Domain Decomposition Methods for Partial Differential Equations

David E. Keyes

david.keyes@columbia.edu

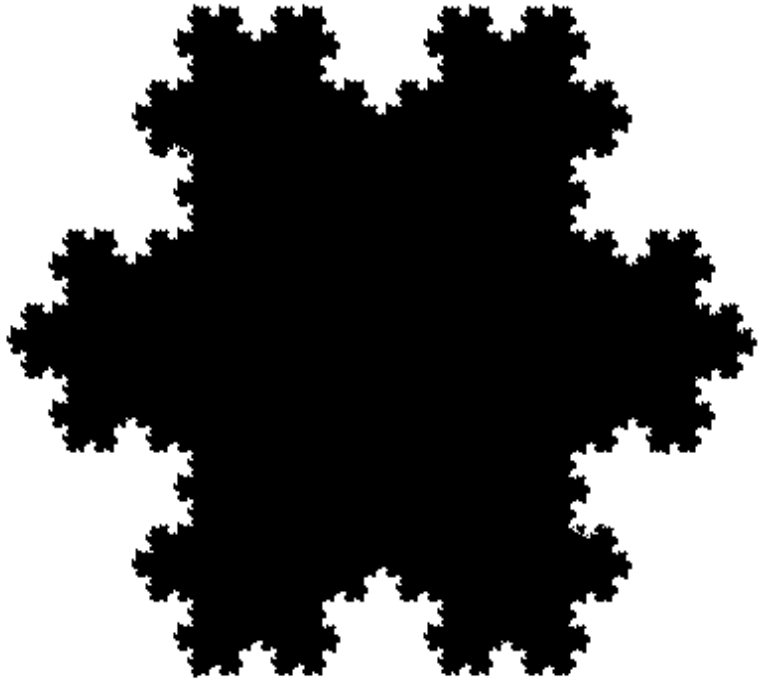
Department of Applied Physics & Applied Mathematics

Columbia University

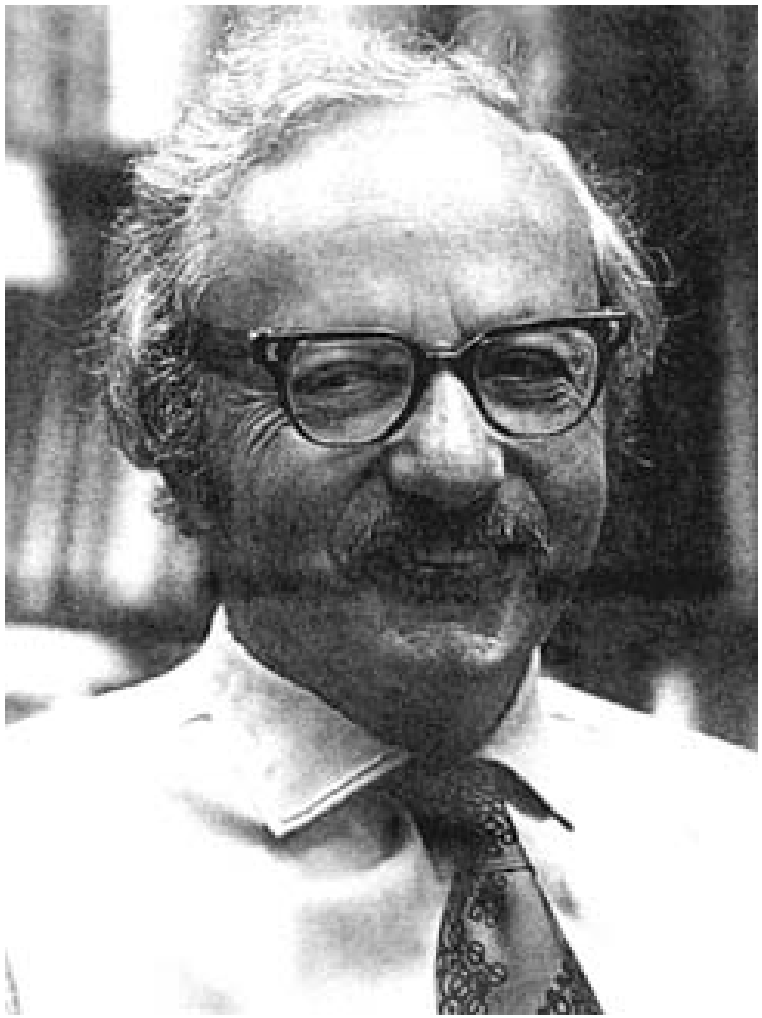
Happy Birthday, Felix Hausdorf!

- Born 8 Nov 1868 in Breslau, Germany (now Wroclaw, Poland)
- Died 26 Jan 1942 in Bonn, Germany

- Developed concept of "Hausdorf" dimension in attempt to apply measures to what we now call fractals, such as the Koch curve (below), whose perimeter has dimension 1.26…



Happy 90th Birthday, George Dantzig!



Born 8 Nov 1914 in Portland, OR

Invented Simplex Method, 1947

"If one would take statistics about which mathematical problem is using up most of the computer time in the world, then ... the answer would probably be linear programming." ***- Laszlo Lovasz***

"For inventing linear programming and discovering methods that led to wide-scale scientific and technical applications to important problems in logistics, scheduling, and network optimization, and to the use of computers in making efficient use of the mathematical theory. ***– National Medal of Science citation by Jimmy Carter***

"The tremendous power of the simplex method is a constant surprise to me."- ***George Dantzig***

Definition and motivation

- **Domain decomposition (DD) is a "divide and conquer" technique for arriving at the solution of problem defined over a domain from the solution of related problems posed on subdomains**
- ***Motivating assumption #1:*** **the solution of the subproblems is qualitatively or quantitatively "easier" than the original**
- ***Motivating assumption #2:*** **the original problem does not fit into the available space**
- ***Motivating assumption #3:*** **the subproblems can be solved with some concurrency**

Remarks on definition

- **"Divide and conquer" is not a fully satisfactory description**
  - **"divide, conquer, and *combine*" is better**
  - **combination is usually through iterative means**
- **True "divide-and-conquer" (only) algorithms are rare in computing (unfortunately)**
- **It might be preferable to focus on "subdomain composition" rather than "domain decomposition"**
  - **generalizes to subproblem composition for multiphysics**

We often think we know all about "two" because two is "one and one". We forget that we have to make a study of "and."

A. S. Eddington (1882-1944)

Remarks on definition, cont.

- **Domain decomposition has generic and specific senses within the universe of parallel computing**
  - **generic sense: any *data decomposition* (considered in contrast to *task decomposition*)**
  - **specific sense: the domain is the *domain of definition of an operator equation* (differential, integral, algebraic)**
- **In general, the process of constructing a parallel program consists of (see J. P. Singh's book ☺):**
  - **Decomposition into tasks**
  - **Assignment of tasks to processes**
  - **Orchestration of processes**
    - **Communication and synchronization**
  - **Mapping of processes to processors**

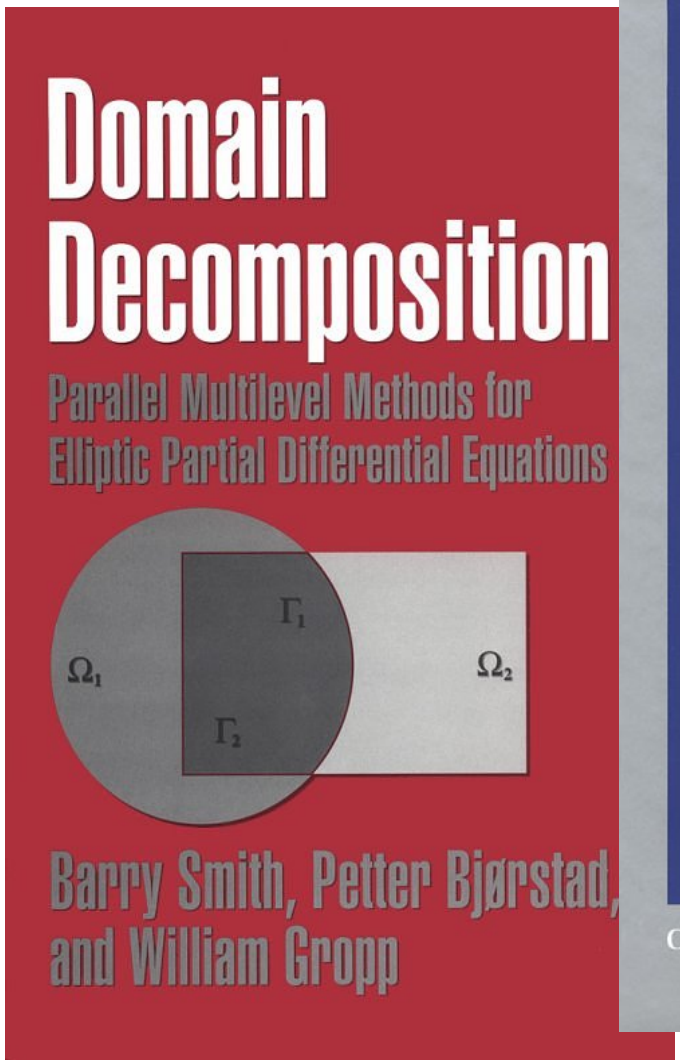
PDE-type domain decomposition leads to bulk synchronous, SPMD codes, with mostly near-neighbor communication, with some global reductions and small global tasks

Plan of presentation

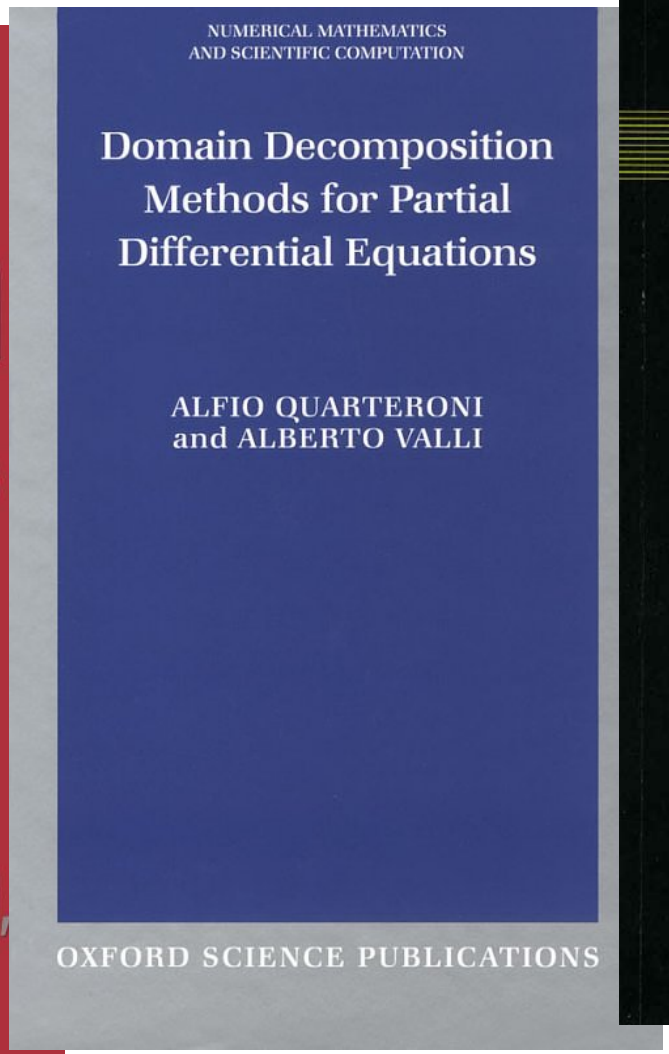
- **Imperative of domain decomposition (DD) for terascale computing**
- **Basic DD algorithmic concepts**
  - **Schwarz**
  - **Schur**
  - **Schwarz-Schur combinations**
- **Basic DD convergence and scaling properties**
- **Some research agendas in DD**
  - **Jacobian-free Newton-Krylov-Schwarz**
  - **Nonlinear Schwarz**

