Strategies for Solving Linear Systems of Equations Using Chapel

(Early Experiences Using Chapel)

Richard Barrett and Stephen Poole

Future Technologies Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory

http://ft.ornl.gov

Cray User Group 2007, Seattle WA
May 7, 2007

Chapel Status

- Compiler version 0.4.481 released April 15.
  - running on my Mac; also Linux, SunOS, CygWin
  - Initial release December 15, 2006.
- Spec version 0.702
- Development team "optimally" responsive.

Productivity

Programmability, performance, portability, and robustness

Programmability: Motivation for "expressiveness"

"By their training, the experts in iterative methods expect to collaborate with users. Indeed, the combination of user, numerical analyst, and iterative method can be incredibly effective. Of course, by the same token, inept use can make any iterative method not only slow but prone to failure. Gaussian elimination, in contrast, is a classical black box algorithm demanding no cooperation from the user.

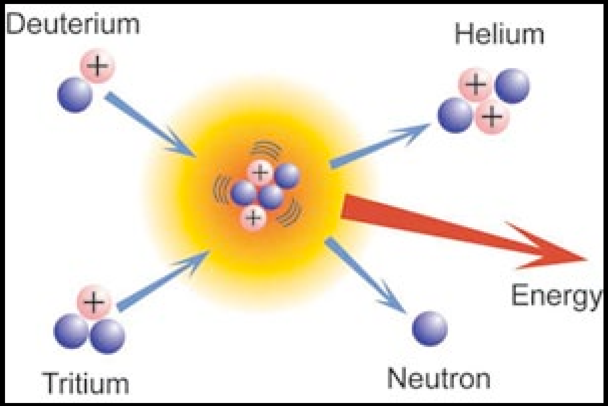
Surely the moral of the story is not that iterative methods are dead,
but that too little attention has been paid to the user's current needs?"

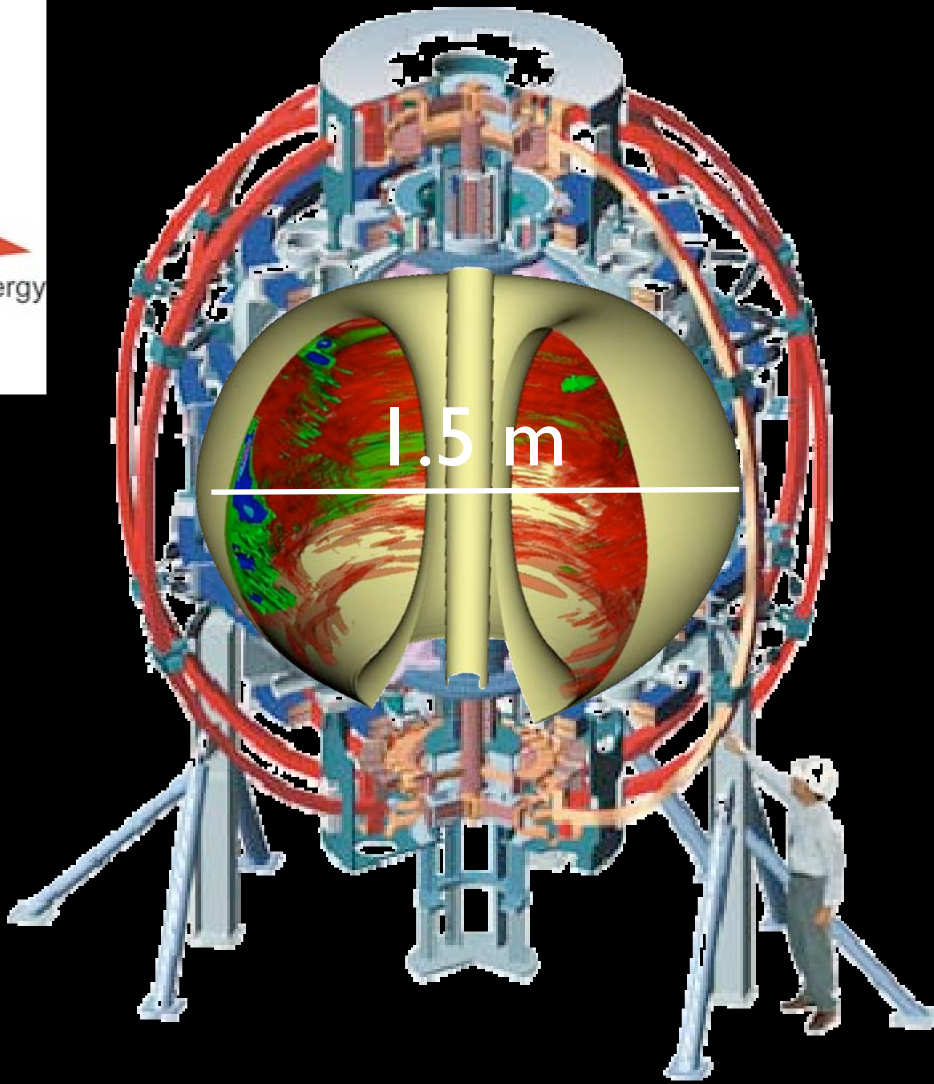
"Progress in Numerical Analysis",
Beresford N. Parlett,
SIAM Review, 1978.

