Second international Workshop on Deepening Performance Models for Automatic Tuning (DPMAT 2017), August 28-31, Nagoya, Japan.

Optimizing Forward and Backward computations in the adjoint method via Multi-level Blocking

Tomoya Ikeda[1], Shin-ichi Ito[2], Hiromichi Nagao[2, 3], Takahiro Katagiri[4], Toru Nagai[4], and Masao Ogino[4]

[1] *Graduate School of Information Science, Nagoya University*

[2] *Earthquake Research Institute, The University of Tokyo*

[3] *Graduate School of Information Science and Technology, The University of Tokyo*

[4] *Information Technology Center, Nagoya University*

Outlines

1. Summary of Data Assimilation
2. Summary of the Adjoint Method
3. Optimization via Multi-level blocking (our proposed method)
4. Experiments
5. Results
6. Conclusion and future works

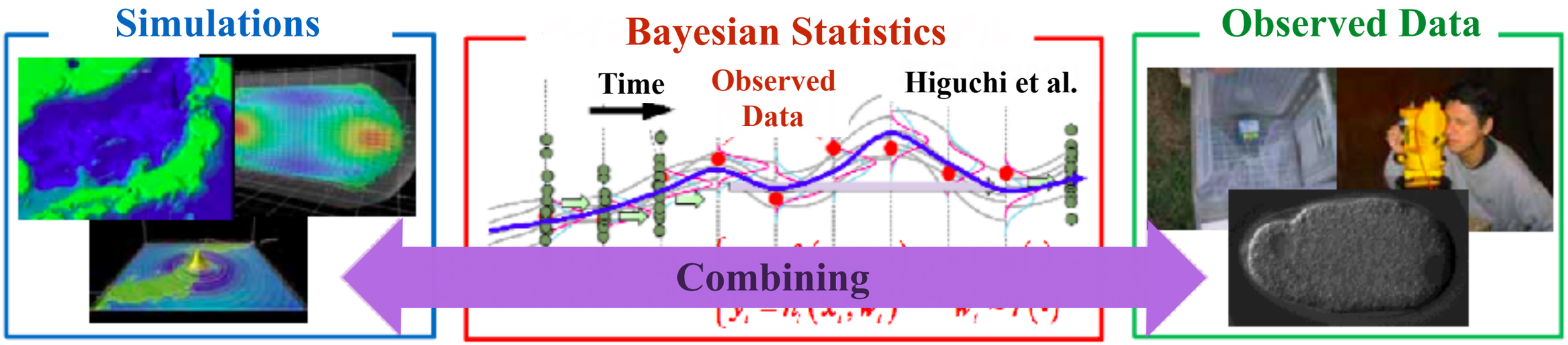
1

Summary of Data Assimilation

Data Assimilation (DA)

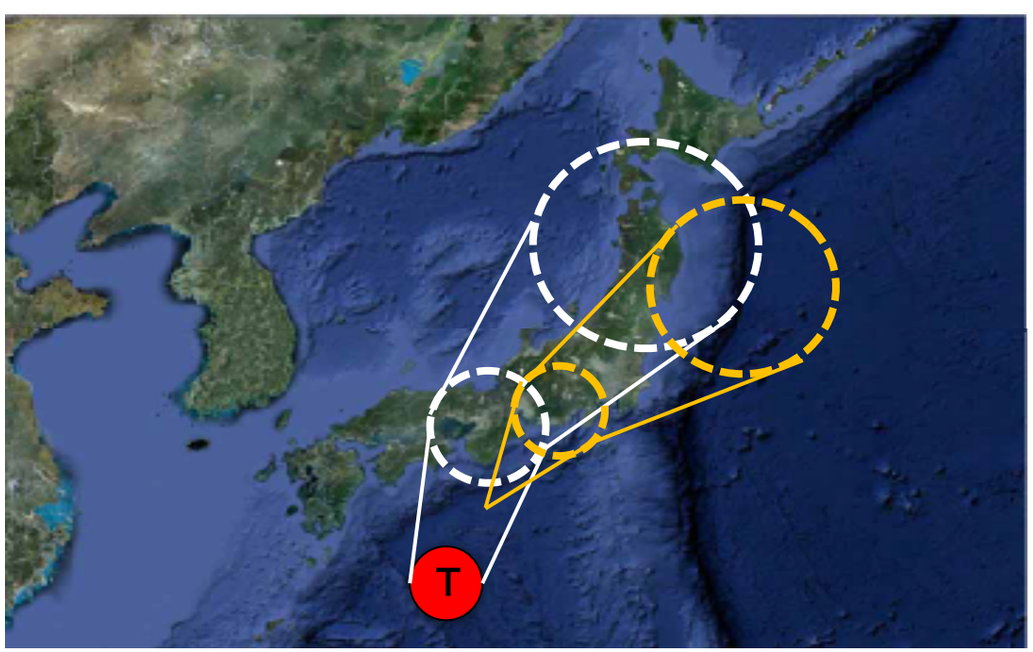
Data Assimilation (DA) :

a method of combining computer simulations and observed data based on Bayesian statistics.

