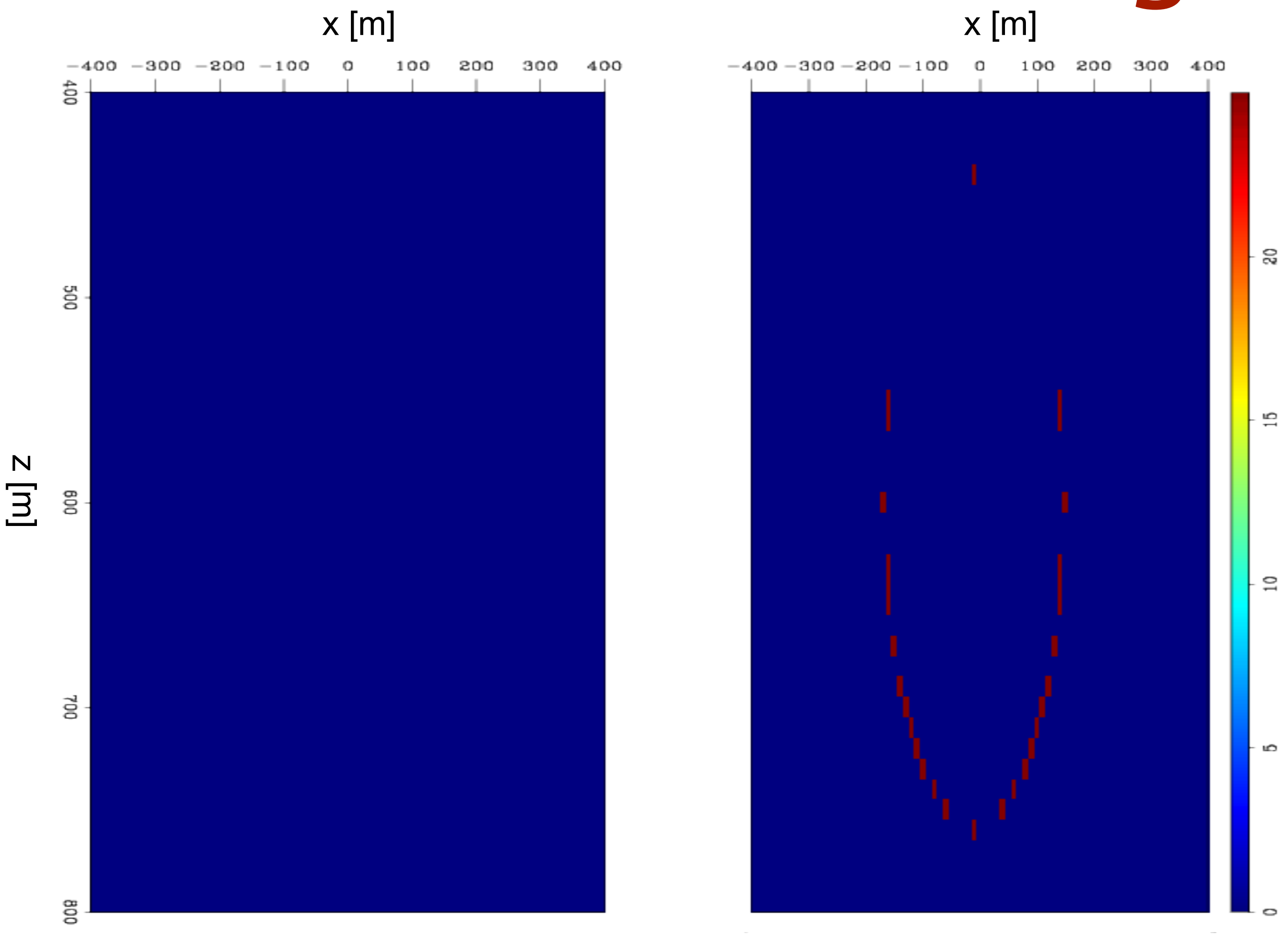
Does it ever converge?



Split algorithm **General algorithm**

iteration = 125

Does it ever converge?



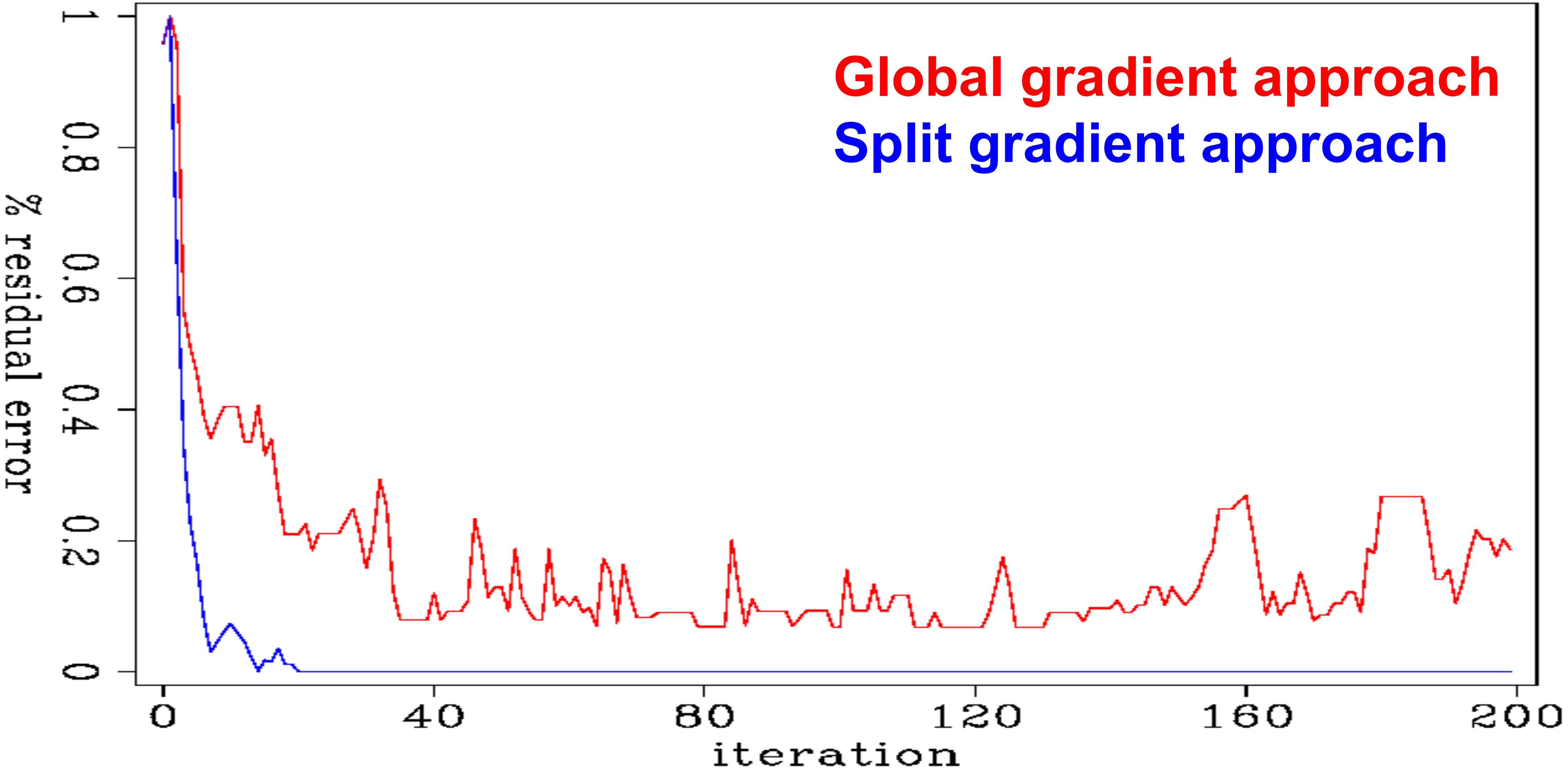
Split algorithm

General algorithm

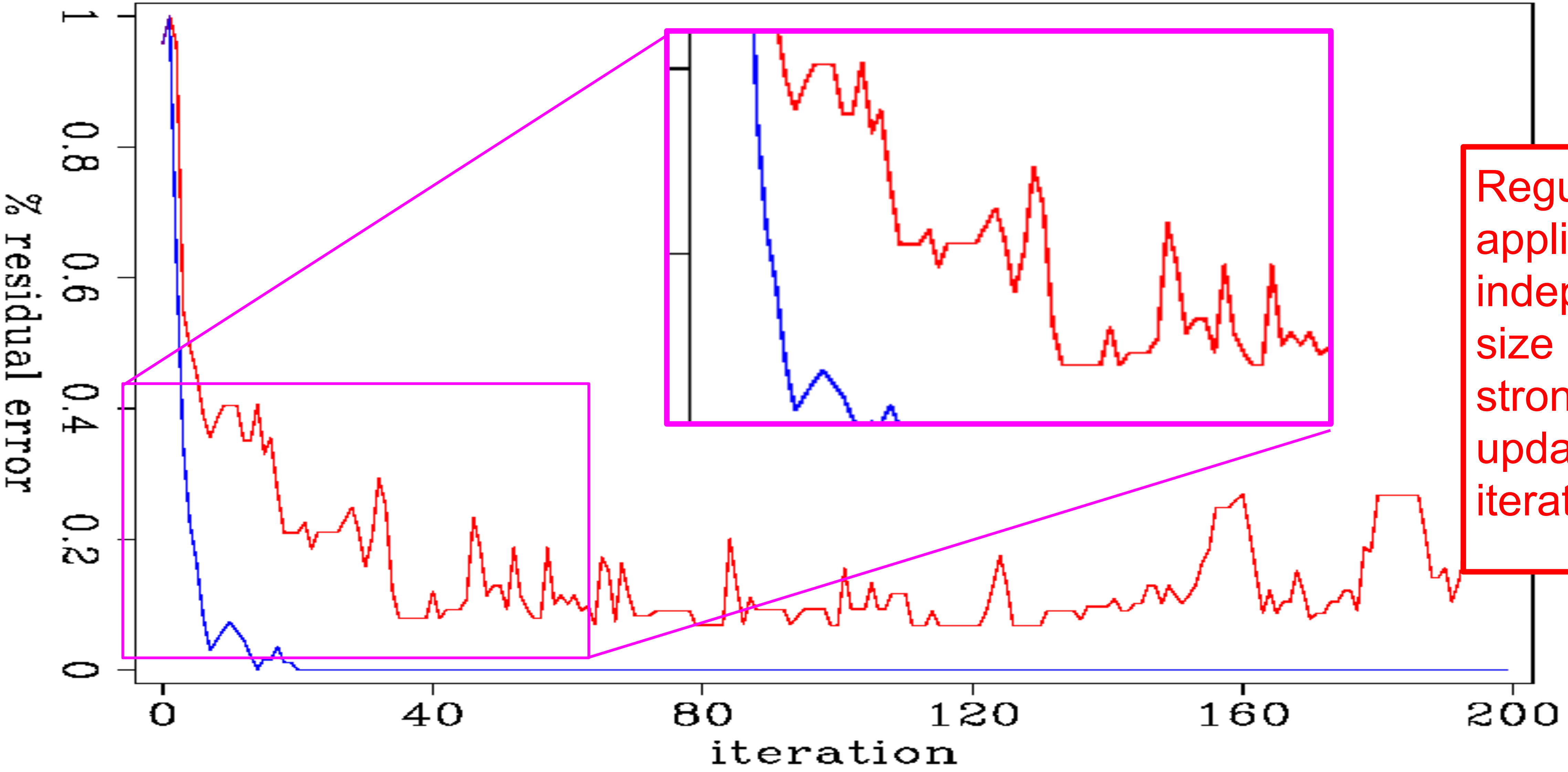
The general algorithm hasn’t converged after 175 iterations, while the split algorithm converges in 20.

iteration = 175

Objective function comparison

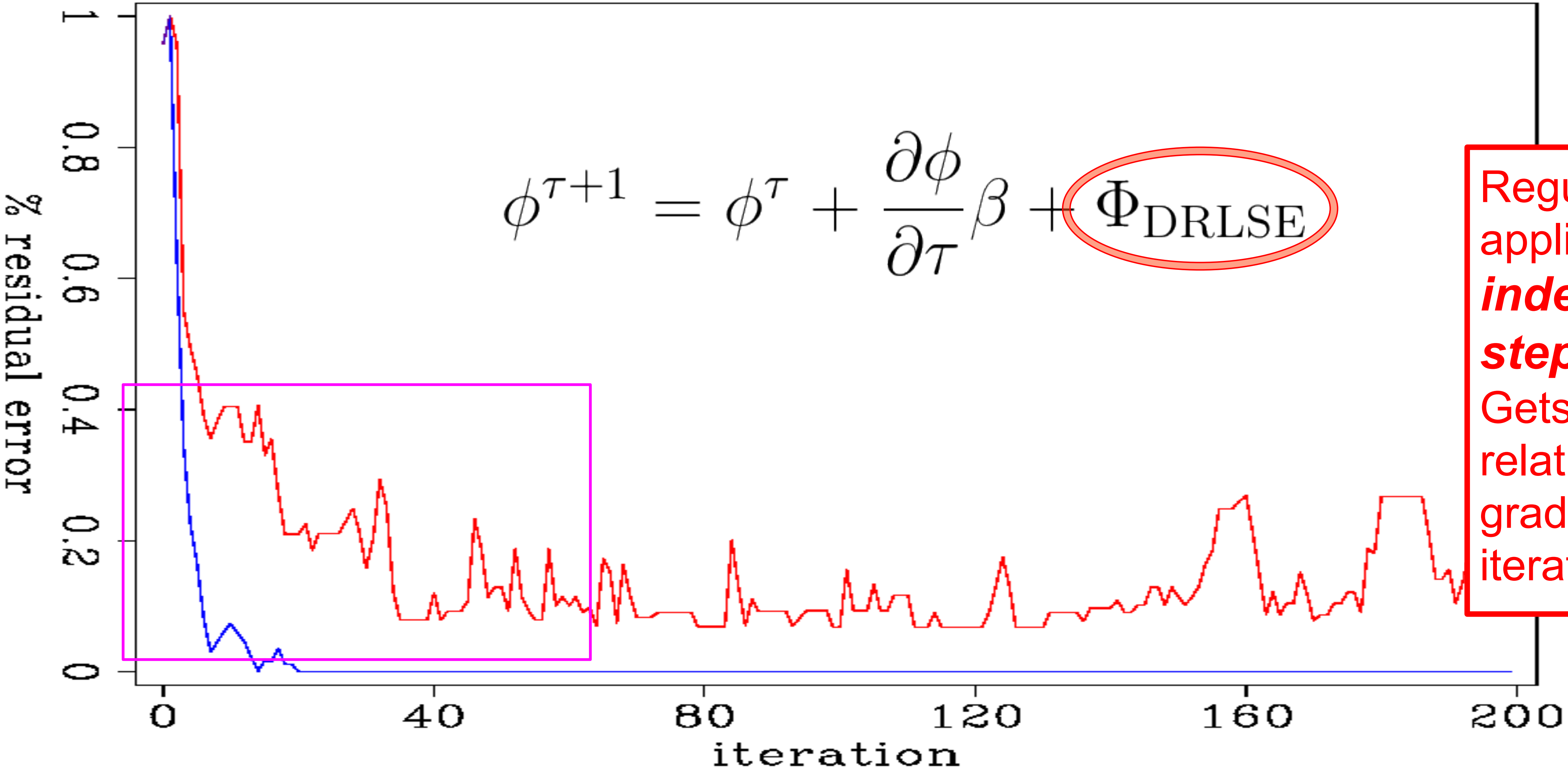


Why so spiky?



Regularization is applied independent of step size beta. Gets stronger relative to update gradient as iterations progress.

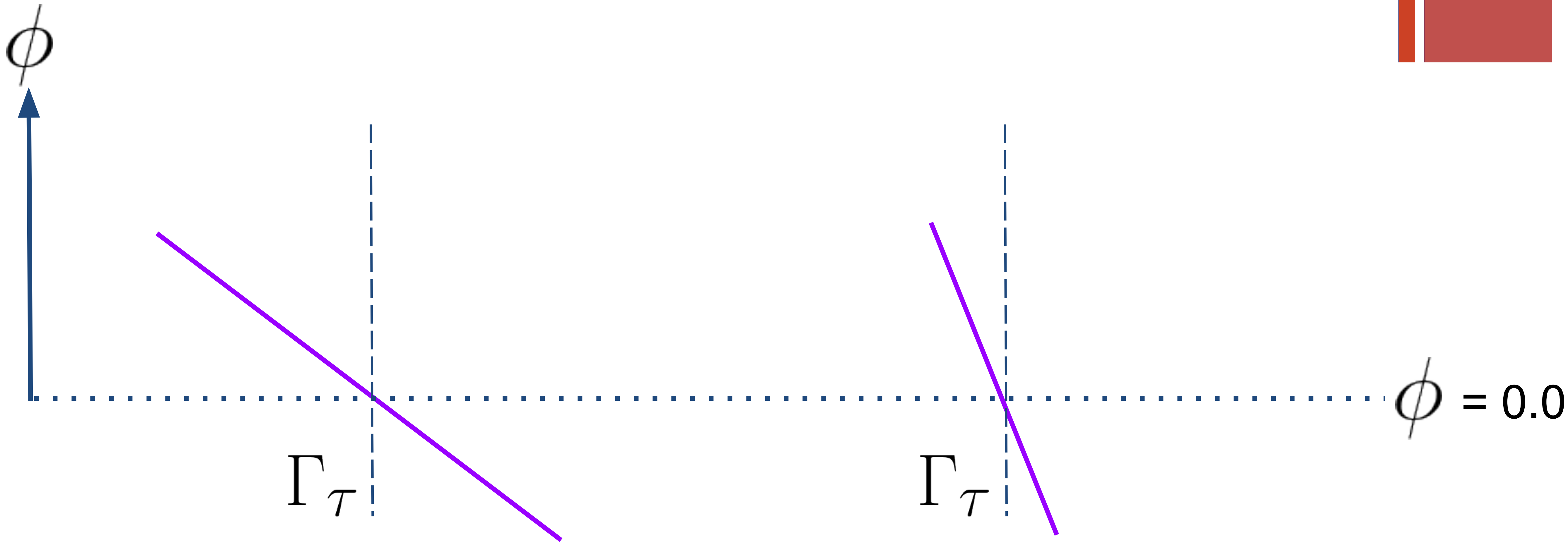
Why so spiky?



$$\phi^{\tau+1} = \phi^{\tau} + \frac{\partial \phi}{\partial \tau}\beta + \Phi_{\mathrm{DRLSE}}$$

Regularization is applied ***independent of step size beta*** Gets stronger relative to update gradient as iterations progress.

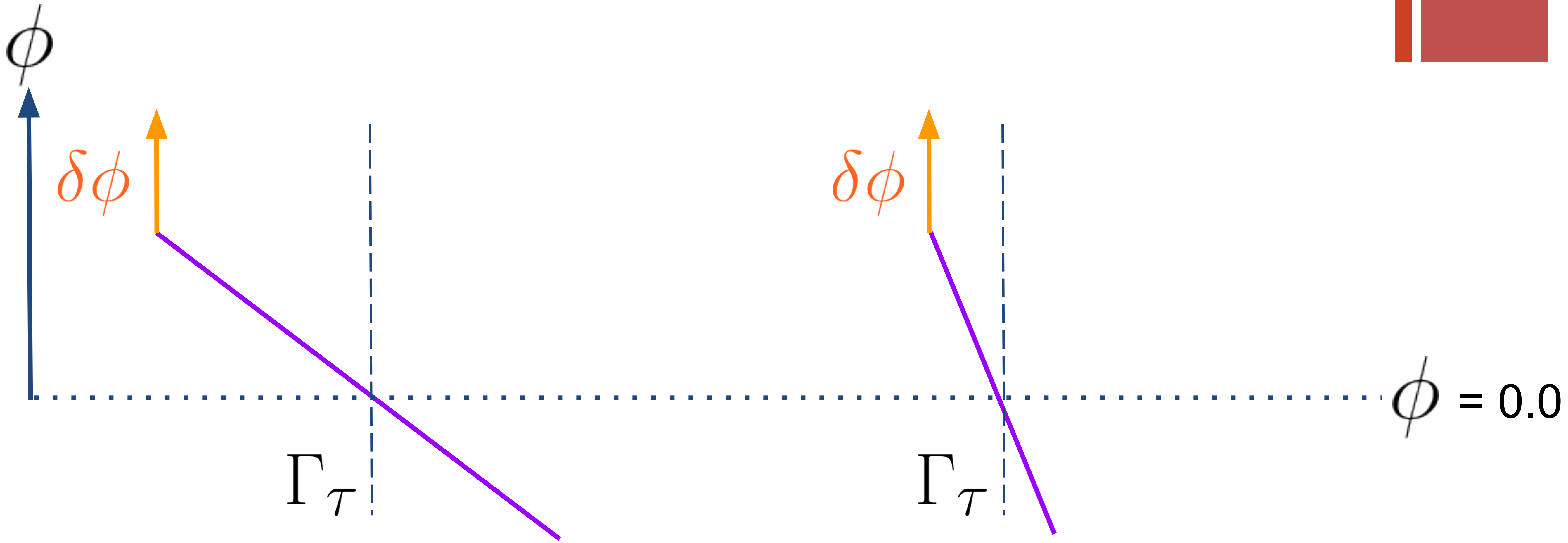
Why regularization?



