7th ITER International School, Aix-en-Provence, 25-28, Aug. 2014

Numerical Methods Used in Fusion Science Numerical Modeling

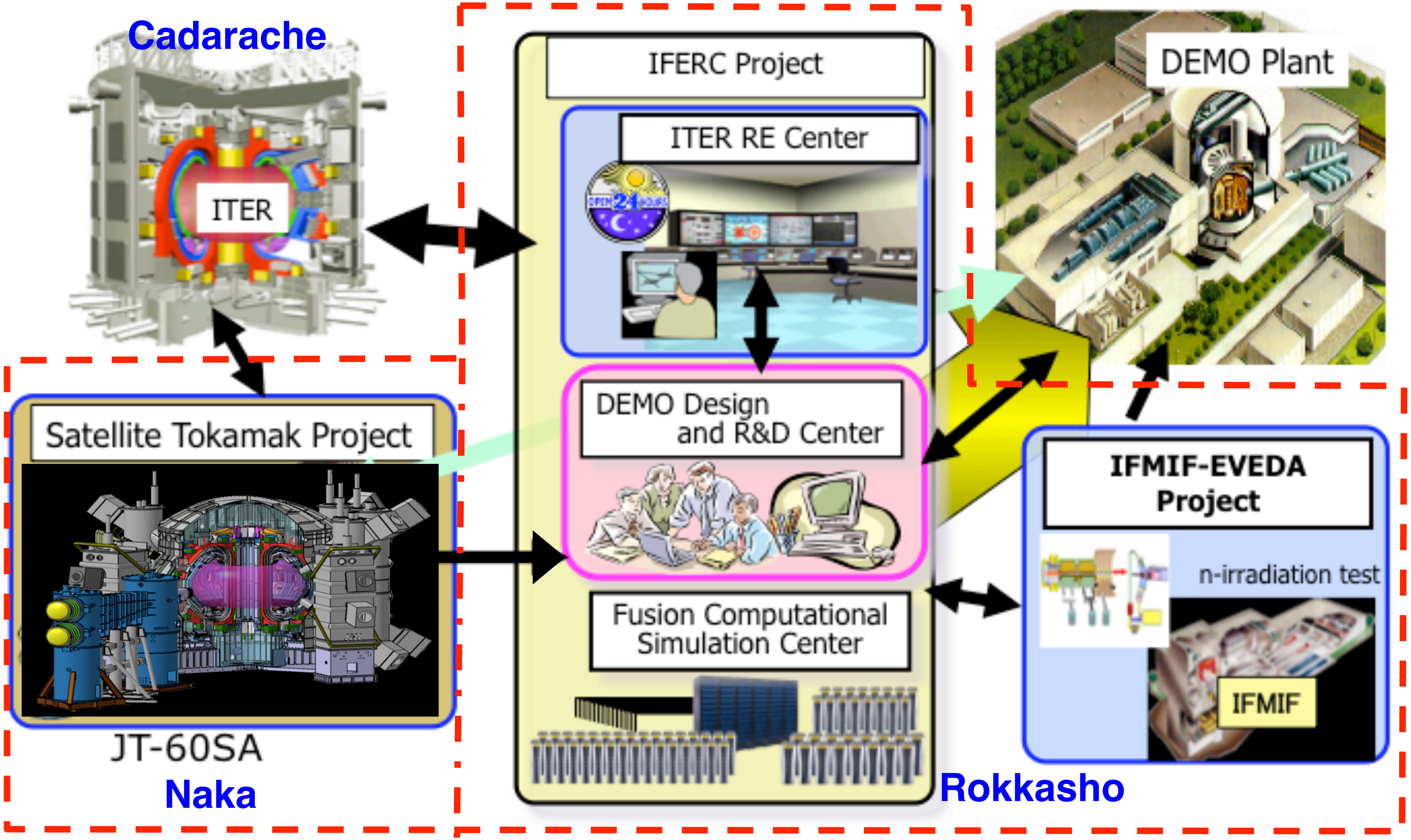
Masatoshi Yagi

Japan Atomic Energy Agency, Rokkasho Fusion Institute

Quick Tour in BA Rokkasho Site

Broader Approach (BA) Activities

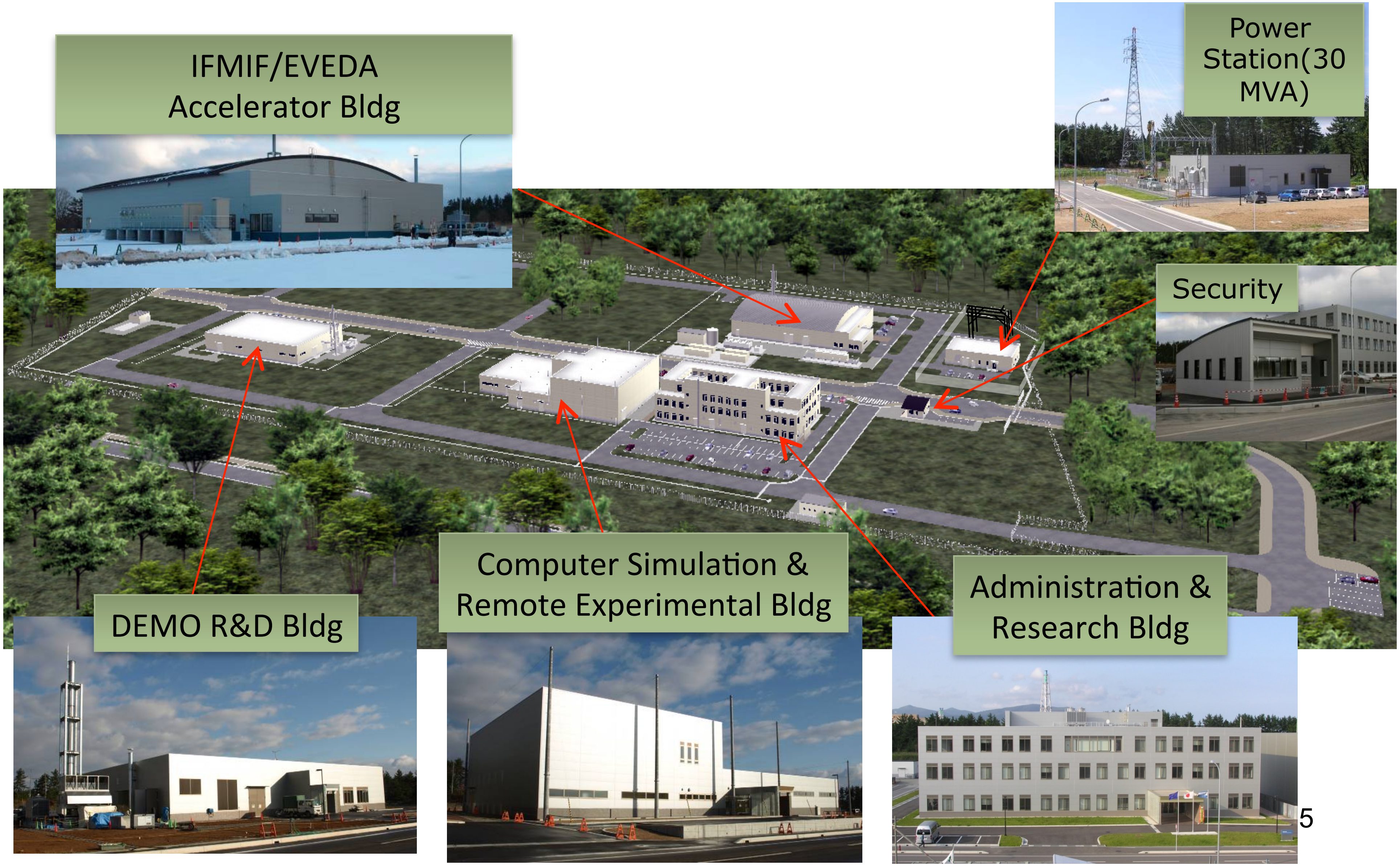
In parallel to the ITER program, BA activities are being implemented by the EU and Japan, aiming at early realization of the fusion energy