Performance Expectations

- ~~If we had a compiler we could "know".~~
- "Domains" define data structures; coupled with operators.
- Distribution options (including user defined)
- Inter-process communication flexibility
- Memory Model
- Diversity of Architectures emerging.
- Strong funding model

Fusion energy

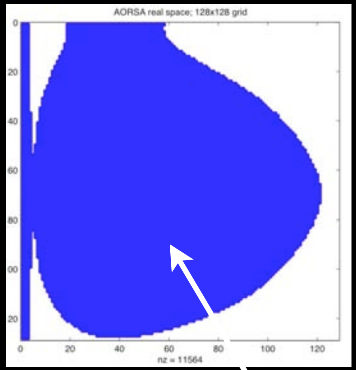




NSTX

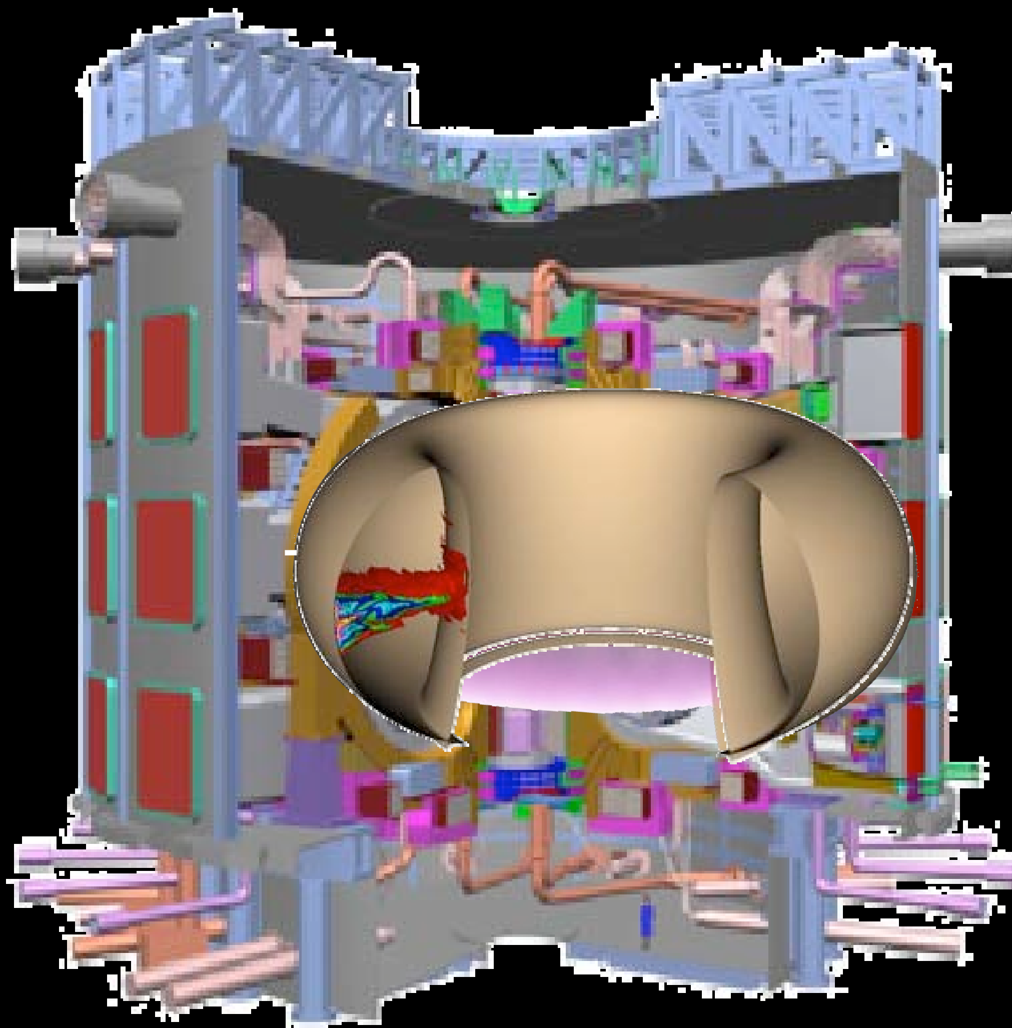
AORSA
rf-heating of plasma

Fourier space



“Real” space