Regularized implicit surface

Unregularized implicit surface

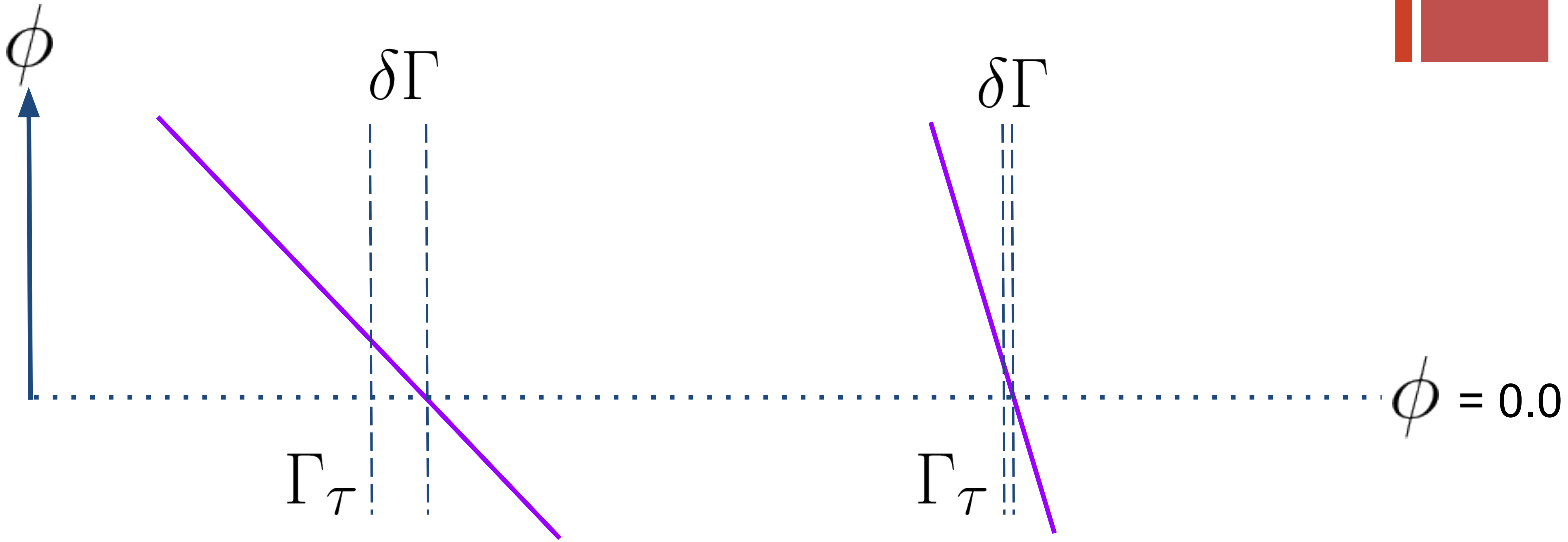
Why regularization?