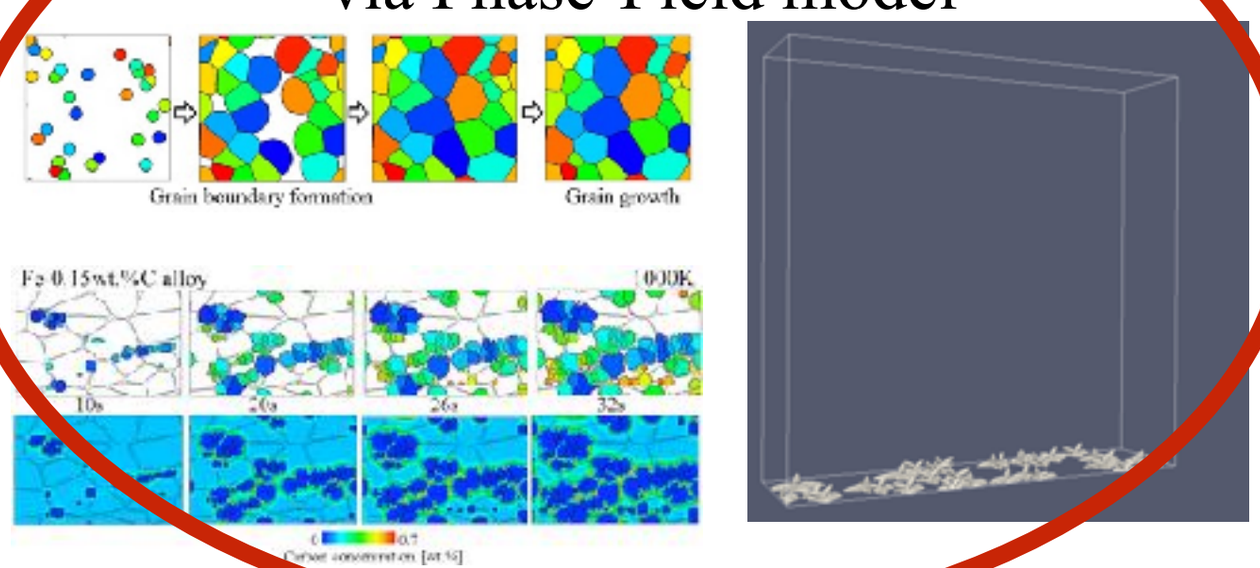


Application Examples

Prediction of a tornado's path



Prediction of physical properties via Phase-Field model



Reference : http://web.tuat.ac.jp/~yamanaka/research.html

Sequential DA vs Non-sequential DA

- **Sequential DA :** Search for **the entire** state and model parameter space and find the estimate of state and uncertainty at each time steps.

  Pros : Not only the state, but also uncertainty can be obtained.

  Cons : Computational cost is $e^{(O(N))}$.

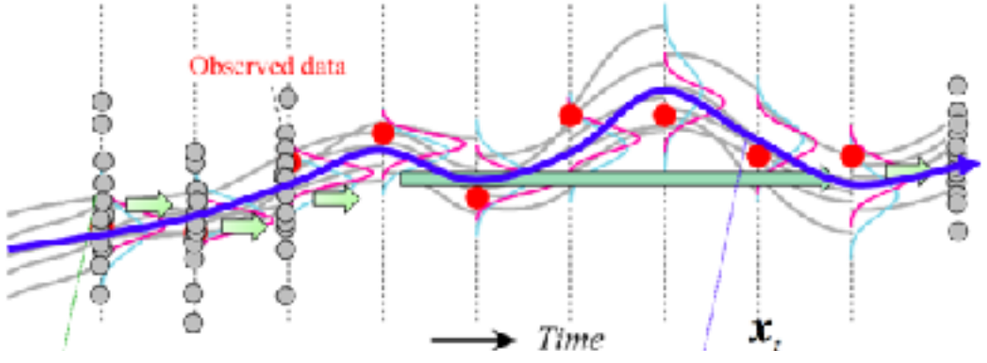
  the degree of freedom $\boldsymbol{N}$ = state variables + model parameters

  Example :
  the Kalman Filter (KF), the Ensemble Kalman Filter (EnKF), the Particle Filter (PF).

- **Non-sequential DA :** Search for **a certain point** in the state and model parameter space where the posterior distribution becomes the maximum and find the estimate of state within a given time window.

  Pros : Computational cost is $O(N)$.

  Cons : Uncertainty cannot be obtained.

  Example : the adjoint method (AM) a.k.a. 4-D var.

Motivation : Necessity the performance improvement

- As the number of the degree of freedom increases, computation becomes very larger scale.

- Even the AM, computations cannot be executed <u>within a realistic time</u> without improvement of performance.

- As for the AM, performance implementation through computation blocking has <u>rarely been considered</u> thus far.

- In order to achieve better performance, we propose **Multi-level Blocking** for the AM.