Prime sources for domain decomposition

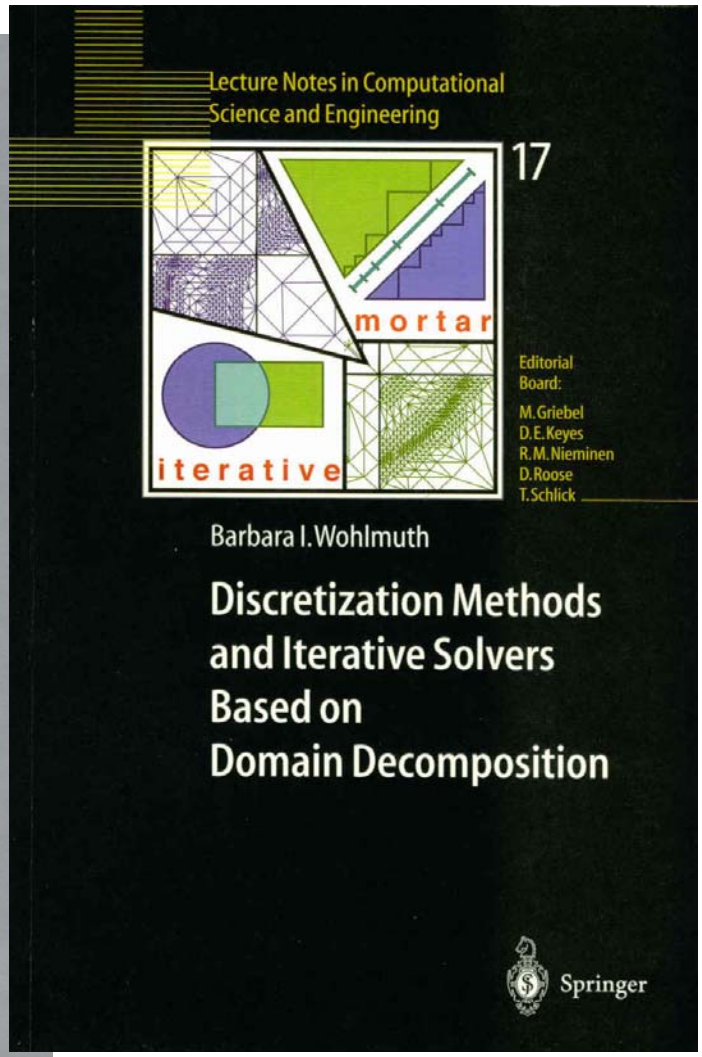
1992



1997



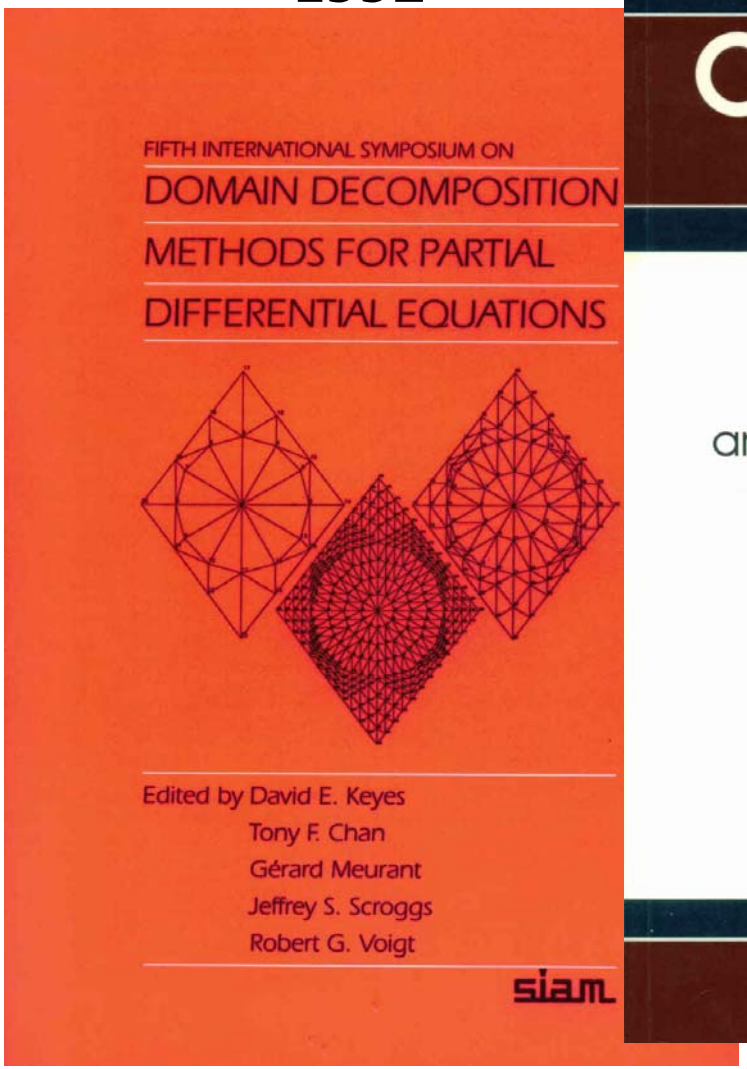
2001



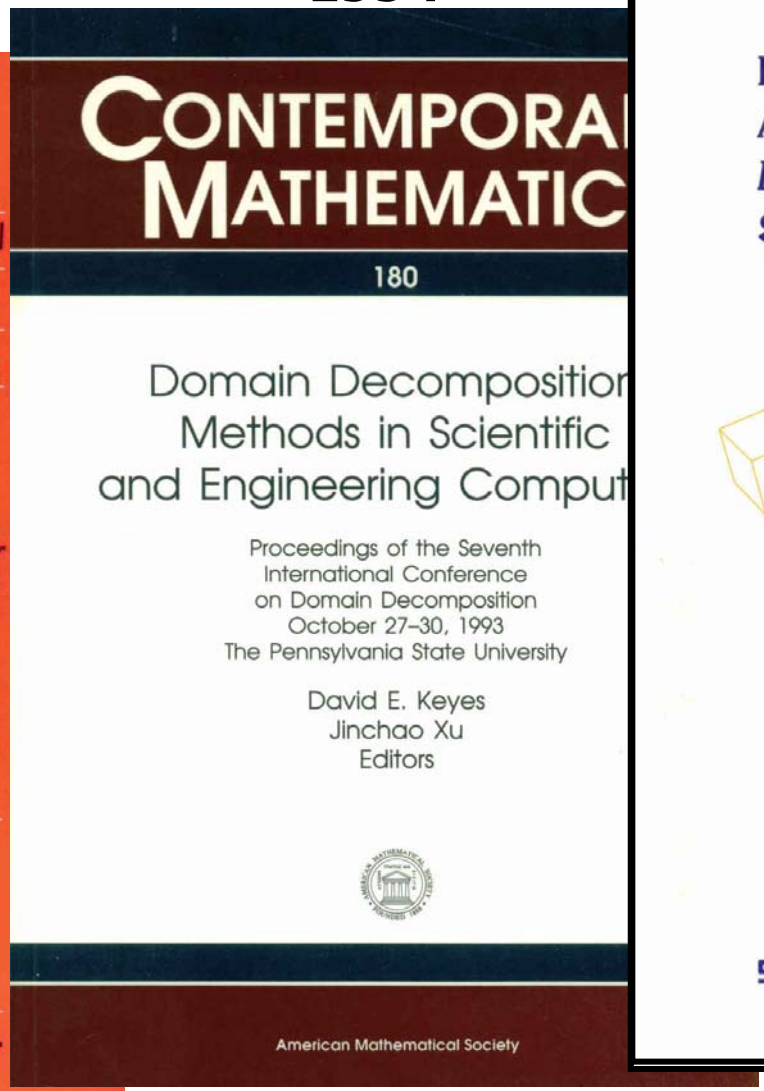
+ O. B . Widlund and A. Toselli (2004)

Other sources for domain decomposition

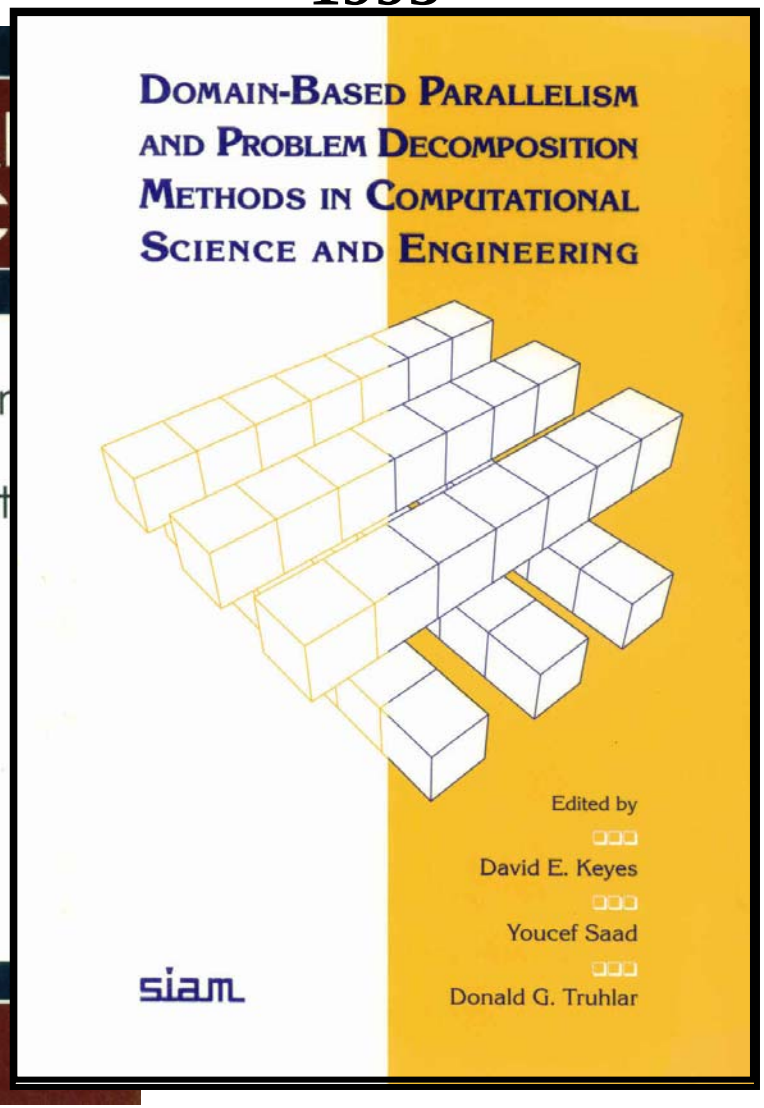
1992



1994

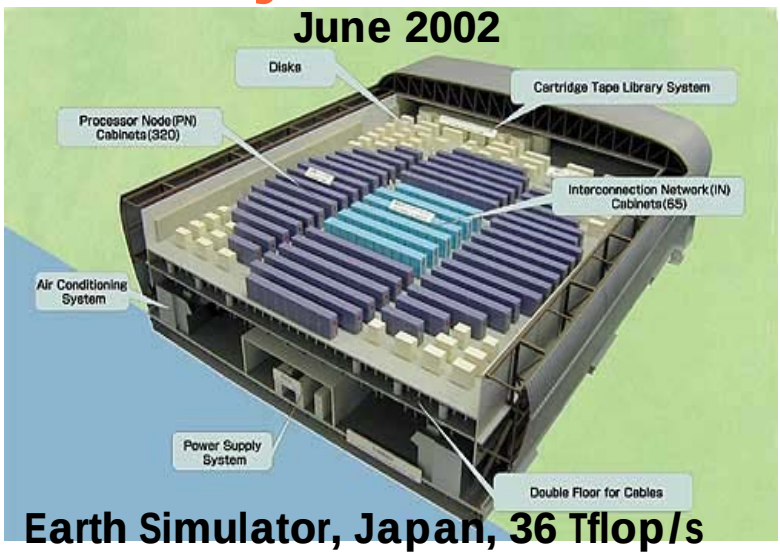


1995



+ misc proceedings volumes, 1988-2004

Why care? Recent high-end systems!



"High performance computing is the backbone of the nation's science and technology enterprise" – Energy Secretary Spencer Abraham

Other platforms for high-end simulation

- DOE's ASC roadmap is to go to 100 Teraflop/s by 2005
- Use variety of vendors
  - Compaq
  - Cray
  - Intel
  - IBM
  - SGI
- Rely on commodity processor/memory units, with tightly coupled network
- Massive software project to rewrite physics codes for distributed shared memory???


NPACI
Blue Horizon


ASCI White


ASCI Q


ASCI
Blue Mountain


ASCI Cplant


ASCI Blue-Pacific

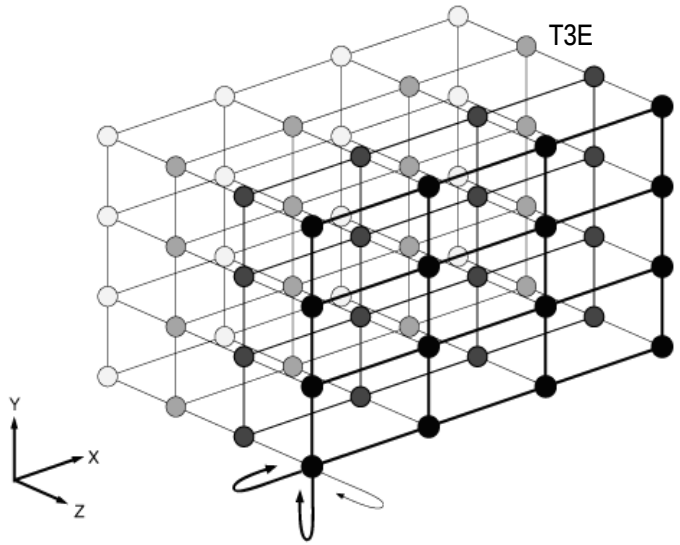

ASCI
Whitecap


ASCI Red

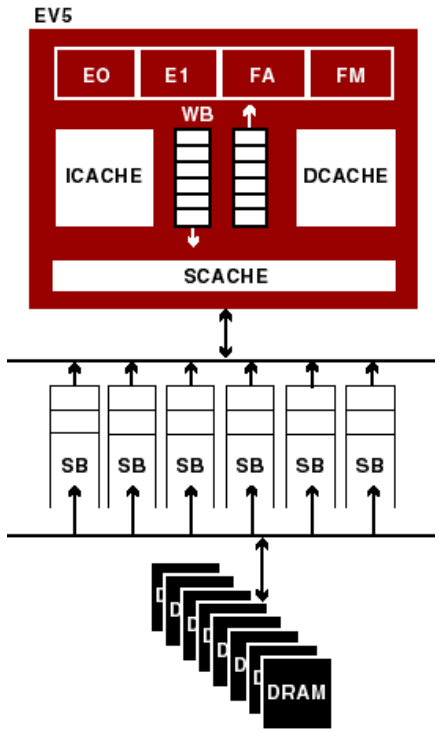
Algorithmic requirements from architecture

- **Must run on physically distributed memory units connected by message-passing network, each serving one or more processors with multiple levels of cache**

"horizontal" aspects



"vertical" aspects



Building platforms is the “easy” part

- **Algorithms must be**
  - **highly concurrent and straightforward to load balance**
  - **latency tolerant**
  - **cache friendly (good temporal and spatial locality)**
  - **highly scalable (in the sense of convergence)**
- **Domain decomposition “natural” for all of these**
- **Domain decomposition also “natural” for software engineering**
- **Fortunate that its theory was built in advance of requirements!**

The earliest DD paper?

Gesammelte

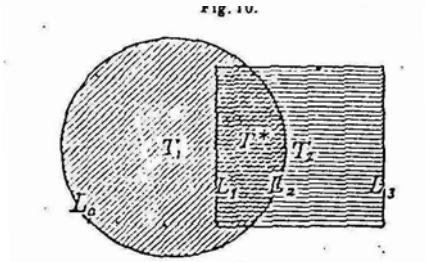
Mathematische Abhandlungen

von

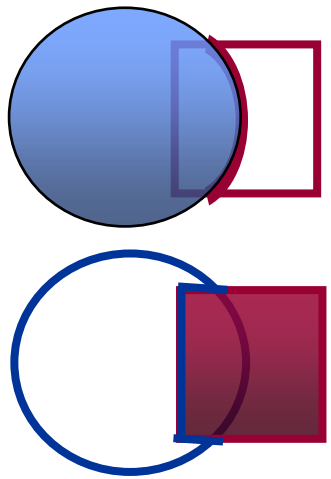
H. A. Schwarz.

Ueber einen Grenzübergang durch alternirendes Verfahren.

Zürich, im August 1870.



What Schwarz proposed…



Solve PDE in circle with BC taken from interior of square

Solve PDE in square with BC taken from interior of circle

And iterate!

Rationale

- **Convenient analytic means (separation of variables) are available for the regular problems in the subdomains, but not for the irregular "keyhole" problem defined by their union**
- **Schwarz iteration defines a functional map from the values defined along (either) artificial interior boundary segment completing a subdomain to an updated set of values on the same segment, through intermediate subdomain solves**
- **A *contraction map* is derived for the error**
- **Rate of convergence is not necessarily rapid – this was not a concern of Schwarz**
- **Subproblems are not solved concurrently – neither was this Schwarz' concern**

Other early DD papers

Journal
of
Applied Physics

Volume 24, Number 8 August, 1953

A Set of Principles to Interconnect the Solutions of Physical Systems

GABRIEL KRON*
General Electric Company, Schenectady, New York
(Received December 31, 1952)

Matrix Structural Analysis of Substructures

J. S. PRZEMIENIECKI*
Air Force Institute of Technology

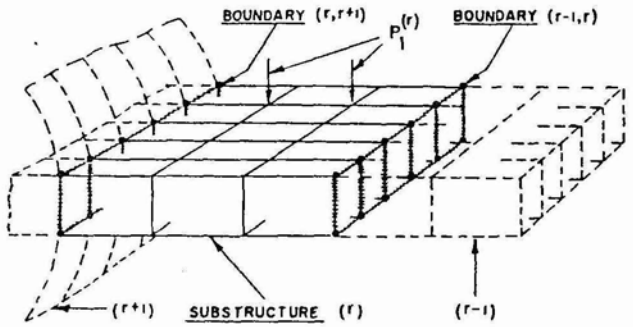


FIG. 1. Typical substructure with fixed boundaries.

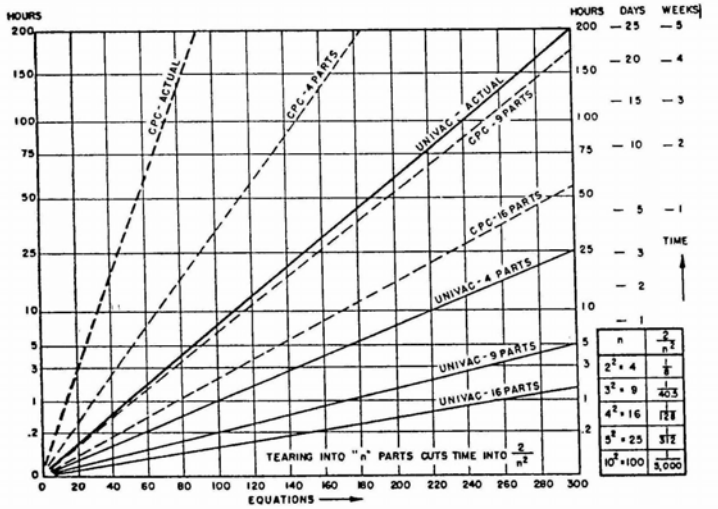


TABLE I. Computer time for matrix inversion.

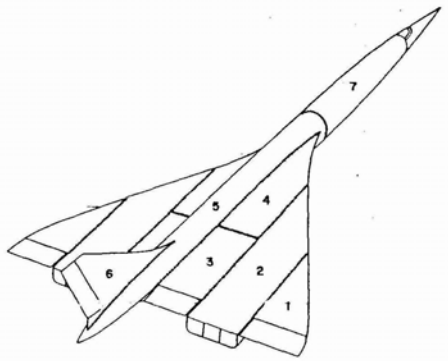


FIG. 3. Typical substructure arrangement for delta aircraft.

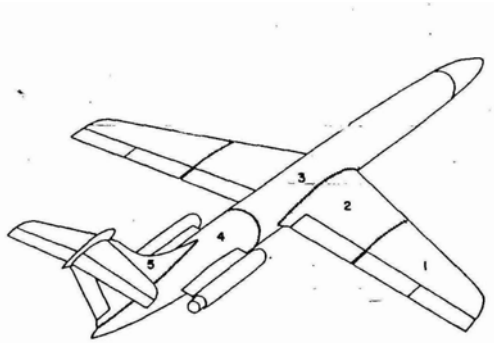


FIG. 4. Typical substructure arrangement for conventional aircraft.

Rationale

- ***For Kron:*** **direct Gaussian elimination has superlinear complexity**
  - **union of subproblems and the connecting problem (each also superlinear) could be solved in fewer overall operations than one large problem**
- ***For Przemieniecki:*** **full airplane structural analysis would not fit in memory of available computers**
  - **individual subproblems fit in memory**

Rationale

- Let problem size be N, number of subdomains be P, and memory capacity be M
- Let problem solution complexity be N^a
- Then subproblem solution complexity is $(N/P)^a$
- Let the cost of connecting the subproblems be $c(N,P)$
- Kron wins if $P\,(N/P)^a + c(N,P) < N^a$

  or $c(N,P) < N^a\,[1-(1/P^{a-1})]$
- Przemieniecki wins if

$$(N/P) < M$$

NB: Kron does not win directly if $a=1$!