Regularized implicit surface

Unregularized implicit surface

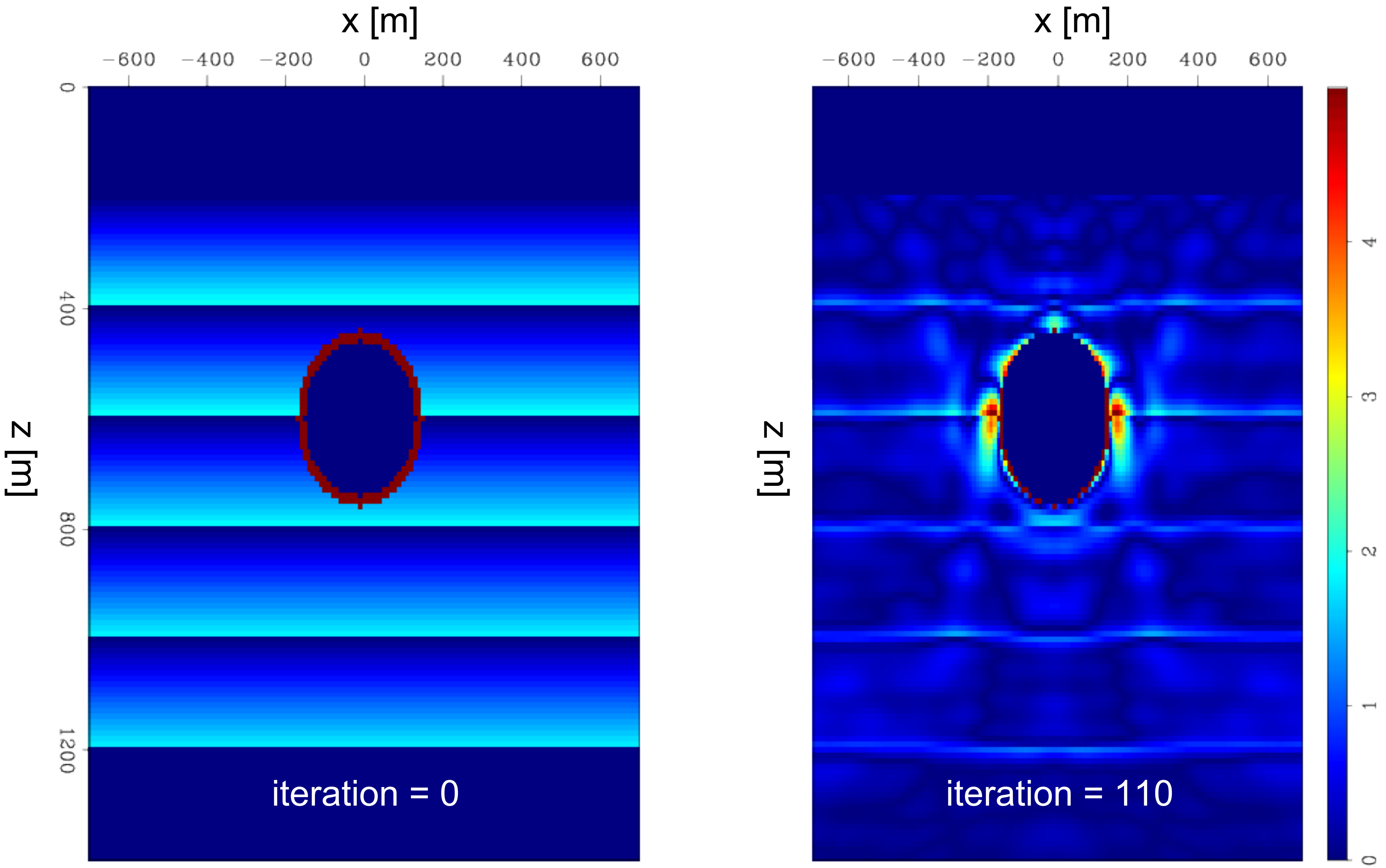
No regularization = stalling



Regularized implicit surface

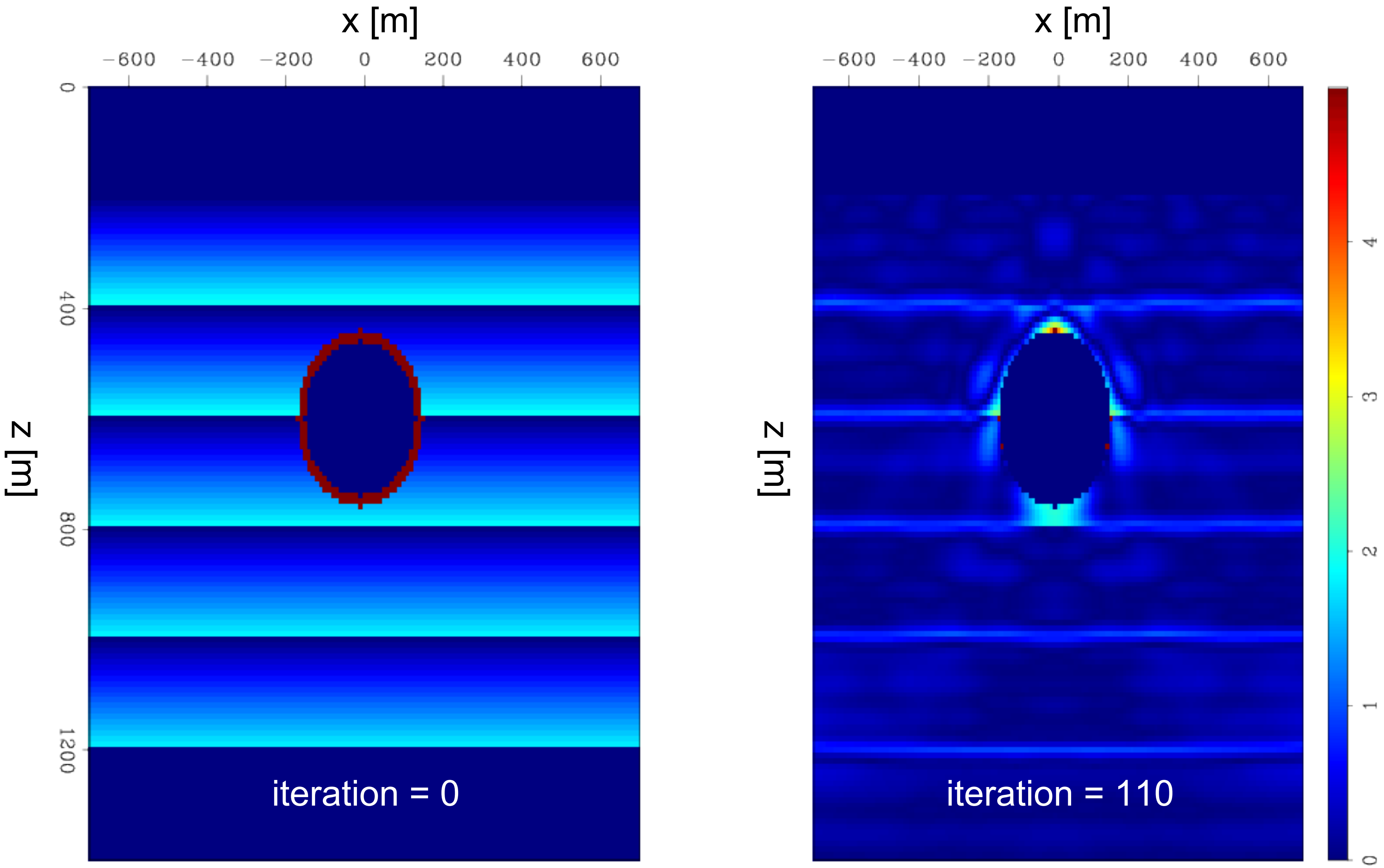
Unregularized implicit surface

Algorithm comparison: w/ tomography



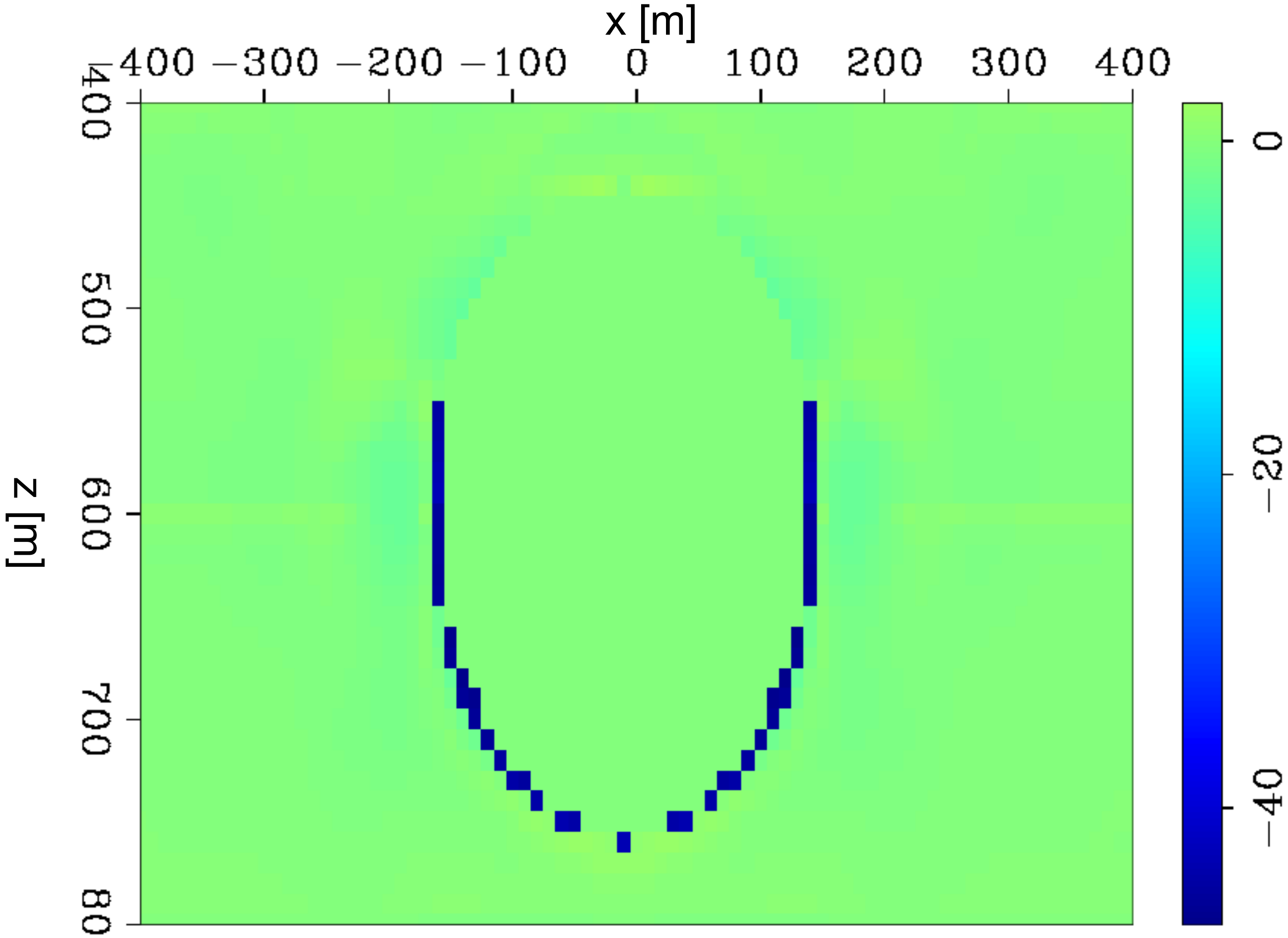
General algorithm

Algorithm comparison: w/ tomography



Domain decomposition algorithm

Algorithm comparison: w/ tomography



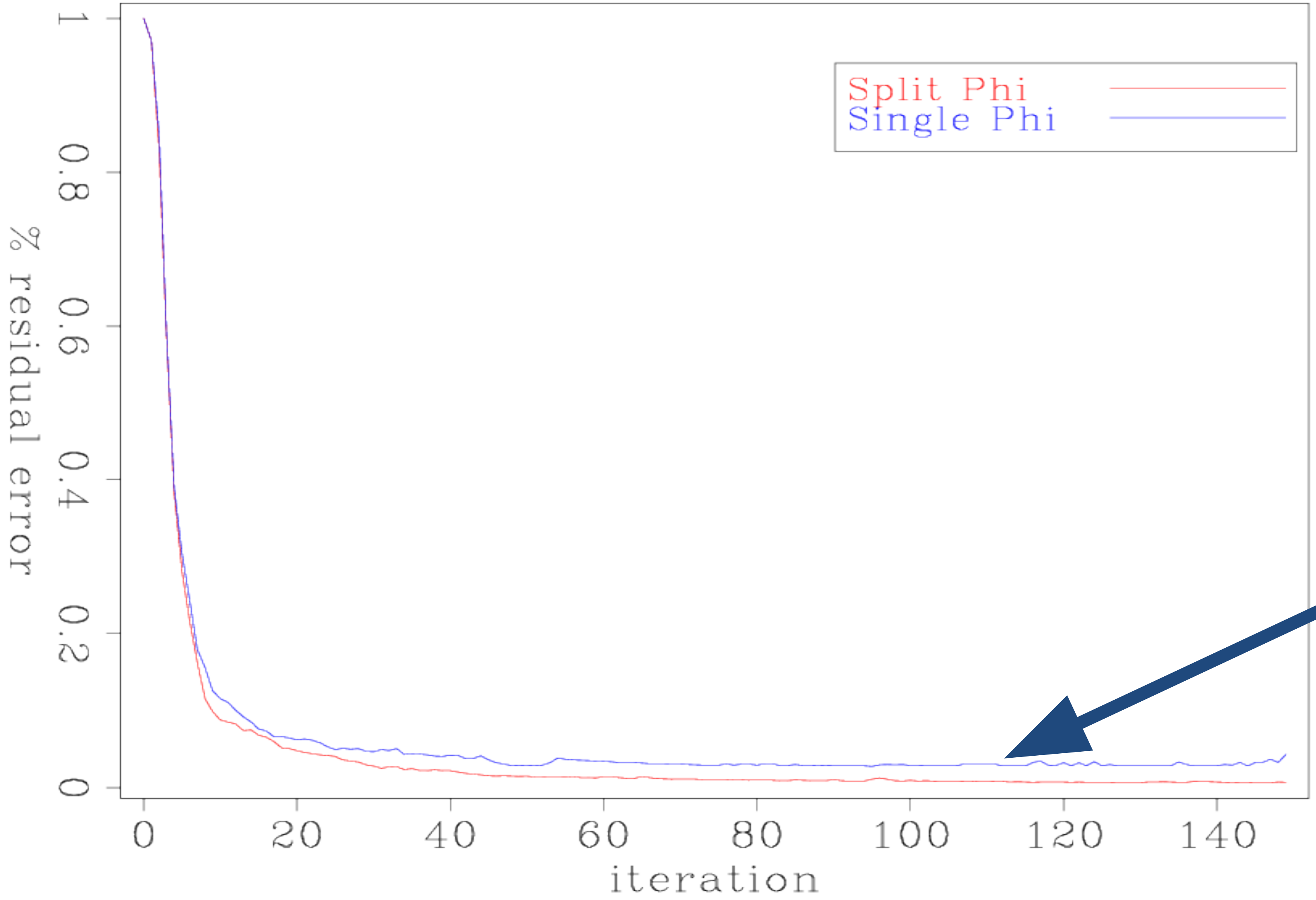
% velocity error

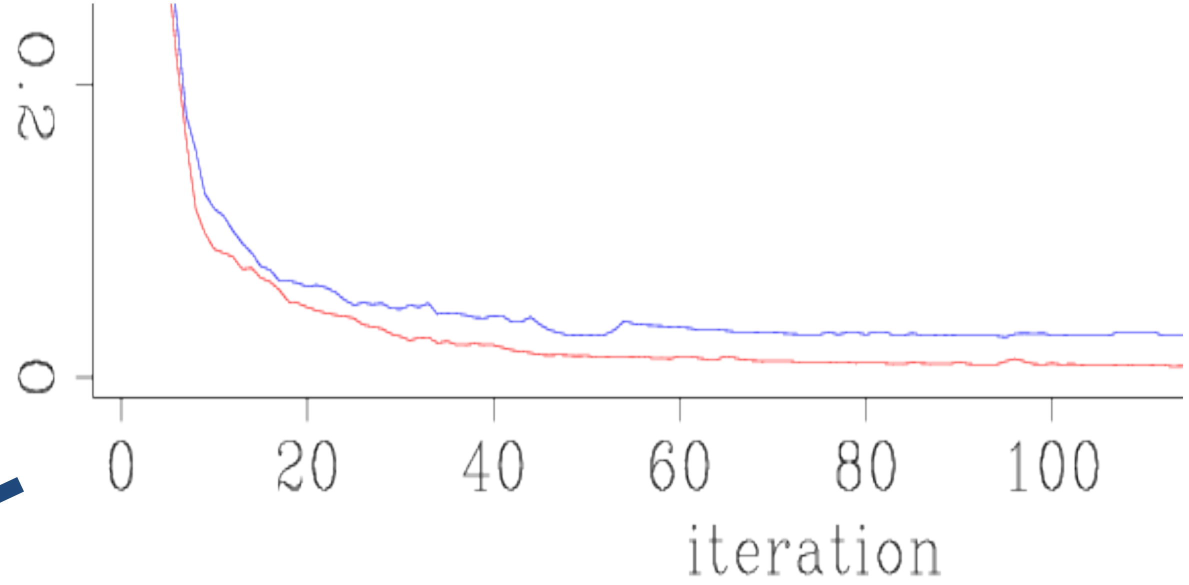
iteration = 110

RED: Split phi does **WORSE** job

BLUE: Split phi does **BETTER** job

Algorithm comparison: w/ tomography





Lower residual error

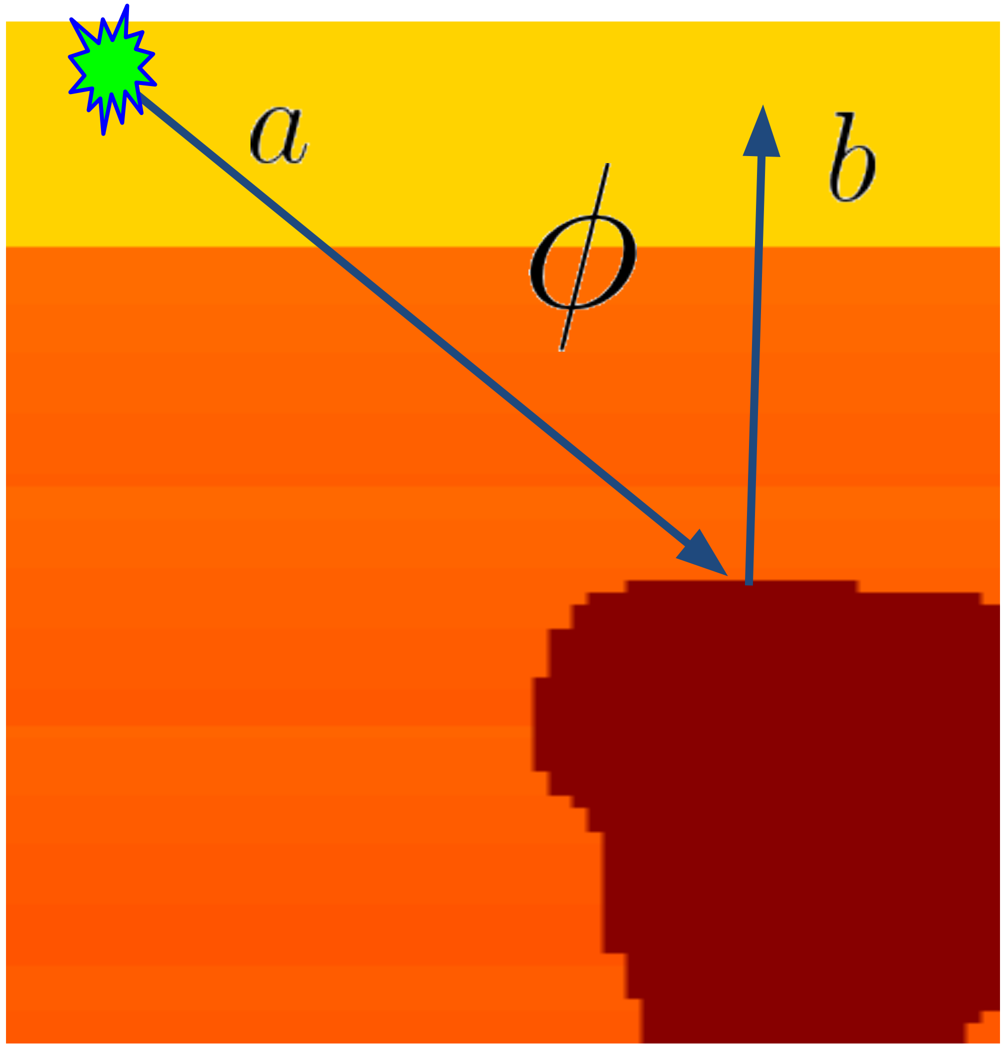
Generalized gradient splitting

$$a \cdot b = |a| \, |b| \, cos(\theta)$$