Fusion energy

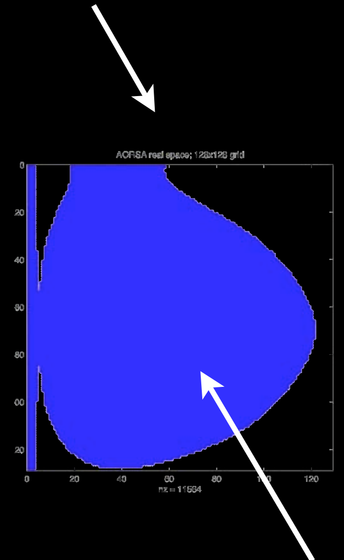


ITER

AORSA arrays in Chapel

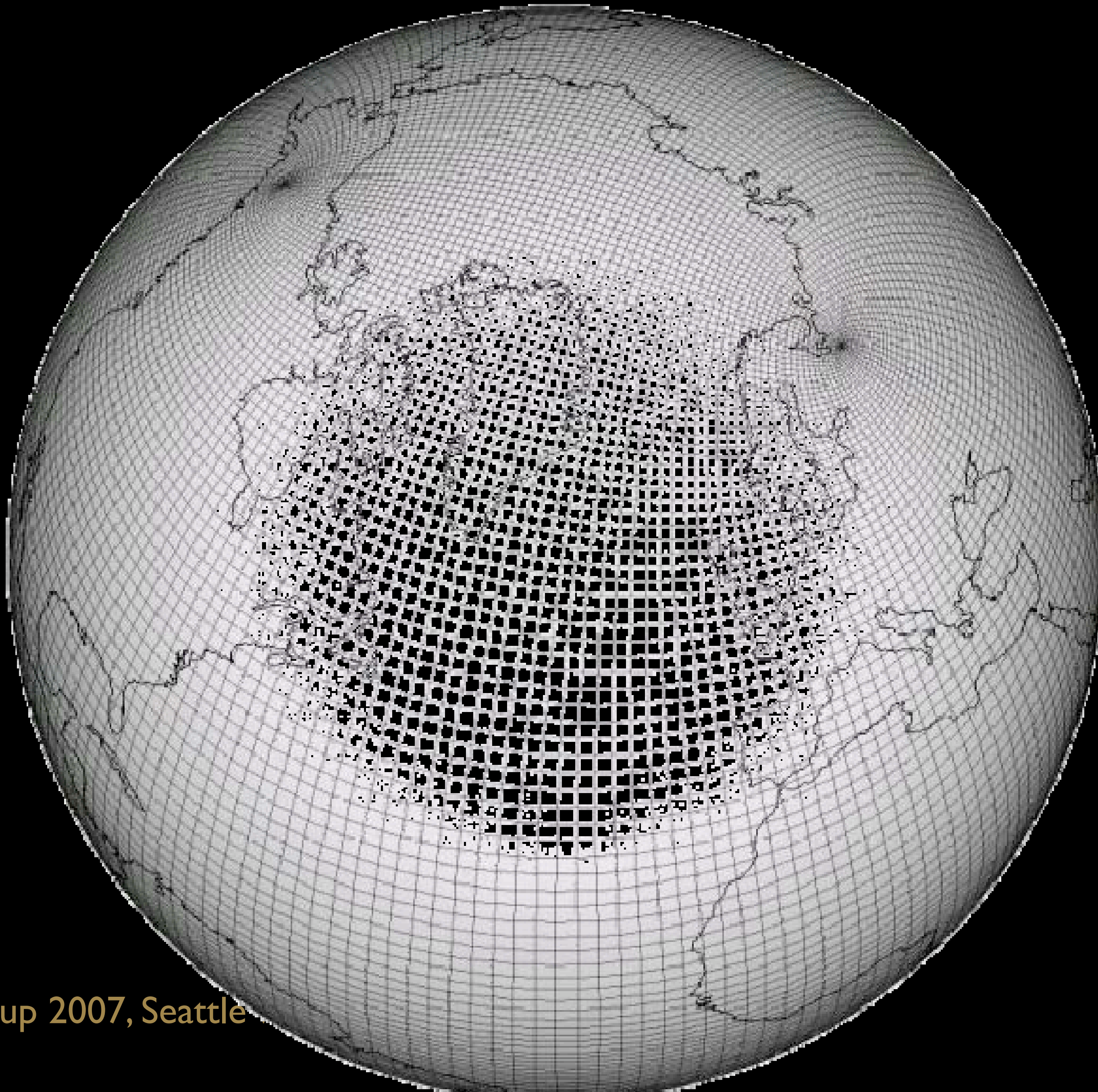
```
const
   FourierSpace : domain(2) distributed ( Block ) = [1.. nnodex, 1.. nnodey];

var
   fgrid,
   mask
          : [FourierSpace] real;

var
   PhysSpace: sparse subdomain (FourierSpace) =
      [i in FourierSpace]  if mask(i) == 1 then i;

var
   pgrid
      : [PhysSpace] real;
```

Dense linear solve, so inter-operability needed.