2

Summary of the Adjoint Method

Algorithm of the AM

Begin

Set appropriate initial value Θ

Forward Computation (FC)

Perform a computation for simulation F with Θ

Backward Computation (BC)

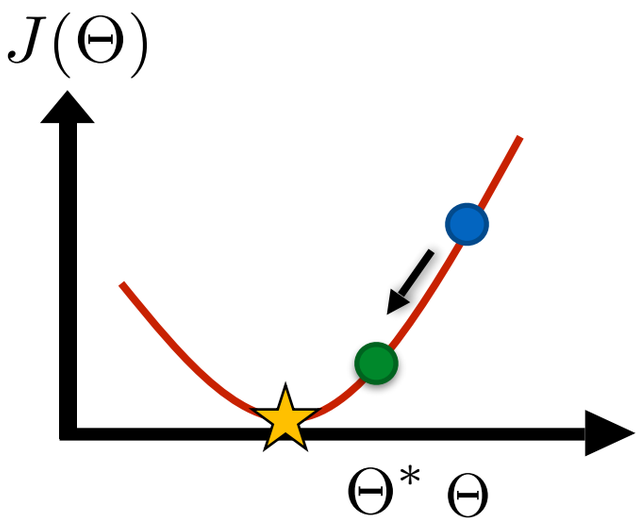
Compute the difference between simulation F and observed data

Obtain gradient vector $\dfrac{\partial J}{\partial \Theta}$

update Θ by applying gradient method

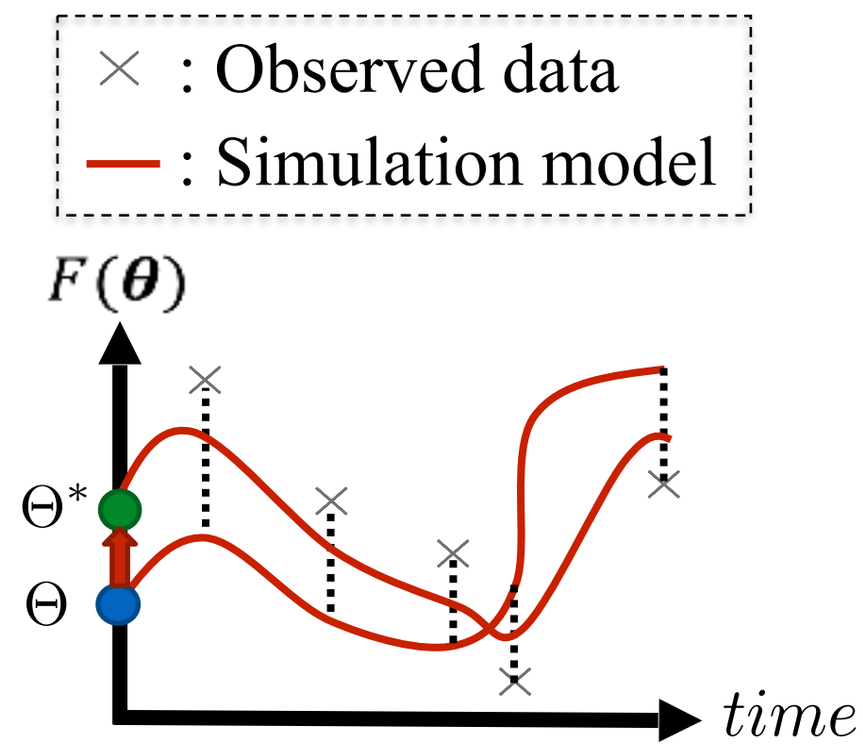
Minimize a cost function $J(\Theta)$

Obtain a modified initial value Θ^*



$\dfrac{\partial J}{\partial \Theta}$

Θ^*



Target model : Phase-field model

A simple phase-field model

R. Kobayashi (1993)

$$\tau \frac{\partial \phi}{\partial t} = \epsilon^2 \Delta \phi + \phi(1-\phi)\left(\phi - \frac{1}{2} + m\right) \quad , \quad |m| < \frac{1}{2}$$

Diffusion term

Reaction term

Model parameters

τ	Time unit
ϵ	Space unit
m	Growth speed

(Constant value in all spaces)

Given parameters : τ , ϵ

<u>Unknown parameter</u> : m

Phase $\phi(x,t)$

- $\phi(x,t)$ represents the state of grid point *x* at time *t*.

$$\begin{cases} m = const. \\ \phi(-\infty, t) = 1, \phi(\infty, t) = 0 \end{cases}$$

$$\phi(x,t) = \frac{1}{2}[1 - tanh(\frac{x}{2\sqrt{2}\epsilon} - \frac{mt}{2\tau})]$$