Contemporary interest

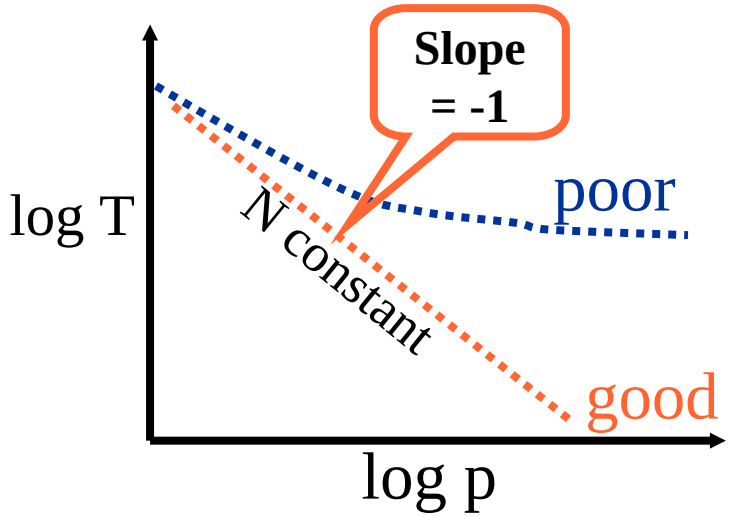
- **Goal is algorithmic scalability:**

  fill up memory of arbitrarily large machines to increase resolution, *while preserving nearly constant* running times* with respect to proportionally smaller problem on one processor

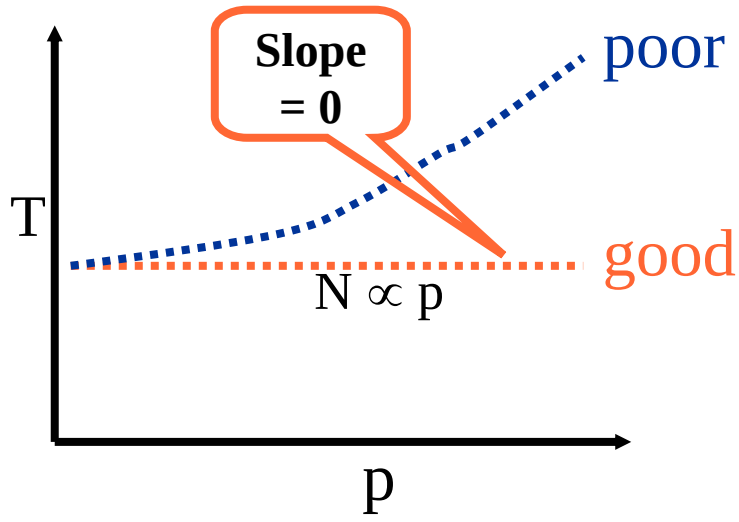
*at worst logarithmically growing

Two definitions of scalability

- **"Strong scaling"**
  - **execution time decreases in inverse proportion to the number of processors**
  - ***fixed size problem overall***

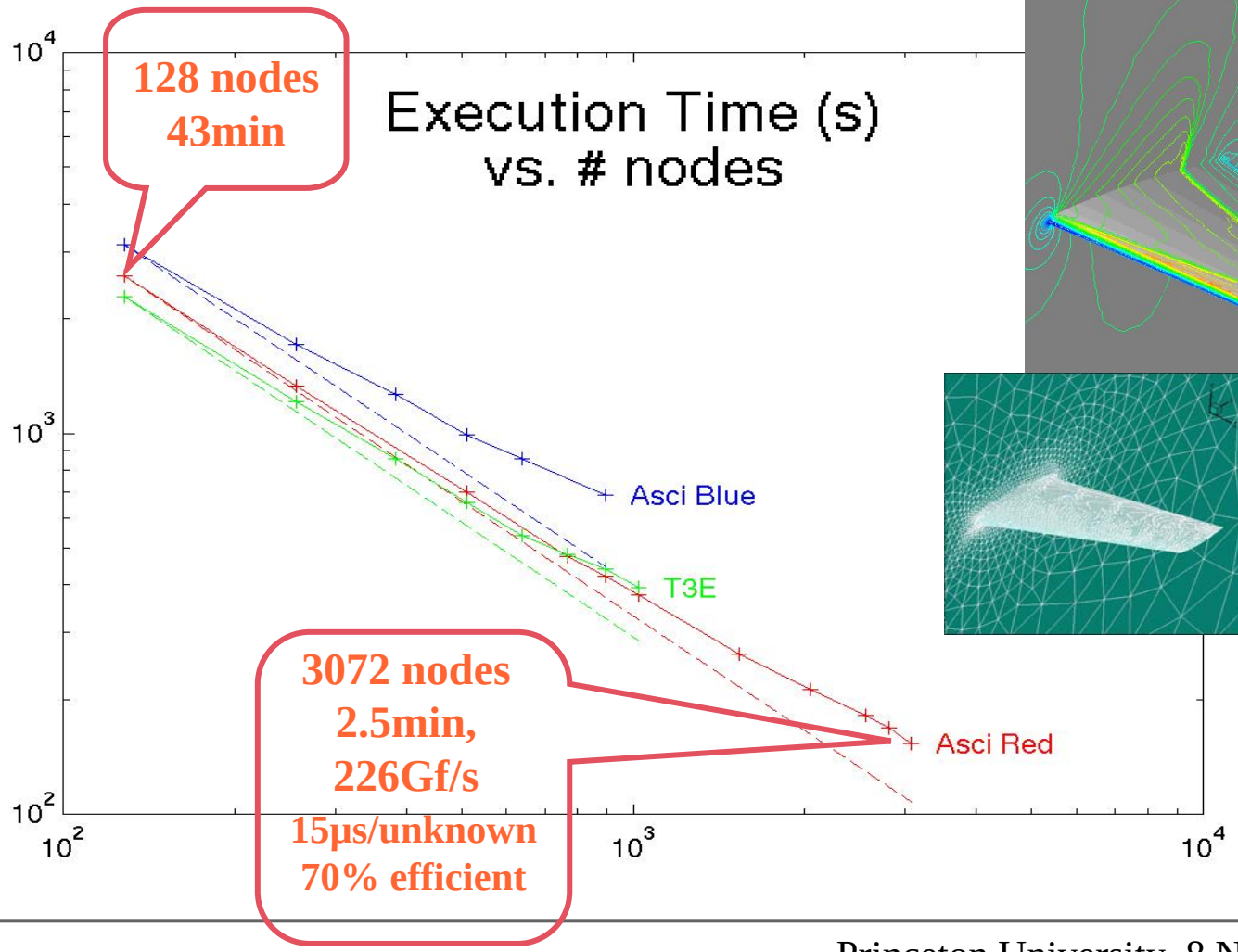


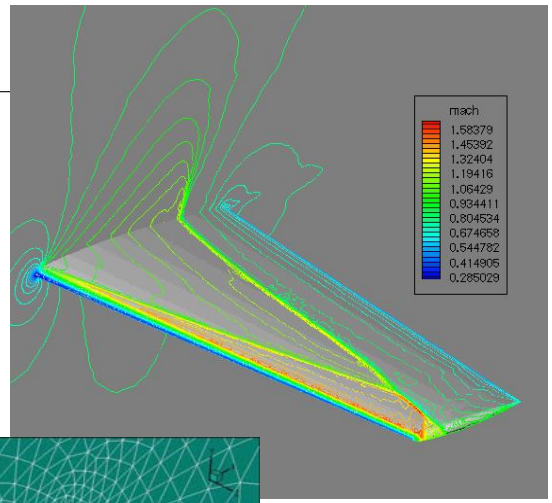
- **"Weak scaling"**
  - **execution time remains constant, as problem size and processor number are increased in proportion**
  - ***fixed size problem per processor***
  - **also known as "Gustafson scaling"**



Strong scaling illus. (1999 Bell Prize)

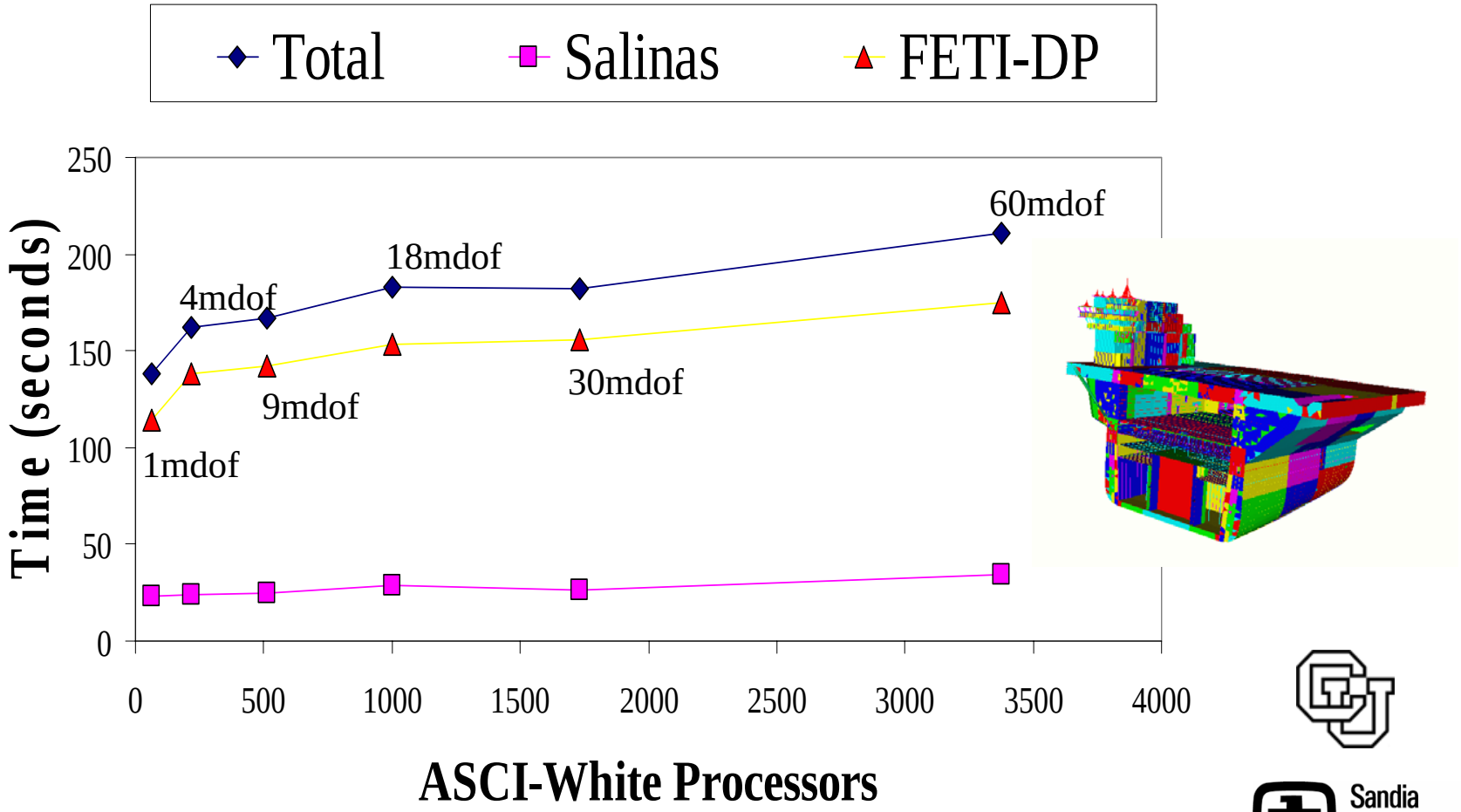
- Newton-Krylov-Schwarz (NKS) algorithm for compressible and incompressible Euler and Navier-Stokes flows
- Used in NASA application FUN3D (M6 wing results below with 11M dof)





Weak scaling illus. (2002 Bell Prize)

- **Finite Element Tearing and Interconnection (FETI) algorithm for solid/shell models**
- **Used in Sandia applications Salinas, Adagio, Andante**



Sandia National Laboratories

Decomposition strategies for $\mathcal{L}u=f$ in Ω

- **Operator decomposition**

$$\mathcal{L} = \sum_k \mathcal{L}_k$$

- **Function space decomposition**

$$f = \sum_k f_k \Phi_k, u = \sum_k u_k \Phi_k$$

- **Domain decomposition**

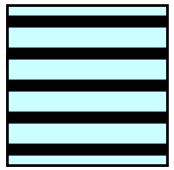
$$\Omega = \bigcup_k \Omega_k$$

Consider the implicitly discretized parabolic case

$$[\frac{I}{\tau} + \mathcal{L}_x + \mathcal{L}_y]u^{(k+1)} = \frac{I}{\tau}u^{(k)} + f$$

Operator decomposition

- **Consider ADI**



$$[\frac{I}{\tau/2}+\mathcal{L}_x]u^{(k+1/2)}=[\frac{I}{\tau/2}-\mathcal{L}_y]u^{(k)}+f$$

$$[\frac{I}{\tau/2}+\mathcal{L}_y]u^{(k+1)}=[\frac{I}{\tau/2}-\mathcal{L}_x]u^{(k+1/2)}+f$$

- **Iteration matrix consists of four multiplicative substeps per timestep**
  - **two sparse matrix-vector multiplies**
  - **two sets of unidirectional bandsolves**
- **Parallelism *within* each substep**
- **But global data exchanges *between* bandsolve substeps**

Function space decomposition

- **Consider a spectral Galerkin method**

$$u(x,y,t) = \sum_{j=1}^{N} a_j(t)\Phi_j(x,y)$$

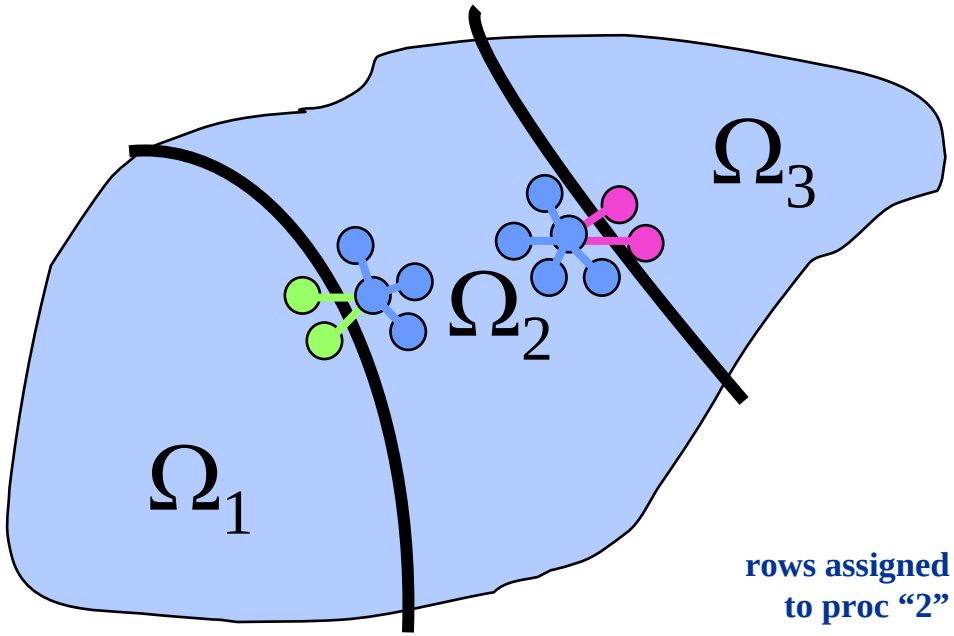
$$\frac{d}{dt}(\Phi_i, u) = (\Phi_i, \mathcal{L}u) + (\Phi_i, f), i = 1,\ldots,N$$

$$\sum_j (\Phi_i, \Phi_j)\frac{da_j}{dt} = \sum_j (\Phi_i, \mathcal{L}\Phi_j)a_j + (\Phi_i, f), i = 1,\ldots,N$$

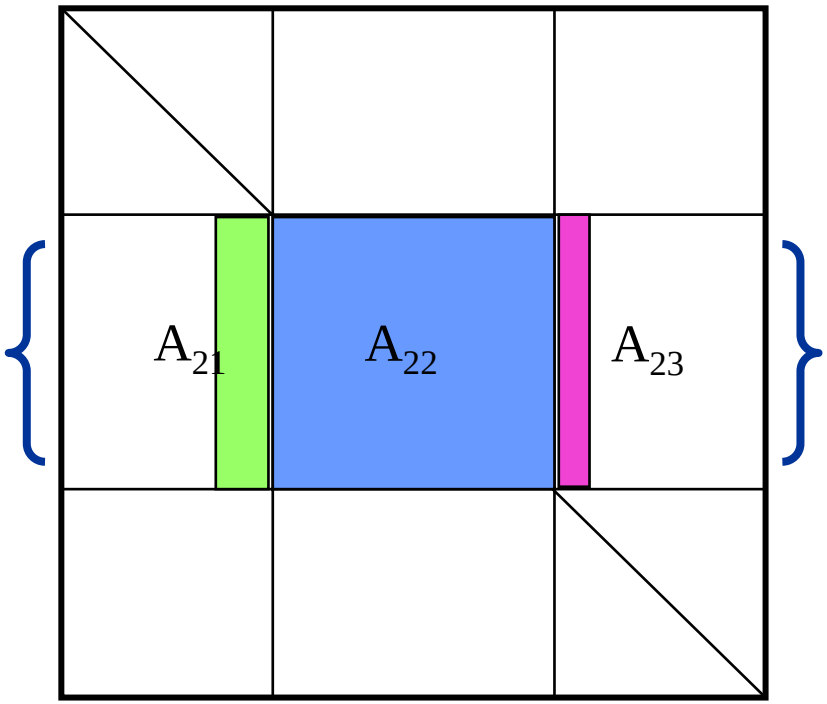
$$\frac{da}{dt} = M^{-1}Ka + M^{-1}f$$

- **Method-of-lines system of ODEs**
- **Perhaps** $M \equiv [(\Phi_j, \Phi_i)], K \equiv [(\Phi_j, \mathcal{L}\Phi_i)]$ **are diagonal matrices**
- **Parallelism across spectral index**
- **But global data exchanges to *transform back* to physical variables at each step**