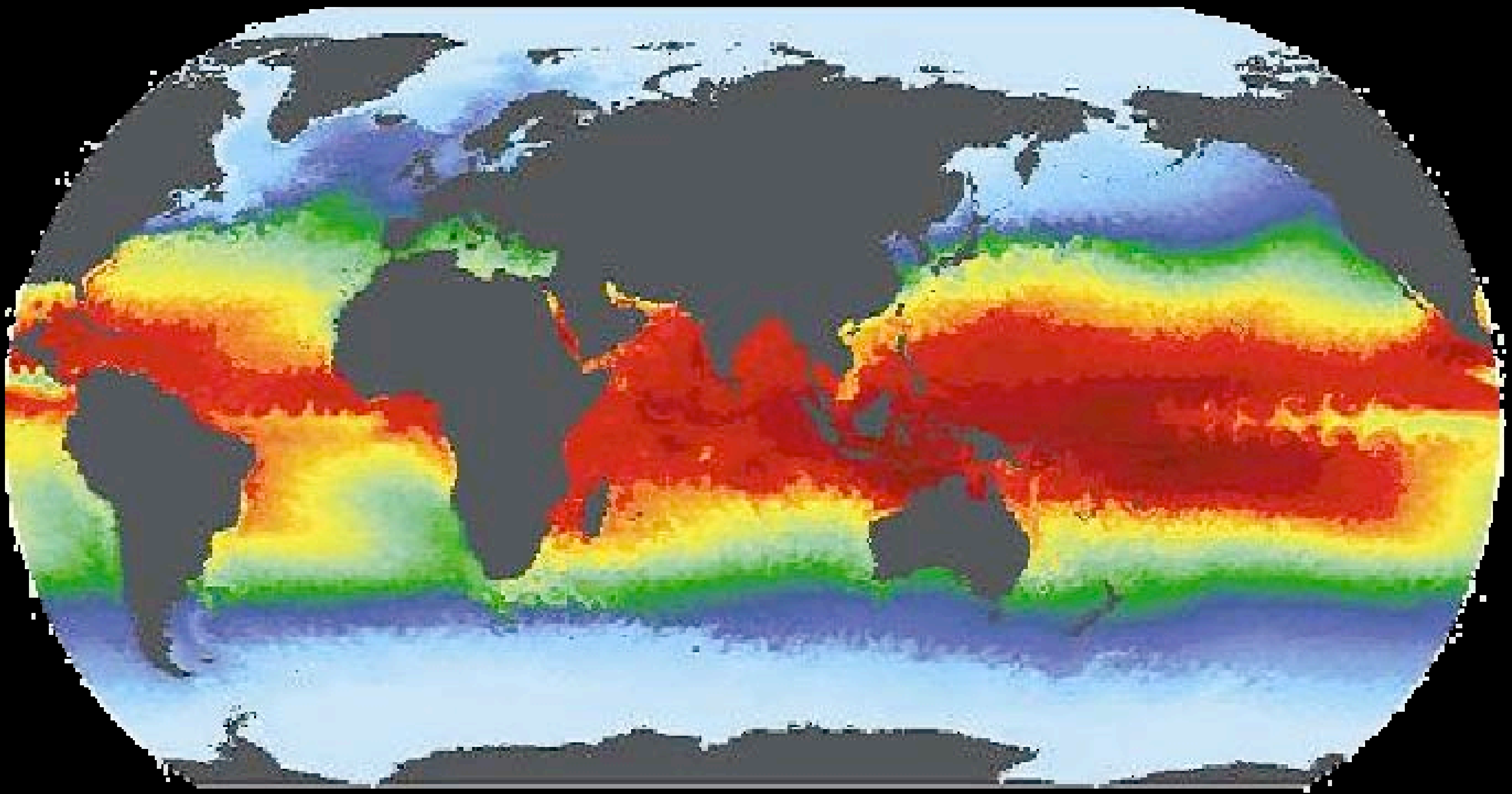


Ocean circulation

POP's tripole grid



POP Output

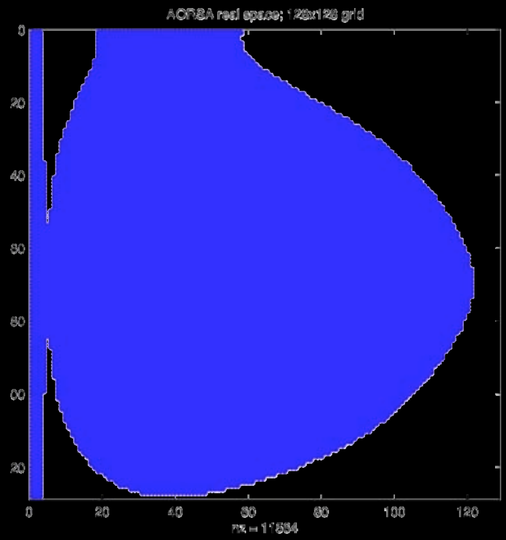


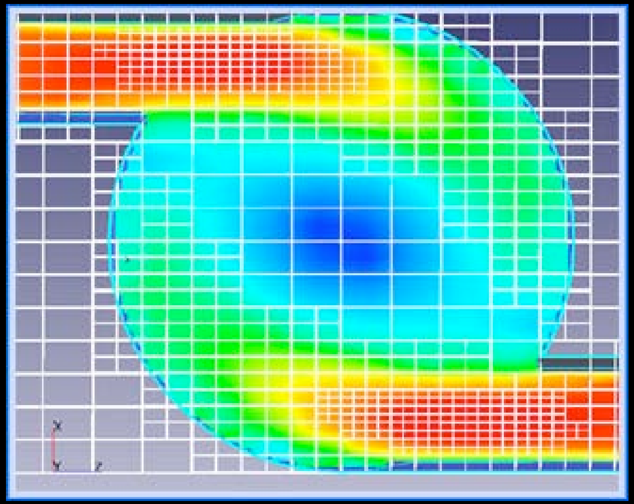
Solving Ax=b
Method of Conjugate Gradients

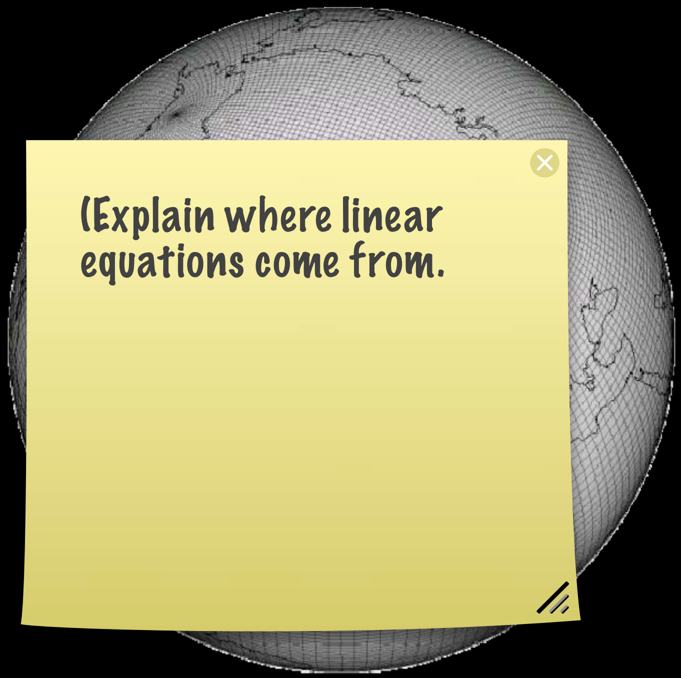
$M^{-1}Ax=M^{-1}b$

```
for i = 1, 2, ...
    solve Mz^(i-1) = r^(i-1)
    ρ_{i-1} = r^(i-1)T z^(i-1)
    if ( i = 1 )
        p = z^(0)
    else
        β = ρ_{i-1} / ρ_{i-2}
        p = z^(i-1) + βp^(i-1)
    end if
    q = Ap
    α = ρ_{i-1} / p^T q
    x = x^(i-1) + α p
    r = r^(i-1) - α q
    check convergence; continue if necessary
end
```