ϕ < 90.0

$$a \cdot b > 0.0$$

So considered a 'top' section



Direct ray paths from shot

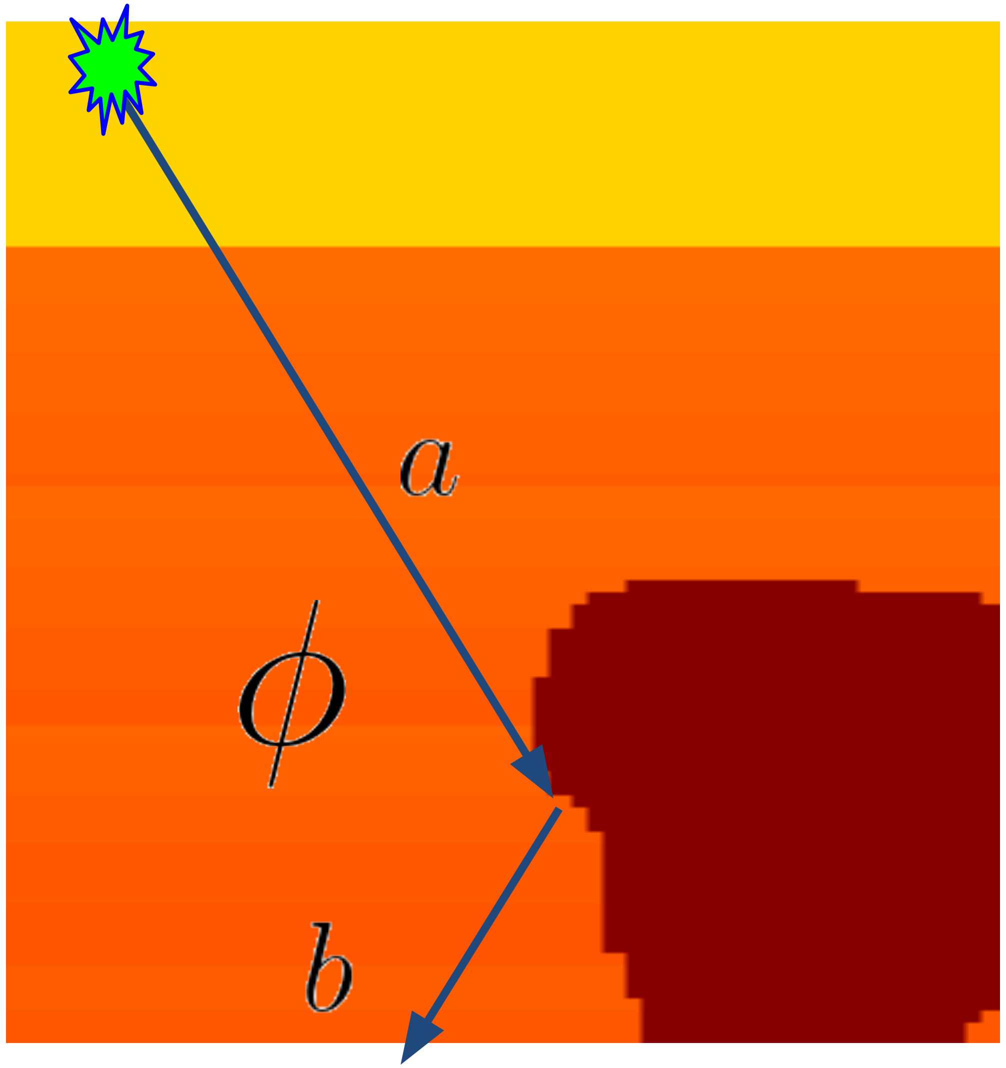
Generalized gradient splitting

$$a \cdot b = |a|\,|b|\,cos(\theta)$$

ϕ > 90.0

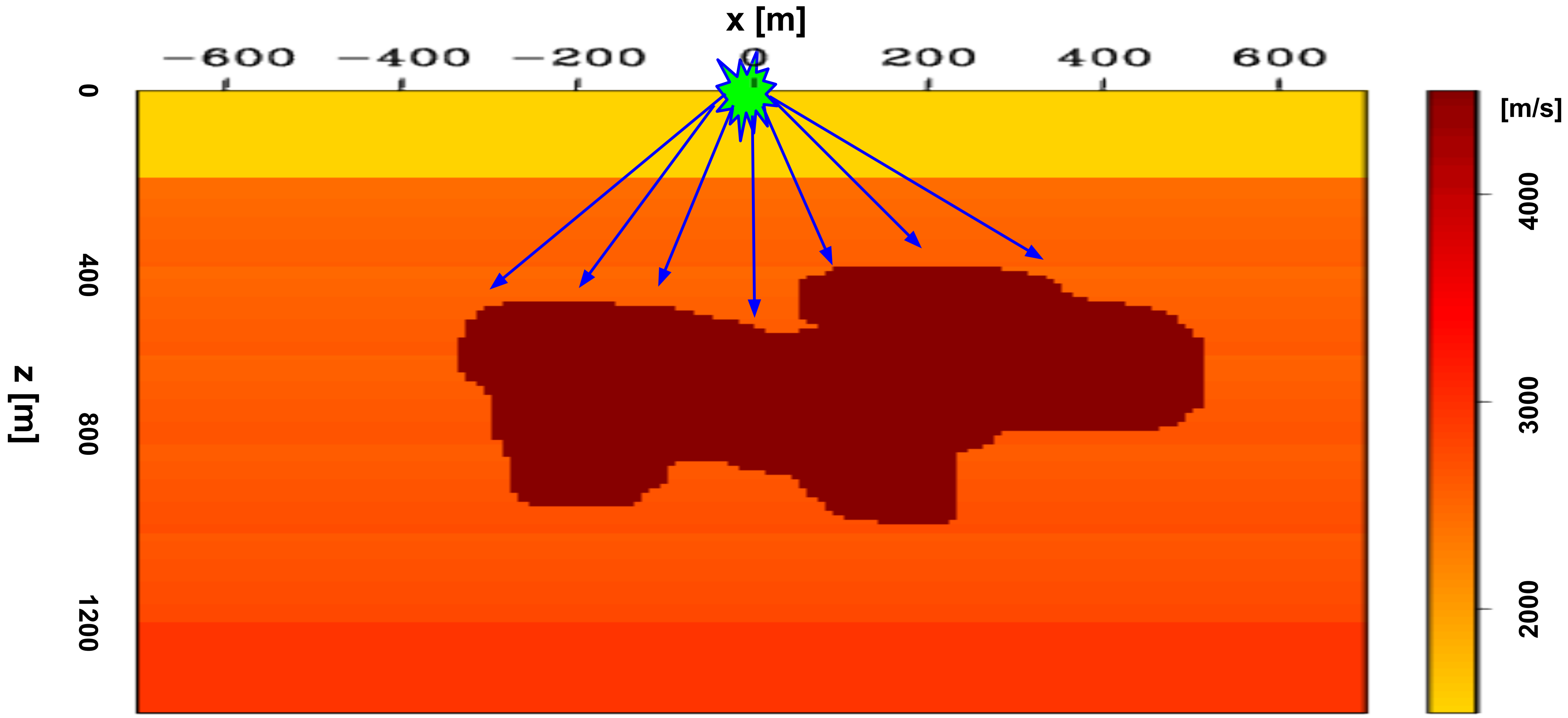
$$a \cdot b < 0.0$$

So considered a ‘bottom’ section for that shot



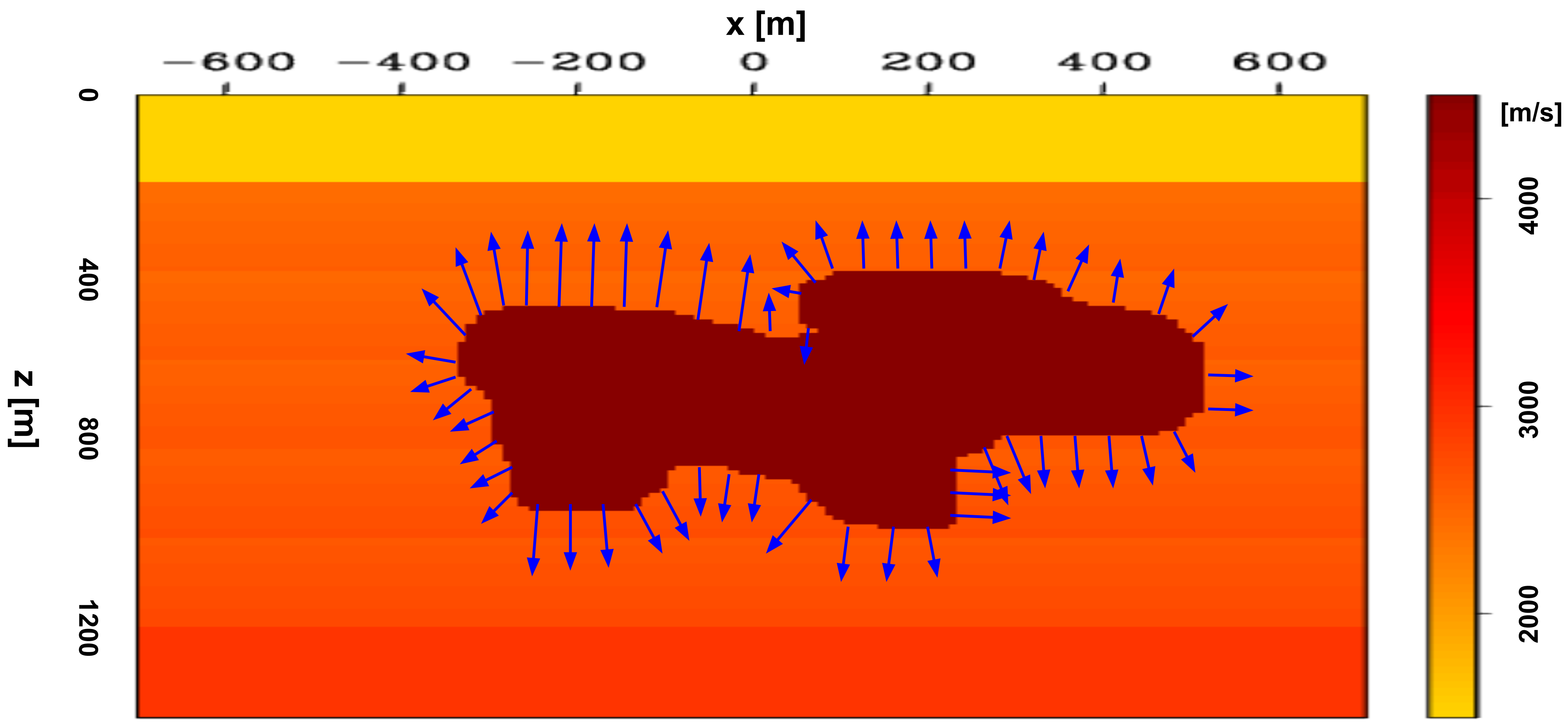
Direct ray paths from shot

Generalized gradient splitting



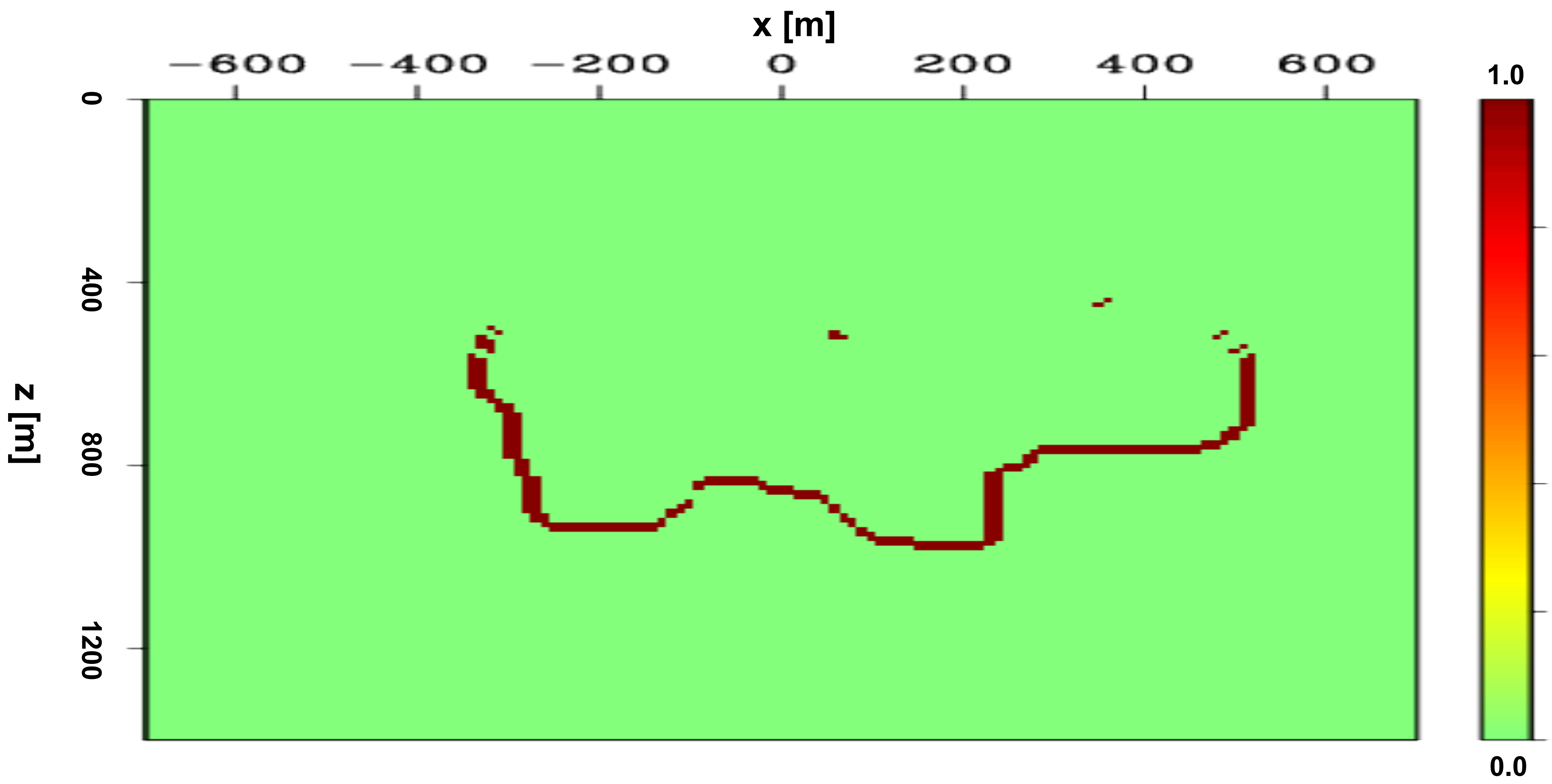
Direct ray paths from shot

Generalized gradient splitting



Boundary normal vector field

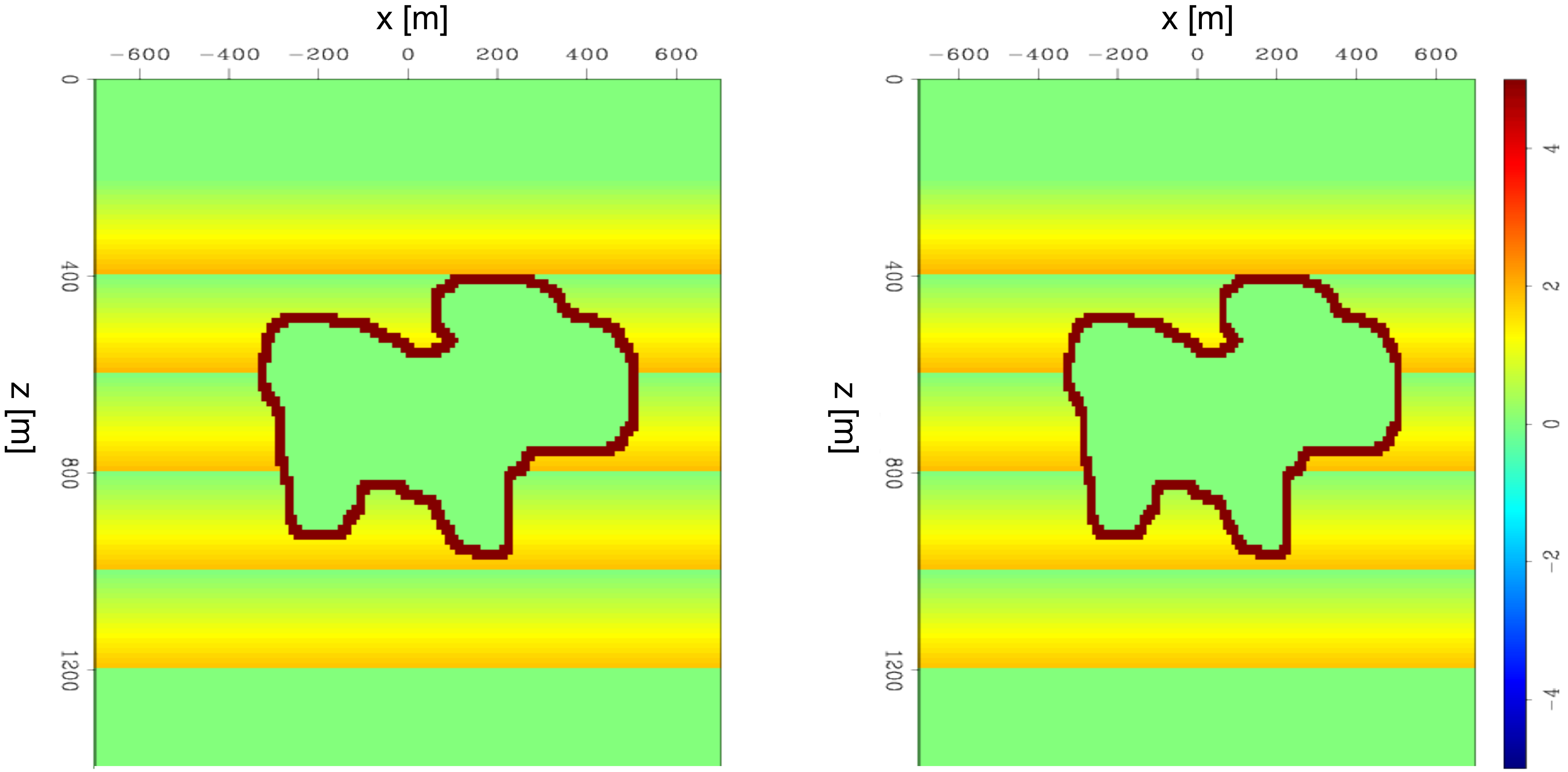
Generalized gradient splitting



‘Bottom’ gradient domain partition

- Take dot product of both vector fields to get a weighting map
- Repeat, and sum for all shots
- Threshold to make binary selector.