Domain decomposition



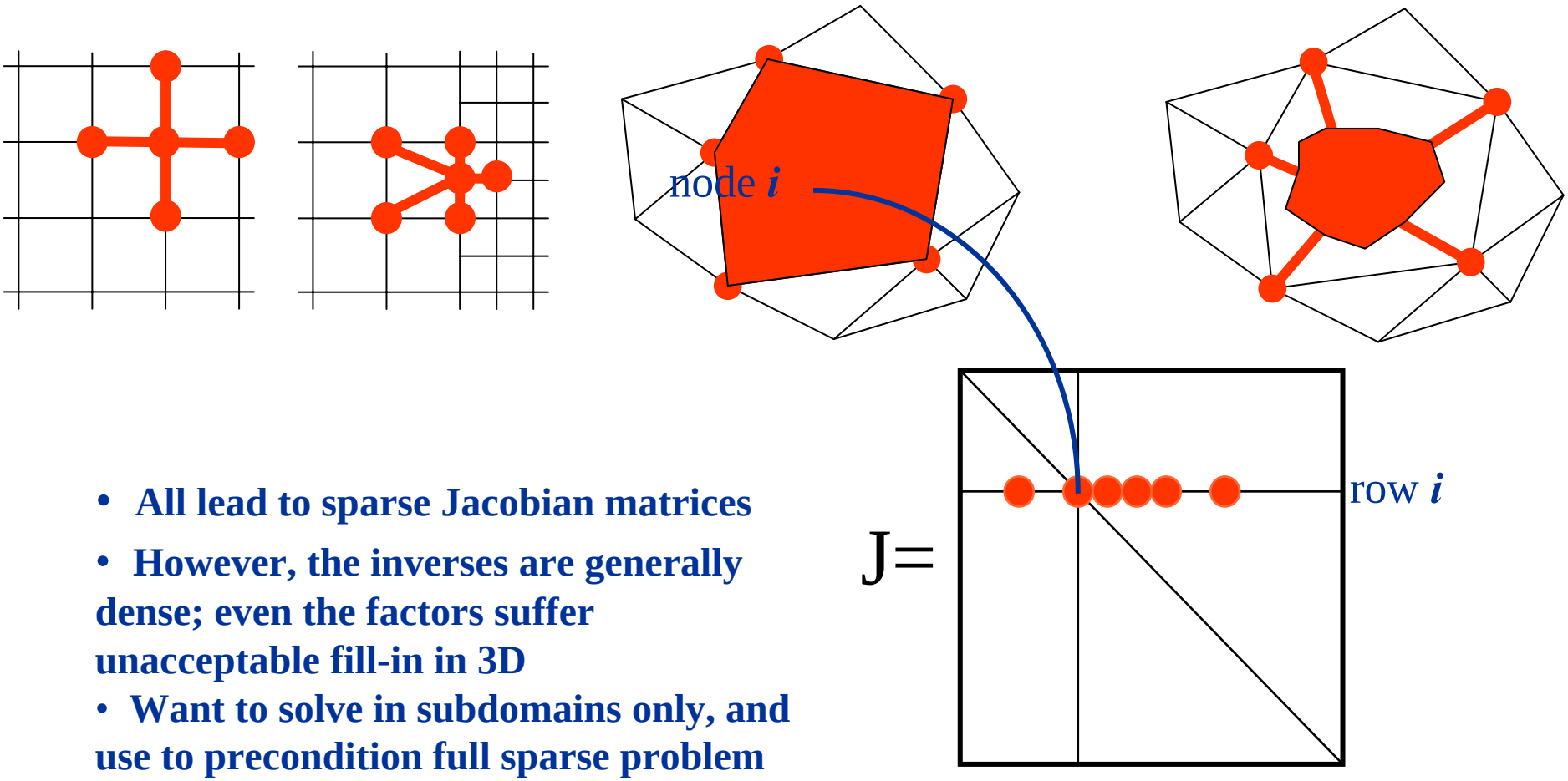
rows assigned to proc "2"



Partitioning of the grid induces block structure on the system matrix (Jacobian)

DD relevant to any local stencil formulation

finite differences finite elements finite volumes



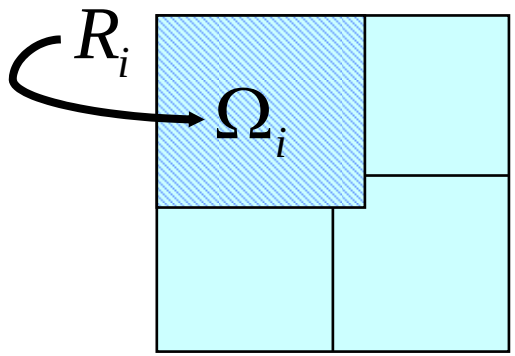
- **All lead to sparse Jacobian matrices**
- **However, the inverses are generally dense; even the factors suffer unacceptable fill-in in 3D**
- **Want to solve in subdomains only, and use to precondition full sparse problem**

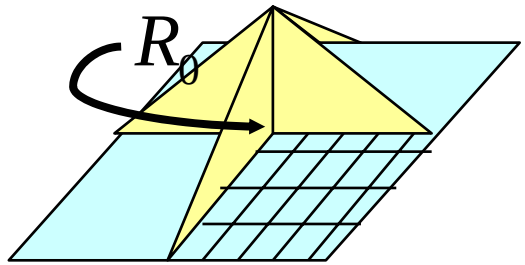
Schwarz domain decomposition method

- **Consider restriction and extension operators for subdomains,** R_i, R_i^T, **(and for possible coarse grid,** R_0, R_0^T**)**
- **Replace discretized** $Au = f$ **with**

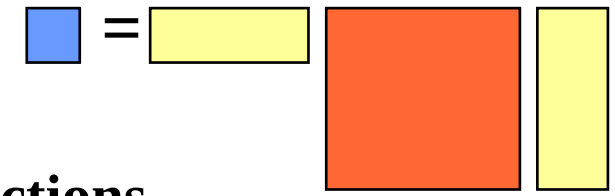
$$B^{-1}Au = B^{-1}f$$

$$B^{-1} = R_0^T A_0^{-1} R_0 + \sum_i R_i^T A_i^{-1} R_i$$





$$A_i = R_i A R_i^T$$



- **Solve by a Krylov method**
- **Matrix-vector multiplies with**
  - **parallelism on each subdomain**
  - **nearest-neighbor exchanges, global reductions**
  - **possible small global system (not needed for parabolic case)**

Remember this formula of Schwarz …

For B^{-1}, to approximate A^{-1} :

$$B^{-1} = \sum_i R_i^T (R_i A R_i^T)^{-1} R_i$$

Krylov bases for sparse systems

- **E.g., conjugate gradients (CG) for symmetric, positive definite systems, and generalized minimal residual (GMRES) for nonsymmetry or indefiniteness**
- **Krylov iteration is an algebraic projection method for converting a high-dimensional linear system into a lower-dimensional linear system**

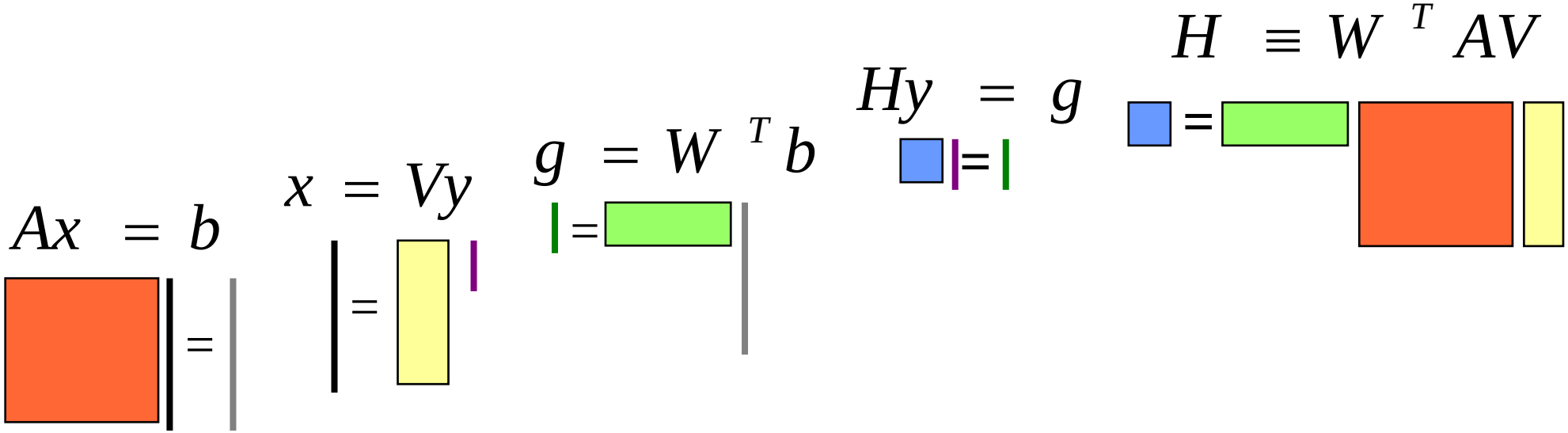
$Ax = b$

$x = Vy$

$g = W^T b$

$Hy = g$

$H \equiv W^T AV$



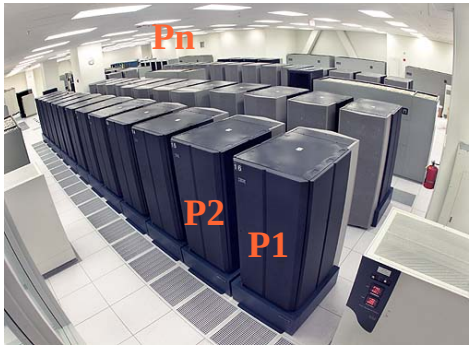
Now, let's compare!

- **Operator decomposition (ADI)**
  - **natural row-based assignment requires *global all-to-all, bulk* data exchanges in each step (for transpose)**
- **Function space decomposition (Fourier)**
  - **Natural mode-based assignment requires *global all-to-all, bulk* data exchanges in each step (for transform)**
- **Domain decomposition (Schwarz)**
  - **Natural domain-based assignment requires *local surface* data exchanges, *global reductions*, and *optional small global* problem**

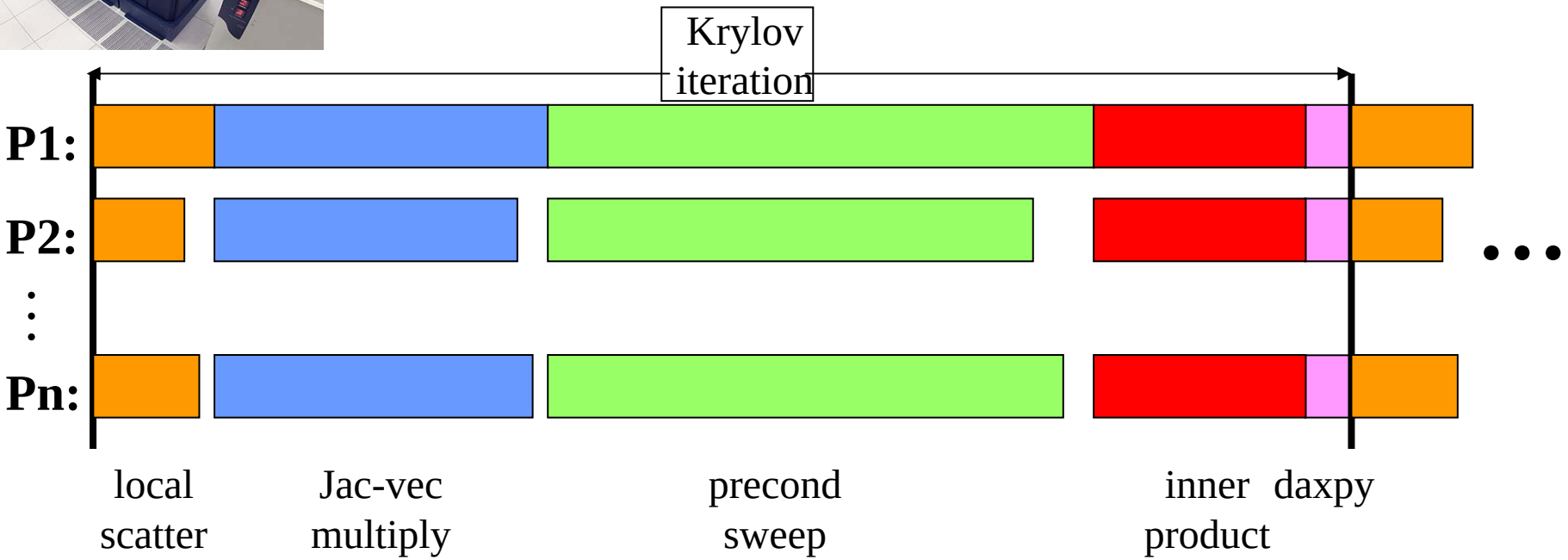
(Of course, domain decomposition can be interpreted as a *special* operator or function space decomposition)

Krylov-Schwarz parallelization summary

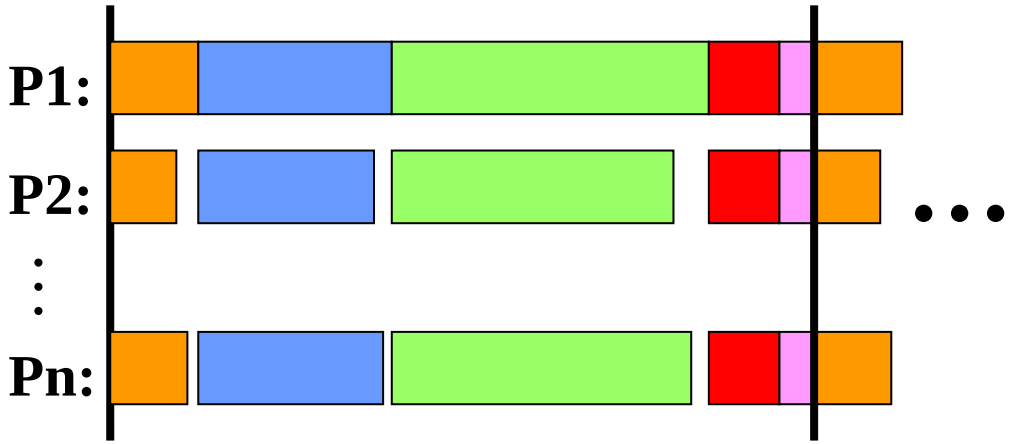
- **Decomposition into concurrent tasks**
  - **by domain**
- **Assignment of tasks to processes**
  - **typically one subdomain per process**
- **Orchestration of communication between processes**
  - **to perform sparse matvec – near neighbor communication**
  - **to perform subdomain solve – nothing**
  - **to build Krylov basis – global inner products**
  - **to construct best fit solution – global sparse solve (redundantly)**
- **Mapping of processes to processors**
  - **typically one process per processor**



Krylov-Schwarz kernel in parallel

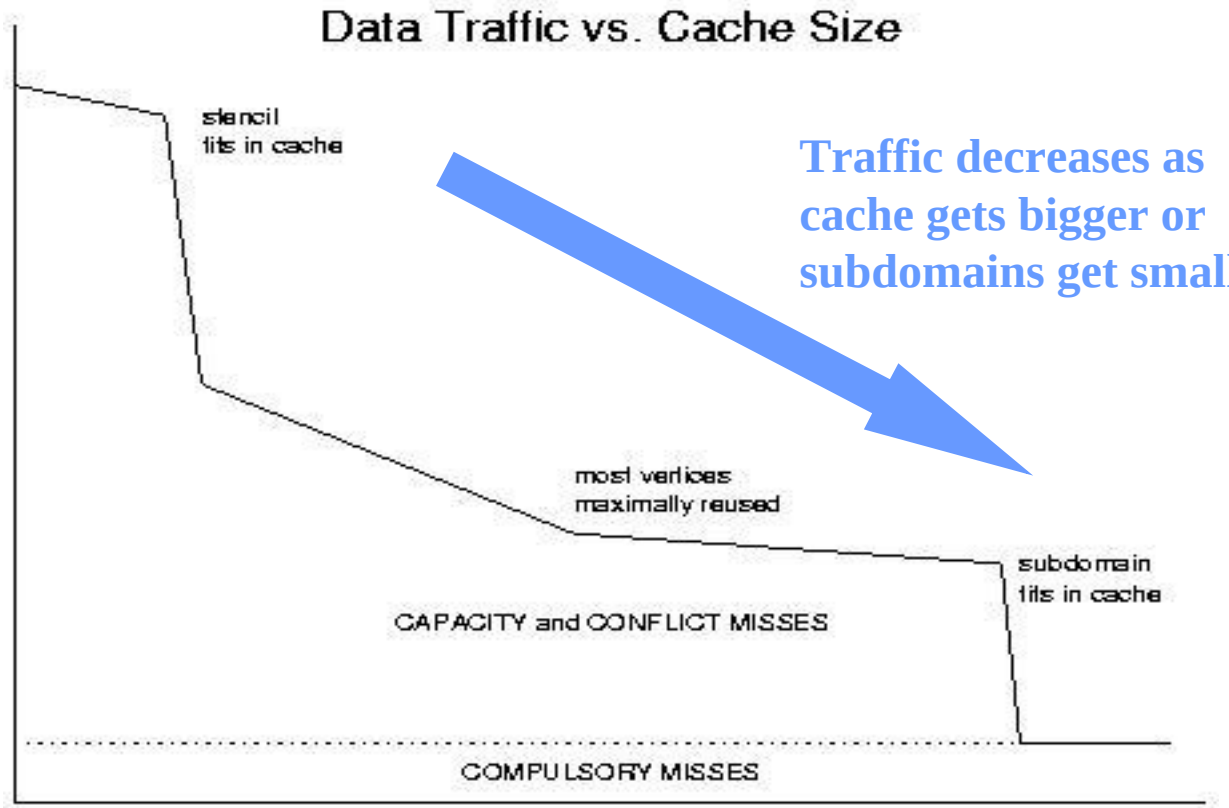


What happens if, for instance, in this (schematicized) iteration, arithmetic speed is ***doubled***, scalar all-gather is ***quartered***, and local scatter is ***cut by one-third***? Each phase is considered separately. Answer is to the right.