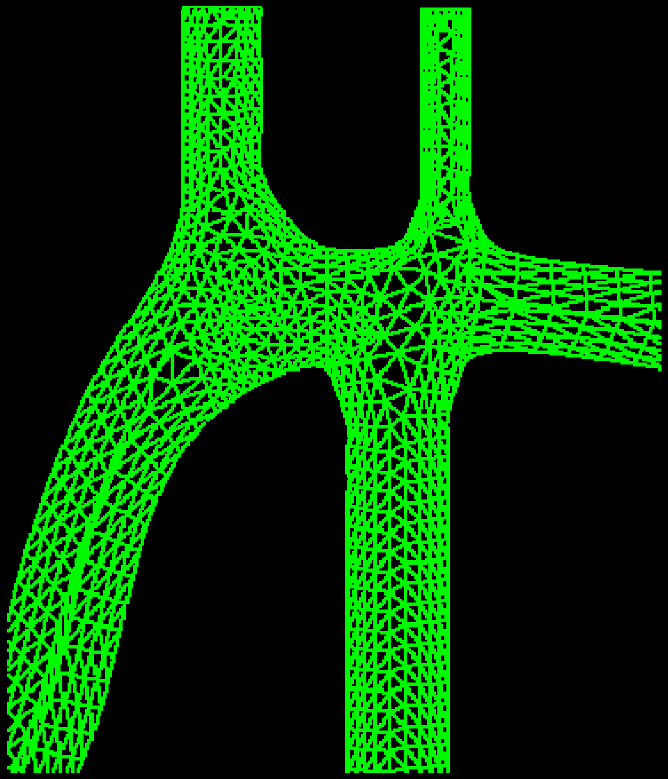
Linear eqns come from the problem space



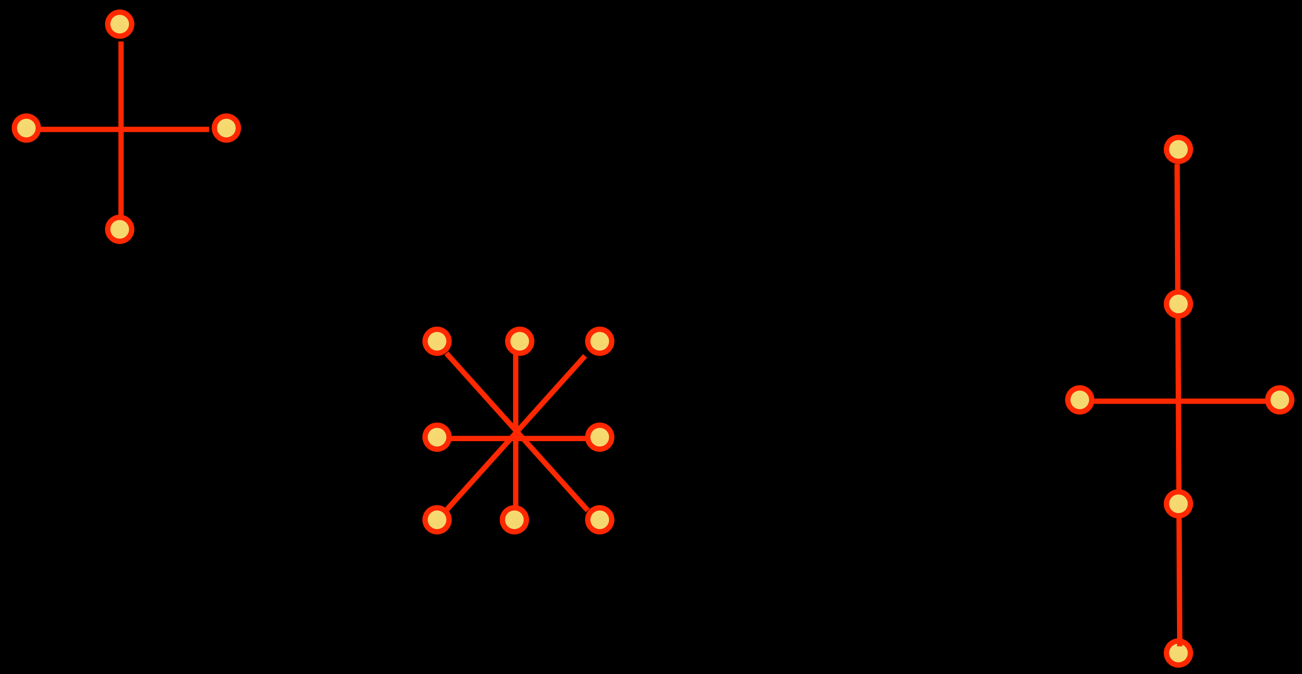




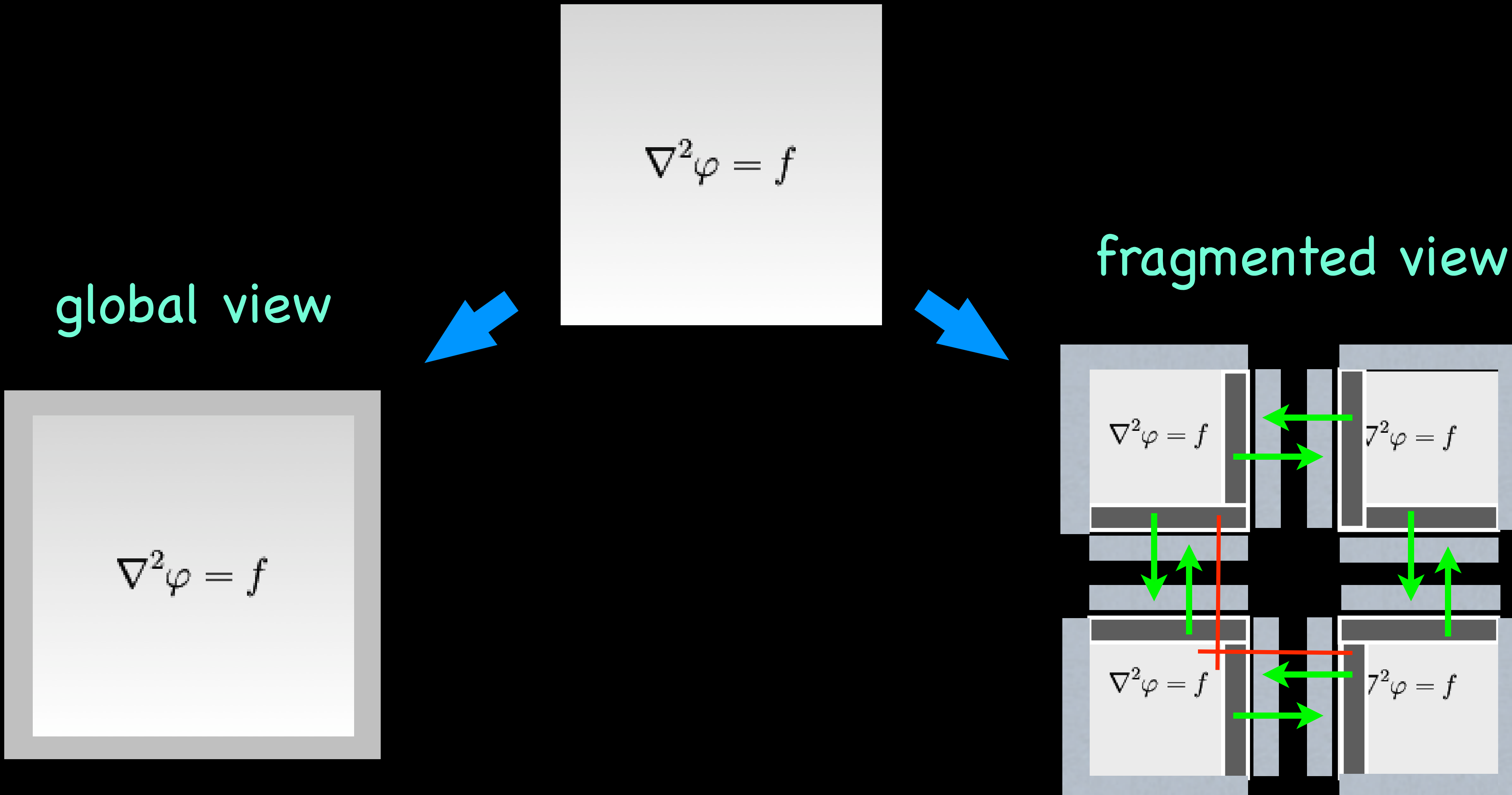
(Explain where linear equations come from.



Linear equations may often be defined as ``stencils” (Matvec, preconditioner)



Finite difference solution of Poisson's Eqn