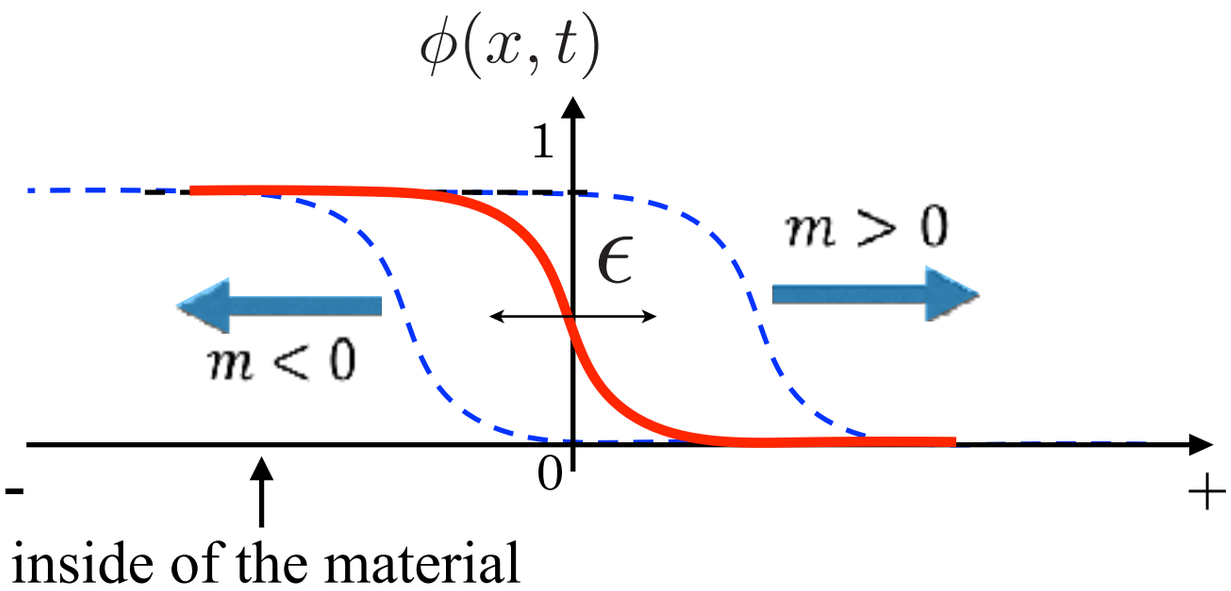
Reference : Ito et al., “Data assimilation for massive autonomous systems based on a second-order adjoint method” (2016)

Illustration of our target model

A simple phase field model

$$\tau \frac{\partial \phi}{\partial t} = \epsilon^2 \Delta \phi + \phi(1-\phi)\left(\phi - \frac{1}{2} + m\right) \quad , \quad |m| < \frac{1}{2}$$

1-Dim



2-Dim

The objective is

To estimate initial states $\phi(x,0)$ and parameter m

Reference : Ito et al., “Data assimilation for massive autonomous systems based on a second-order adjoint method” (2016)

Problem needs to be solved

- In order to estimate initial states and a parameter simultaneously, time evolution equation of parameter *m* is introduced to the model equation :

$$\begin{cases} \dfrac{\partial \phi}{\partial t} = \dfrac{\epsilon^2}{\tau}\Delta\phi + \dfrac{1}{\tau}\phi(1-\phi)\left(\phi - \dfrac{1}{2} + m\right) \\ \dfrac{\partial m}{\partial t} = 0 \end{cases} \qquad |m| < \frac{1}{2} \qquad 0 < \phi(x,0) < 1$$

- Combining phase $\phi(x,t)$ and parameter *m* as follow :

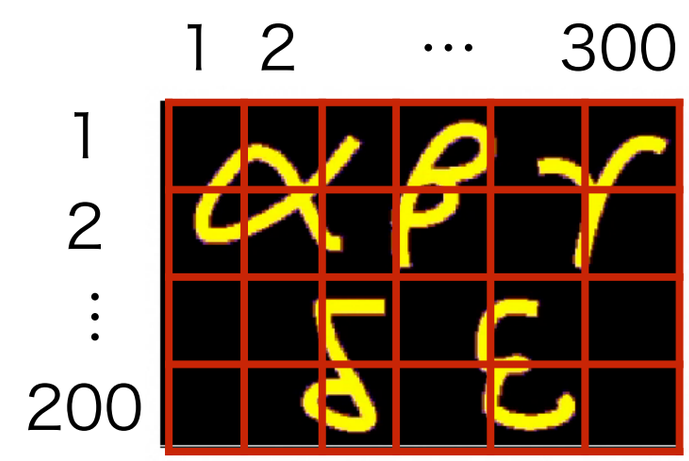
$$\theta(t) = (\phi(t)^T, m + \frac{1}{2})^T$$

s. t. $\quad 0 < \theta_i(0) < 1$

E.g. Discretized space by 60,000 grid points

$$\phi(t) = (\phi_1, \phi_2, \ldots, \phi_{60000})^T$$

The problem is to search $\theta(0) = \Theta$ that best fit the data, subject to $\dfrac{\partial \theta}{\partial t} = F(\theta)$, $0 < \theta_i(0) < 1$



Reference : Ito et al., “Data assimilation for massive autonomous systems based on a second-order adjoint method” (2016)

Discretized equations in FC

$$\begin{cases} \dfrac{\partial \phi}{\partial t} = \dfrac{\epsilon^2}{\tau}\Delta\phi + \dfrac{1}{\tau}\phi(1-\phi)\left(\phi - \dfrac{1}{2} + m\right) \\ \dfrac{\partial m}{\partial t} = 0 \end{cases}$$

Diffusion term

Discretized

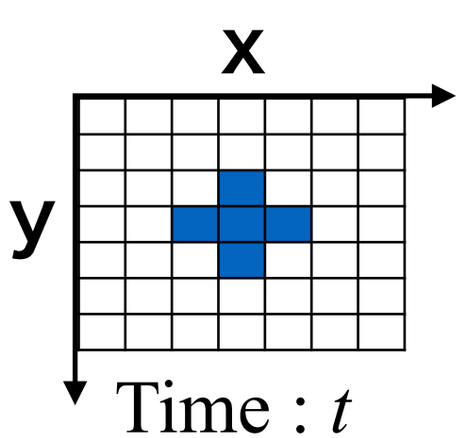
$$\begin{cases} \dfrac{\partial \theta_j}{\partial t} = \displaystyle\sum_{k=1}^{M} \Delta_{jk}\theta_k + \theta_j(1-\theta_j)(\theta_j + \theta_{M+1} - 1) \\ \dfrac{\partial \theta_{M+1}}{\partial t} = 0 \end{cases}$$