Domain decomposition comparison



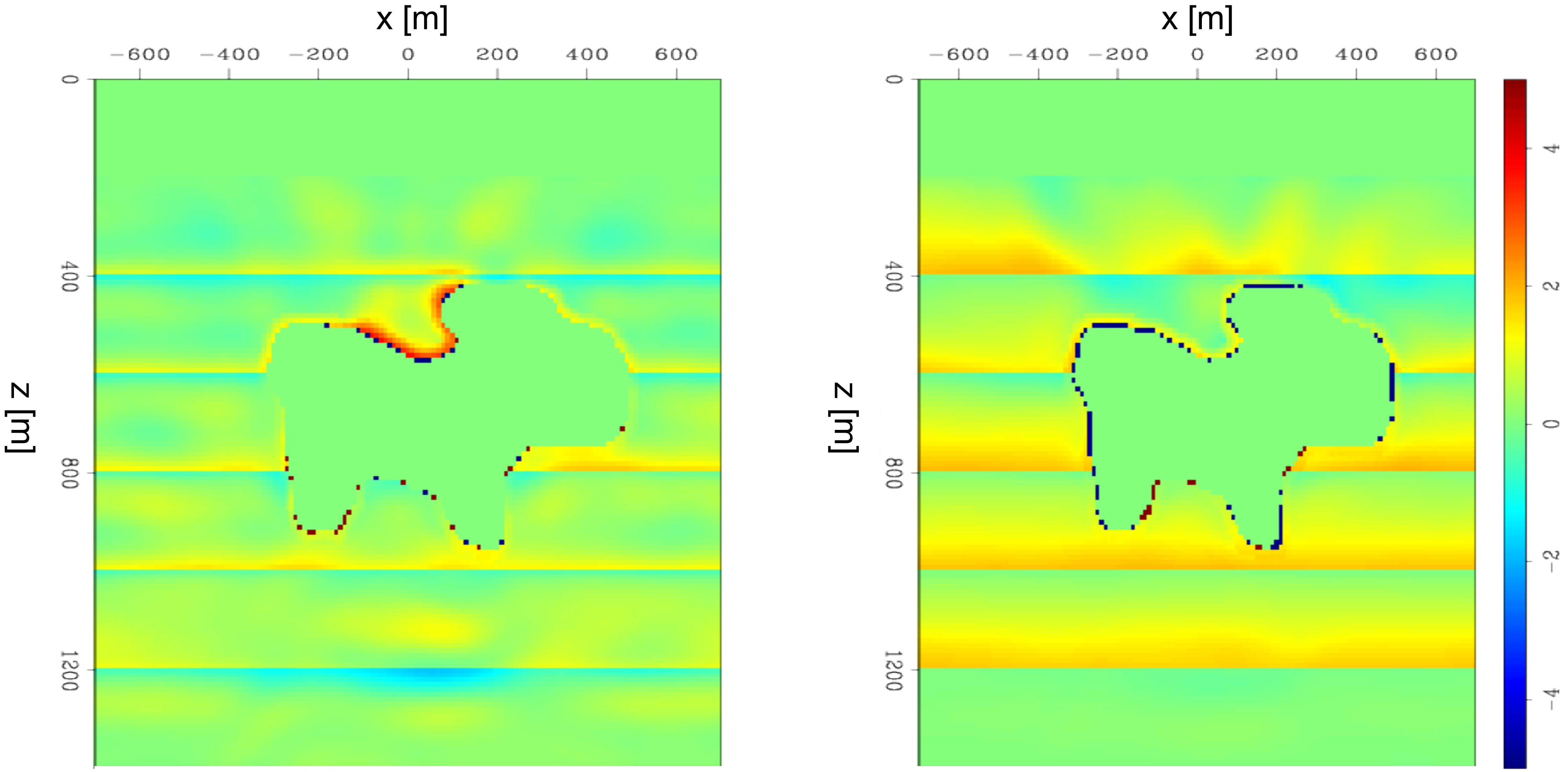
Split algorithm % vel error

General algorithm % vel error

Green = good match

iteration = 0

Domain decomposition comparison



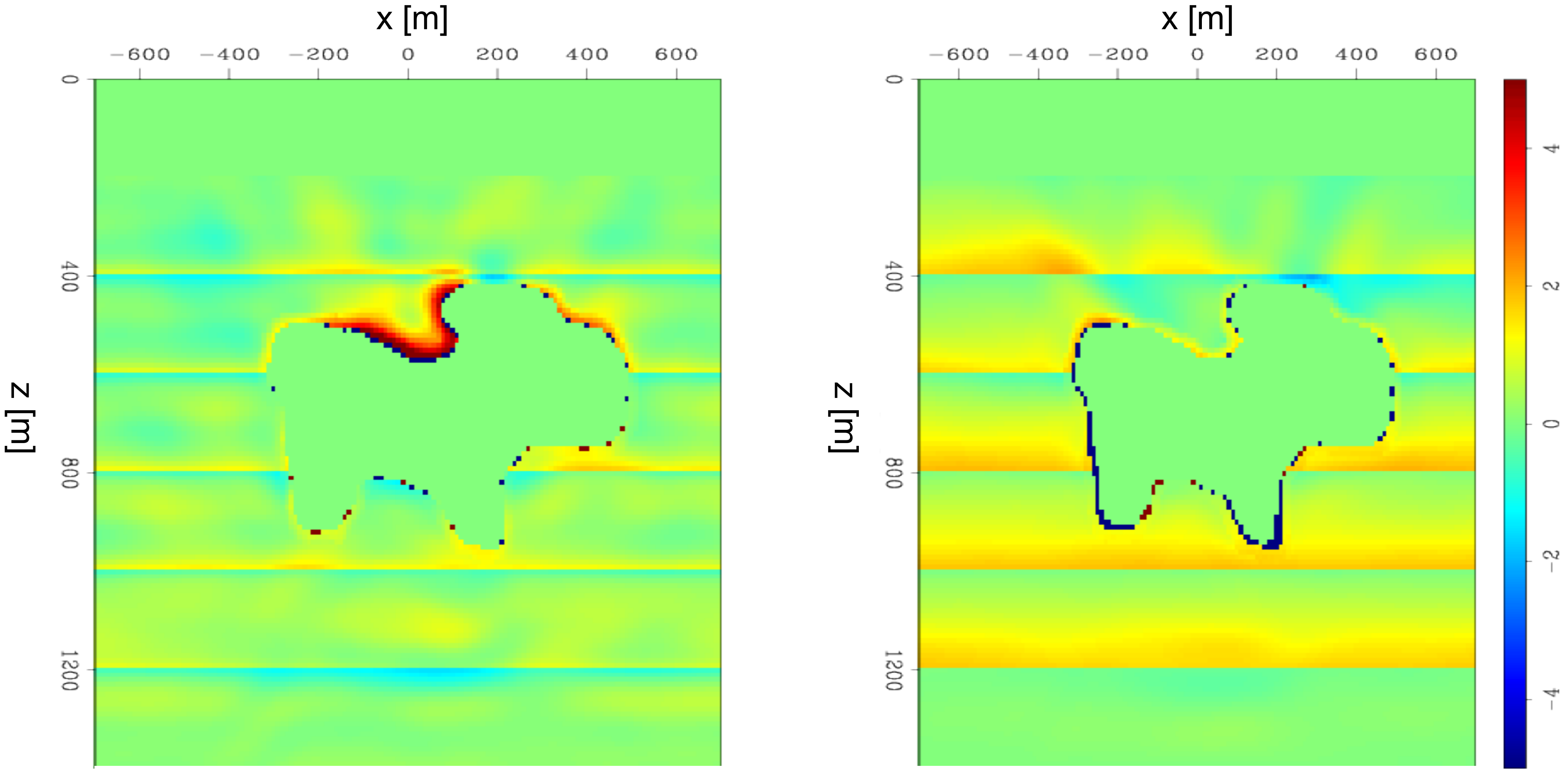
Split algorithm % vel error

General algorithm % vel error

Green = good match

iteration = 65

Domain decomposition comparison



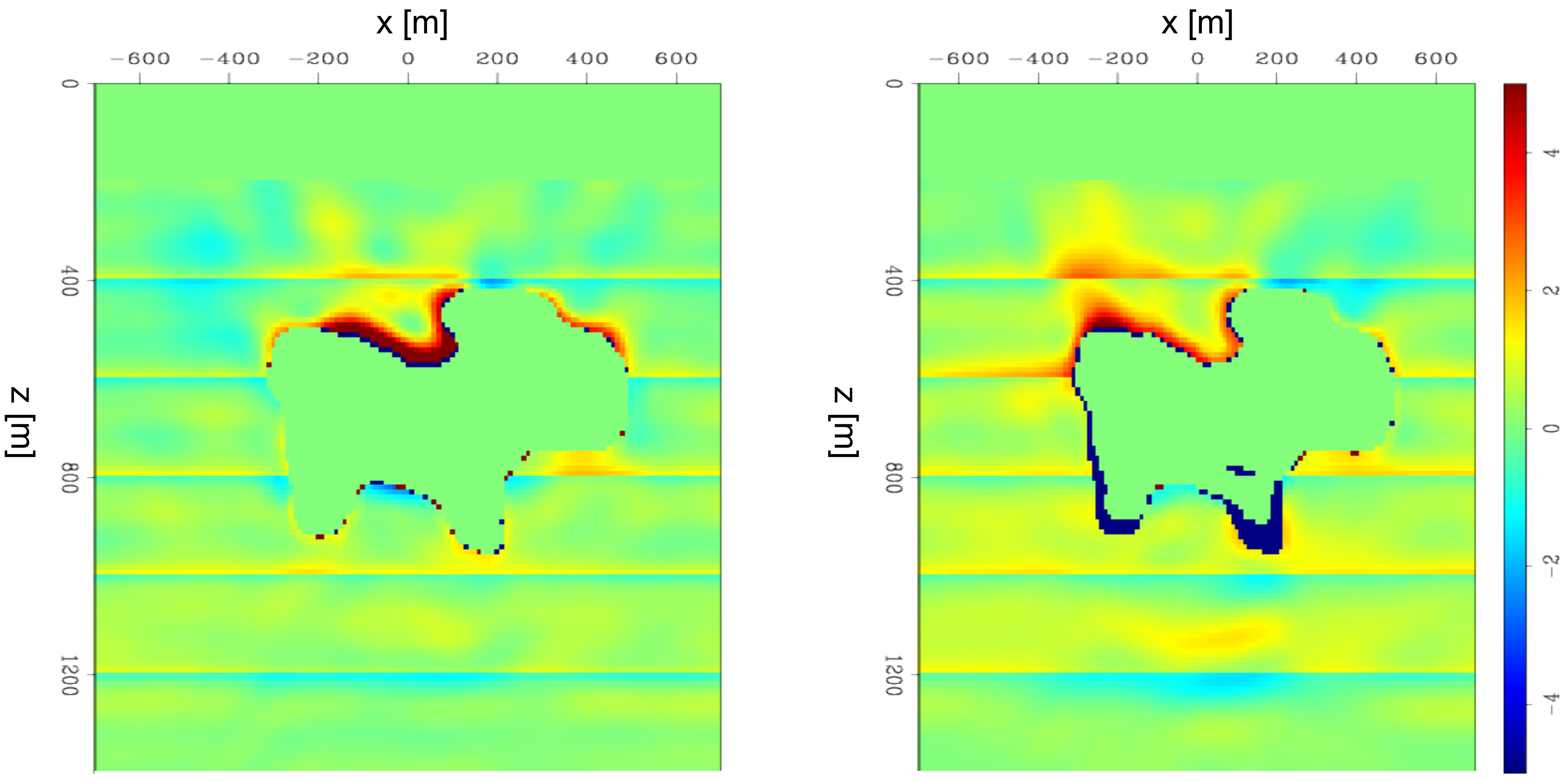
Split algorithm % vel error

General algorithm % vel error

Green = good match

iteration = 120

Domain decomposition comparison



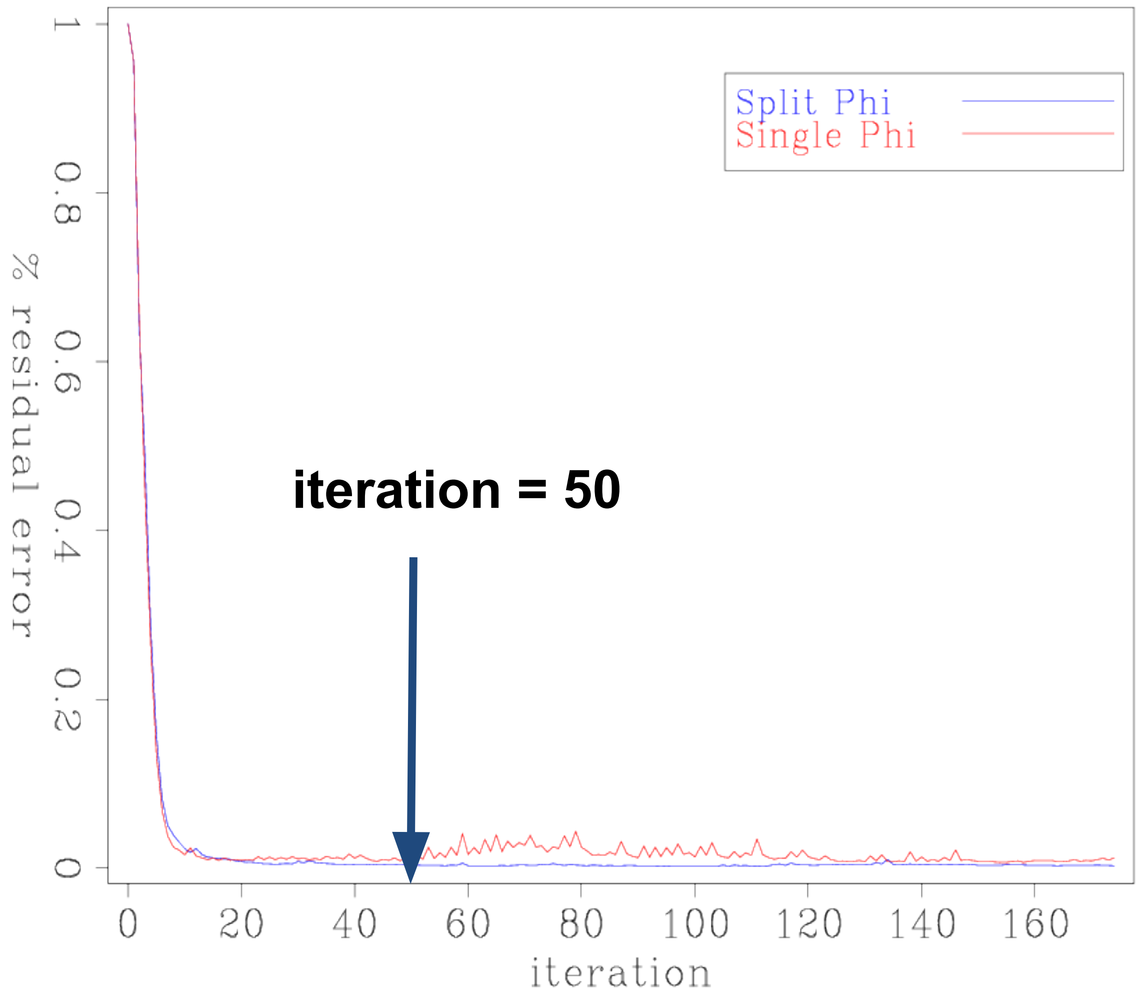
Split algorithm % vel error

General algorithm % vel error

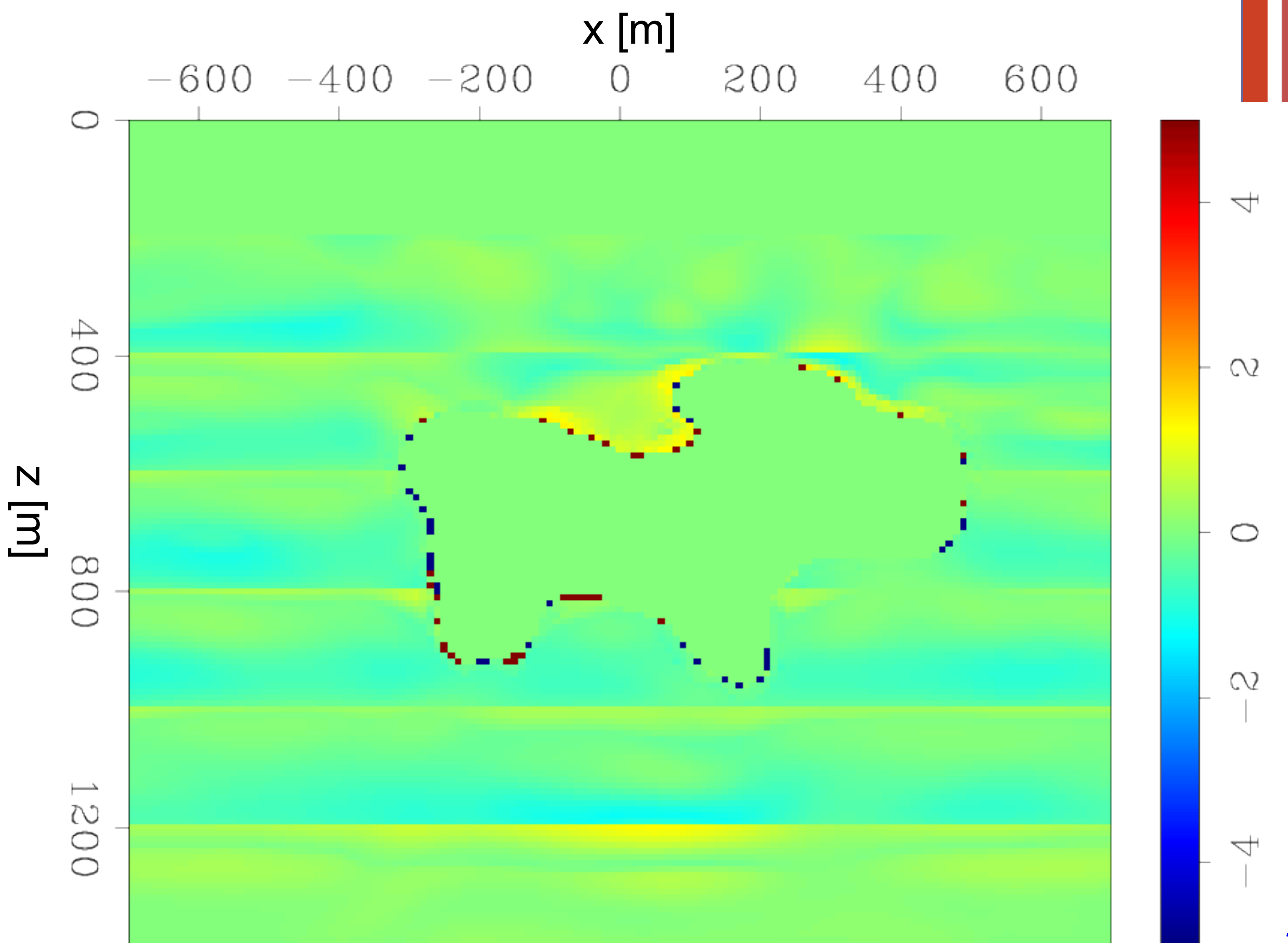
Green = good match

iteration = 175

Domain decomposition comparison



iteration = 50

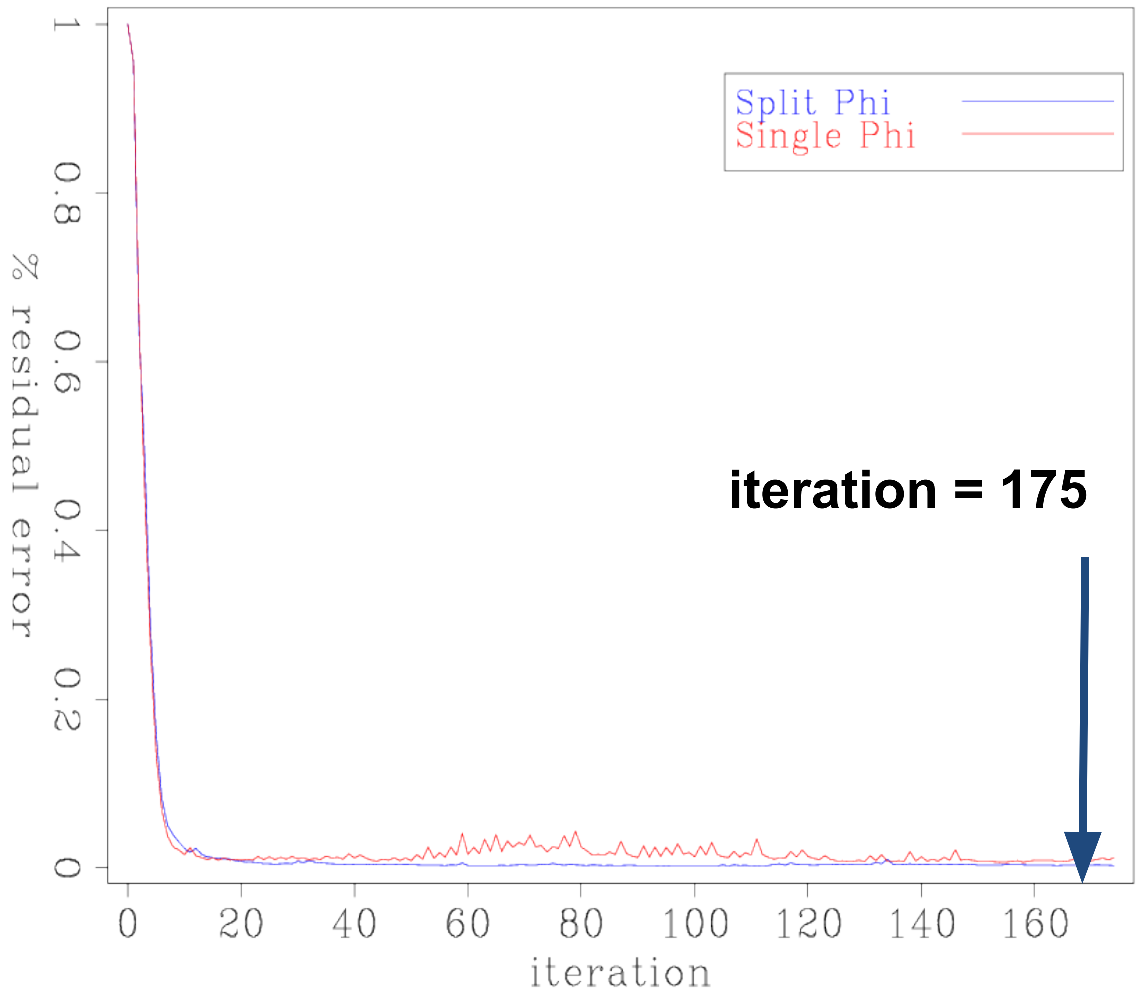


% velocity error differential between methods

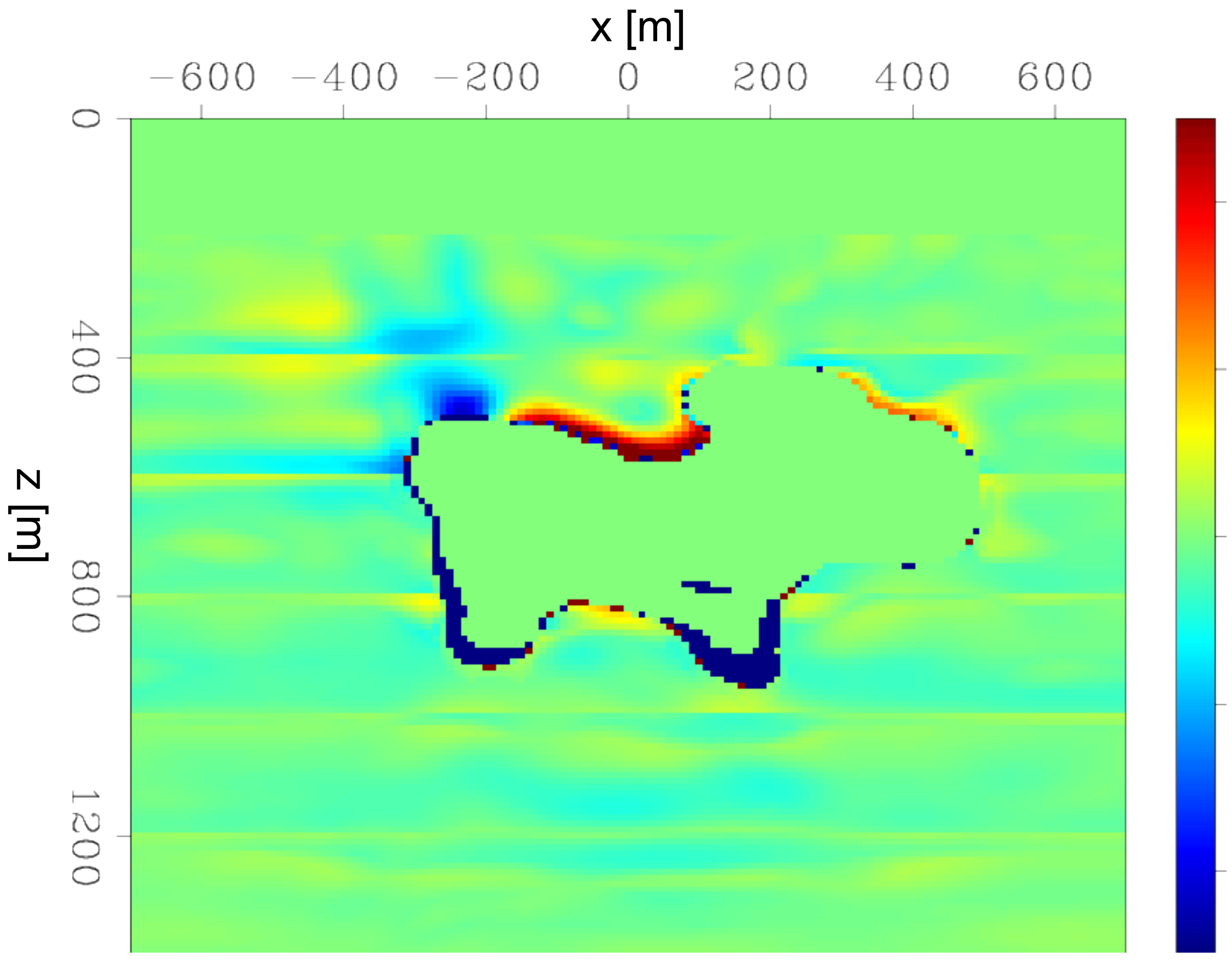
RED: Split phi does **WORSE** job