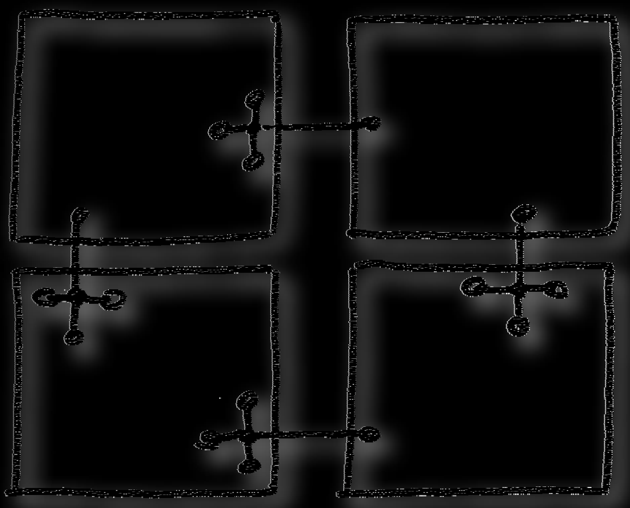
Fortran-MPI

```
CALL BOUNDARY_EXCHANGE ( ... )

DO J = 2, LCOLS+1
   DO I = 2, LROWS+1

      Y(I,J) =

                X(I-1,J-1)  + X(I-1,J) + X(I-1,J+1) +

                 X(I,J-1)   +  X(I,J)   +   X(I,J+1)   +

                X(I+1,J-1) + X(I+1,J) + X(I+1,J+1)

   END DO
END DO
```