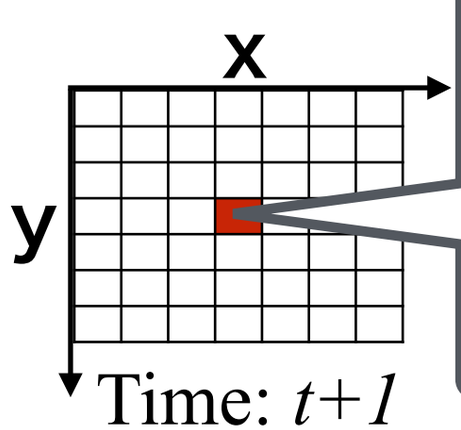
Laplacian operator

$$\begin{pmatrix} -4 & 1 & 0 & 1 \\ 1 & -4 & 1 & 0 \\ 0 & 1 & -4 & 1 \\ 1 & 0 & 1 & -4 \end{pmatrix}$$

- We adopt Finite Difference Method (FDM) to compute these equations.

- In order to solve linear equations with the explicit method of FDM, **stencil computations** are necessary.





Updating the **red grid point** requires **5 blue grid points**

Other discretized equations in the AM

Discretized equations in BC

$$-\tau\frac{\partial\lambda_i}{\partial t}=\begin{cases}\epsilon^2\Delta_i\lambda_i+\{-3\theta_i^2+(4-2\theta_{M+1})\theta_i+\theta_{M+1}-1\}\lambda_i+\frac{\partial\mathcal{J}}{\partial\theta_i} & \textit{for i=1,...,M,}\\ \sum_{j=1}^{M}\theta_j(1-\theta_j)\lambda_j & \textit{otherwise,}\end{cases}$$

$$\lambda(0)=\frac{\partial J}{\partial\Theta}\quad,\quad\lambda(t_f)=0$$

where J and t_f represents a cost function and the end time of simulation, respectively.

Equation of the cost function

$$J=\int_0^{t_f}dt\mathcal{J},$$

$$\mathcal{J}=\frac{1}{2}\sum_{t_s\in\tau}\delta(t-t_s)\sum_{i=1}^{M}(\phi_i^{obs}(t_s)-\phi_i(t_s))^2$$

- The value of cost function represents the amount of divergence between simulation model and observed data.
- We search the optimal initial value by minimizing the value.