BLUE: Split phi does **BETTER** job

Domain decomposition comparison



iteration = 175

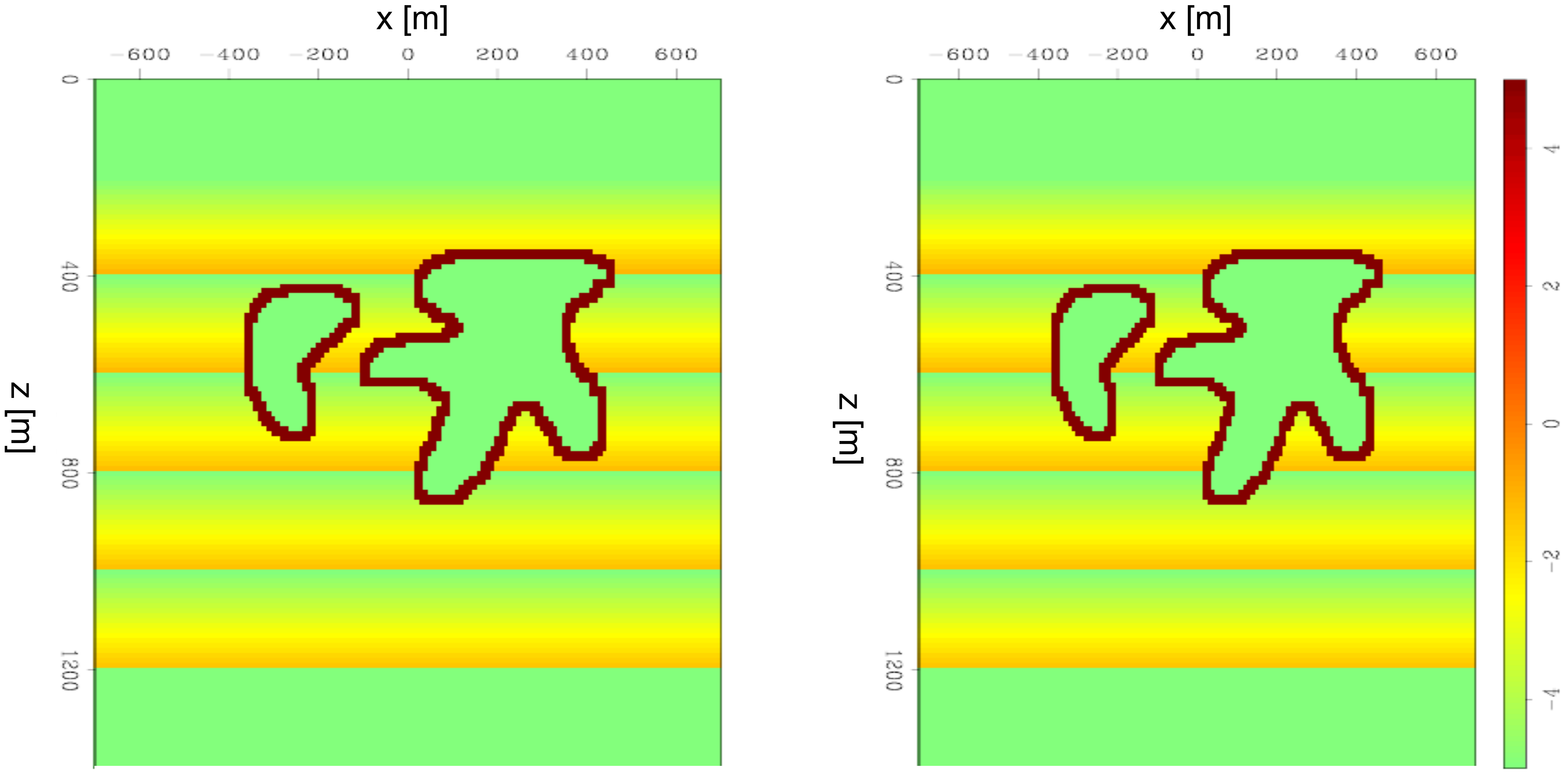


% velocity error differential between methods

RED: Split phi does **WORSE** job

BLUE: Split phi does **BETTER** job

Domain decomposition comparison



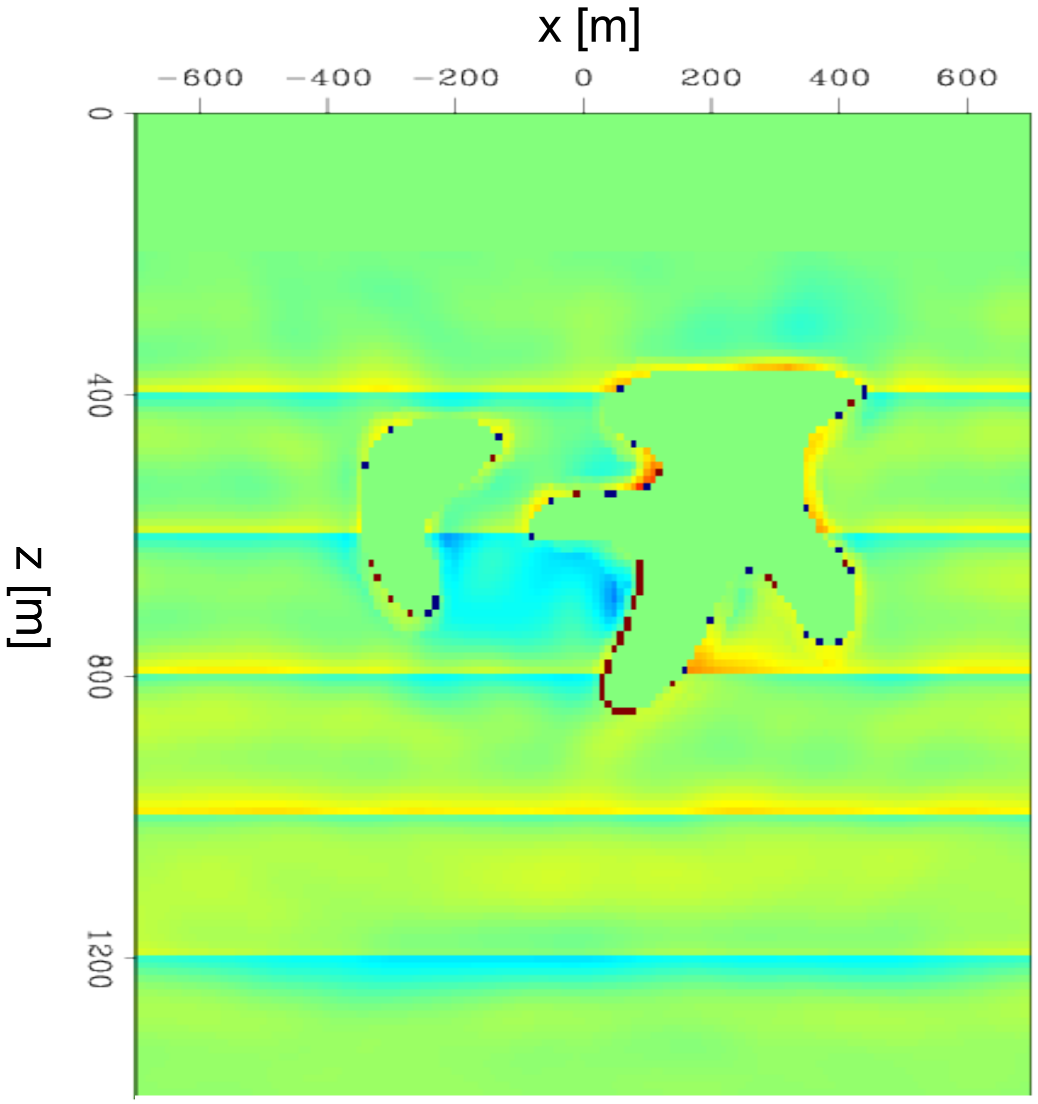
Split algorithm % vel error

General algorithm % vel error

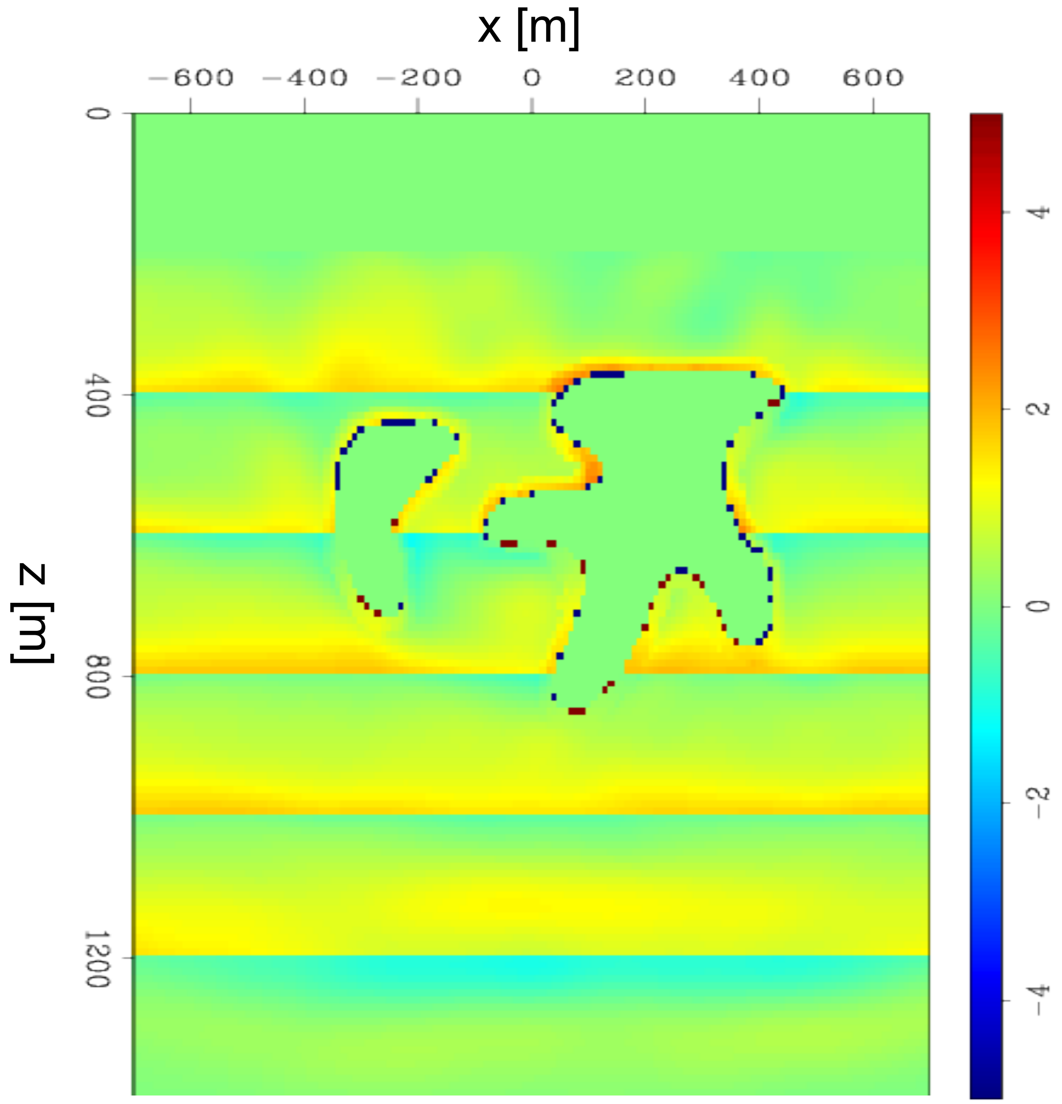
Green = good match

iteration = 0

Domain decomposition comparison



Split algorithm % vel error

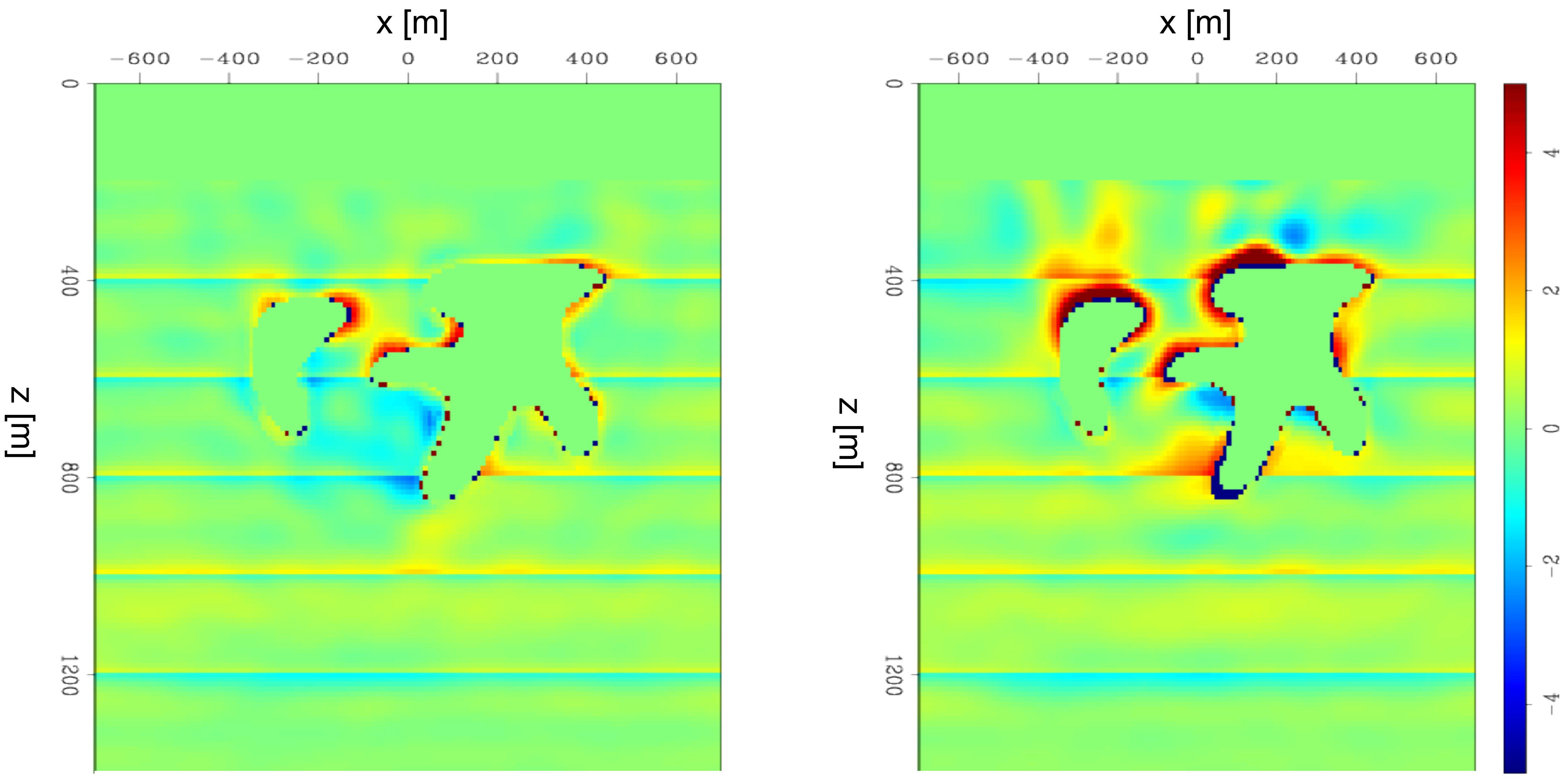


General algorithm % vel error

Green = good match

iteration = 80

Domain decomposition comparison



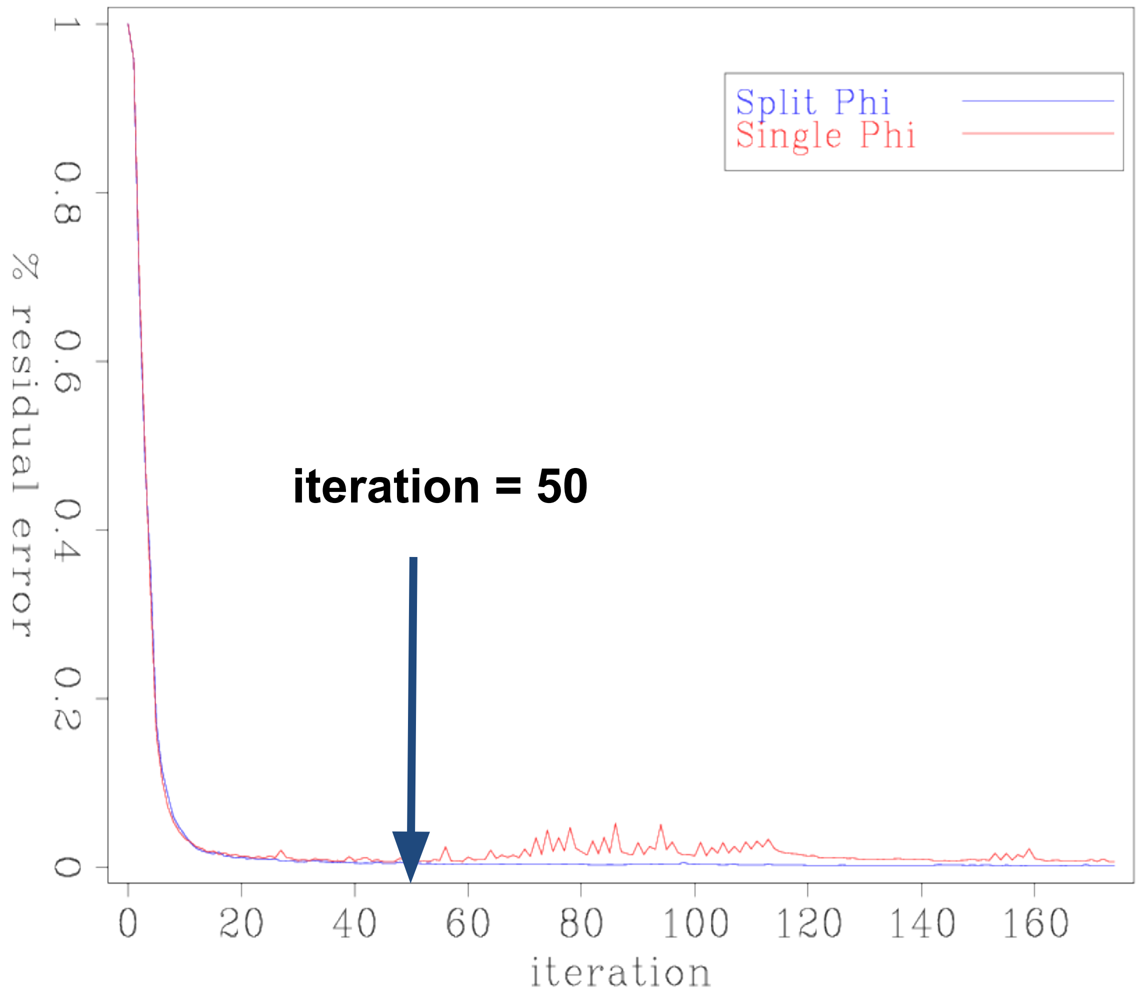
Split algorithm % vel error

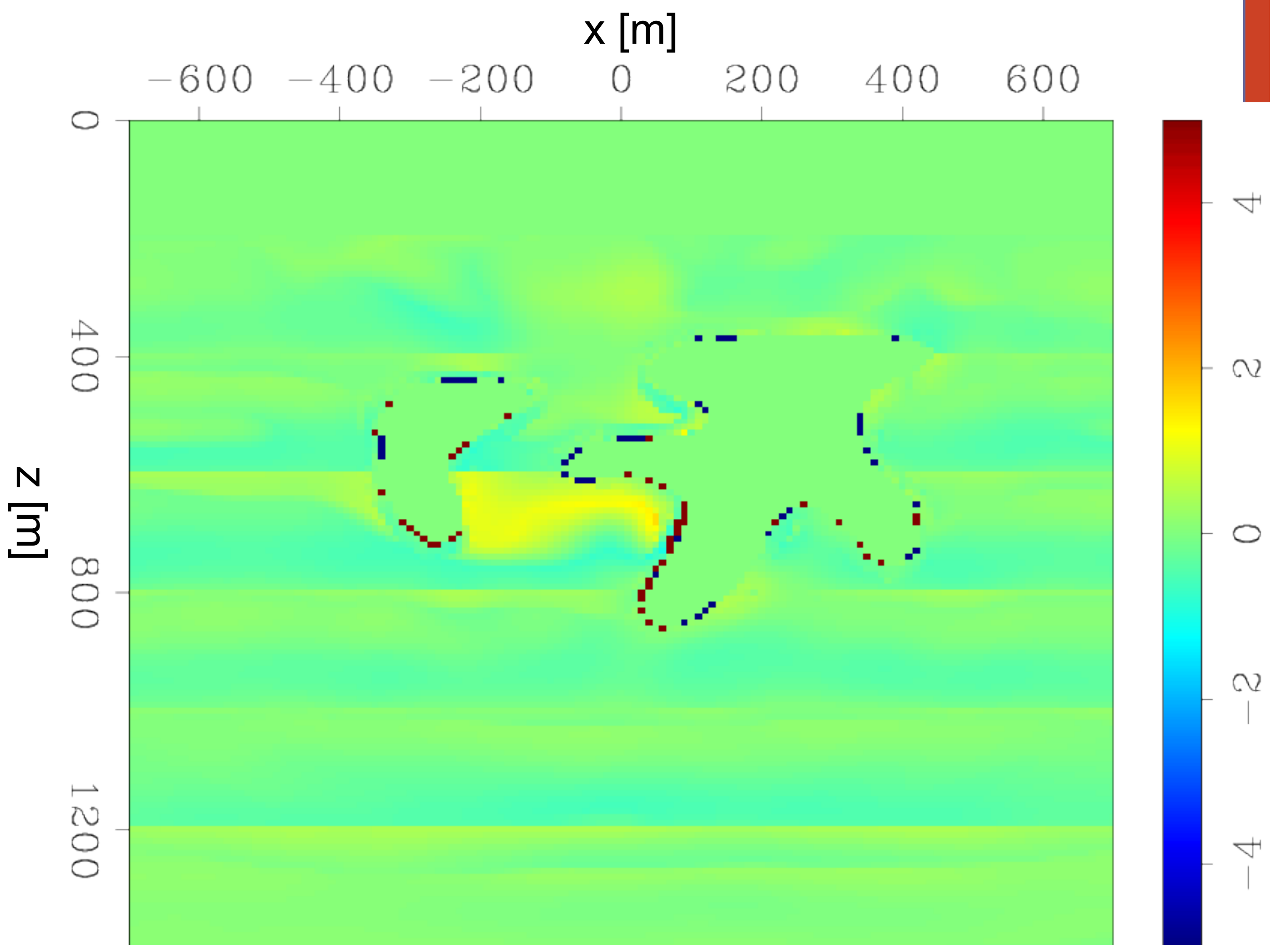
General algorithm % vel error

Green = good match

iteration = 175

Domain decomposition comparison





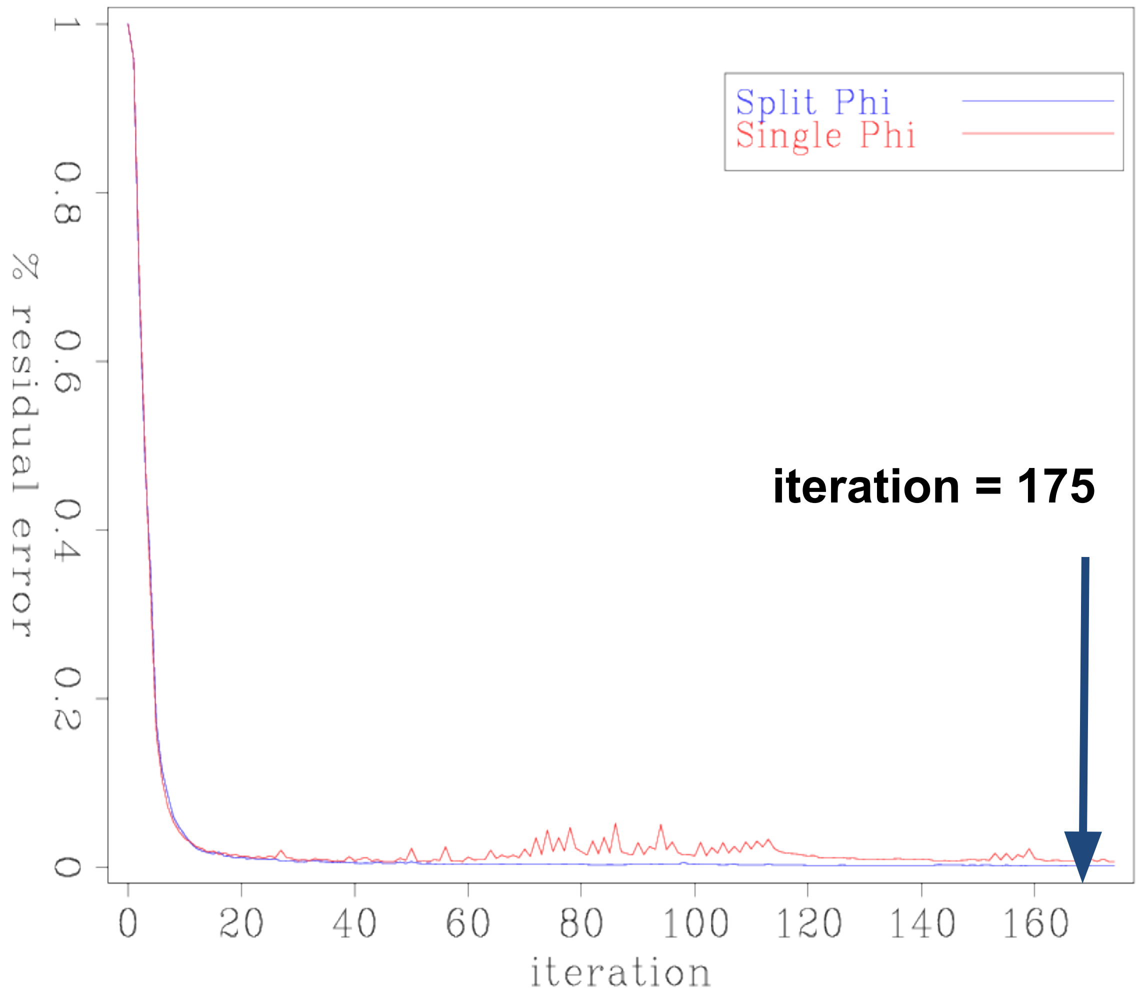
% velocity error differential between methods