Krylov-Schwarz compelling in serial, too

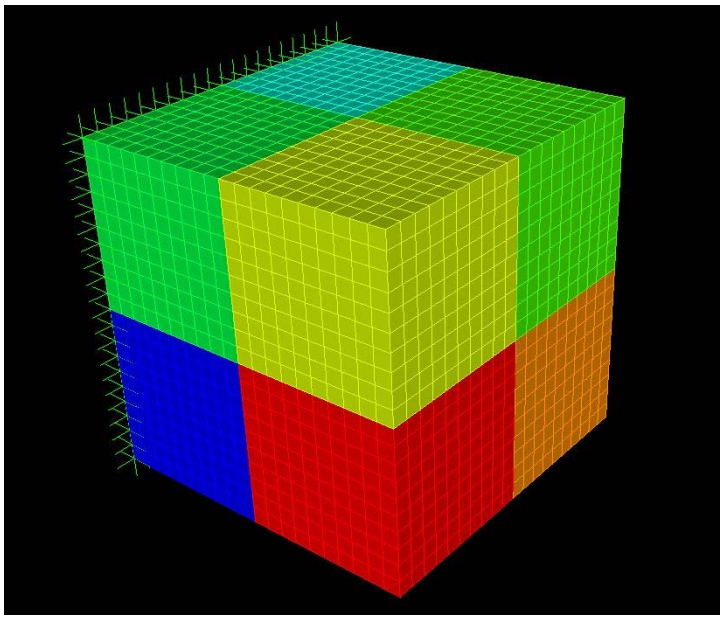
- As successive workingsets “drop” into a level of memory, capacity (and with effort conflict) misses disappear, leaving only compulsory misses, reducing demand on main memory bandwidth
- Cache size is not easily manipulated, but domain size *is*



Estimating scalability of stencil computations

- **Given complexity estimates of the leading terms of:**
  - **the concurrent computation (per iteration phase)**
  - **the concurrent communication**
  - **the synchronization frequency**
- **And a bulk synchronous model of the architecture including:**
  - **internode communication (network topology and protocol reflecting horizontal memory structure)**
  - **on-node computation (effective performance parameters including vertical memory structure)**
- **One can estimate optimal concurrency and optimal execution time**
  - **on per-iteration basis, or overall (by taking into account any granularity-dependent convergence rate)**
  - **simply differentiate time estimate in terms of *(N,P)* with respect to *P*, equate to zero and solve for *P* in terms of *N***

Estimating 3D stencil costs (per iteration)



- grid points in each direction n, total work $N=O(n^3)$
- processors in each direction p, total procs $P=O(p^3)$
- memory per node requirements $O(N/P)$

- concurrent execution time per iteration $A\, n^3/p^3$
- grid points on side of each processor subdomain n/p
- Concurrent neighbor commun. time per iteration $B\, n^2/p^2$
- cost of global reductions in each iteration $C \log p$ or $C\, p^{(1/d)}$
  - C includes synchronization frequency
- same dimensionless units for measuring A, B, C
  - e.g., cost of scalar floating point multiply-add

3D stencil computation illustration

Rich local network, tree-based global reductions

- **total wall-clock time per iteration**

$$T(n,p) = A\frac{n^3}{p^3} + B\frac{n^2}{p^2} + C\log p$$

- **for optimal p,** $\frac{\partial T}{\partial p} = 0$ **, or** $-3A\frac{n^3}{p^4} - 2B\frac{n^2}{p^3} + \frac{C}{p} = 0,$

or (with $\theta \equiv \frac{32B^3}{243A^2C}$ **),**

$$p_{opt} = \left(\frac{3A}{2C}\right)^{1/3}\left(\left[1+(1-\sqrt{\theta})\right]^{1/3} + \left[1-(1-\sqrt{\theta})\right]^{1/3}\right)\cdot n$$

- **without "speeddown," p can grow with n**
- **in the limit as** $B/C \to 0$

$$p_{opt} = \left(\frac{3A}{C}\right)^{1/3}\cdot n$$

3D stencil computation illustration

Rich local network, tree-based global reductions

- **optimal running time**

$$T(n, p_{opt}(n)) = \frac{A}{\rho^3} + \frac{B}{\rho^2} + C\log(\rho n),$$

where

$$\rho = \left(\frac{3A}{2C}\right)^{1/3}\left(\left[1+(1-\sqrt{\theta})\right]^{1/3} + \left[1-(1-\sqrt{\theta})\right]^{1/3}\right)$$

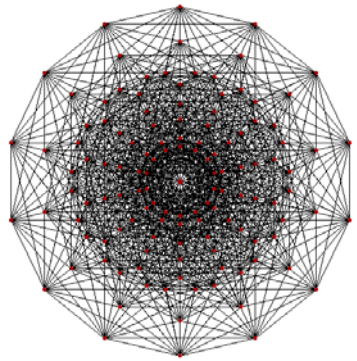
- **limit of infinite neighbor bandwidth, zero neighbor latency ($B \to 0$)**

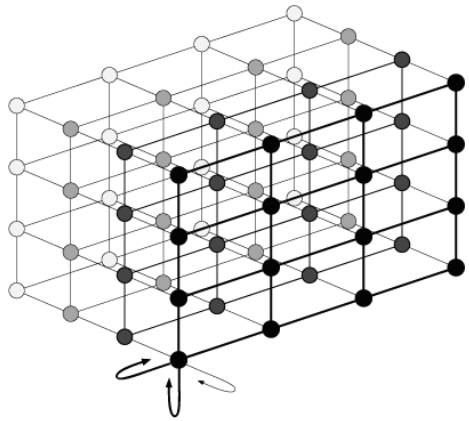
$$T(n, p_{opt}(n)) = C\left[\log n + \frac{1}{3}\log\frac{A}{C} + const.\right]$$

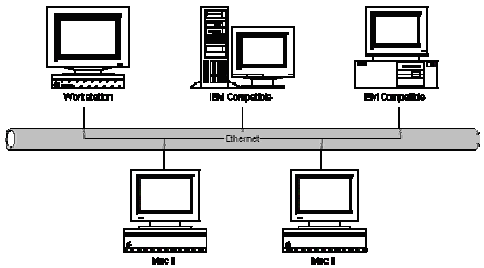
(This analysis is on a per iteration basis; complete analysis multiplies this cost by an iteration count estimate that generally depends on *n* and *p*.)

Scalability results for DD stencil computations

- **With tree-based (logarithmic) global reductions and scalable nearest neighbor hardware:**
  - **optimal number of processors scales *linearly* with problem size**
- **With 3D torus-based global reductions and scalable nearest neighbor hardware:**
  - **optimal number of processors scales as *three-fourths* power of problem size (almost "scalable")**
- **With common network bus (heavy contention):**
  - **optimal number of processors scales as *one-fourth* power of problem size (not "scalable")**





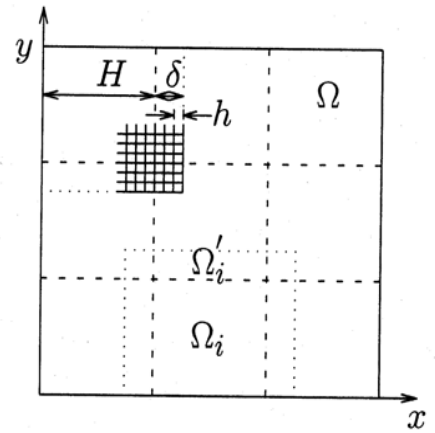


Resource scaling for PDEs

- **For 3D problems, work is often proportional to the four-thirds power of memory, because**
  - **for equilibrium problems, work scales with problem size times number of iteration steps -- proportional to resolution in single spatial dimension**
  - **for evolutionary problems, work scales with problems size times number of time steps -- CFL arguments place latter on order of spatial resolution, as well**
- **Proportionality constant can be adjusted over a very wide range by both discretization (high-order implies more work per point and per memory transfer) and by algorithmic tuning**
- **Machines designed for PDEs can be "memory-thin"**
- **If frequent time frames are to be captured, other resources -- disk capacity and I/O rates -- must both scale linearly with work, more stringently than for memory.**

Factoring convergence into estimates

- **Krylov-Schwarz iterative methods typically converge in a number of iterations that scales as the square-root of the condition number of the Schwarz-preconditioned system**
- **In terms of *N* and *P*, where for *d*-dimensional isotropic problems, $N=h^{-d}$ and $P=H^{-d}$, for mesh parameter *h* and subdomain diameter *H*, iteration counts may be estimated as follows:**



Preconditioning Type	in 2D	in 3D
Point Jacobi	$O(N^{1/2})$	$O(N^{1/3})$
Domain Jacobi (δ=0)	$O((NP)^{1/4})$	$O((NP)^{1/6})$
1-level Additive Schwarz	$O(P^{1/2})$	$O(P^{1/3})$
2-level Additive Schwarz	$O(1)$	$O(1)$

Where do these results come from?

- **Point Jacobi is well known (see any book on the numerical analysis of elliptic problems)**
- **Subdomain Jacobi has interesting history (see ahead a few slides)**
- **Schwarz theory is neatly and abstractly summarized in Section 5.2 of book by Smith, Bjorstad & Gropp ("Widlund School")**
  - **condition number of preconditioned operator, $\kappa(B^{-1}A) \leq \omega\,[1+\rho(\mathcal{E})]\,C_0^2$**
  - **C_0^2 is a splitting constant for the subspaces of the decomposition**
  - **$\rho(\mathcal{E})$ is a measure of the orthogonality of the subspaces**
  - **ω is a measure of the approximation properties of the subspace solvers (can be unity for exact subdomain solves)**
  - **obtained by Rayleigh quotient estimates for extremal eigenvalues of $B^{-1}A$ and theorem bounding sums of projections**
  - **upper and lower bounds are estimated for different subspaces, different operators, and different subspace solvers and the "crank" is turned**

Comments on the Schwarz results

- **Basic Schwarz estimates are for:**
  - ***self-adjoint*** **elliptic operators**
  - ***positive definite*** **operators**
  - ***exact*** **subdomain solves,** A_i^{-1}
  - ***two-way*** **overlapping with** R_i, R_i^T
  - ***generous*** **overlap,** $\delta = O(H)$ **(otherwise 2-level result is** $O(1+H/\delta)$**)**
- **Extensible to:**
  - ***nonself-adjointness*** **(e.g, convection)**
  - ***indefiniteness*** **(e.g., wave Helmholtz)**
  - ***inexact*** **subdomain solves**
  - ***one-way*** **overlap communication ("restricted additive Schwarz")**
  - ***small*** **overlap**

Comments on the Schwarz results, cont.

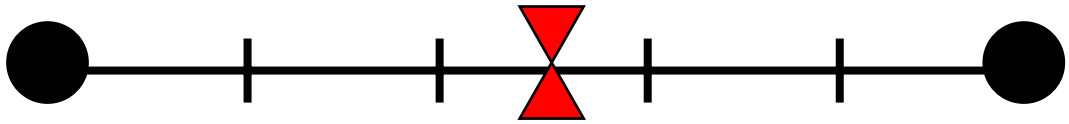
- **Theory still requires "sufficiently fine" coarse mesh**
  - **However, coarse space need *not* be nested in the fine space or in the decomposition into subdomains**
- **Practice is better than one has any right to expect**

"In theory, theory and practice are the same ...
In practice they're not!"
— *Yogi Berra*

- **Wave Helmholtz (e.g., acoustics) is delicate at high frequency:**
  - **standard Schwarz Dirichlet boundary conditions can lead to undamped resonances within subdomains,** $u_\Gamma = 0$
  - **remedy involves Robin-type transmission boundary conditions on subdomain boundaries,** $(u + \alpha\, \partial u / \partial n)_\Gamma = 0$

Block Jacobi preconditioning: 1D example

Consider the scaled F.D. Laplacian on an interval:

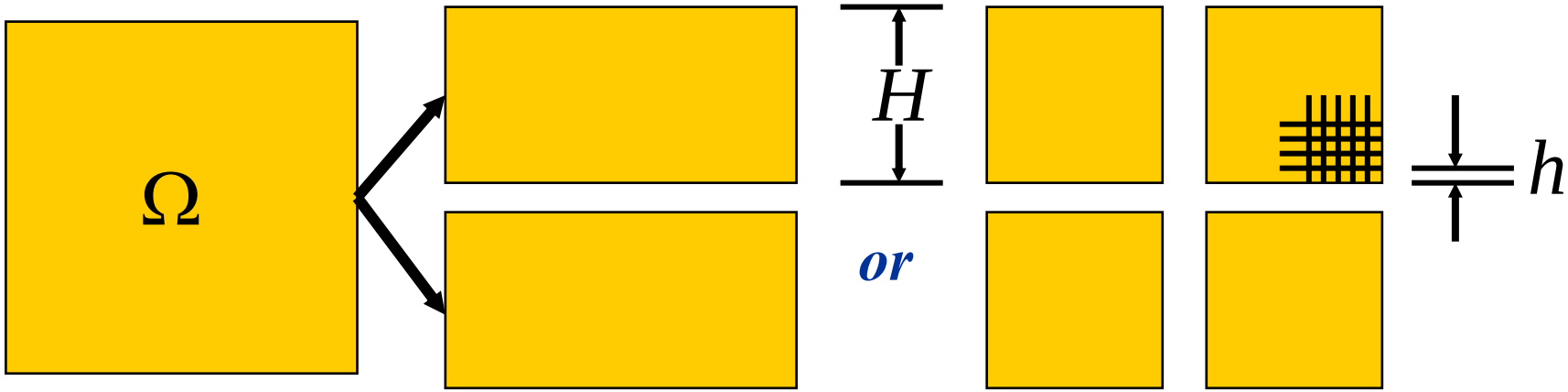


$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \bigcirc & \\ & \bigcirc & 2 & -1 \\ & & -1 & 2 \end{bmatrix}$$

$$B^{-1}A = \begin{bmatrix} 1 & & \tfrac{1}{3} & \\ & 1 & \tfrac{2}{3} & \\ & \tfrac{2}{3} & 1 & \\ & \tfrac{1}{3} & & 1 \end{bmatrix} = I - \begin{bmatrix} & & \tfrac{1}{3} & \\ & & \tfrac{2}{3} & \\ & \tfrac{2}{3} & & \\ & \tfrac{1}{3} & & \end{bmatrix}$$

Bound on block Jacobi preconditioning

- **Consider decomposition of 1D, 2D, or 3D domain into subdomains by cutting planes**



- **Using functional analysis, Dryja & Widlund (1987) showed that zero-overlap Schwarz on improves conditioning from $O(h^{-2})$ for native elliptic problem to $O(H^{-1}h^{-1})$**

Mirror result from linear algebra

- **Chang & Schultz (1994) proved same result from algebraic approach, from eigenanalysis of $(B^{-1}A)$, where A is F.D. Laplacian in 1D, 2D, or 3D, and B is A with entries removed by arbitrary cutting planes**
- **Their Theorem 2.4.7: Given $n \times n \times n$ grid, cut by**
  - **q planes in x (slabs)**
  - **q planes in x or y (beams)**
  - **q planes in $x, y,$ or z (subcubes)**

  (with cuts anywhere) then $\kappa(B^{-1}A) \leq qn + q + 1$
- **Note: $q = O(H^{-1})$ and $n = O(h^{-1})$ if cut evenly**
- **Proof: eigenanalysis of low-rank matrices $I - (B^{-1}A)$**

Mirror results from graph theory

- **Boman & Hendrickson (2003) proved same result from graph-theoretic approach, using their new "support theory"**
- **Section 9 of their SIMAX paper "Support Theory for Preconditioning," using *congestion-dilation* lemma from graph theory (Vaidya *et al.*) derives $O(h^{-2})$, for *point* Jacobi**
- **Extended by B & H to *block* Jacobi, to get $O(H^{-1}h^{-1})$**
- **Many different mathematical tools can be used to explore this divide-and-conquer preconditioning idea!**

"Unreasonable effectiveness" of Schwarz

- **When does the sum of partial inverses equal the inverse of the sums? When the decomposition is right!**
  Let $\{r_i\}$ be a complete set of orthonormal row eigenvectors for A : $r_i A = a_i r_i$ **or** $a_i = r_i A r_i^T$

  Then
  $$A = \Sigma_i r_i^T a_i r_i$$

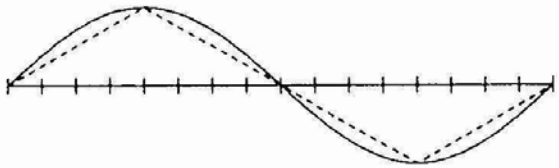
  and
  $$A^{-1} = \Sigma_i r_i^T a_i^{-1} r_i = \Sigma_i r_i^T (r_i A r_i^T)^{-1} r_i$$

  — the Schwarz formula!