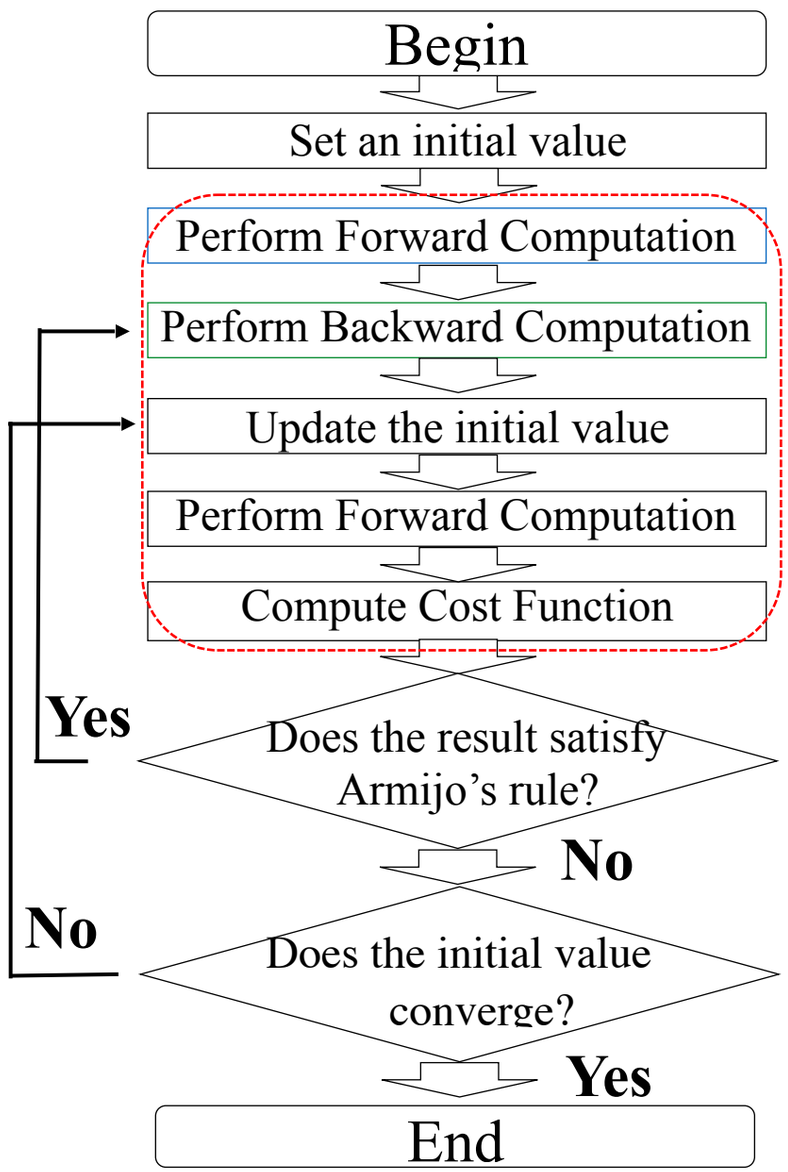
3

Optimization via Multi-level Blocking

Proposed method : Multi-level Blocking

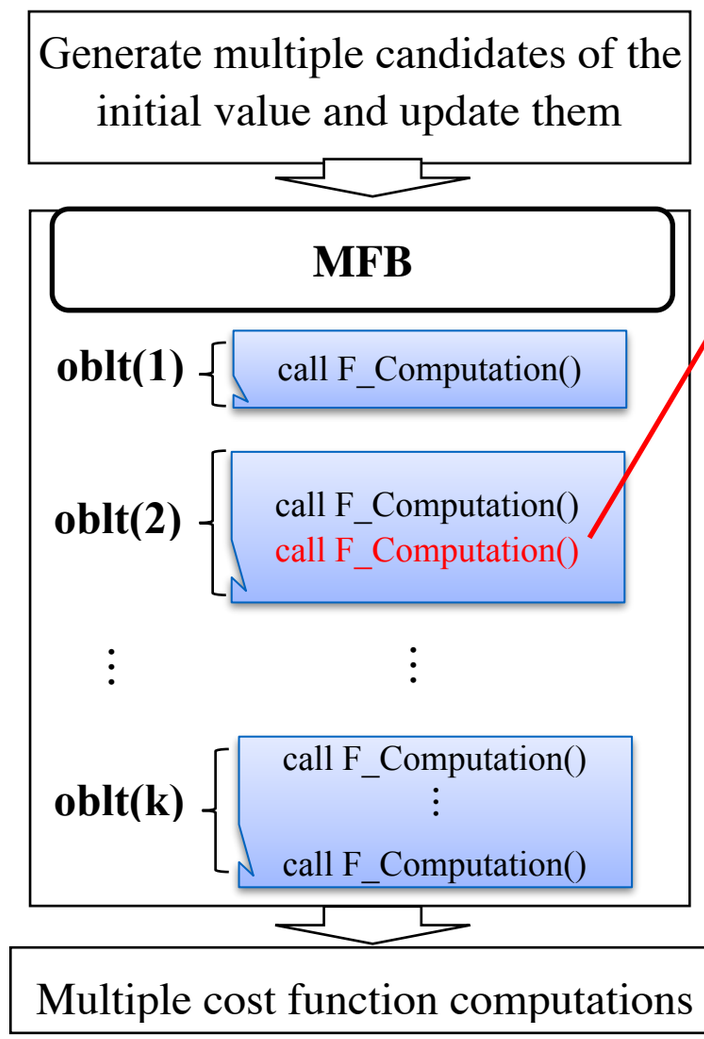
Conventional method

These processes will be executed sequentially



Proposed method

Multi-level Blocking = Spatial and Temporal Blocking (**STB**) + Multiple Forwards Blocking (**MFB**)



Apply **STB** to stencils in FC and BC

FC

```
subroutine F_Computation()
 do
   update grid point values
 end do
end subroutine F_Computation
```

BC

```
subroutine B_Computation()
  do
    update grid point values
    if (Observed data exist) then
     compute the difference between
     a simulation and observed data
    end if
  end do
end subroutine B_Computation
```

Spatial and Temporal Blocking (STB)

Naïve

```
subroutine f_computation_naïve()
    do it = 0, nda
        do y = 1, ny
            do x = 1, nx
                update A(x,y,it+1) by using A(x-1,y,it), A(x+1,y,it),
                    A(x,y-1,it), A(x,y+1,it) and A(x,y,it)
```

nda : The number of total time steps
ny : The number of y-axis grid points
nx : The number of x-axis grid points

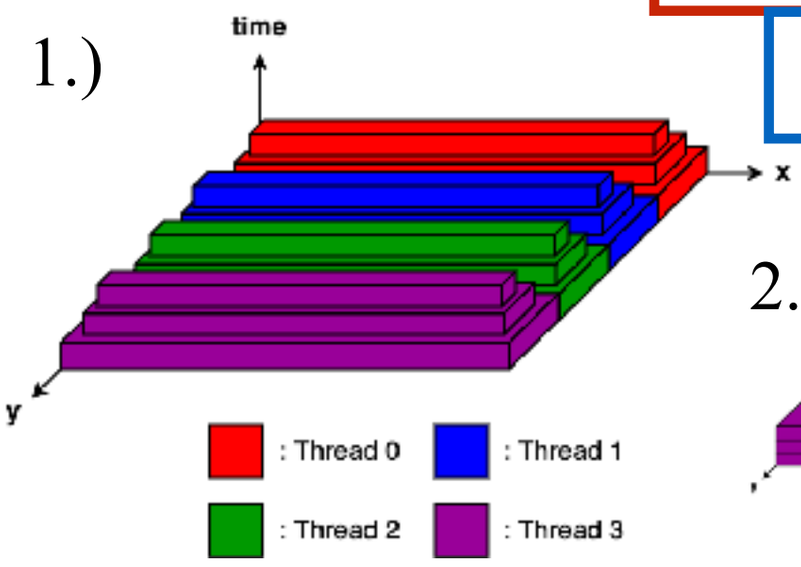
After applying STB

```
subroutine f_computation_propose()
    do it = 0, nda, blt
        do yy = 1, y_tiles
            do xx = 1, x_tiles
                do t = it, it+iblt-1
                    do y = yhead(yy), ytail(yy)
                        do x = xhead(xx), xtail(xx)
                            update A(x,y,t+1) by using A(x-1,y,t), A(x+1,y,t),
                                A(x,y-1,t), A(x,y+1,t) and A(x,y,t)
```