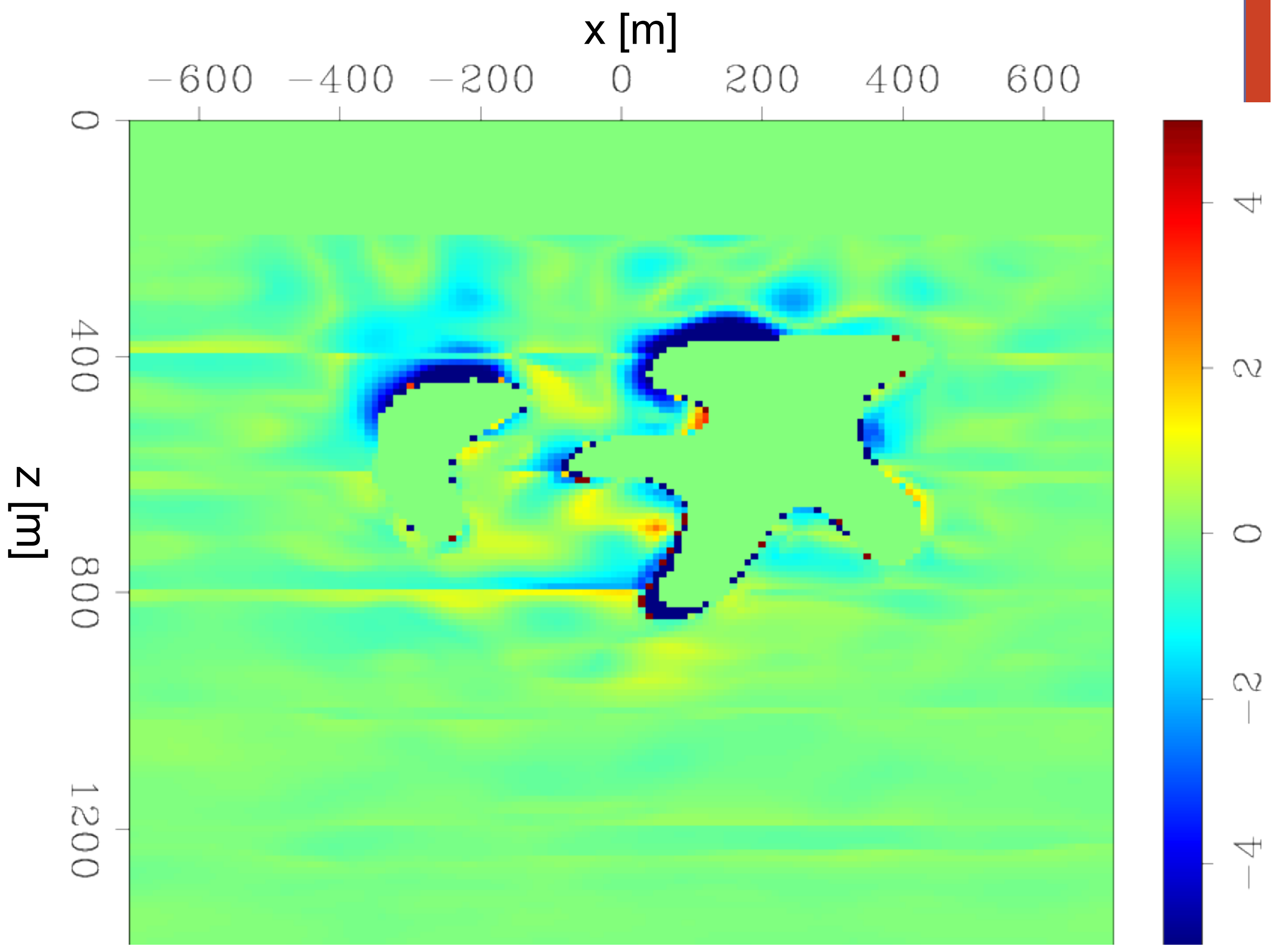
RED: Split phi does **WORSE** job

BLUE: Split phi does **BETTER** job

+

Domain decomposition comparison



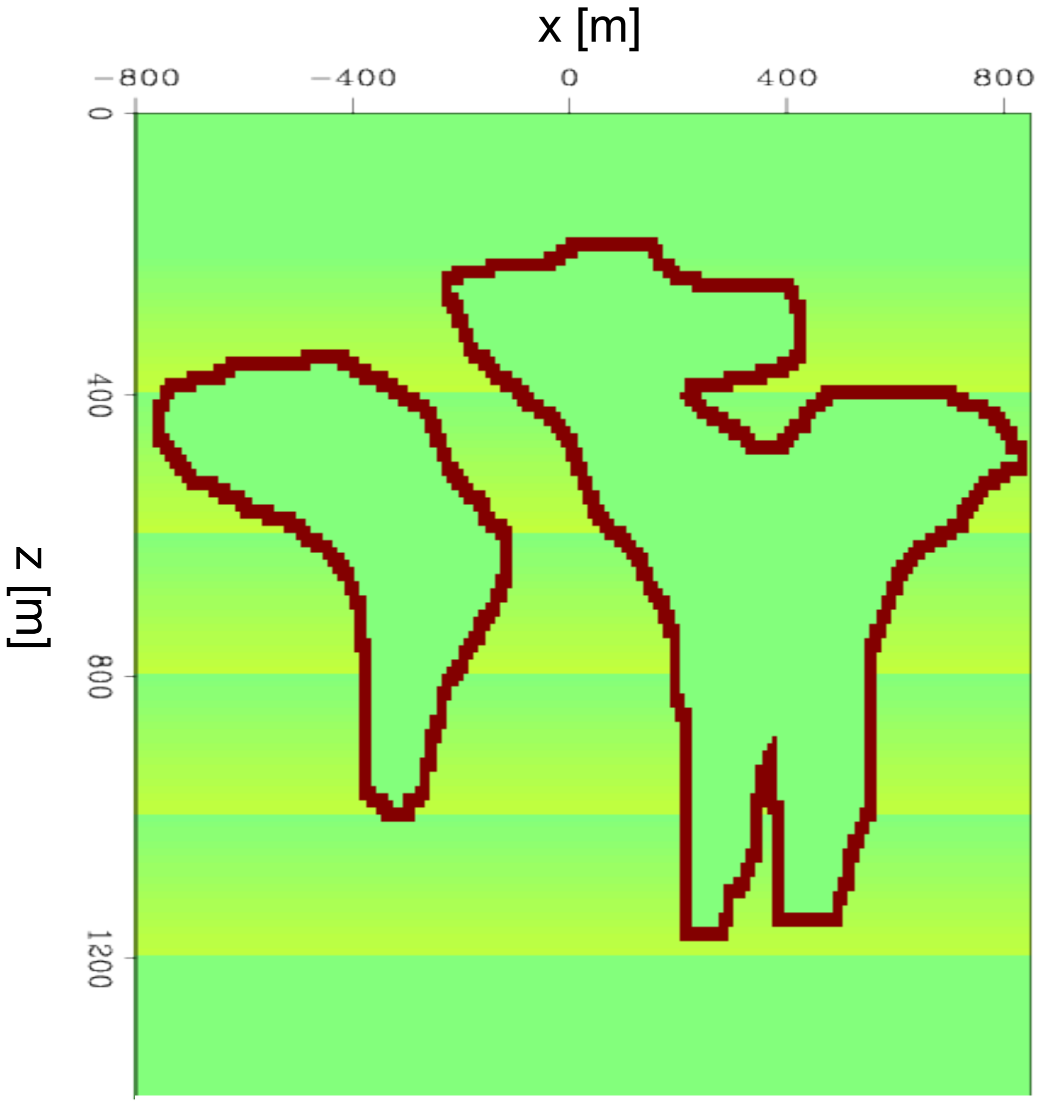


% velocity error differential between methods

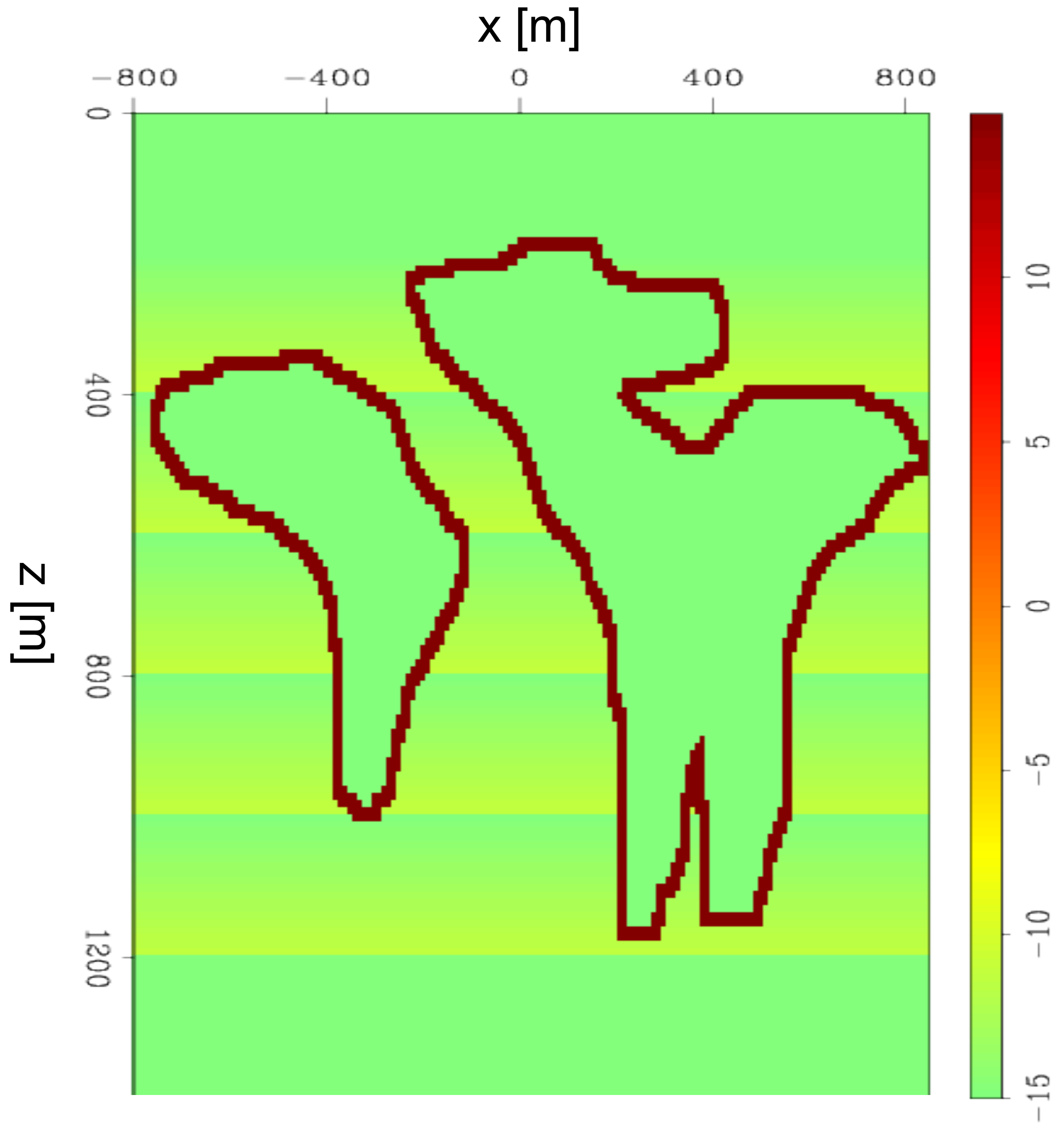
RED: Split phi does **WORSE** job

BLUE: Split phi does **BETTER** job

Domain decomposition comparison



Split algorithm % vel error

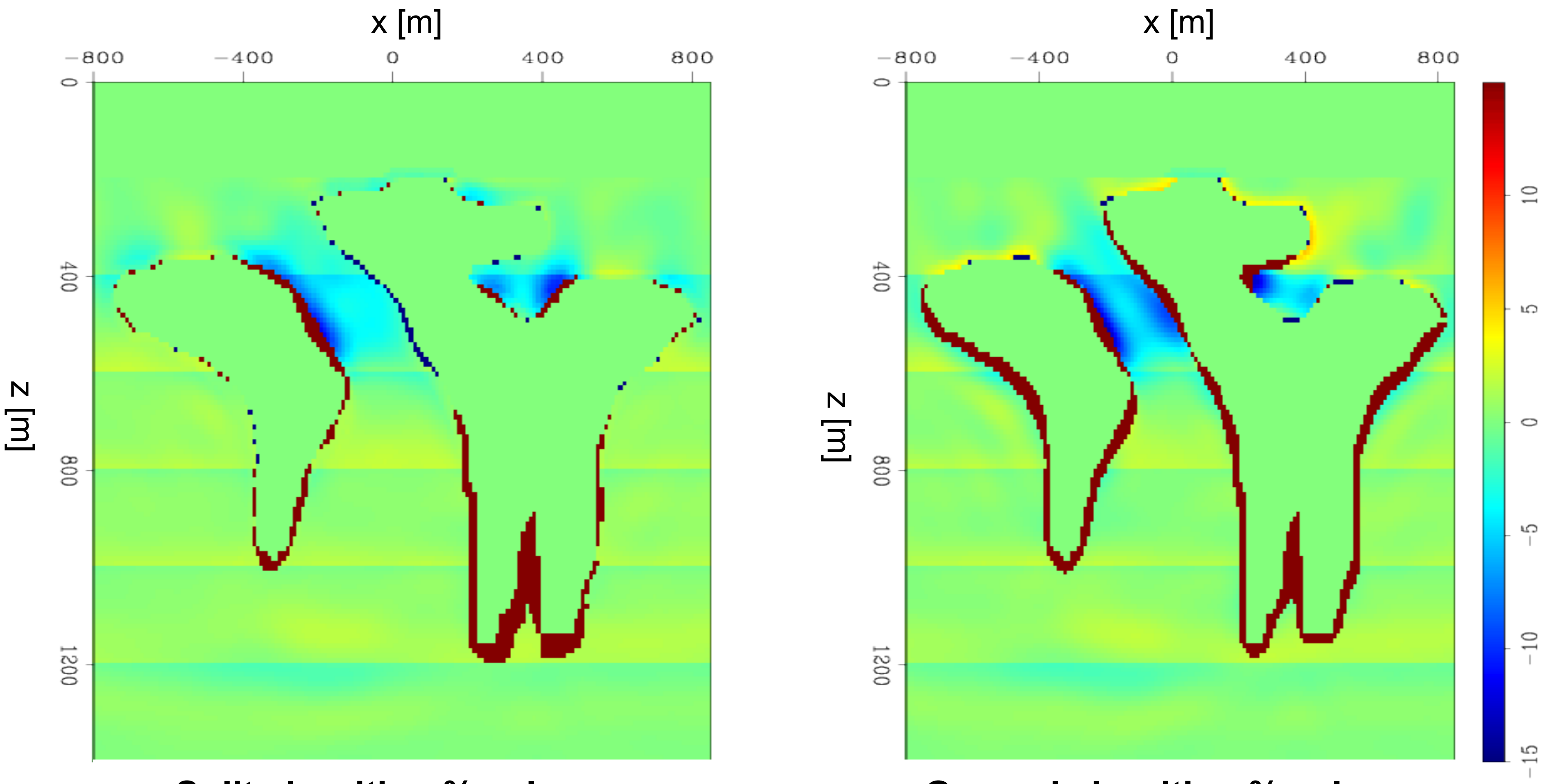


General algorithm % vel error

Green = good match

iteration = 0

Domain decomposition comparison



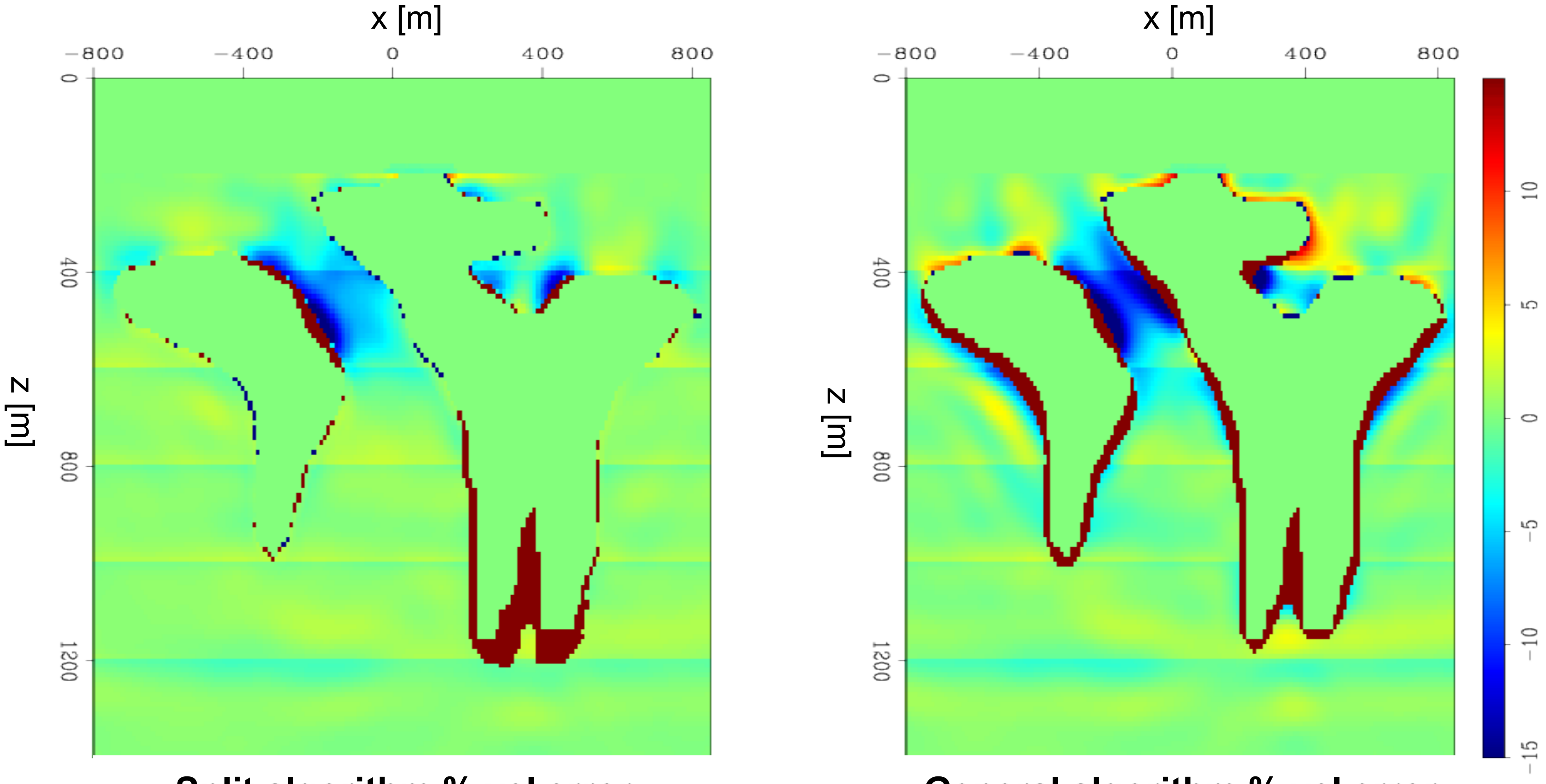
Split algorithm % vel error

General algorithm % vel error

Green = good match

iteration = 50

Domain decomposition comparison



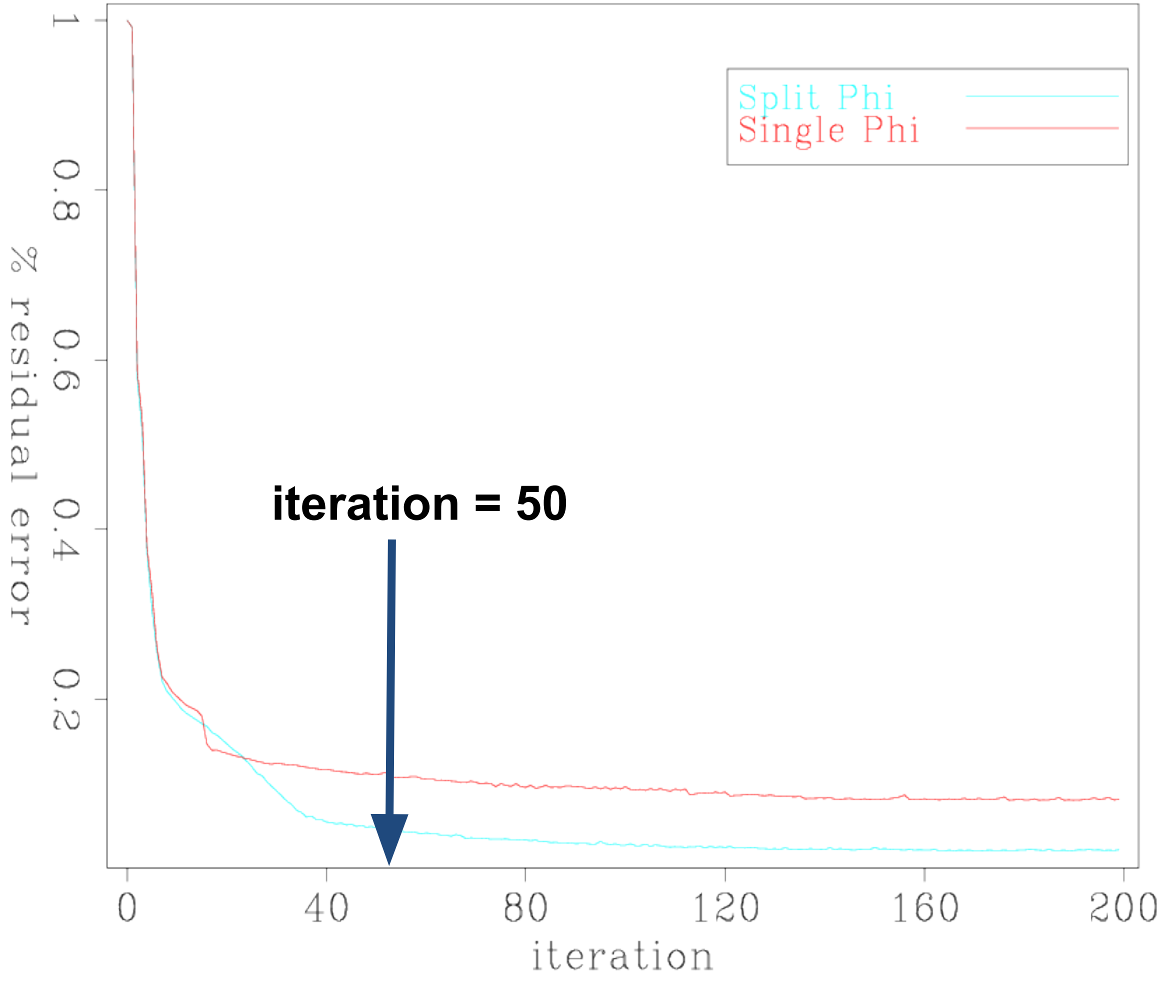
Split algorithm % vel error

General algorithm % vel error

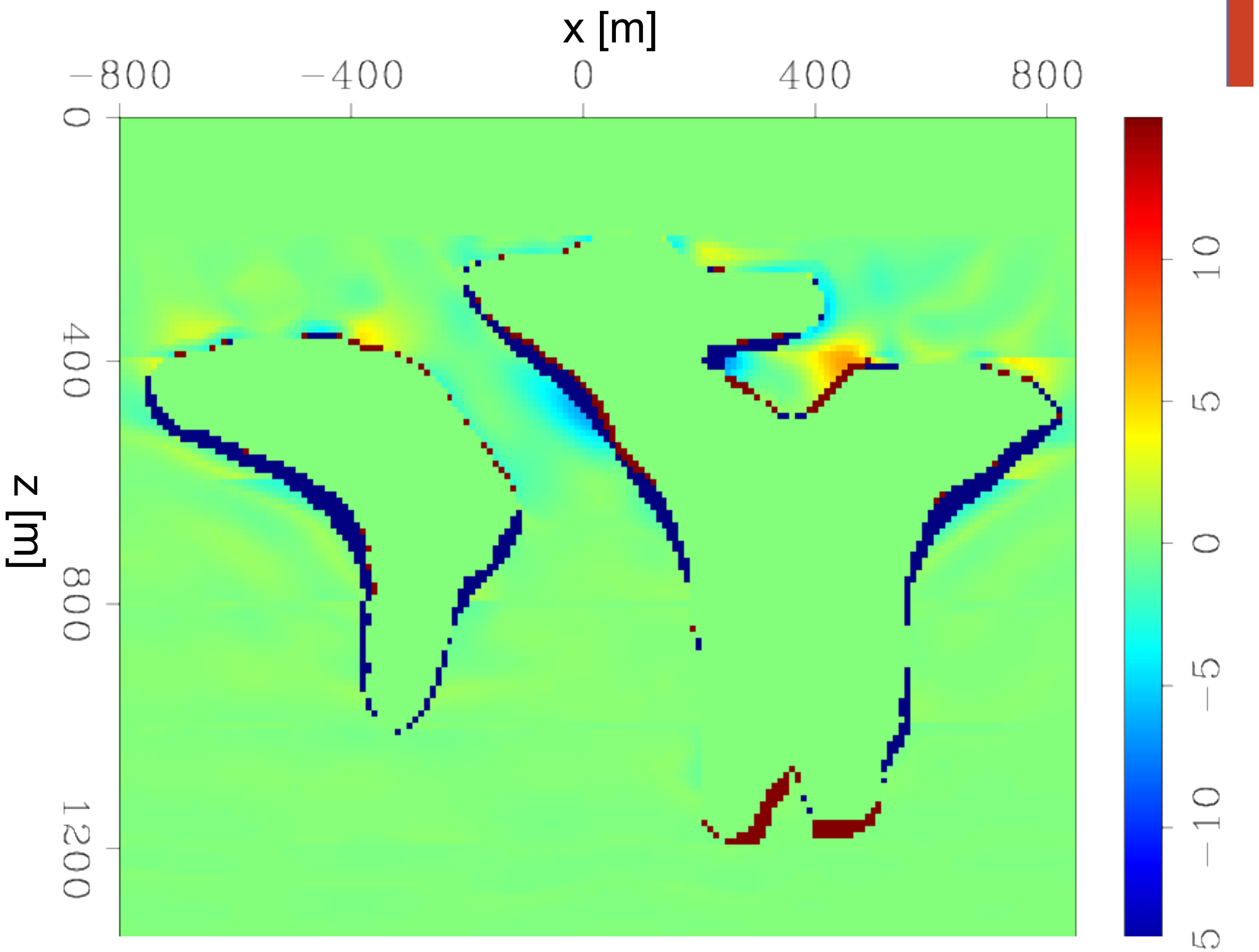
Green = good match

iteration = 100

Domain decomposition comparison



iteration = 50

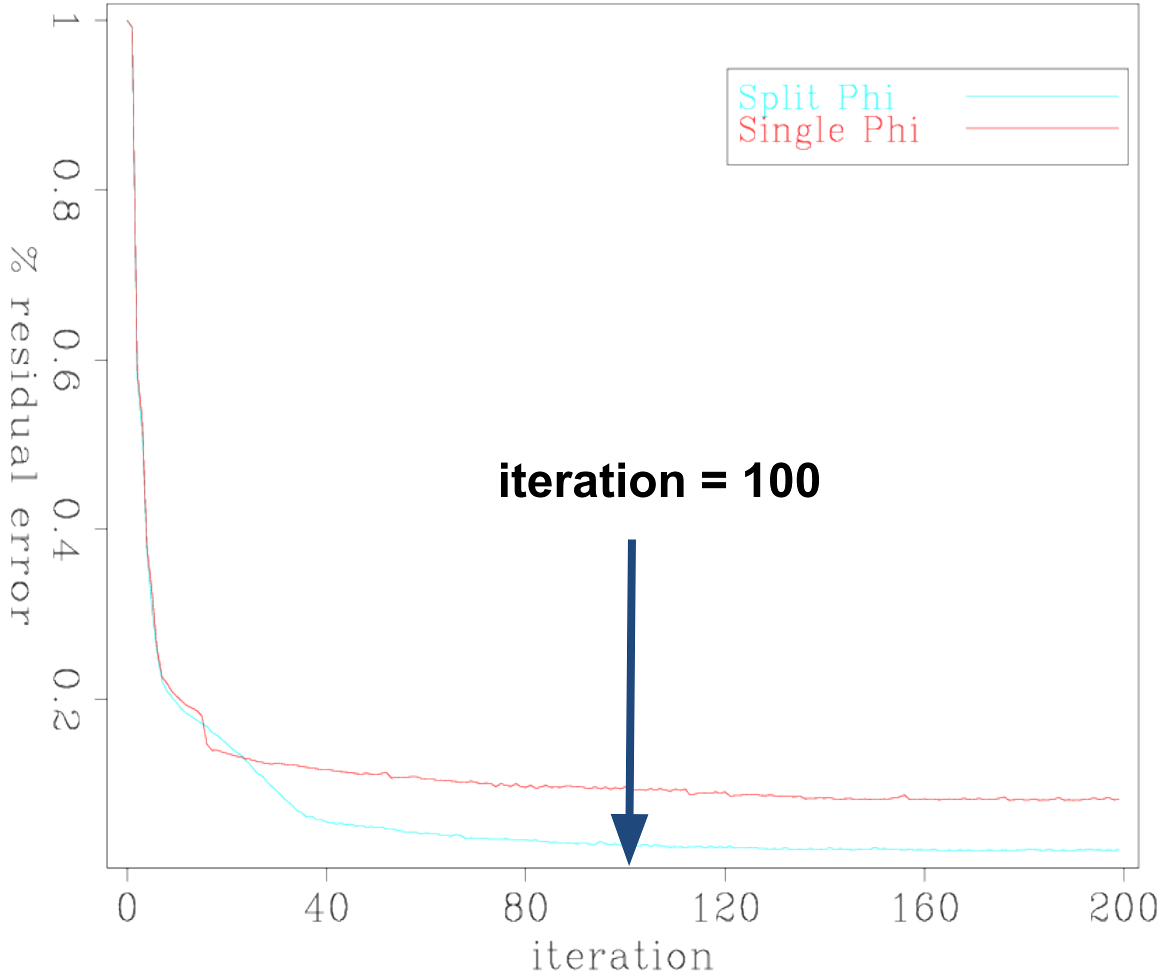


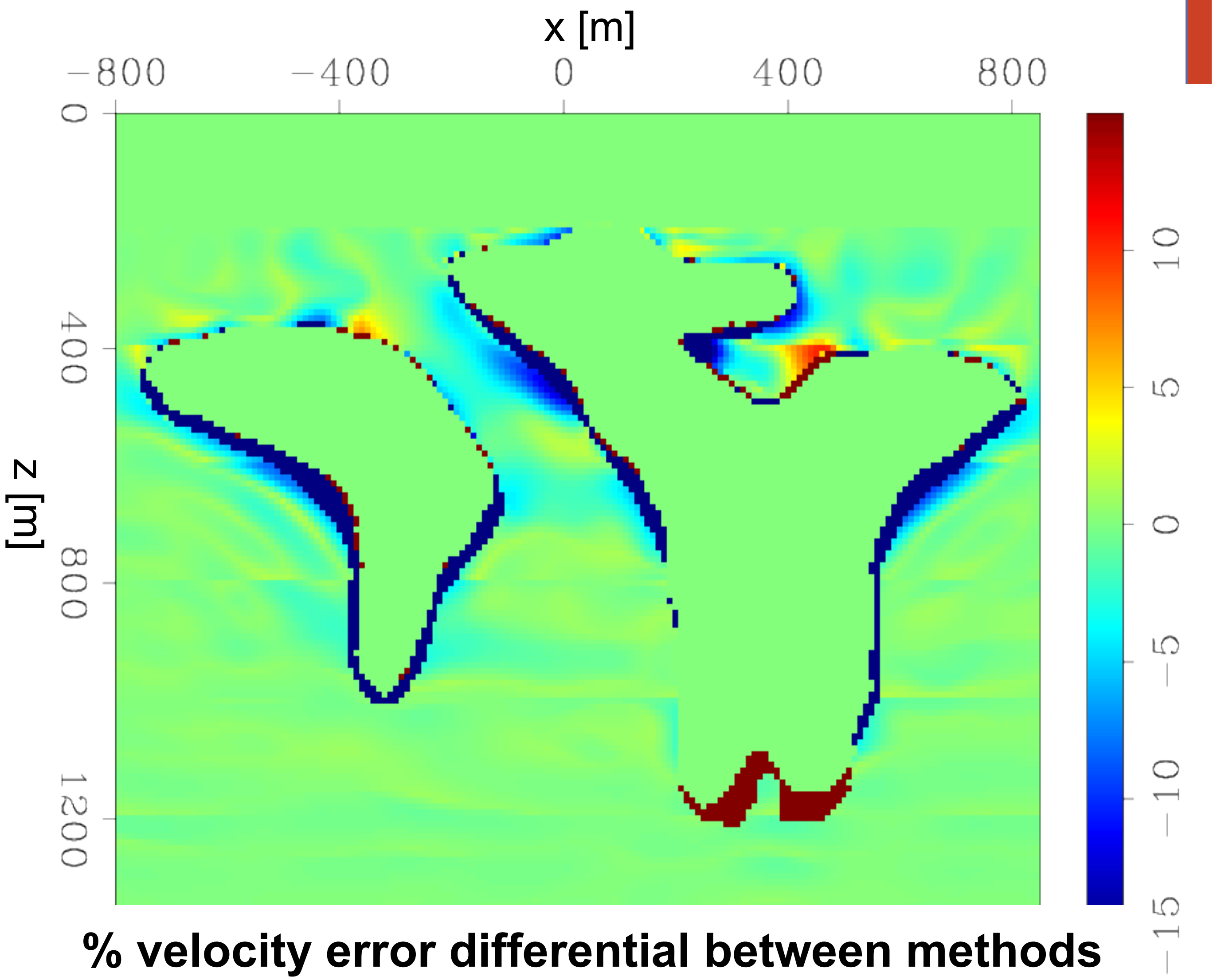
% velocity error differential between methods