Rokkasho BA site

Rokkasho BA site

◆ **Constructions of all buildings were completed by March 2010.**



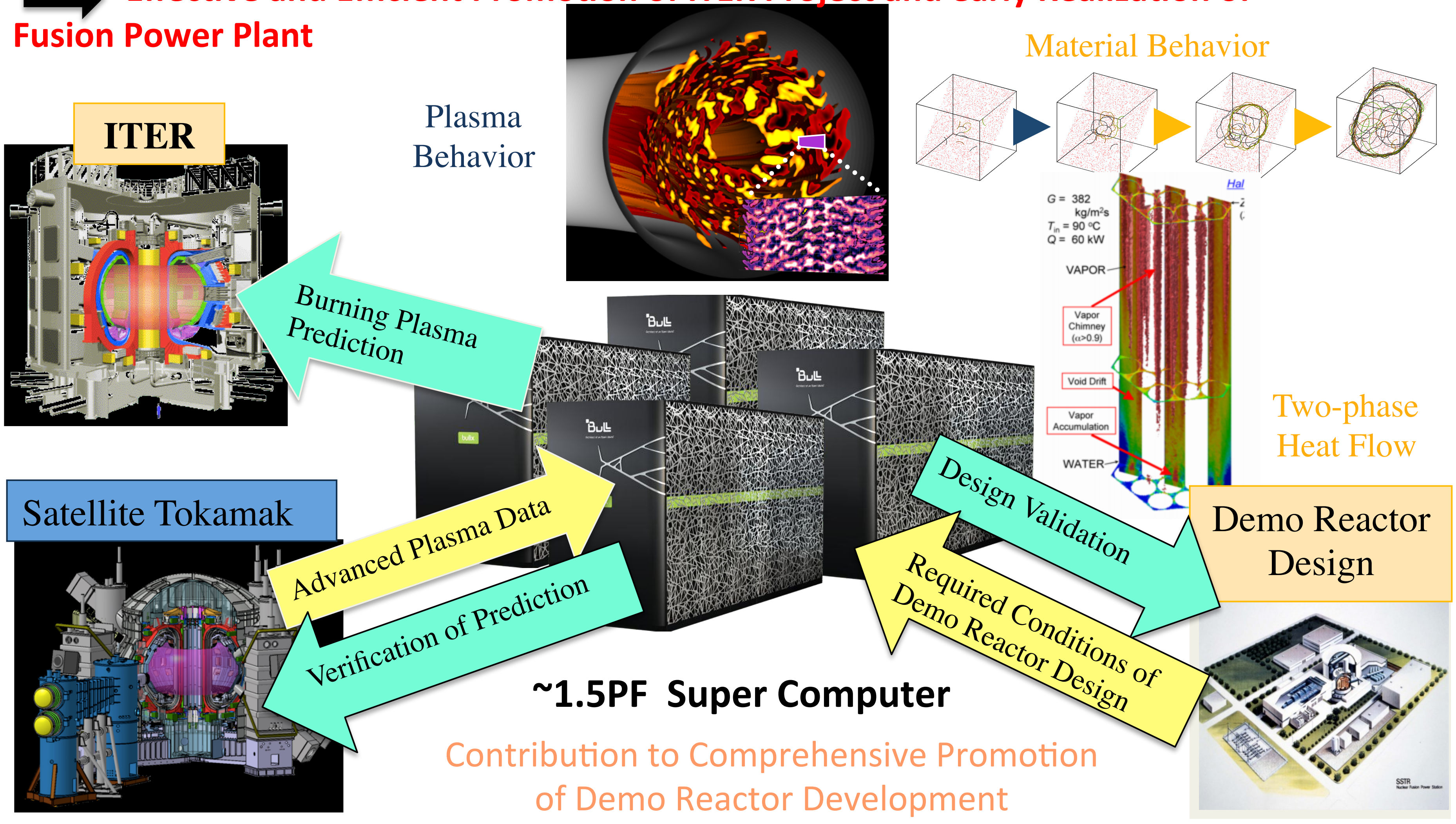
IFERC

International Fusion Energy Research Center
Computer Simulation Center

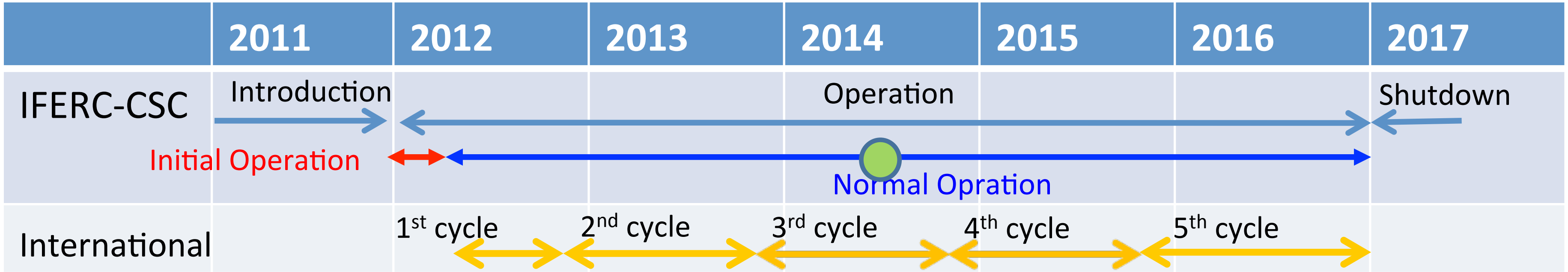
JAEA FUSION FOR ENERGY

Simulation Research on Burning and Steady State Advanced Plasma Behaviors for ITER/Satellite Tokamak, Demo Reactor Design, Advanced Fusion Material Development, etc

➡ **Effective and Efficient Promotion of ITER Project and early Realization of Fusion Power Plant**



HELIOS in IFERC-CSC



TOP500 Super Computer List on 2012/11

Rank	Site	Rmax (Tflops)	Rpeak (Tflops)
1	DOE/SC/ORNL, United States	17590	27112.5
2	DOE/NNSA/LLNL United States	16324.75	20132.66
3	RIKEN KEI, Japan	10510.00	11280.38
:	:	:	:
15	IFERC-CSC HELIOS, Japan	1237.0	1524.1

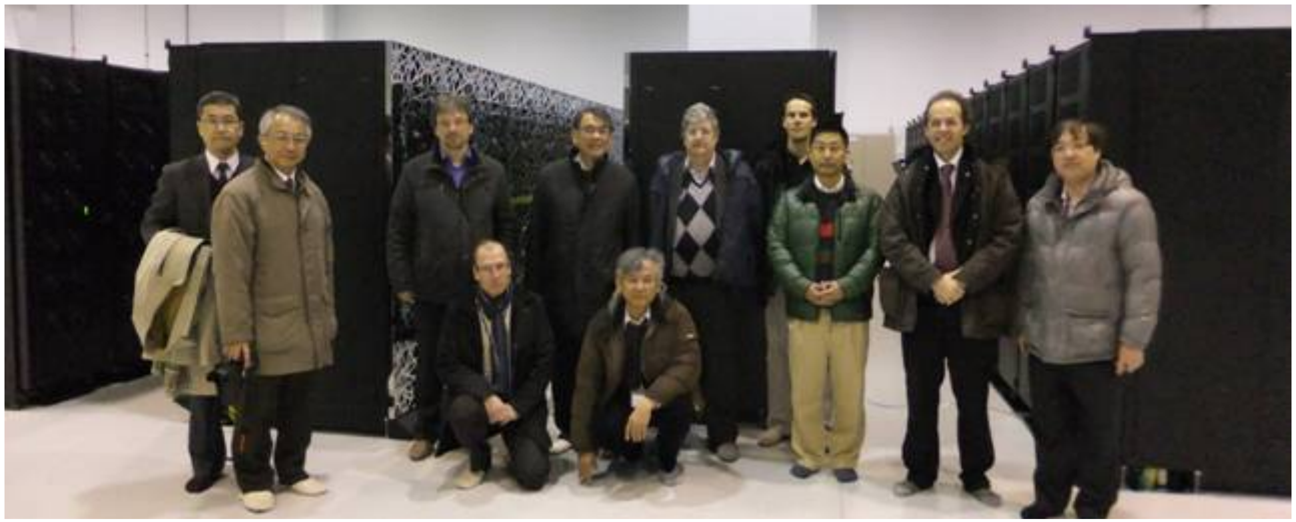
Bull computers provided by F4E/CEA

- TOP500 Super Computing Ranking on 2013/11 World 24th, Domestic 3rd

- 2014/1 System Enhancement
  Intel Xeon Phi, Many Integrated Core (MIC) architecture)
  - Theoretical performance 427Tflops
  - Linpack 225.1Tflops

flops (Floating-point Operations Per Second)

Giga (G): 10^9, Tera (T): 10^{12}, Peta (P): 10^{15}, Exa (E): 10^{18}



- **Boundary Value Problem**
  - ✓ **Explicit Space Discretization**
  - ✓ **Tomas Algorithm**
- **Initial Value Problem**
  - ✓ **Explicit Time Discretization**
  - ✓ **Von Neumann Analysis**
  - ✓ **Implicit Time Discretization**
- **Eigen Value Problem**
  - ✓ **Power Method**
  - ✓ **Inverse Shifted Power Method**

- **High Performance Computing**
  - ✓ **Vectorization**
  - ✓ **Open MP Programming**
  - ✓ **Message Passing Interface (MPI) Programming**
  - ✓ **Hybrid Programming**

■ Boundary Value Problem (BVP)

Example: One-Dimensional Boundary Value Problem

$$\theta''(x) = q(x),\ \theta(0) = \theta(1) = 0$$

We will solve this problem, numerically.

