AMath 483/583 — Lecture 22 — May 16, 2011

Today:

- Gauss-Seidel and SOR iterative methods
- Totalview debugging

Next:

- GPUs, Python

Read: Class notes and references

Jacobi iteration

$$(U_{i-1} - 2U_i + U_{i+1}) = -\Delta x^2 f(x_i)$$

Solve for U_i:

$$U_i = \frac{1}{2}\left(U_{i-1} + U_{i+1} + \Delta x^2 f(x_i)\right).$$

Note: With no heat source, $f(x) = 0$,
the temperature at each point is average of neighbors.

Jacobi iteration

$$(U_{i-1} - 2U_i + U_{i+1}) = -\Delta x^2 f(x_i)$$

Solve for U_i:

$$U_i = \frac{1}{2}\left(U_{i-1} + U_{i+1} + \Delta x^2 f(x_i)\right).$$

Note: With no heat source, $f(x) = 0$,
the temperature at each point is average of neighbors.

Suppose $U^{[k]}$ is a approximation to solution. Set

$$U_i^{[k+1]} = \frac{1}{2}\left(U_{i-1}^{[k]} + U_{i+1}^{[k]} + \Delta x^2 f(x_i)\right) \quad \text{for } i = 1,\ 2,\ \ldots,\ n.$$

Repeat for $k = 0,\ 1,\ 2,\ \ldots$ until convergence.

Can be shown to converge (eventually... very slow!)

Jacobi iteration in Fortran

```
uold = u  ! starting values before updating

do iter=1,maxiter

    dumax = 0.d0

    do i=1,n
      u(i) = 0.5d0*(uold(i-1) + uold(i+1) + dx**2*f(i))
      dumax = max(dumax, abs(u(i)-uold(i)))
      enddo

    ! check for convergence:
    if (dumax .lt. tol) exit

    uold = u  ! for next iteration
    enddo
```

Note: we must use old value at $i-1$ for Jacobi.

Otherwise we get the Gauss-Seidel method.

Gauss-Seidel iteration in Fortran

```
do iter=1,maxiter

    dumax = 0.d0

    do i=1,n
      uold = u(i)
      u(i) = 0.5d0*(u(i-1) + u(i+1) + dx**2*f(i))
      dumax = max(dumax, abs(u(i)-uold))
      enddo

    ! check for convergence:
    if (dumax .lt. tol) exit

    enddo
```

Note: Now `u(i)` depends on value of `u(i-1)` that has already been updated for previous `i`.

Gauss-Seidel iteration in Fortran

```
do iter=1,maxiter

    dumax = 0.d0

    do i=1,n
      uold = u(i)
      u(i) = 0.5d0*(u(i-1) + u(i+1) + dx**2*f(i))
      dumax = max(dumax, abs(u(i)-uold))
      enddo

    ! check for convergence:
    if (dumax .lt. tol) exit

    enddo
```

Note: Now `u(i)` depends on value of `u(i-1)` that has already been updated for previous `i`.

Good news: This converges about twice as fast as Jacobi!

Gauss-Seidel iteration in Fortran

```
do iter=1,maxiter

    dumax = 0.d0

    do i=1,n
      uold = u(i)
      u(i) = 0.5d0*(u(i-1) + u(i+1) + dx**2*f(i))
      dumax = max(dumax, abs(u(i)-uold))
      enddo

    ! check for convergence:
    if (dumax .lt. tol) exit

    enddo
```

Note: Now `u(i)` depends on value of `u(i-1)` that has already been updated for previous `i`.

Good news: This converges about twice as fast as Jacobi!

But... loop carried dependence! Cannot parallelize so easily.

Red-black ordering

We are free to write equations of linear system in any order...

reordering rows of coefficient matrix, right hand side.

Can also number unknowns of linear system in any order...

reordering elements of solution vector.

Red-black ordering

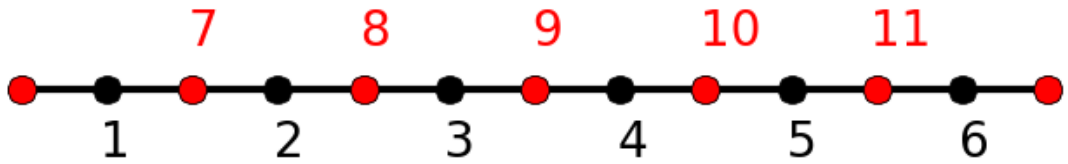
We are free to write equations of linear system in any order...
reordering rows of coefficient matrix, right hand side.

Can also number unknowns of linear system in any order...
reordering elements of solution vector.

Red-black ordering: Iterate through points with odd index first ($i = 1,\ 3,\ 5,\ \ldots$) and then even index points ($i = 2,\ 4,\ 6,\ \ldots$).

Then all black points can be updated in any order,
all red points can then be updated in any order.

Same asymptotic convergence rate as natural ordering.