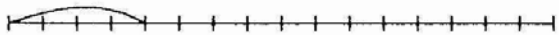
- **Good decompositions are a compromise between conditioning and parallel complexity, in practice**

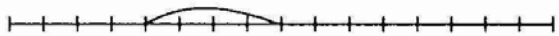
Schwarz subspace decomposition

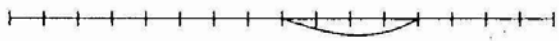
Consider a one-dimensional example. The function $u(x)$ sketched below

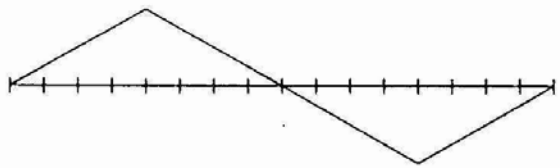


can be decomposed into the sum of the following five functions:



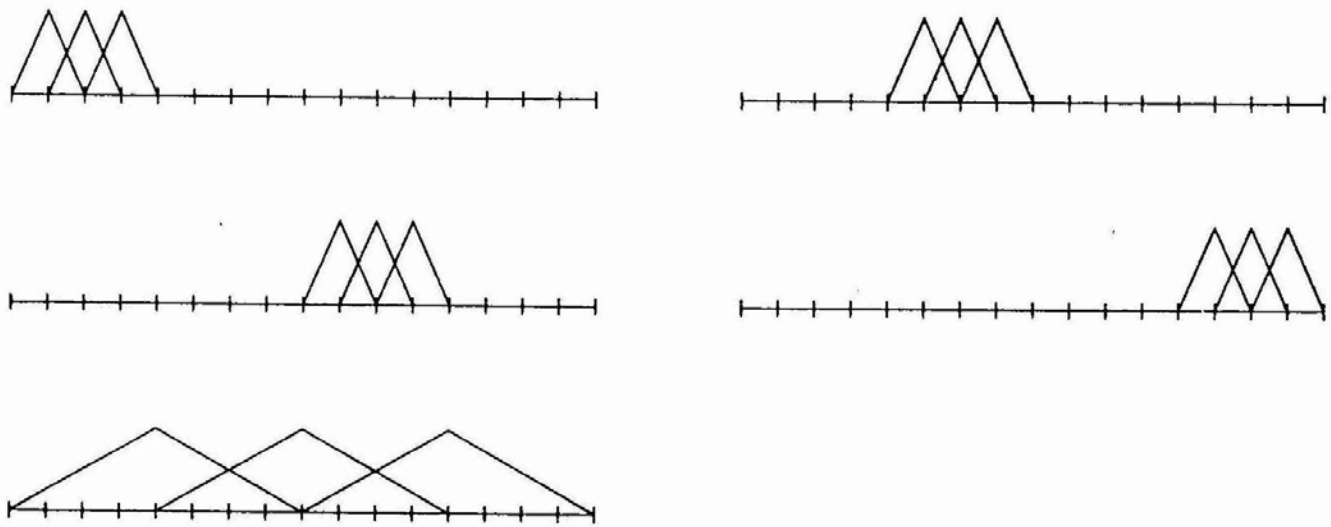






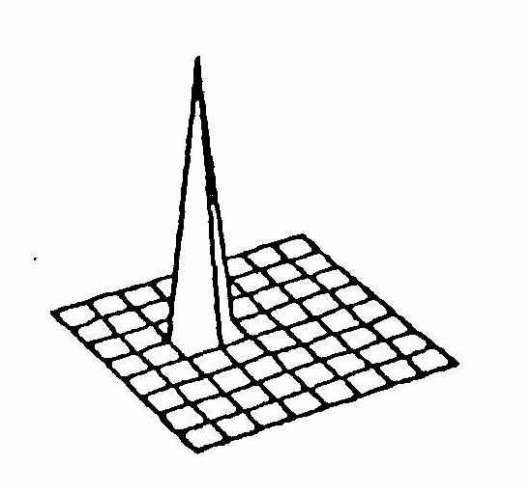
Schwarz subspace decomposition

Piecewise linear finite element bases for each of the five functions are shown below:

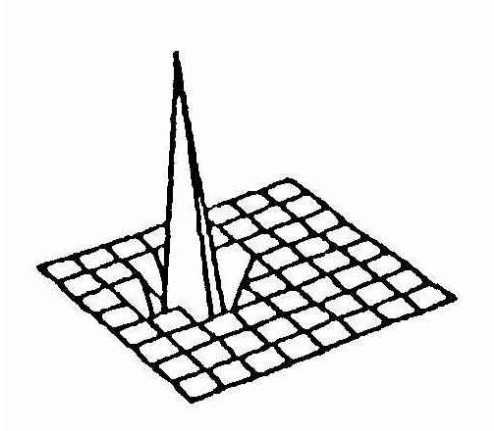


The first four of these subspaces are **mutually orthogonal**. The last one is **not orthogonal** to any of the others.

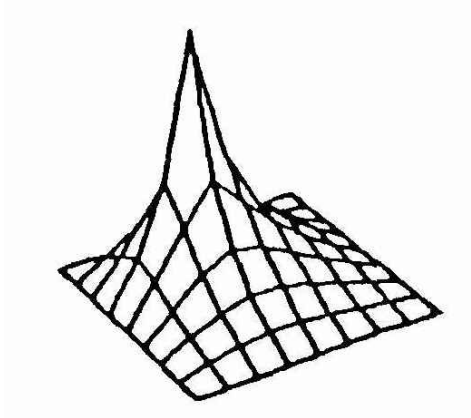
“Unreasonable effectiveness” of Schwarz, cont.



Delta function, $\delta(x)$



$A\ \delta(x)$



$A^{-1}\ \delta(x)$

- **Forward operator is localized and sparse**
- **Inverse operator is dense but locally concentrated**

Basic domain decomposition concepts

- **Iterative correction**
- **Schwarz preconditioning**
- **Schur preconditioning**
- **Polynomial combinations of Schwarz projections**
- **Schwarz-Schur combinations**

Iterative correction

- **The most basic idea in iterative methods:**

$$u \leftarrow u + B^{-1}(f - Au)$$

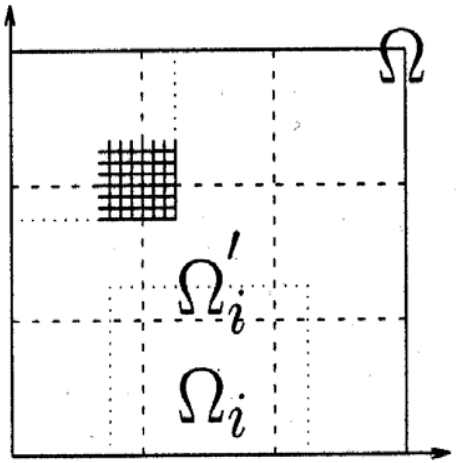
- **Evaluate residual accurately, but solve approximately, where B^{-1} is an approximate inverse to A**
- **A sequence of complementary approximate solves can be used, e.g., with B_1 and B_2 one has**

$$u \leftarrow u + [B_1^{-1} + B_2^{-1} - B_2^{-1} A B_1^{-1}](f - Au)$$

- **Scale recurrence, e.g., with $B_2^{-1} = R^T (RAR^T)^{-1} R$, leads to *multilevel methods***
- **Optimal polynomials of $(B^{-1}A)$ leads to various *preconditioned Krylov methods***

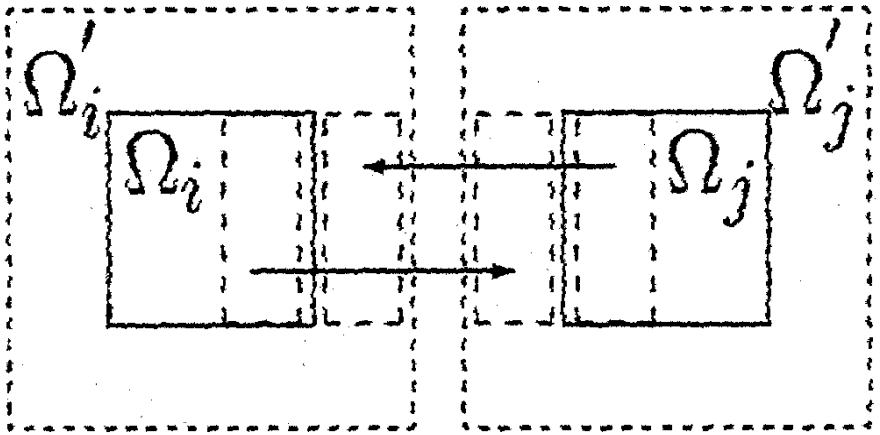
Schwarz preconditioning

- **Given $A x = b$, partition x into subvectors, corresp. to subdomains Ω_i of the domain Ω of the PDE, nonempty, possibly overlapping, whose union is all of the elements of $x \in \Re^n$**



- **Let Boolean rectangular matrix R_i extract the i^{th} subset of x :**

$$x_i = R_i x$$



- **Let $A_i = R_i A R_i^T$**

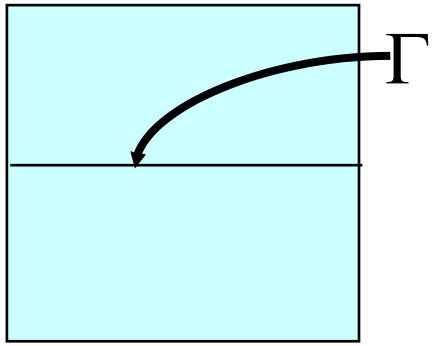
$$B^{-1} = \sum_i R_i^T A_i^{-1} R_i$$

The Boolean matrices are gather/scatter operators, mapping between a global vector and its subdomain support

Schur complement substructuring

- **Given a partition**

$$\begin{bmatrix} A_{ii} & A_{i\Gamma} \\ A_{\Gamma i} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_i \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_i \\ f_\Gamma \end{bmatrix}$$



- **Condense:**

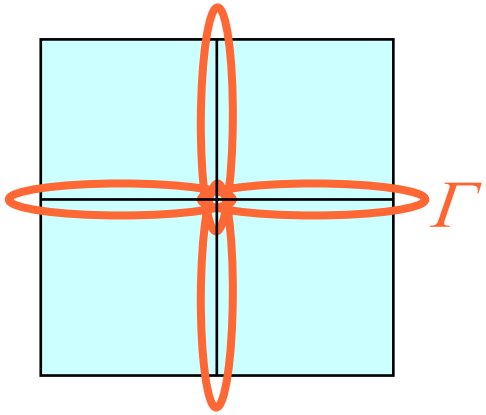
$$Su_\Gamma = g \qquad S \equiv A_{\Gamma\Gamma} - A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma} \qquad g \equiv f_\Gamma - A_{\Gamma i} A_{ii}^{-1} f_i$$

- **Properties of the Schur complement:**
  - **smaller than original A, but generally dense**
  - **expensive to form, to store, to factor, and to solve**
  - **better conditioned than original A**
- **Therefore, solve iteratively, with action of S on each Krylov vector, using a preconditioner M^{-1}**
- **In continuous form, S is a Steklov-Poincaré operator**

Schur preconditioning in global system

- Let M^{-1} be a good preconditioner for S
- Let

$$B^{-1} = \left(\begin{bmatrix} A_{ii} & 0 \\ A_{\Gamma i} & I \end{bmatrix} \begin{bmatrix} I & A_{ii}^{-1} A_{i\Gamma} \\ 0 & M \end{bmatrix} \right)^{-1}$$



- Then B^{-1} is a preconditioner for A
- So, instead of $M^{-1} S u_\Gamma = M^{-1} g$, use full system

$$B^{-1} \begin{bmatrix} A_{ii} & A_{i\Gamma} \\ A_{\Gamma i} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_i \\ u_\Gamma \end{bmatrix} = B^{-1} \begin{bmatrix} f_i \\ f_\Gamma \end{bmatrix}$$

- Here, solves with A_{ii} may be done approximately since all degrees of freedom are retained

Schwarz polynomials

- **Polynomials of Schwarz projections that are hybrid combinations of additive and multiplicative may be appropriate for certain implementations**
- **We may solve the fine subdomains concurrently and follow with a coarse grid (redundantly/cooperatively)**

$$u \leftarrow u + \Sigma_i B_i^{-1}(f - Au)$$

$$u \leftarrow u + B_0^{-1}(f - Au)$$

- **This leads to algorithm "Hybrid II" in S-B-G'96:**

$$B^{-1} = B_0^{-1} + (I - B_0^{-1}A)(\Sigma_i B_i^{-1})$$

- **Convenient for SPMD programming model**

Schwarz-on-Schur

- **Preconditioning the Schur complement is complex in and of itself; Schwarz is used on the reduced problem**
- **Neumann-Neumann**

$$M^{-1} = \Sigma_i D_i R_i^T S_i^{-1} R_i D_i$$

- **Balancing Neumann-Neumann**

$$M^{-1} = M_0^{-1} + (I - M_0^{-1} S)(\Sigma_i D_i R_i^T S_i^{-1} R_i D_i)(I - S M_0^{-1})$$

- **Other variants:**
  - **Bramble-Pasciak-Schatz**
  - **multigrid on the Schur complement**

Newton-Krylov-Schwarz: a nonlinear PDE "workhorse"

$$F(u) \approx F(u_c) + F'(u_c)\delta u = 0$$
$$u = u_c + \lambda\, \delta u$$

$$J\delta u = -F$$
$$\delta u = \underset{x \in V \equiv \{F, JF, J^2F, \cdots\}}{\text{argmin}} \{Jx + F\}$$

$$M^{-1} J\delta u = -M^{-1} F$$
$$M^{-1} = \sum_i R_i^T (R_i J R_i^T)^{-1} R_i$$

Newton

nonlinear solver

asymptotically quadratic

Krylov

accelerator

spectrally adaptive

Schwarz

preconditioner

parallelizable

Jacobian-free Newton-Krylov

- **In the Jacobian-Free Newton-Krylov (JFNK) method, a Krylov method solves the linear Newton correction equation, requiring Jacobian-vector products**
- **These are approximated by the Fréchet derivatives**

$$J(u)v \approx \frac{1}{\varepsilon}[F(u+\varepsilon v) - F(u)]$$

(where ε is chosen with a fine balance between approximation and floating point rounding error) or automatic differentiation, so that the actual Jacobian elements are *never explicitly needed*

- **One builds the Krylov space on a true $F'(u)$ (to within numerical approximation)**