Explicit Space Discretization

Assuming an equidistant grid,

$$\theta_{i\pm1} \equiv \theta(x \pm \Delta x) = \theta(x) \pm \Delta x \theta_x(x) + \frac{\Delta x^2}{2}\theta_{xx}(x) \pm \cdots$$

Here the *x* subscript denotes differentiation, and the *i* subscript refers to the index of data points.

Three types of differences:

Forward: $$\left(\theta_x\right)_i = \frac{\theta_{i+1} - \theta_i}{\Delta x} + \mathrm{O}(\Delta x)$$

Backward: $$\left(\theta_x\right)_i = \frac{\theta_i - \theta_{i-1}}{\Delta x} + \mathrm{O}(\Delta x)$$

Central: $$\boxed{\left(\theta_x\right)_i = \frac{\theta_{i+1} - \theta_{i-1}}{2\Delta x} + \mathrm{O}(\Delta x^2)}$$

Second derivative

$$\boxed{\left(\theta_{xx}\right)_i = \frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{\Delta x^2} + O(\Delta x^2)}$$

Formula for high-order approximation for interior points and for boundary points

One-sided, second-order finite difference

$$\left(\theta_x\right)_i = \frac{a\theta_i + b\theta_{i-1} + c\theta_{i-2}}{\Delta x} + O(\Delta x^2)$$

$$\theta_{i-2} = \theta_i - 2\Delta x(\theta_x)_i + 2\Delta x^2\left(\theta_{xx}\right)_i - \frac{\left(2\Delta x\right)^3}{6}\left(\theta_{xxx}\right)_i + \cdots$$

$$\theta_{i-1} = \theta_i - \Delta x(\theta_x)_i + \frac{\Delta x^2}{2}\left(\theta_{xx}\right)_i - \frac{\left(\Delta x\right)^3}{6}\left(\theta_{xxx}\right)_i + \cdots$$

$$\theta_i = \theta_i$$

$$\Delta x\left(\theta_x\right)_i = (a+b+c)\theta_i - \Delta x(2c+b)\left(\theta_x\right)_i + \frac{\Delta x^2}{2}(4c+b)\left(\theta_{xx}\right)_i + O(\Delta x^3)$$

$$\begin{cases} a+b+c=0 \\ 2c+b=-1 \\ 4c+b=0 \end{cases} \Rightarrow a = \frac{3}{2},\ b=-2,\ c=\frac{1}{2}$$

$$\boxed{\left(\theta_x\right)_i = \frac{3\theta_i - 4\theta_{i-1} + \theta_{i-2}}{2\Delta x} + O(\Delta x^2)}$$

Similarly, $$\boxed{\left(\theta_x\right)_i = \frac{-3\theta_i + 4\theta_{i+1} - \theta_{i+2}}{2\Delta x} + O(\Delta x^2)}$$

$$\frac{1}{\Delta x^2}(\theta_{i+1} - 2\theta_i + \theta_{i-1}) = q_i,\ i = 1, \cdots, N\text{-}1$$

$\theta_0 = \theta_N = 0$ **Dirichlet boundary condition**

$$\begin{bmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & . & . & . & & & \\ & & 1 & -2 & 1 & & \\ & & & . & . & . & \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ . \\ \theta_{N-3} \\ \theta_{N-2} \\ \theta_{N-1} \end{bmatrix} = \begin{bmatrix} \Delta x^2 q_1 - u_0 \\ \Delta x^2 q_2 \\ \Delta x^2 q_3 \\ . \\ \Delta x^2 q_{N-3} \\ \Delta x^2 q_{N-2} \\ \Delta x^2 q_{N-1} - u_N \end{bmatrix}$$

Tomas Algorithm for Tridiagonal Systems

Solution of BVP reduces to solving the linear system $\mathrm{A}\theta = \mathbf{q}$

where the matrix A is tridiagonal if the boundary condition are Dirichlet

1. **LU decomposition of matrix A=LU,**
   L: lower trianglar matrix, U: upper triangular matrix
2. **Forward substitution Ly=q**
3. **Backward substitution Ux=y**

Step 1: A=LU

$$\begin{bmatrix} a_1 & c_1 & & & & & \\ b_2 & a_2 & c_2 & & & & \\ & b_3 & a_3 & c_3 & & & \\ & & . & . & . & & \\ & & & . & . & . & \\ & & & & b_{N-1} & a_{N-1} & c_{N-1} \\ & & & & & b_N & a_N \end{bmatrix} = \begin{bmatrix} 1 & & & & & \\ l_2 & 1 & & & & \\ & l_3 & 1 & & & \\ & & . & . & & \\ & & & . & . & \\ & & & & l_{N-1} & 1 & \\ & & & & & l_N & 1 \end{bmatrix} \begin{bmatrix} d_1 & u_1 & & & & \\ & d_2 & u_2 & & & \\ & & d_3 & u_3 & & \\ & & & . & . & \\ & & & & . & . \\ & & & & d_{N-1} & u_{N-1} \\ & & & & & d_N \end{bmatrix}$$

$$\boxed{d_1 = a_1,\ u_1 = c_1} \quad \begin{cases} l_i d_{i-1} = b_i \\ l_i u_{i-1} + d_i = a_i \\ u_i = c_i \end{cases} \Rightarrow \boxed{\begin{cases} l_i = b_i / d_{i-1} \\ d_i = a_i - l_i u_{i-1},\ (i = 2, \cdots N-1) \\ u_i = c_i \end{cases}} \quad \boxed{l_N = b_N / d_{N-1},\ d_N = a_N - l_N u_{N-1}}$$