Fortran-MPI

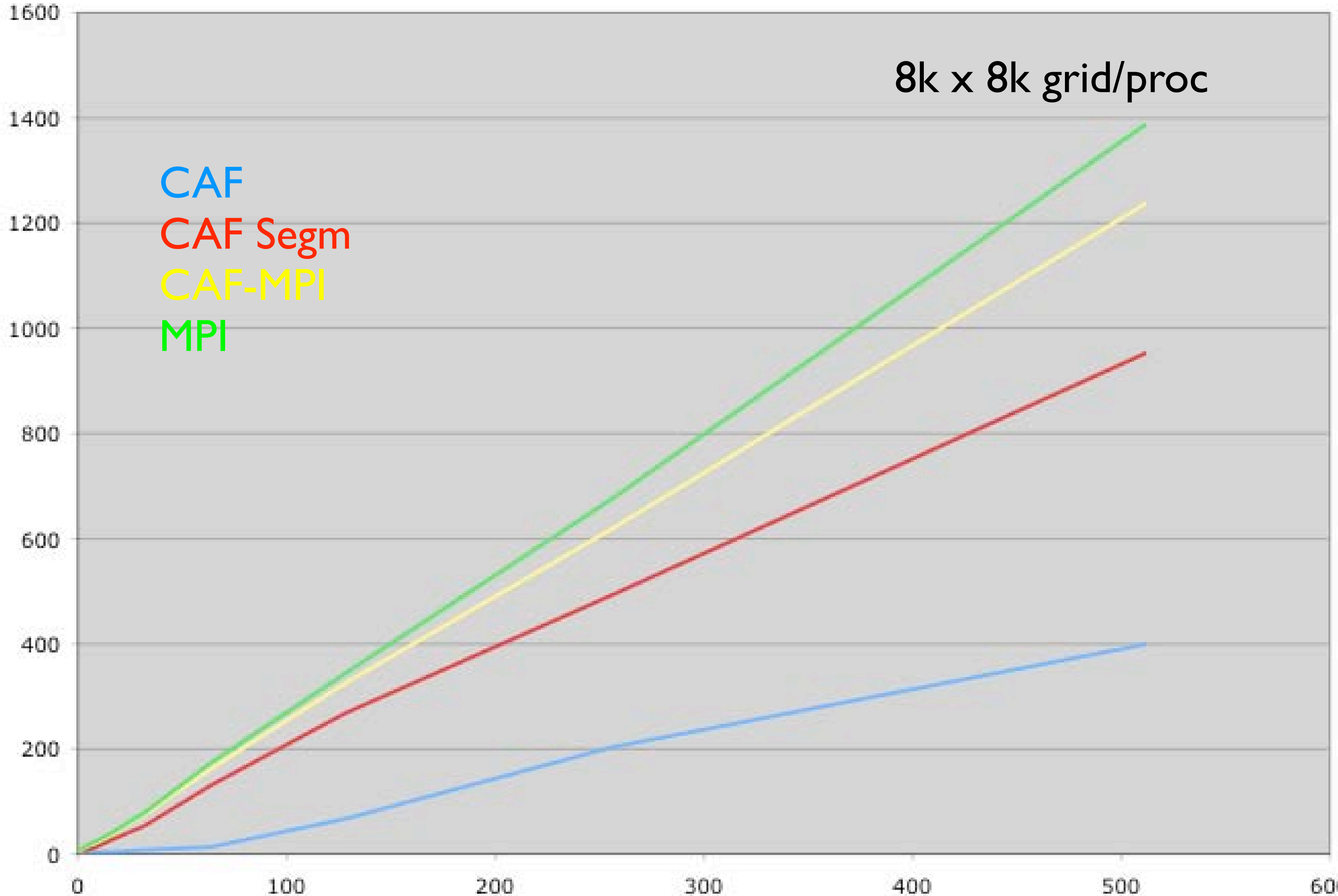
```
CALL BOUNDARY_EXCHANGE ( ... )

DO J = 2, LCOLS+1
   DO I = 2, LROWS+1

      Y(I,J) =

         A(I-1,J-1) *X(I-1,J-1)  + A(I-1,J) *X(I-1,J) + A(I-1,J+1) X(I-1,J+1) +

         A (I,J-1)*X(I,J-1)      +  A(I,J)*X(I,J)      +   A (I,J+1) *X(I,J+1)  +

         A(I+1,J-1) X(I+1,J-1) + A(I+1,J)*X(I+1,J) + A(I+1,J+1)*X(I+1,J+1)


   END DO
END DO
```

Co-Array Fortran:
Load-it-when-you-need-it



```
DO J = 2, LCOLS+1
   DO I = 2, LROWS+1

      NW = X(II(I-1,J-1),JJ(I-1,J-1))[IMG_LOC(I-1,J-1)]
      W  = X(II(I  ,J-1),JJ(I  ,J-1))[IMG_LOC(I  ,J-1)]
      SE = X(II(I+1,J-1),JJ(I+1,J-1))[IMG_LOC(I+1,J-1)]

      N  = X(II(I-1,J  ),JJ(I-1,J  ))[IMG_LOC(I-1,J)]
      C  = X(I,J)
      S  = X(II(I+1,J  ),JJ(I+1,J  ))[IMG_LOC(I+1,J)]

      NE = X(II(I-1,J+1),JJ(I-1,J+1))[IMG_LOC(I-1,J+1)]
      E  = X(II(I  ,J+1),JJ(I,  J+1))[IMG_LOC(I  ,J+1)]
      SW = X(II(I+1,J+1),JJ(I+1,J+1))[IMG_LOC(I+1,J+1)]

      Y(I,J) = ( NW + N  + NE  +   &
                 W  + C  + E   +   &
                 SE  + S + SW )

   END DO
END DO
```

9-pt stencil; weak scaling



CRay X1E, 2006

Chapel:
"Direct translation"

```
const
  PhysicalSpace: domain(2) distributed(Block) = [1..m, 1..n],
  AllSpace = PhysicalSpace.expand(1);

var
  X, Y : [AllSpace] : real;        // Arrays

const
  NW = ( -1, -1 ), N = ( -1, 0 ), NE = ( -1, 1 ),
   W = ( 0, -1),                   E = ( 0, 1 ),
  SW = ( 1, -1 ), S = ( 1, 0 ),   SE = ( 1, 1 );

forall i in PhysicalSpace do

    Y(I) =

          X(I+NW) + X(I+N) + X(I+NE) +

          X(I+W)   +   X(I)   + X(I+E) +

          X(I+SW)  + X(I+S) + X(I+SE);
```

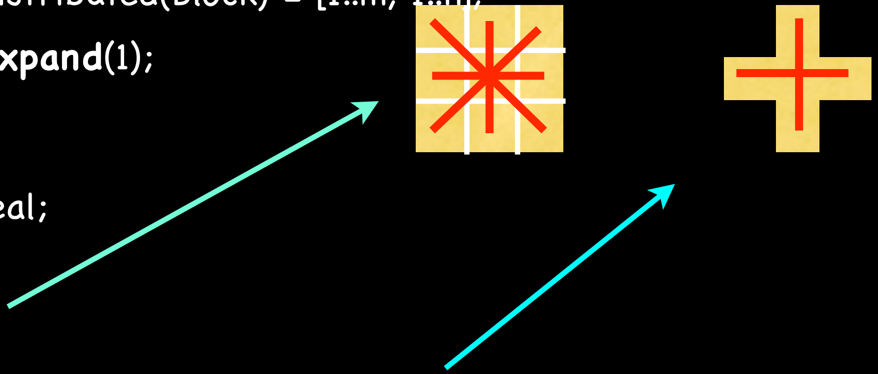
Chapel: Reduction implementation

```
const
   PhysicalSpace: domain(2) distributed(Block) = [1..m, 1..n],
   AllSpace = PhysicalSpace.expand(1);

var
   Coeff, X, Y : [AllSpace] : real;

var
   Stencil = [ -1..1, -1..1 ];

forall i in PhysicalSpace do

     Y(i) = ( + reduce  [k in Stencil]  Coeff (i+k) * X (i+k) );
```

```
forall i in Stencil do
```