Recall idea of preconditioning

- **Krylov iteration is expensive in memory and in function evaluations, so subspace dimension *k* must be kept small in practice, through preconditioning the Jacobian with an approximate inverse, so that the product matrix has low condition number in**

$$(B^{-1}A)x = B^{-1}b$$

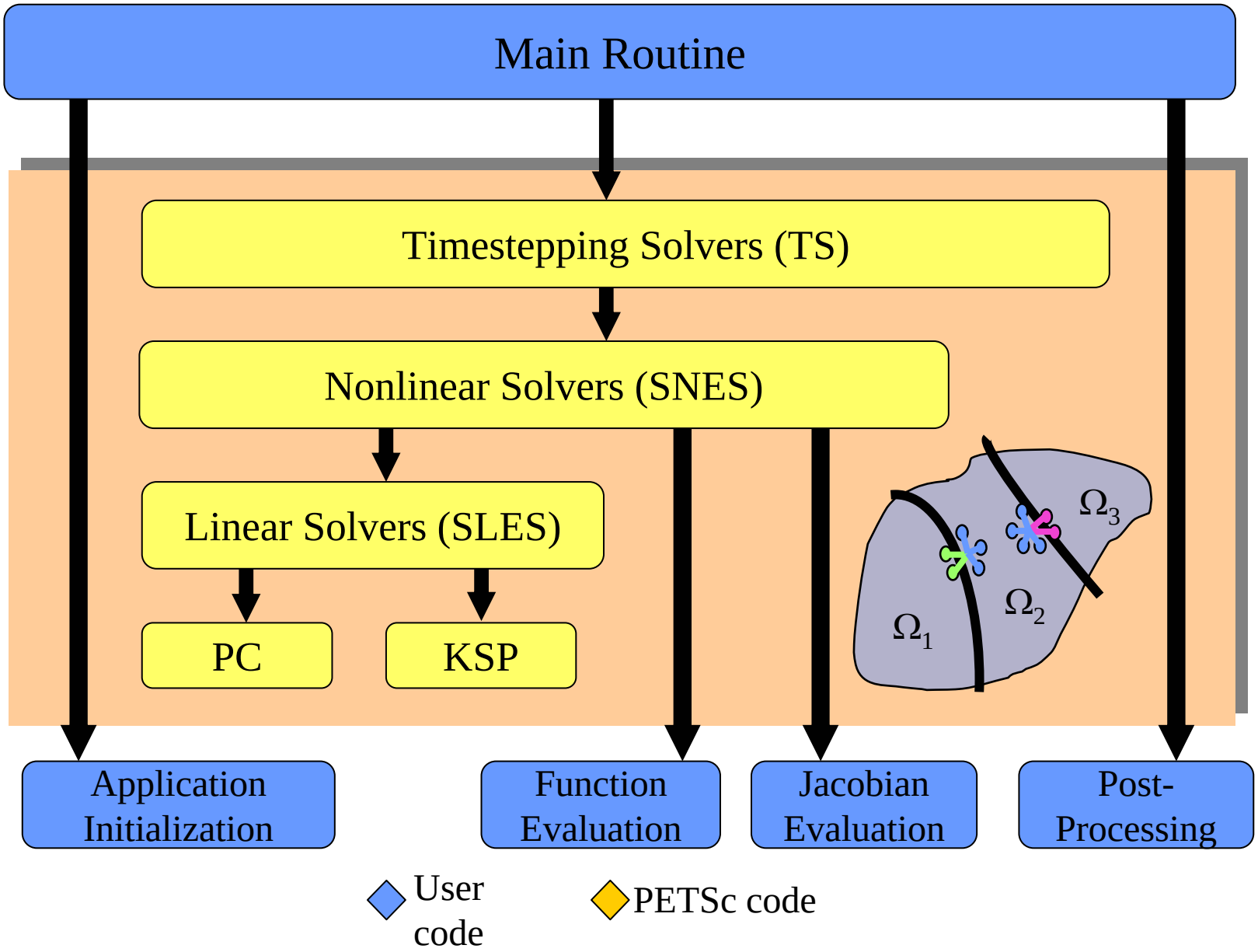
- **Given the ability to apply the action of B^{-1} to a vector, preconditioning can be done on either the left, as above, or the right, as in, e.g., for matrix-free:**

$$JB^{-1}v \approx \frac{1}{\varepsilon}[F(u+\varepsilon B^{-1}v) - F(u)]$$

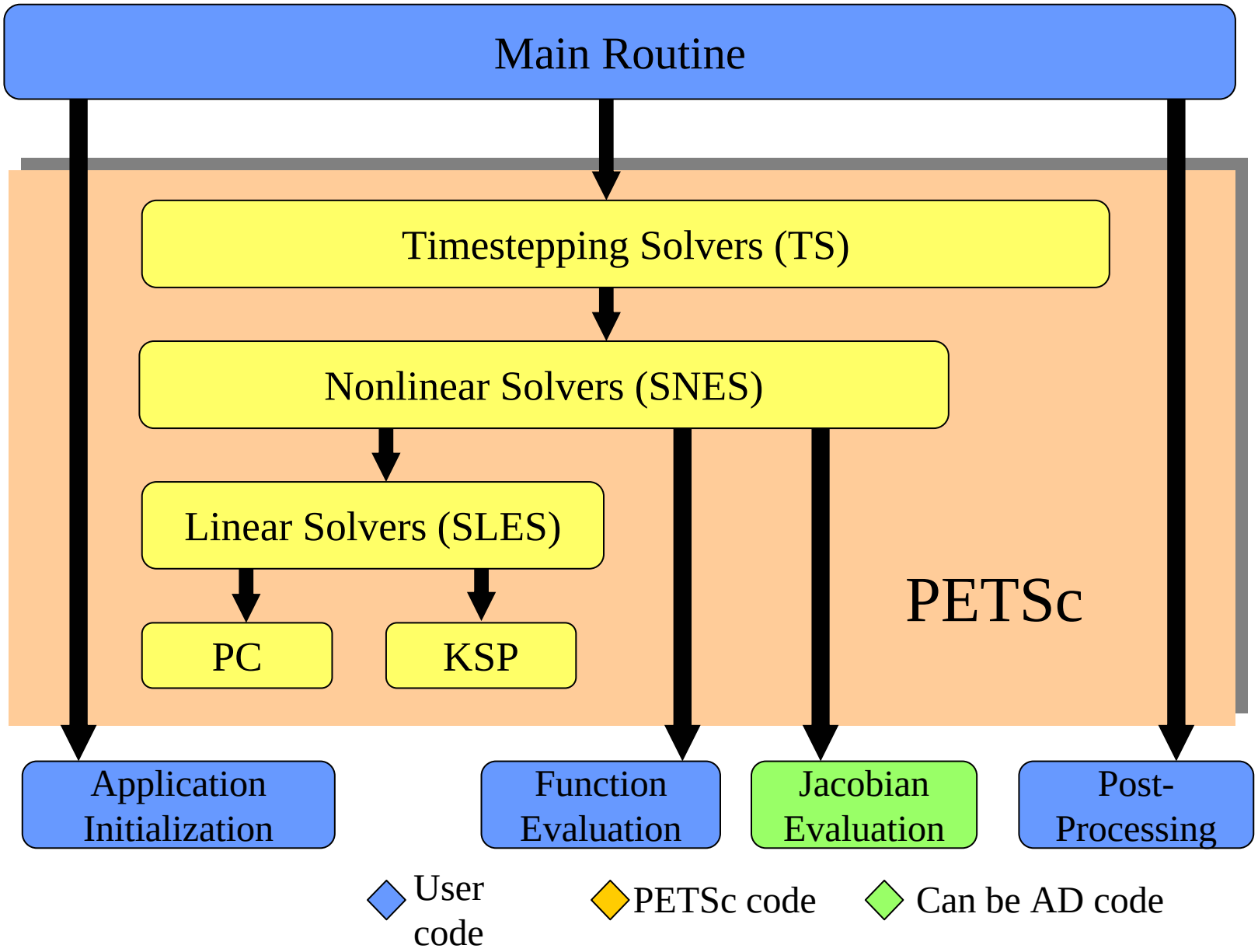
Philosophy of Jacobian-free NK

- **To *evaluate* the linear residual, we use the true $F'(u)$, giving a true Newton step and asymptotic quadratic Newton convergence**
- **To *precondition* the linear residual, we do anything convenient that uses understanding of the dominant physics/mathematics in the system and respects the limitations of the parallel computer architecture and the cost of various operations:**
  - **Jacobian blocks decomposed for parallelism (Schwarz)**
  - **Jacobian of lower-order discretization**
  - **Jacobian with "lagged" values for expensive terms**
  - **Jacobian stored in lower precision**
  - **Jacobian of related discretization**
  - **operator-split Jacobians**
  - **physics-based preconditioning**

NKS efficiently implemented in PETSc's MPI-based distributed data structures



User code/PETSc library interactions



Nonlinear Schwarz preconditioning

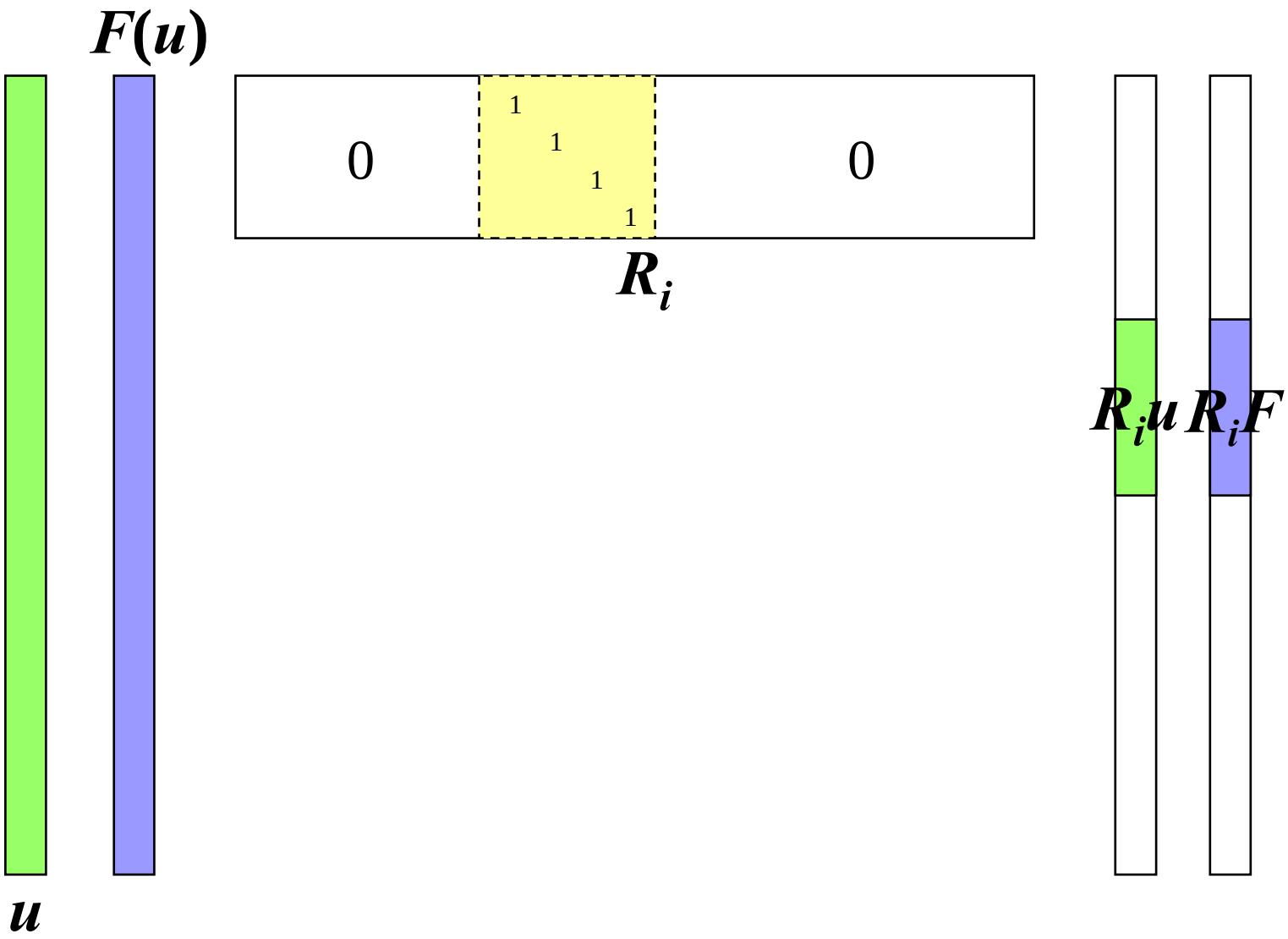
- **Nonlinear Schwarz has Newton both *inside* and *outside* and is fundamentally Jacobian-free**
- **It replaces $F(u)=0$ with a new nonlinear system possessing the same root, $\Phi(u)=0$**
- **Define a correction $\delta_i(u)$ to the i^{th} partition (e.g., subdomain) of the solution vector by solving the following local nonlinear system:**

$$R_i F(u+\delta_i(u))=0$$

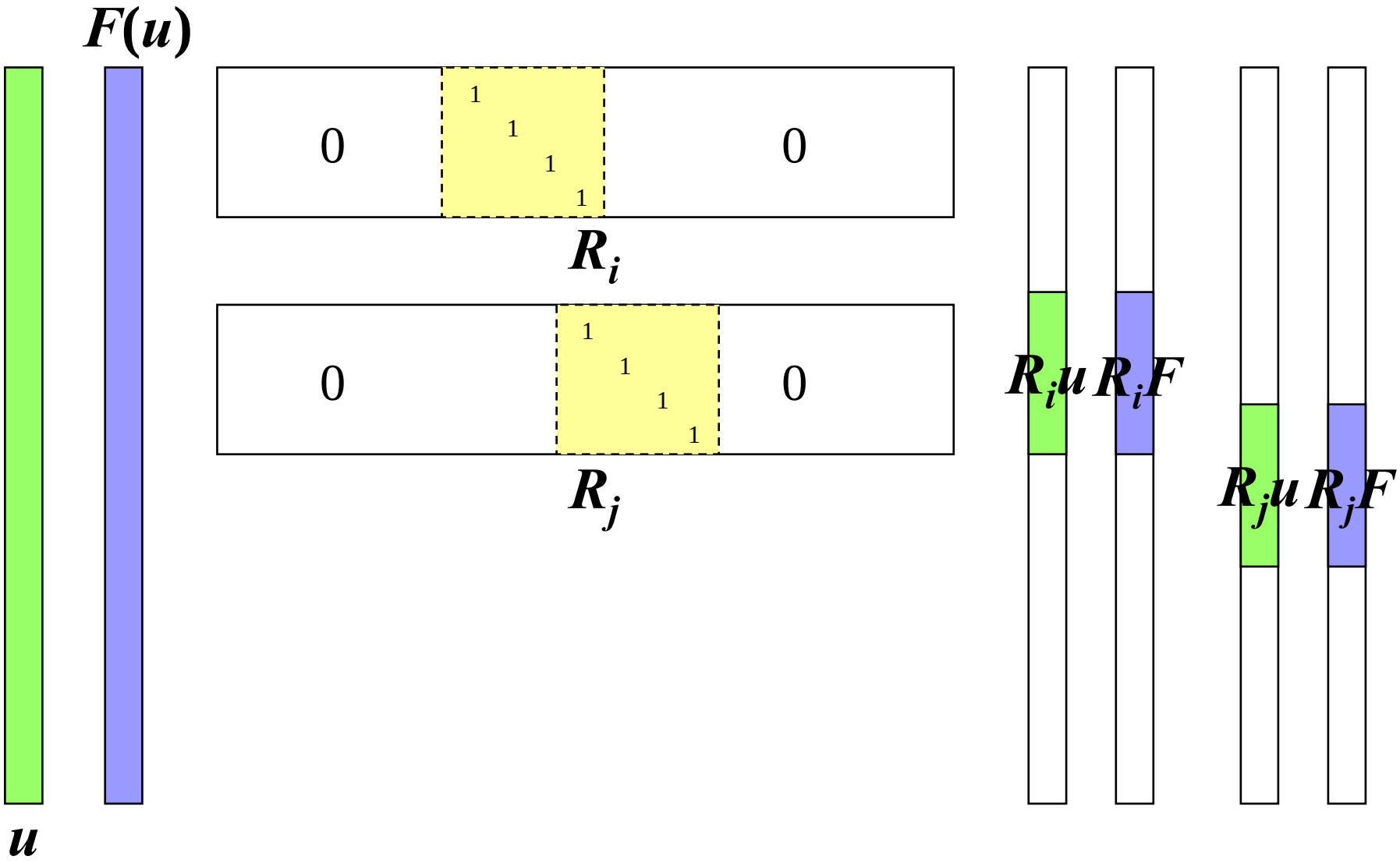
where $\delta_i(u)\in\Re^n$ is nonzero only in the components of the i^{th} partition

- **Then sum the corrections: $\Phi(u)=\sum_i \delta_i(u)$ to get an implicit function of u**