Step 2: Ly=q

$$\begin{bmatrix} 1 & & & & & & \\ l_2 & 1 & & & & & \\ & l_3 & 1 & & & & \\ & & \cdot & \cdot & & & \\ & & & \cdot & \cdot & & \\ & & & & l_{N-1} & 1 & \\ & & & & & l_N & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \cdot \\ \cdot \\ y_{N-1} \\ y_N \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \cdot \\ \cdot \\ q_{N-1} \\ q_N \end{bmatrix}$$

$$\boxed{y_1 = q_1}$$

$$l_i y_{i-1} + y_i = q_i$$

$$\Rightarrow \boxed{y_i = q_i - l_i y_{i-1}, \ (i = 2, \cdots, N)}$$

Step 3: Ux=y

$$\begin{bmatrix} d_1 & u_1 & & & & & \\ & d_2 & u_2 & & & & \\ & & d_3 & u_3 & & & \\ & & & \cdot & \cdot & & \\ & & & & \cdot & \cdot & \\ & & & & & d_{N-1} & u_{N-1} \\ & & & & & & d_N \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \cdot \\ \cdot \\ \theta_{N-1} \\ \theta_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \cdot \\ \cdot \\ y_{N-1} \\ y_N \end{bmatrix}$$

$$\theta_N = y_N / d_N,$$

$$d_i \theta_i + u_i \theta_{i+1} = y_i$$

$$\Rightarrow \boxed{\theta_i = (y_i - u_i \theta_{i+1}) / d_i, \ (i = N\text{-}1, \cdots, 1)}$$

Tomas algorithm will always converge if

$$|a_k| \geq |b_k| + |c_k|, \ k = 2, \cdots, N-1$$

$$|a_1| > |c_1|, \ \& \ |a_N| > |b_N|$$

■ Advection Equation

$$\frac{\partial\Theta}{\partial t}+U\frac{\partial\Theta}{\partial x}=0$$

with periodic boundary condition $\Theta(x=0,t)=\Theta(x=1,t)$

and initial condition $\Theta(x,t=0)=\sin 2\pi x$

We will solve this equation, numerically

Euler-Forward/Central-Difference Scheme (EF/CD) [Explicit Scheme]

$$\frac{\Theta_j^{n+1}-\Theta_j^n}{\Delta t}+U\frac{\Theta_{j+1}^n-\Theta_{j-1}^n}{2\Delta x}=0,\quad j=1,\cdots,N-1 \qquad \Theta_0^n=\Theta_N^n \quad \textbf{and} \quad \Theta_j^0=\sin 2\pi x_j$$

Von Neumann Stability Analysis

$$\Theta_j^n=\sum_{k=-\infty}^{\infty}a_k^n e^{2\pi i k x_j}$$

$$a_k^{n+1}=a_k^n(1-iC\sin 2\pi k\Delta x)$$ **where** $$C=\frac{U\Delta t}{\Delta x}$$

CFL number
(Courant, Friedrichs and Lewy)

$$\boxed{\frac{\left|a_k^{n+1}\right|}{\left|a_k^n\right|}=\left|1+C^2\sin^2 2\pi k\Delta x\right|^{1/2}>1}$$

EF/CD scheme is *absolutely unstable*

Euler-Forward/Upwind-Differencing Scheme (EF/UD) [Explicit Scheme]

$$\frac{\Theta_j^{n+1}-\Theta_j^n}{\Delta t}+U\frac{\Theta_j^n-\Theta_{j-1}^n}{\Delta x}=0,\quad j=1,\cdots,N-1$$

which can be rewritten as

$$\Theta_j^{n+1}=\Theta_j^n-\frac{C}{2}(\Theta_{j+1}^n-\Theta_{j-1}^n)+\underbrace{\frac{C}{2}(\Theta_{j+1}^n-2\Theta_j^n+\Theta_{j-1}^n)}_{\text{numerical diffusion}}$$

Von Neumann Stability Analysis

$$a_k^{n+1}=a_k^n[1-C(1-e^{-2\pi ik\Delta x})]$$

$$\frac{\left|a_k^{n+1}\right|}{\left|a_k^n\right|}=[1+2(C^2-C)(1-\cos 2\pi k\Delta x)]^{1/2}$$

Stability Condition $\boxed{C\leq 1}$

Stability Region in Complex Plane

Assuming $a^n\propto e^{\lambda t_n},\ a^{n+1}\propto e^{\lambda t_{n+1}}=e^{\lambda(t_n+\Delta t)},$

$$\frac{a_k^{n+1}-a_k^n}{\Delta t}=\lambda a_k^n=-\frac{1-e^{-2\pi ik\Delta x}}{\Delta x}a_k^n$$

$$\lambda\Delta t=-C(1-\cos 2\pi k\Delta x)-iC\sin 2\pi k\Delta x$$

Crank-Nicolson/Center-Differencing Scheme (CN/CD) [Implicit Scheme]

$$\frac{\Theta_j^{n+1}-\Theta_j^n}{\Delta t}+\frac{U}{2}\frac{\Theta_{j+1}^{n+1}-\Theta_{j-1}^{n+1}}{2\Delta x}+\frac{U}{2}\frac{\Theta_{j+1}^{n}-\Theta_{j-1}^{n}}{2\Delta x}=0,\quad j=1,\cdots,N-1$$