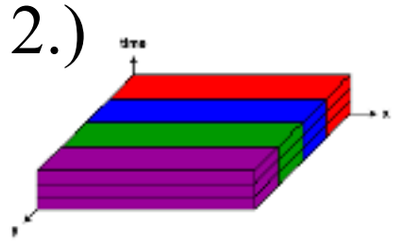
Applying temporal blocking

Applying spatial blocking for y-axis and x-axis

1.)



2.)

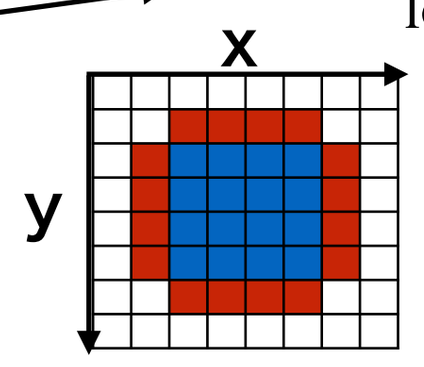


Perform 2 STEPs:

1.) Compute grid point values pyramidally

2.) Compute the rest of grid point values

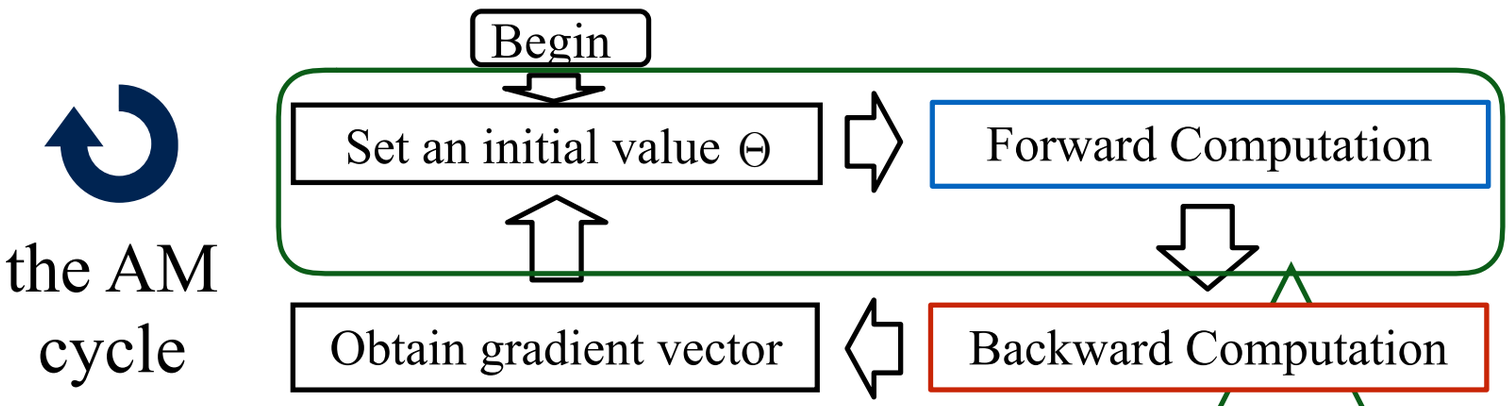
- Computation peformed without grid points located in halo region