RED: Split phi does **WORSE** job

BLUE: Split phi does **BETTER** job

Domain decomposition comparison



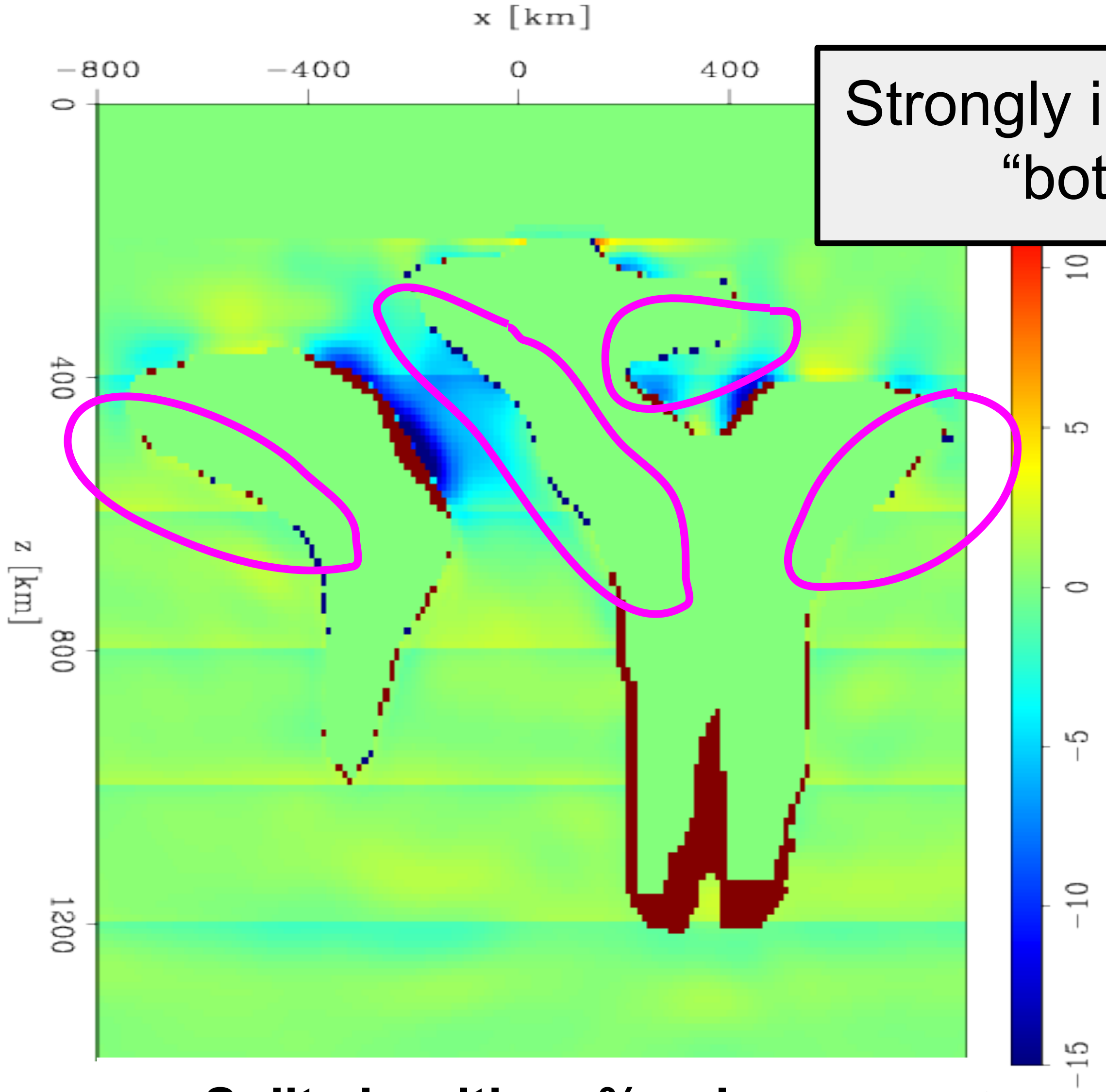


% velocity error differential between methods

RED: Split phi does **WORSE** job

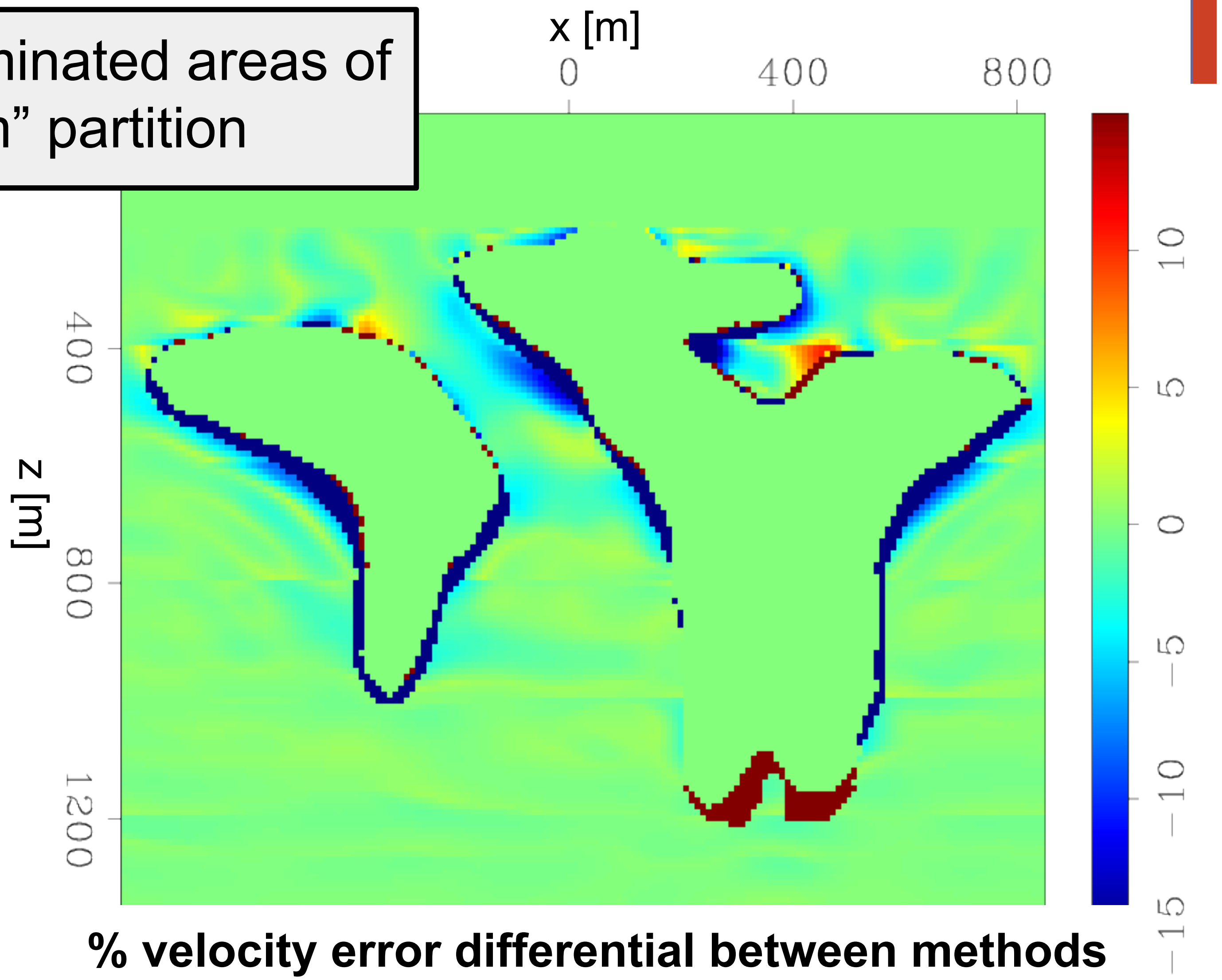
BLUE: Split phi does **BETTER** job

Domain decomposition comparison



Strongly illuminated areas of “bottom” partition

Split algorithm; % vel error

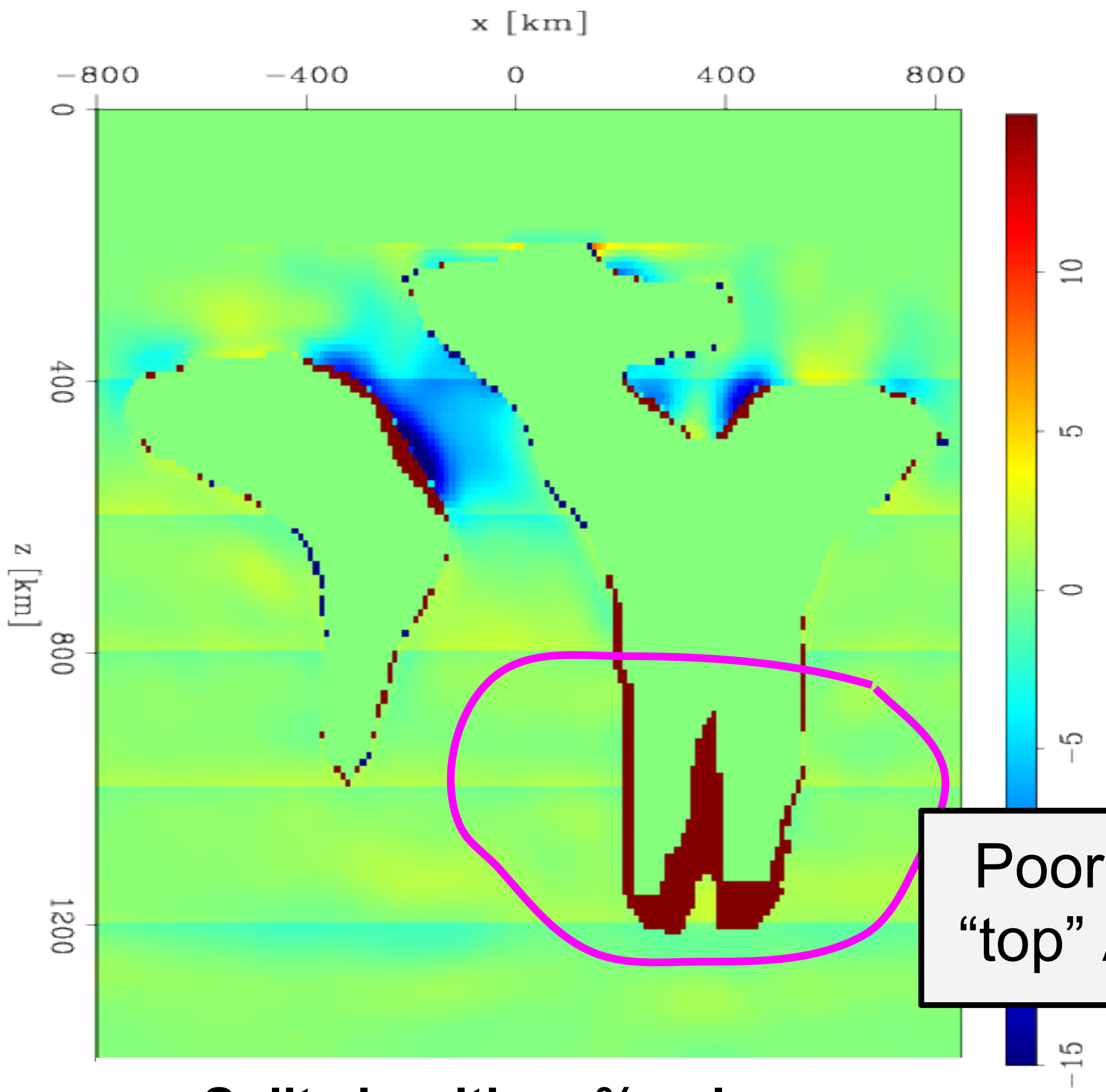


% velocity error differential between methods

RED: Split phi does **WORSE** job

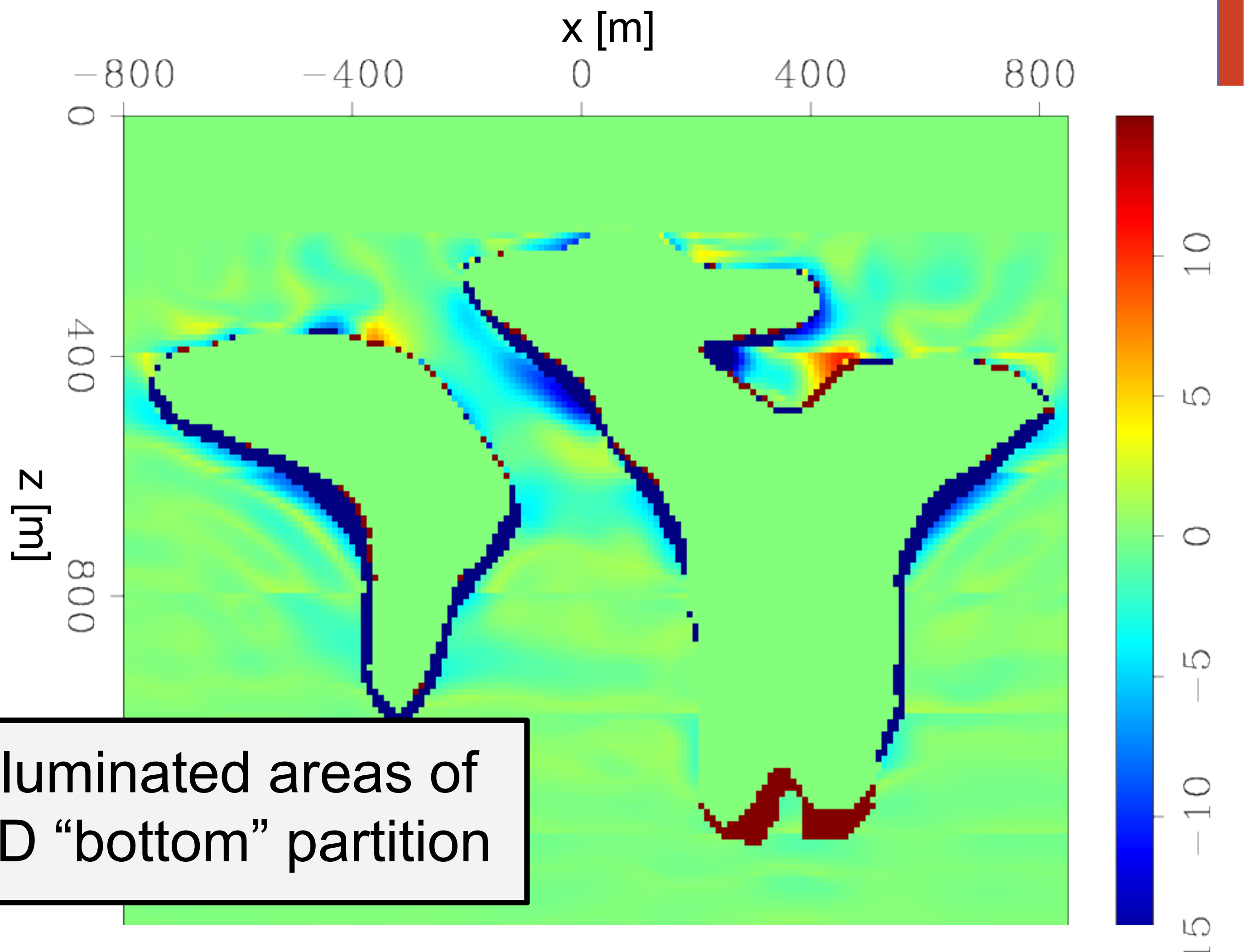
BLUE: Split phi does **BETTER** job

Domain decomposition comparison



Split algorithm; % vel error

Poorly illuminated areas of “top” AND “bottom” partition



% velocity error differential between methods

RED: Split phi does **WORSE** job

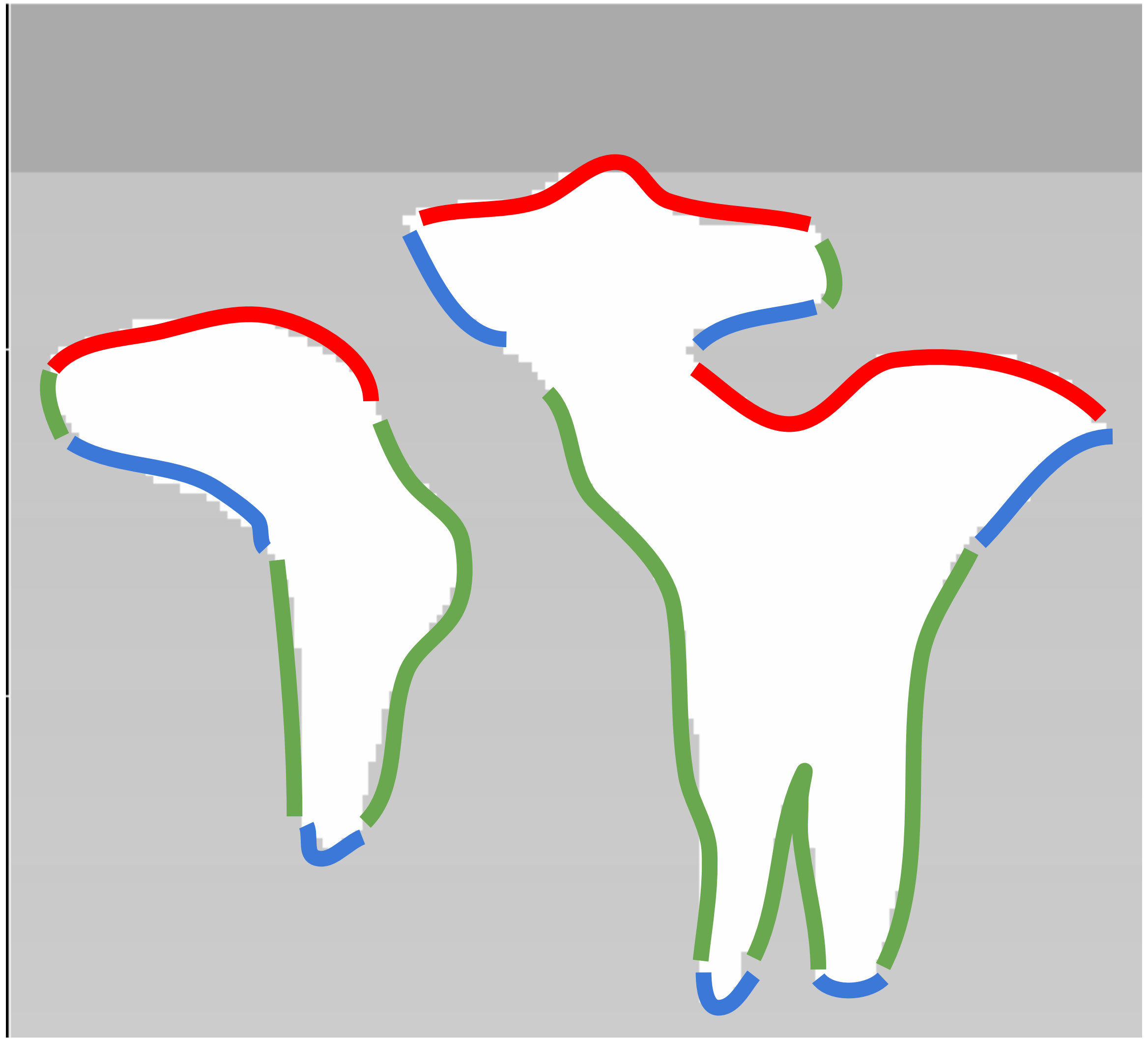
BLUE: Split phi does **BETTER** job

Future work

- Better splitting / domain decomposition method (takes into account illumination, etc).

- More than two decomposed domains.

    - For example; **top**, **base**, **flanks.**



Summary

- Level set theory offers distinct advantages to identifying salt boundaries.
- Domain decomposition allows for more accurate convergence on true salt.
- Expanding this method to more partitions could further improve the convergence on the flanks and base of salt.

Acknowledgements

- Current SEP students (Ali, Musa, Yang, Ohad).
- Recent SEP alumni (Sjoerd, Adam, Mandy, Xukai, Elita).