Von Neumann Stability Analysis

$$a_k^{n+1}=\left(\frac{1-i\dfrac{C}{2}\sin 2\pi k\Delta x}{1+i\dfrac{C}{2}\sin 2\pi k\Delta x}\right)a_k^n,\qquad \frac{\left|a_k^{n+1}\right|}{\left|a_k^n\right|}=1$$

CN/CD scheme is neutrally stable

Eigen Values

$$\lambda\Delta t=\frac{-\dfrac{C^2}{4}\sin^2 2\pi k\Delta x-i\dfrac{C}{2}\sin 2\pi k\Delta x}{1-\dfrac{C^2}{4}\sin^2 2\pi k\Delta x}$$

which are on the left half-plane for any positive value of C

$$
\begin{bmatrix}
1 & C/4 & & & & & \\
-C/4 & 1 & C/4 & & & & \\
 & . & . & . & & & \\
 & & -C/4 & 1 & C/4 & & \\
 & & & . & . & . & \\
 & & & & -C/4 & 1 & C/4 \\
 & & & & & -C/4 & 1
\end{bmatrix}
\begin{bmatrix}
\Theta_1^{n+1} \\
\Theta_2^{n+1} \\
\Theta_3^{n+1} \\
. \\
\Theta_{N-3}^{n+1} \\
\Theta_{N-2}^{n+1} \\
\Theta_{N-1}^{n+1}
\end{bmatrix}
$$

$$
=\begin{bmatrix}
\overbrace{C\Theta_0^n/4}+\Theta_1^n-C\Theta_2^n/4+\overbrace{C\Theta_0^{n+1}/4} \\
C\Theta_1^n/4+\Theta_2^n-C\Theta_3^n/4 \\
C\Theta_2^n/4+\Theta_3^n-C\Theta_4^n/4 \\
. \\
C\Theta_{N-4}^n/4+\Theta_{N-3}^n-C\Theta_{N-2}^n/4 \\
C\Theta_{N-3}^n/4+\Theta_{N-2}^n-C\Theta_{N-1}^n/4 \\
C\Theta_{N-2}^n/4+\Theta_{N-1}^n-\underbrace{C\Theta_N^n/4}+\underbrace{C\Theta_N^{n+1}/4}
\end{bmatrix}
$$

Tomas Algorithm is available to solve CN/CD scheme

Exercise #1 **Consider full implicit scheme for advection equation and perform von Neumann stability analysis and calculate eigenvalue**

Effects of Boundary Conditions

Example $\frac{\partial \Theta}{\partial t} + U\frac{\partial \Theta}{\partial x} = 0,\ \Theta(x,t=0)\sin 2\pi x,\ 0<x<1,\ \Theta(x=0,t) = -\sin 2\pi t$

CN/CD Scheme

$$\frac{\Theta_j^{n+1} - \Theta_j^n}{\Delta t} + \frac{1}{2}\left(\frac{\Theta_{j+1}^{n+1} - \Theta_{j-1}^{n+1}}{2\Delta x} + \frac{\Theta_{j+1}^{n} - \Theta_{j-1}^{n}}{2\Delta x}\right) = 0,\ j = 1,\cdots,N-1$$

$$\Theta_0^{n+1} = -\sin 2\pi t^{n+1},\ \Theta_j^0 = \sin 2\pi x_j$$

Fictitious Node $j = N+1$

Linear extrapolation $\Theta_{N+1} = \Theta_N + \Delta x \frac{\Theta_N - \Theta_{N-1}}{\Delta x} = 2\Theta_N - \Theta_{N-1}$

$$\Rightarrow \boxed{\frac{\Theta_N^{n+1} - \Theta_N^n}{\Delta t} + \frac{1}{2}\left(\frac{\Theta_N^{n+1} - \Theta_{N-1}^{n+1}}{\Delta x} + \frac{\Theta_N^{n} - \Theta_{N-1}^{n}}{\Delta x}\right) = 0}$$

Addition of upwind derivative does not influence the stability of CN scheme

Eigenvalue Problem

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

Local Eigensolvers

Power Method: **simple method to obtain the maximum eigenvalue**

$$\mathbf{x}^{k+1} = c\mathbf{A}\mathbf{x}^{k} \qquad C\text{: normalization constant}$$

Assume there exists an eigenvalue λ_1 **that dominates**

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \ldots \geq |\lambda_n|$$

Initial Guess $\mathbf{x}^0 = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_n\mathbf{v}_n$

$$\mathbf{x}^k = \mathbf{A}\mathbf{x}^{k-1} = \cdots = \mathbf{A}^k\mathbf{x}^0 = c_1\lambda_1^k\mathbf{v}_1 + \cdots + c_n\lambda_n^k\mathbf{v}_n$$

$$\frac{\mathbf{x}^k}{c_1\lambda_1^k} = \mathbf{v}_1 + \frac{c_2}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k\mathbf{v}_2 + \cdots + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k\mathbf{v}_n \Rightarrow \mathbf{v}_1 \text{ for } k \to \infty$$

Convergence Rate $\dfrac{|\lambda_2|}{|\lambda_1|}$

Pseudo-code

Initialize: $\mathbf{x}^0$

Begin Loop: for k=1,2,…

$$\hat{\mathbf{x}}^k = \mathbf{A}\mathbf{x}^{k-1}$$

$$\mathbf{x}^k = \frac{\hat{\mathbf{x}}^k}{\max(\hat{\mathbf{x}}^k)}$$

$\max(\mathbf{y})$ **returns the entry of y with *maximum modulus***

endfor

Ex. $\mathbf{y} = (4.32, -9.88, 2.9)^T, \ \max(\mathbf{y}) = -9.88$

End Loop:

$$\boxed{\max(\hat{\mathbf{x}}^k) \to \lambda_1, \ \mathbf{x}^k \longrightarrow \mathbf{v}_1}$$