Blue : Computational region

Red : Halo region

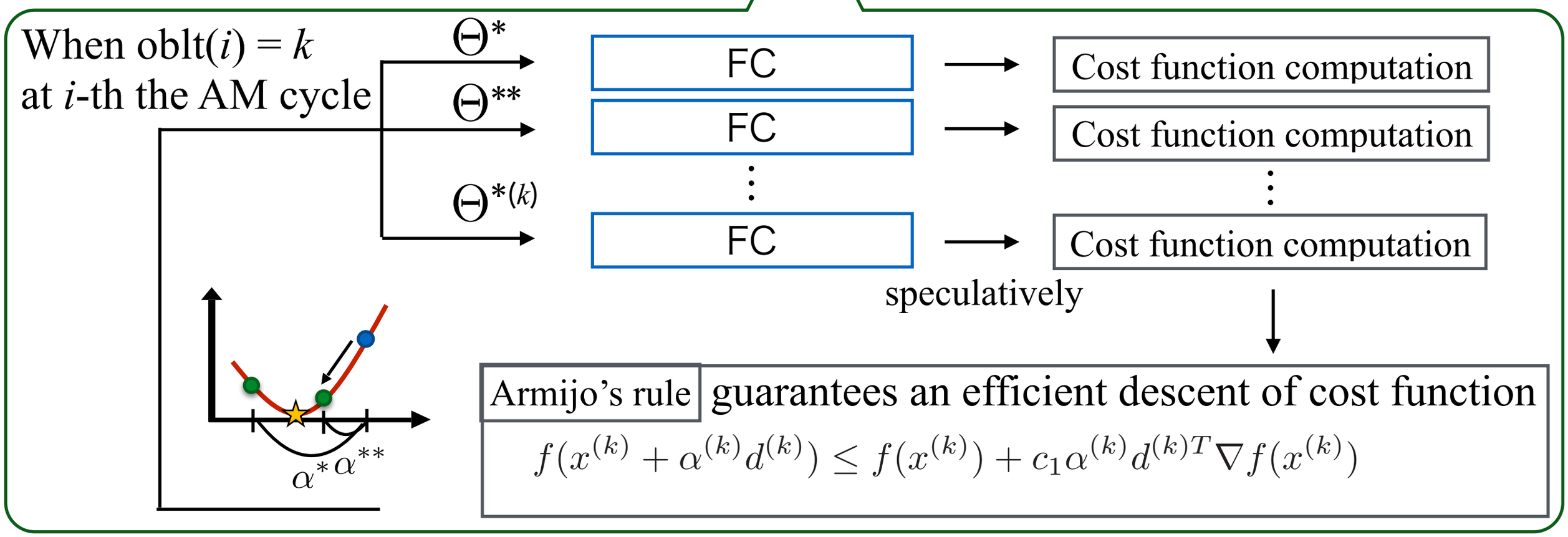
Multiple Forwards Blocking (MFB)



The MFB size is represented as follow :

$$\mathrm{oblt}(i) = j$$

j is the block size at the i-th AM cycle.



- Since there is no dependency between multiple initial values, multiple candidates of the initial value can be generated.

→ Multiple FCs and cost function computations can be executed in parallel.

4

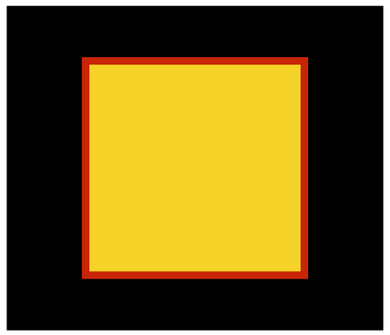
Experiments

Experimental and Problem Settings

Problem settings

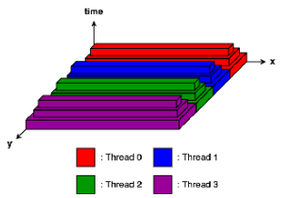
The parameter effects on the growth speed of boundary

- We set twin problem to estimate the parameter m of the square shaped phase.



Outside phase : 0.00
Inside phase : 0.10
Boundary : 0.90

- Problem size was (nx×ny) = (1600×1600), the number of time step was 128.
- Initial guess of the parameter was set to -0.10, 0.00 and 0.09.
- True value of the parameter was set to 0.10 (the boundary moves to positive direction) .



Experimental settings

- Spatial blocking size : **y_tiles** was set to the number of threads and **x_tiles** was set to 1. The shape of division will be rectangular.
- Temporal blocking size : **blt** was chosen from 1 to 50 exhaustively.
  (Note that maximum **blt** is 32 and 25 when the number of threads is 25 and 32, respectively.)
- MFB blocking size : **oblt**(1) = 1, **oblt**(i) = 2 (i=2,3,4), and **oblt**(j) = 3 (j > 4)

Target Machine : Fujitsu PRIMEHPC FX100

Fujitsu PRIMEHPC FX100

	System Composition
CPU	SPARC64 XIfx, 2.2GHz 32(+2) Cores
Memory	32GB per Node (HMC is adopted)
L1 Cache (4way)	L1 : 64KB (Instruction/Data separated per core)
L2 Cache	L2 : 24MB (Shared)
Theoretical peak performance	1.1264 TFlops (Double precision)
Total number of nodes	2,880 Nodes (92,160 Cores)
Total performance	3.2 PFlops/sec
Type	NUMA (2 sockets per node, 16 cores per socket)
Theoretical Bandwidth	480GB/sec per node
Compiler	Fujitsu Fortran90 Compiler
Compiler Options	-Kfast -Kopenmp

Compiler version :

Fujitsu Fortran Compiler Version 2.0.0 P-id: T01776-01 (Aug 30 2016 16:49:17)

Memory allocation policy : *localalloc* is adopted

5

Results

6

Conclusion and future works

Conclusion and Future works

Conclusion

- We proposed **Multi-level blocking** to improve the performance of the AM.
- The performance was improved by a factor of **7.52** and **7.20** compared to the baseline in FC and BC.
- By applying STB to FC and BC, **1.87** times and **1.37** times speed-ups were achieved, respectively.
- As the total improvement of the AM application, **1.18** times speed-up was gained.

Future works

- To establish Hybrid-MPI, we implement the process-level parallelism via MPI.
  - The performance may be improved even if we apply Pure-MPI.
- Making computations in MFB executed simultaneously.
  - Since computations in MFB is executed sequentially in the current implementation.