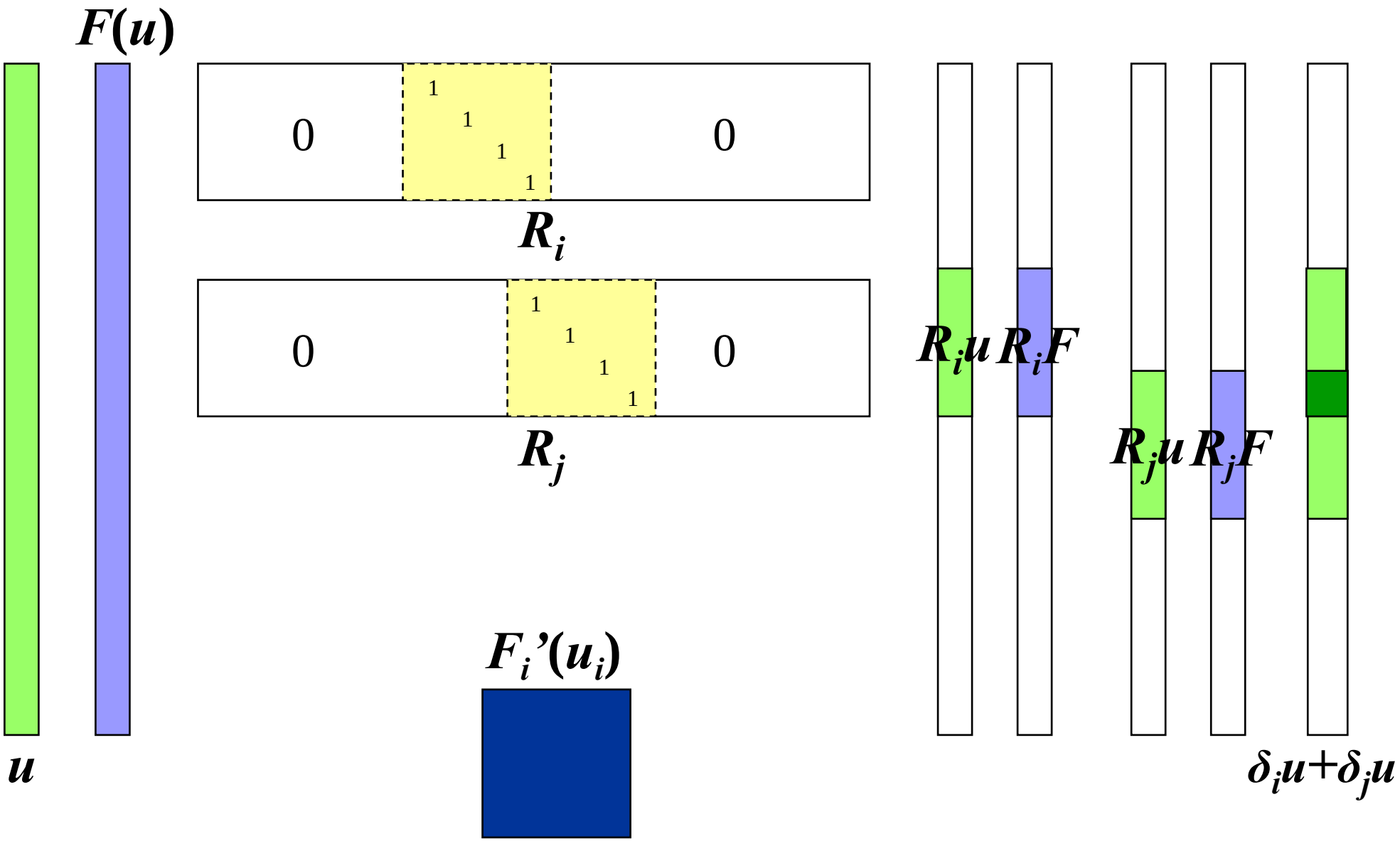
Nonlinear Schwarz – picture



Nonlinear Schwarz – picture



Nonlinear Schwarz – picture



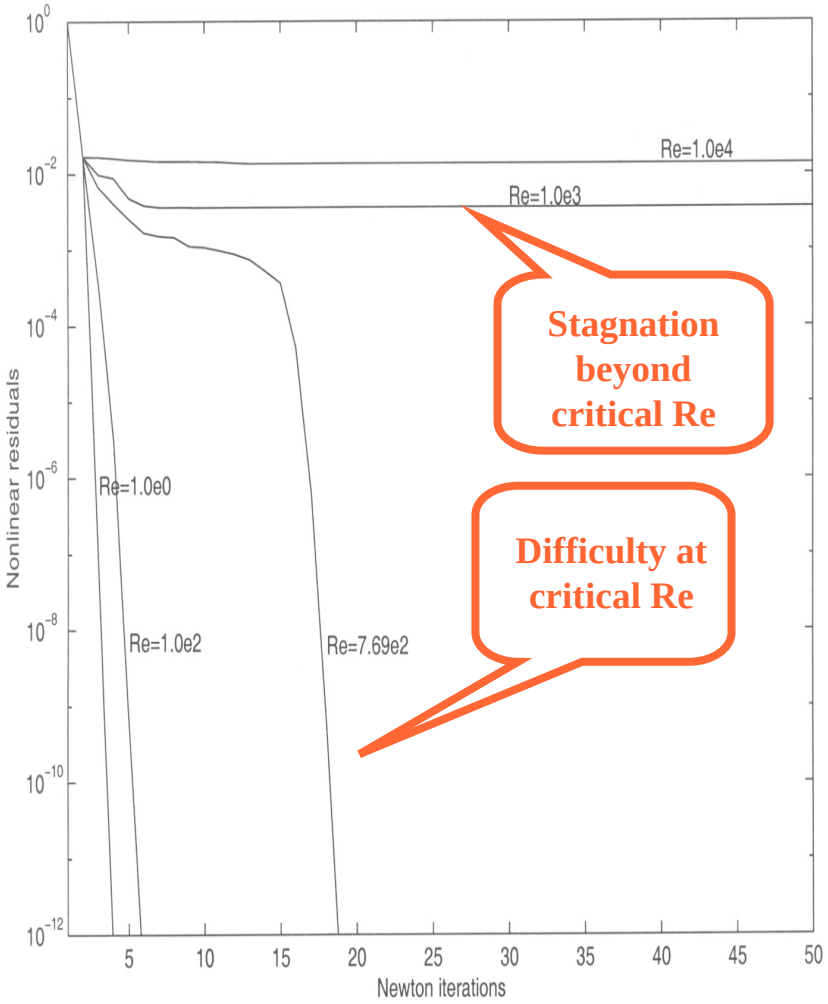
Nonlinear Schwarz, cont.

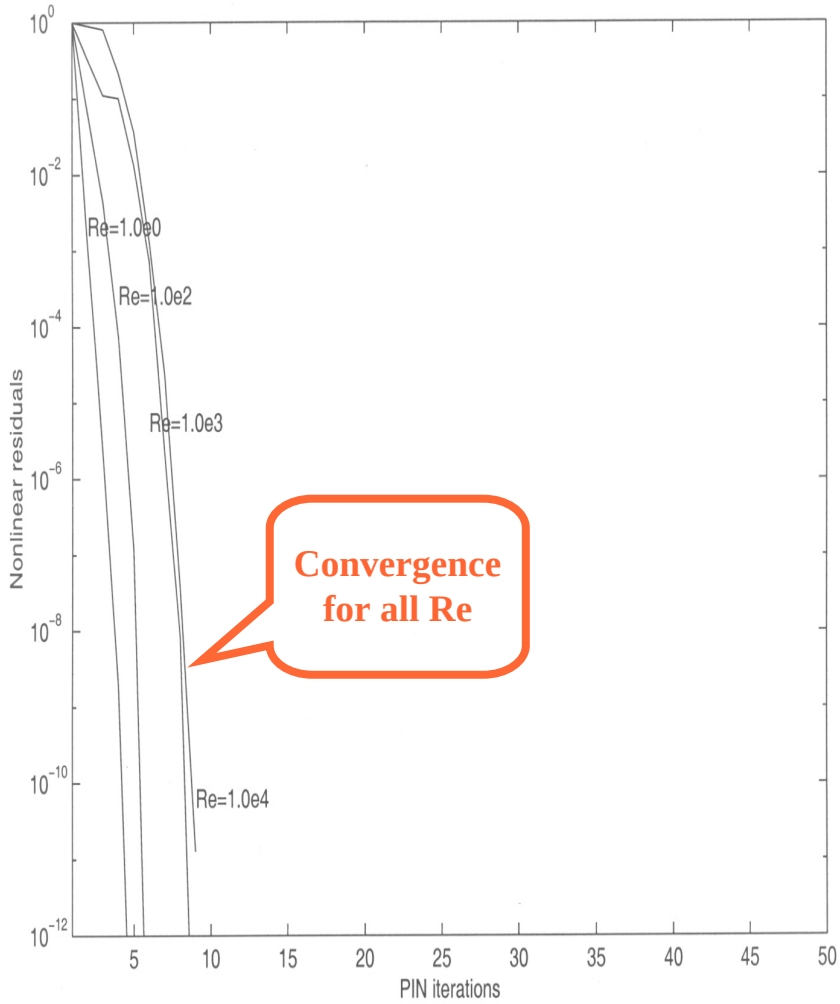
- It is simple to prove that if the Jacobian of $F(u)$ is nonsingular in a neighborhood of the desired root then $\Phi(u) = 0$ and $F(u) = 0$ have the same unique root
- To lead to a Jacobian-free Newton-Krylov algorithm we need to be able to evaluate for any $u, v \in \Re^n$:
  - The residual $\Phi(u) = \sum_i \delta_i(u)$
  - The Jacobian-vector product $\Phi(u)'v$
- Remarkably, (Cai-Keyes, 2000) it can be shown that

$$\Phi'(u)v \approx \sum_i (R_i^T J_i^{-1} R_i) Jv$$

  where $J = F'(u)$ and $J_i = R_i J R_i^T$
- All required actions are available in terms of $F(u)$!

Experimental example of nonlinear Schwarz





Vanilla Newton's method

Nonlinear Schwarz

Multiphysics coupling: partial elimination

- **Consider system** $F(u)=0$ **partitioned by physics as**

$$\begin{cases} F_1(u_1,u_2)=0 \\ F_2(u_1,u_2)=0 \end{cases}$$

- **Can formally solve for** u_1 **in** $F_1(u_1,u_2)=0$

$$u_1 \equiv G(u_2)$$

- **Then second equation is** $F_2(G(u_2),u_2)=0$
- **Jacobian**

$$\frac{dF_2}{du_2}=\frac{\partial F_2}{\partial u_1}\frac{\partial G}{\partial u_2}+\frac{\partial F_2}{\partial u_2}$$

can be applied to a vector in matrix-free manner

Multiphysics coupling: nonlinear GS

- **In previous notation, given initial iterate** $\left\{u_1^0, u_2^0\right\}$
- **For *k=1, 2, …,* until convergence, do**
  - Solve for v in $F_1(v, u_2^{k-1}) = 0$
  - Solve for w in $F_2(v, w) = 0$
- **Then**

$$\left\{u_1^k, u_2^k\right\} = \left\{v, w\right\}$$

Multiphysics coupling: nonlinear Schwarz

- **Given initial iterate** $\left\{u_1^0, u_2^0\right\}$
- **For** $k=1, 2, \ldots,$ **until convergence, do**
  - Define $G_1(u_1, u_2) \equiv \delta u_1$ by $F_1(u_1^{k-1} + \delta u_1, u_2^{k-1}) = 0$
  - Define $G_2(u_1, u_2) \equiv \delta u_2$ by $F_2(u_1^{k-1}, u_2^{k-1} + \delta u_2) = 0$
- **Then solve** $\begin{cases} G_1(u,v) = 0 \\ G_2(u,v) = 0 \end{cases}$ **in matrix-free manner**
- **Jacobian:** $\begin{bmatrix} \frac{\partial G_1}{\partial u} & \frac{\partial G_1}{\partial v} \\ \frac{\partial G_2}{\partial u} & \frac{\partial G_2}{\partial v} \end{bmatrix} \approx \begin{bmatrix} I & \left(\frac{\partial F_1}{\partial u}\right)^{-1} \frac{\partial F_1}{\partial v} \\ \left(\frac{\partial F_2}{\partial v}\right)^{-1} \frac{\partial F_2}{\partial u} & I \end{bmatrix}$
- **Finally** $\left\{u_1^k, u_2^k\right\} = \{v, w\}$

Physics-based preconditioning

- **In Newton iteration, one seeks to obtain a correction ("delta") to solution, by inverting the Jacobian matrix on (the negative of) the nonlinear residual:**

$$\delta u^{k} = -[J(u^{k})]^{-1} F(u^{k})$$

- **A typical operator-split code also derives a "delta" to the solution, by some implicitly defined means, through a series of implicit and explicit substeps**

$$F(u^{k}) \mapsto \delta u^{k}$$

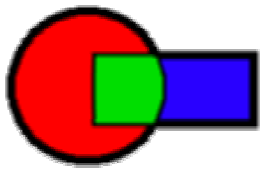
- **This implicitly defined mapping from residual to "delta" is a *natural* preconditioner**
- **Software must accommodate this!**

Physics-based preconditioning

- **We consider a standard "dynamical core," the shallow-water wave splitting algorithm, *as a solver***
- **Leaves a first-order in time splitting error**
- **In the Jacobian-free Newton-Krylov framework, this solver, which maps a residual into a correction, can be regarded *as a preconditioner***
- **The true Jacobian is never formed yet the time-implicit nonlinear residual at each time step can be made as small as needed for nonlinear consistency in long time integrations**

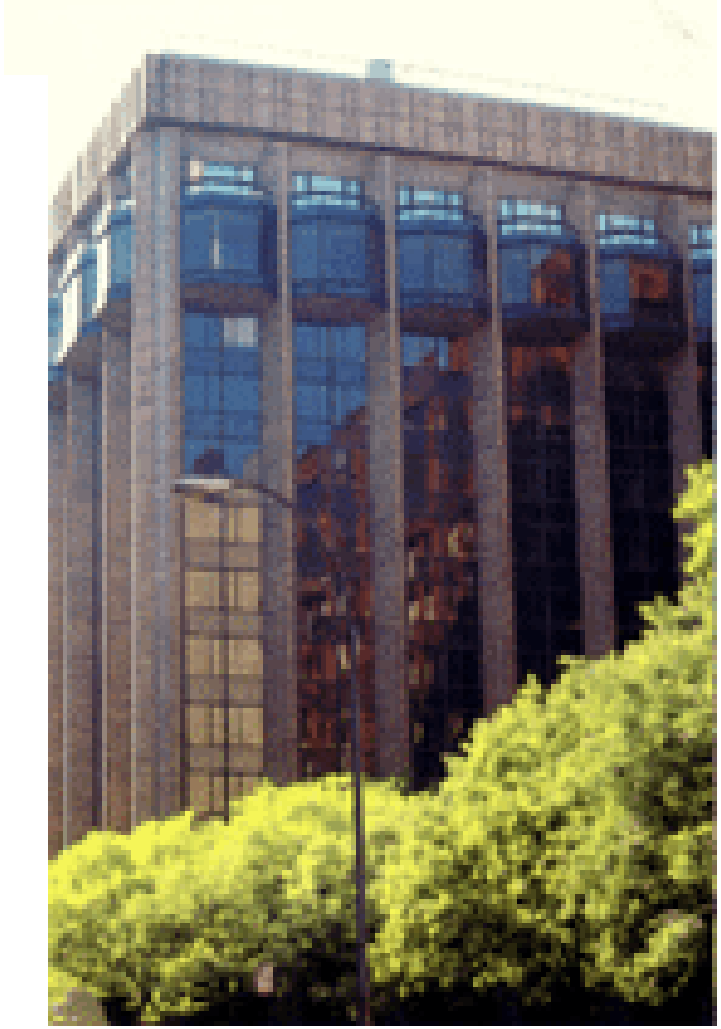
State of the art

- **Domain decomposition is the dominant paradigm in contemporary terascale PDE simulation**
- **Several freely available software toolkits exist, and successfully scale to thousands of tightly coupled processors for problems on quasi-static meshes**
- **Concerted efforts underway to make elements of these toolkits interoperate, and to allow expression of the best methods, which tend to be modular, hierarchical, recursive, and above all — *adaptive*!**
- **Many challenges loom at the "next scale" of computation**
- **Implementation of domain decomposition methods on parallel computers has inspired many useful variants of domain decomposition methods**
- **The past few years have produced an incredible variety of interesting results (in both the continuous and the discrete senses) in domain decomposition methods, with no slackening in sight**



DD-16 in New York City, January 2005

- 3.5-day meeting January 12-15, 2005
- Co-organized by NYU and Columbia
- 14 invited speakers
- 8 participant-organized minisymposia
- Contributed talks
- Poster session
- Pre-workshop short course, January 11, 2005



http://www.cims.nyu.edu/dd16

EOF