Rayleigh Quotient

$$R(\mathbf{A},\mathbf{x}) \equiv \frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} = \frac{(c_1\mathbf{v}_1 + \cdots + c_n\mathbf{v}_n)^T A(c_1\mathbf{v}_1 + \cdots + c_n\mathbf{v}_n)}{c_1^2 + c_2^2 + \cdots + c_n^2}$$

$$= \frac{\lambda_1 c_1^2 + \lambda_2 c_2^2 + \cdots + \lambda_n c_n^2}{c_1^2 + c_2^2 + \cdots + c_n^2} = \lambda_1 \frac{1 + \left(\frac{\lambda_2}{\lambda_1}\right)\left(\frac{c_2}{c_1}\right)^2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)\left(\frac{c_n}{c_1}\right)^2}{1 + \left(\frac{c_2}{c_1}\right)^2 + \cdots + \left(\frac{c_n}{c_1}\right)^2} \longrightarrow \lambda_1$$

Inverse Shifted Power Method: to selectively compute minimum eigenvalue

$$\mathbf{Ax} = \lambda\mathbf{x} \Leftrightarrow \mathbf{A}^{-1}\mathbf{x} = \lambda^{-1}\mathbf{x}$$

$\mathbf{Ax}^{k+1} = c\mathbf{x}^k$ **This method is most effective with a proper shift**

$$\boxed{(\mathbf{A} - \sigma\mathbf{I})\mathbf{x}^{k+1} = c\mathbf{x}^k}$$

Convergence Rate $\dfrac{|\lambda_n - \sigma|}{|\lambda_{n-1} - \sigma|}$

Pseudo-code

Initialize: **Choose** $\mathbf{x}^0$

Choose σ

Factorize $\mathbf{A} - \sigma\mathbf{I} = \mathbf{LU}$

Begin Loop: for k=1,2,…

$$\hat{\mathbf{x}}^k = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{x}^{k-1}$$

$$\mathbf{x}^k = \frac{\hat{\mathbf{x}}^k}{\max(\hat{\mathbf{x}}^k)}$$

if $\left| R(\mathbf{A}, \mathbf{x}^k) - R(\mathbf{A}, \mathbf{x}^{k-1}) \right| < \varepsilon$ **return**

endfor

End Loop:

LU decomposition

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
\vdots \qquad \vdots \qquad\qquad \vdots \qquad &\;\; \vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

First stage of Gaussian elimination

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{22}^{(1)}x_2 + \cdots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\
\vdots \qquad \vdots \qquad\qquad \vdots \qquad &\;\; \vdots \\
a_{n2}^{(1)}x_2 + \cdots + a_{nn}^{(1)}x_n &= b_n^{(1)}
\end{aligned}
$$

$$
a_{ij}^{(1)} = a_{ij} - a_{1j} l_{i1}^{(1)}, \; l_{i1}^{(1)} = \frac{a_{i1}}{a_{11}}, \; b_i^{(1)} = b_i - b_1 \frac{a_{i1}}{a_{11}}
$$

(n-1)th stage gives upper triangular system U

L is constructed from $l_{ij}^{(k)}$ **where *k* corresponds to kth column and i < j**

Modified Inverse Shifted Power Method with Rayleigh Quotient

Pseudo-code

Initialize: Choose $\mathbf{x}^0$

Compute $\sigma_0 = \dfrac{(\mathbf{x}^0)^T \mathbf{A}\mathbf{x}^0}{(\mathbf{x}^0)^T \mathbf{x}^0}$

Begin Loop: for k=0,1,…

$$\mathbf{L}^k\mathbf{U}^k = \mathbf{A} - \sigma_k \mathbf{I}$$

$$\hat{\mathbf{x}}^{k+1} = (\mathbf{U}^k)^{-1}(\mathbf{L}^k)^{-1}\mathbf{x}^k$$

$$\mathbf{x}^{k+1} = \frac{\hat{\mathbf{x}}^{k+1}}{\max(\hat{\mathbf{x}}^{k+1})}$$

$$\sigma_{k+1} = \frac{(\mathbf{x}^{k+1})^T \mathbf{A}\mathbf{x}^{k+1}}{(\mathbf{x}^{k+1})^T \mathbf{x}^{k+1}}$$

if $\left|\sigma_{k+1} - \sigma_k\right| < \varepsilon$ **return**

endfor

End Loop:

Exercise #2 **Develop the code for modified inverse shifted power method, then investigate eigenvalue of the matrix changing initial guess**

$$\mathbf{A} = \begin{pmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{pmatrix} \qquad \mathbf{x}^0 = \begin{pmatrix} 1.0 \\ 1.0 \\ 1.0 \end{pmatrix}, \begin{pmatrix} 0.4 \\ -0.4 \\ -0.4 \end{pmatrix}$$

You may calculate eigenvalue analytically and compare it to numerical result for verification

High Performance Computing

Code Tuning

✓ Intel AVX/AVX2 (Advanced Vector Extension)