Matrix as a "sparse domain" of 5 pt stencils

```
const
   PhysicalSpace: domain(2) distributed(Block) = [1..m, 1..n],
   AllSpace = PhysicalSpace.expand(1);

var
   Coeff, X, Y : [AllSpace] : real;

var
   Stencil9pt = [ -1..1, -1..1 ],
   Stencil = sparse subdomain (Stencil9pt) = [(i,j) in Stencil9pt]
           if ( abs(i) + abs(j) < 2 ) then (i,j);
forall i in PhysicalSpace do

     Y(i) = ( + reduce [k in Stencil] Coeff (i+k) * X (i+k) );
```



27 point stencil in 3d: Reduction implementation

```
const
   PhysicalSpace: domain(2) distributed(Block) = [1..m, 1..n, 1..p],
   AllSpace = PhysicalSpace.expand(1);

var
   Coeff, X, Y : [AllSpace] : real;

var
   Stencil = [ -1..1, -1..1, -1..1 ];

forall i in PhysicalSpace do

     Y(i) = ( + reduce [k in Stencil] Coeff (i+k) * X (i+k) );
```

5-point stencil in 2d:

```
forall i in PhysicalSpace do

    Y(i) = ( + reduce [k in Stencil] Coeff (i+k) * X (i+k) );
```

9-point stencil in 2d:

```
forall i in PhysicalSpace do

    Y(i) = ( + reduce [k in Stencil] Coeff (i+k) * X (i+k) );
```

27-point stencil in 3d:

```
forall i in PhysicalSpace do
    Y(i) = ( + reduce [k in Stencil] Coeff (i+k) * X (i+k) );
```

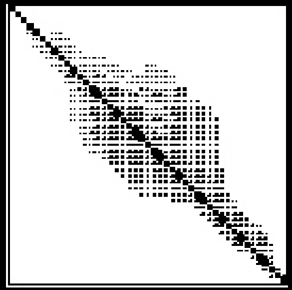
7-point stencil in 3d:

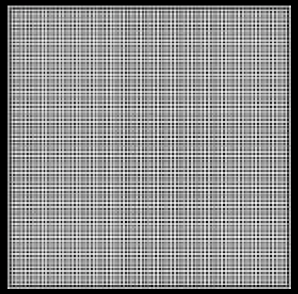
```
forall i in PhysicalSpace do
    Y(i) = ( + reduce [k in Stencil] Coeff (i+k) * X (i+k) );
```

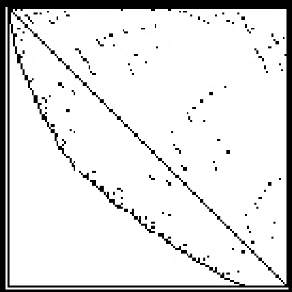
n-point stencil in m-d:

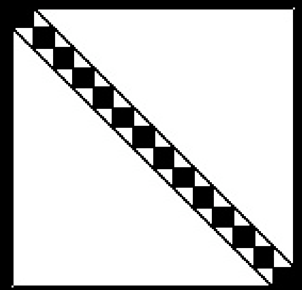
```
forall i in PhysicalSpace do

    Y(i) = ( + reduce [k in Stencil] Coeff (i+k) * X (i+k) );
```

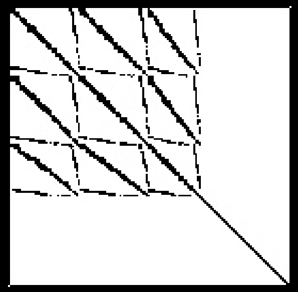
Matrix structures







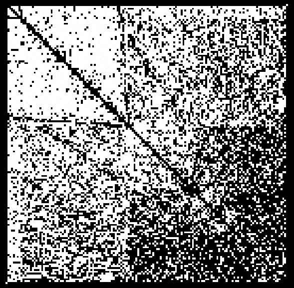


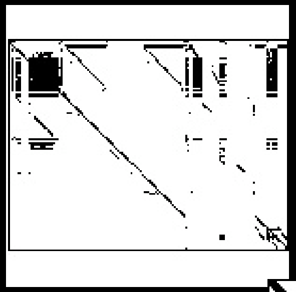


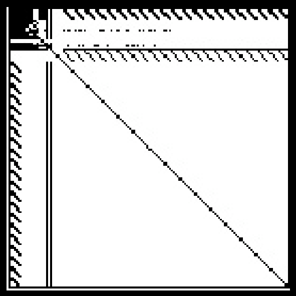
Dense (banded, symmetric)

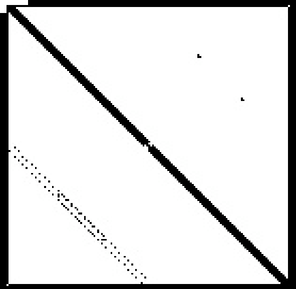
Sparse:

- Special structure:
  - Symmetric
  - Toeplitz
  - Circulant
  - Blocks
  - No (exploitable) structure









Current and Future work

- "Finite Difference Methods Using Chapel", Barrett, Roth, Poole, & Vetter, Tech Report, 2007.
- "Using MPI in a Chapel World", Barrett, Graham, Poole, & Roth; In progress.
- More, better submitted...
- In progress: Sweep3d, sPPM, graph-based algorithms.
- Fortress and X10 work.
- LDRD: "Compilation and Runtime System Support for Global View Languages", Barrett, Graham, Poole, & Roth.

Acknowledgements

- This research was sponsored by the Office of Mathematical, Information, and Computational Sciences, Office of Science, U.S. Department of Energy under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.
- Chapel development team.

OAK RIDGE
National Laboratory