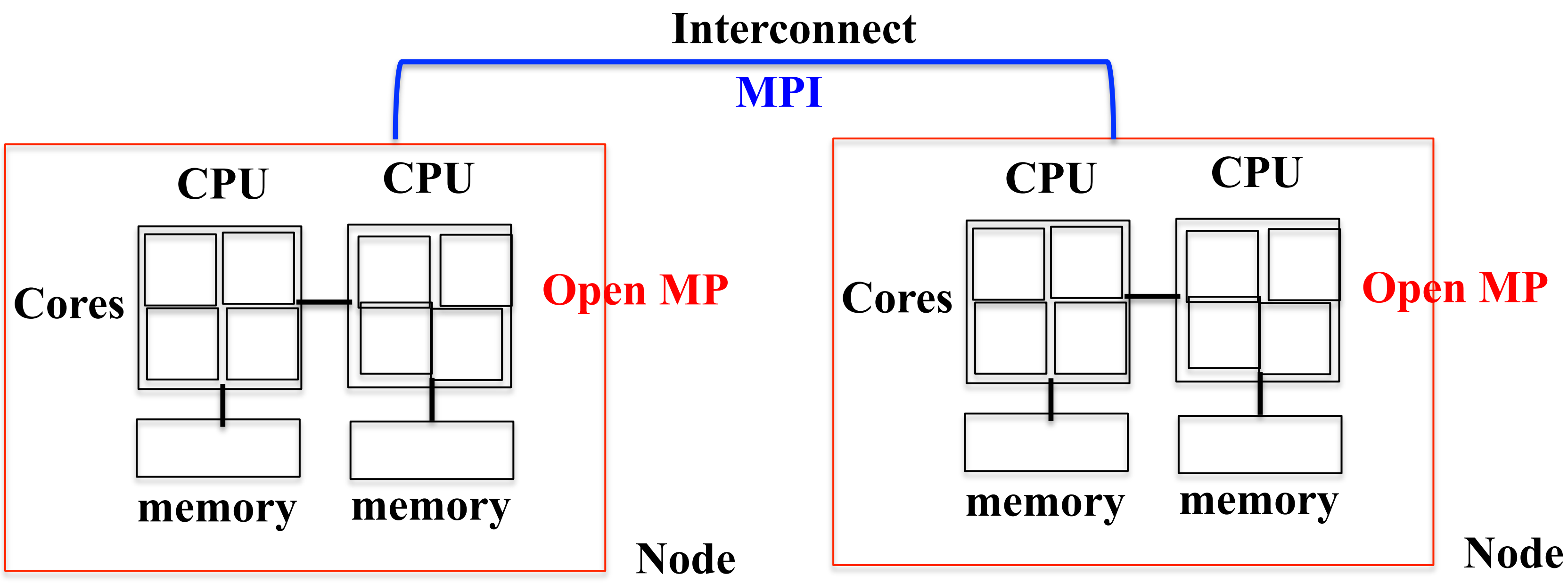
256bit SIMD (Single Instruction Multiple Data)

Intel AVX-512, Xeon Phi Knights Landing (~2015)

✓ Open MP Parallel Programming

✓ MPI Parallel Programming

Open MP/MPI Hybrid Programming



Loop-level Parallelization

```
DO K=1, KMAX
    DO J=1, JMAX
        DO I=1,IMAX
            Numerical Operations
        END DO                Vectorization
    END DO                                      Open MP
END DO                                              MPI
```

- ✓ **Vectorization**

cc/ifort –xAVX

Alignment of 32 bye boundary

float A[1000] __attribute__((aligned(32)));

REAL*4 A(1000)
!DIR$ATTRIBUTES ALIGN: 32:: A

You should use MKL library, which is already optimized for AVX

✓ **Open MP**

```
!$OMP parallel
do istep =1, nstep
!$OMP do
    do j=2, n – 1
        do i = 2, n – 1
g(i,j)=0.25d0 * (f(i-1,j)+f(i+1, j) &
                 + f(i, j-1) + f(i, j+1)
        end do
    end do
!$OMP end do nowait
!$OMP single
   er=0.0d0
!$OMP end single
!$OMP do reduction(+:er)
                .
 end do
!$OMP end parallel
```

```
$ifort –openmp ….

$export OMP_NUM_THREADS=4

$./a.out
```

✓ MPI (Message Passing Interface)

```
call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,myid,ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,nprocs,ierr)
s=1+myid*(n/nprocs)
e=s+(n/nprocs)-1
do j=s,e      ! Domain Decomposition
  do i=1, n
a(i,j)=&
          0.25*(b(i-1,j)+b(i,j+1)+b(i,j-1)+b(i+1,j)) - &
             h*h*f(i,j)
   end do
end do
      call MPI_SENDRECV(
    &          a(1,e), nx, MPI_DOUBLE_PRECISION, nbrtop, 0,
    &          a(1,s-1), nx, MPI_DOUBLE_PRECISION, nbrbottom, 0,
    &          comm1d, status, ierr )
      call MPI_SENDRECV(
    &          a(1,s), nx, MPI_DOUBLE_PRECISION, nbrbottom, 1,
    &          a(1,e+1), nx, MPI_DOUBLE_PRECISION, nbrtop, 1,
    &          comm1d, status, ierr )
.
call MPI_FINALIZE(ierr)
```

```
$mpiifort …..

$mpirun –np 4 ./a.out
```

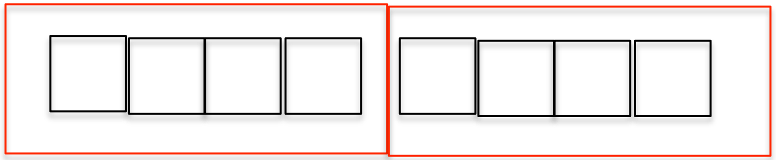
Hybrid Programming

```
call MPI_INT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, myid, ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, numprocs, ierr)
…
s=1+myid*(n/nprocs)
e=s+(n/nprocs)-1
if(myid == numprocs-1) e=n
!$omp parallel do private(i)
do j=s,e
 y(j)=0.0d0
do i=1,n
 y(j)=y(j)+A(j,i)*x(i)
end do
end do
!$omp end parallel do
```

```
$mpiifort -openmp…..

$export OMP_NUM_THREADS=4

$mpirun –np 2 ./a.out
```



2MPI x 4Threads