Red-Black Gauss-Seidel

```
do iter=1,maxiter
    dumax = 0.d0

    ! UPDATE ODD INDEX POINTS:
    !$omp parallel do reduction(max : dumax) &
    !$omp private(uold)
    do i=1,n,2
        uold = u(i)
        u(i) = 0.5d0*(u(i-1) + u(i+1) + dx**2*f(i))
        dumax = max(dumax, abs(u(i)-uold))
        enddo

    ! UPDATE EVEN INDEX POINTS:
    !$omp parallel do reduction(max : dumax) &
    !$omp private(uold)
    do i=2,n,2
        uold = u(i)
        u(i) = 0.5d0*(u(i-1) + u(i+1) + dx**2*f(i))
        dumax = max(dumax, abs(u(i)-uold))
        enddo

    ! check for convergence:
    if (dumax .lt. tol) exit
    enddo
```

Gauss-Seidel method in 2D

If $\Delta x = \Delta y = h$:

$$\frac{1}{h^2}\left(U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{i,j}\right) = -f(x_i, y_j).$$

Solve for $U_{i,j}$ and iterate:

$$u_{i,j}^{[k+1]} = \frac{1}{4}(u_{i-1,j}^{[k+1]} + u_{i+1,j}^{[k]} + u_{i,j-1}^{[k+1]} + u_{i,j+1}^{[k]} - h^2 f_{i,j})$$

Again no need for matrix A.

Gauss-Seidel method in 2D

If $\Delta x = \Delta y = h$:

$$\frac{1}{h^2}\left(U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{i,j}\right) = -f(x_i, y_j).$$

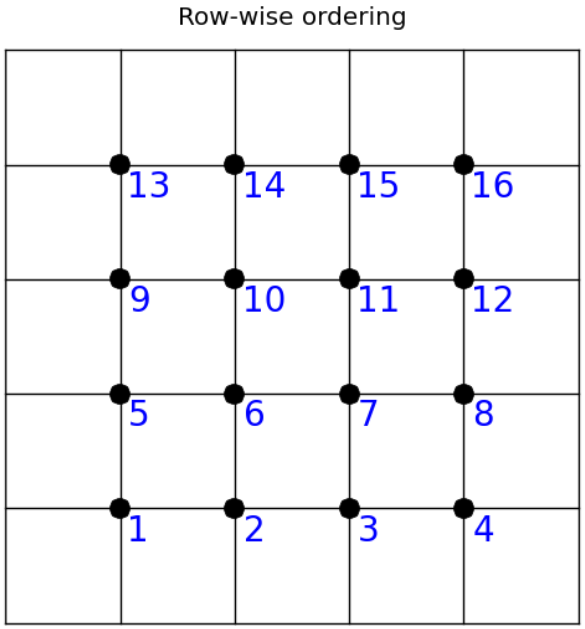
Solve for $U_{i,j}$ and iterate:

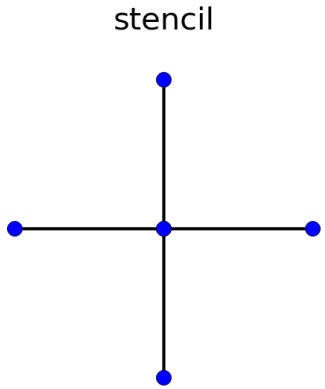
$$u_{i,j}^{[k+1]} = \frac{1}{4}(u_{i-1,j}^{[k+1]} + u_{i+1,j}^{[k]} + u_{i,j-1}^{[k+1]} + u_{i,j+1}^{[k]} - h^2 f_{i,j})$$

Again no need for matrix A.

Note: Above indices for old and new values assumes we iterate in the natural row-wise order.

Gauss-Seidel in 2D



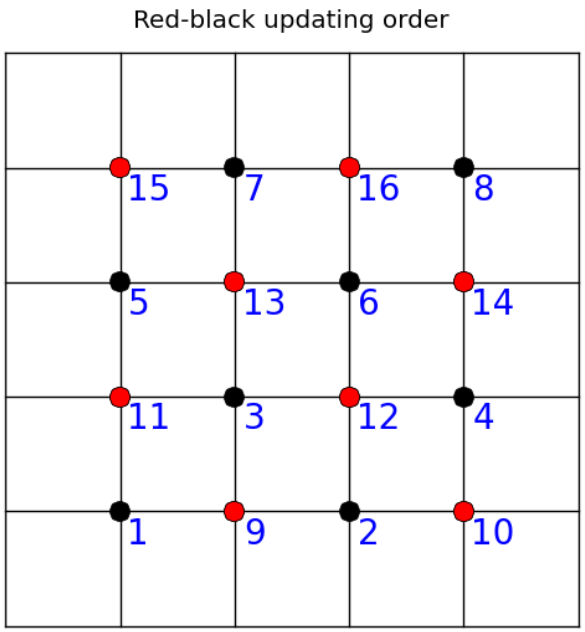


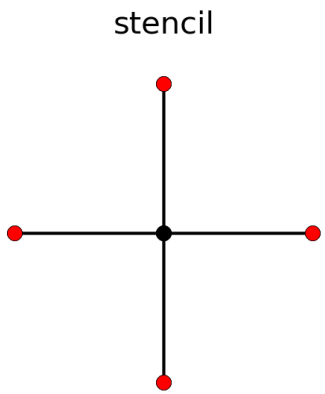
Updating point 7 for example (u_{32}):

Depends on new values at points 6 and 3, old values at points 7 and 10.

$$U_{32}^{[k+1]} = \frac{1}{4}(U_{22}^{[k+1]} + U_{42}^{[k]} + U_{21}^{[k+1]} + U_{41}^{[k]} + h^2 f_{32})$$

Red-black ordering in 2D





Again all black points can be updated in any order:

New value depends only on red neighbors.

Then all red points can be updated in any order:

New value depends only on black neighbors.

SOR method

Gauss-Seidel move solution in right direction but not far enough in general.

Iterates “relax” towards solution.

SOR method

Gauss-Seidel move solution in right direction but not far enough in general.

Iterates "relax" towards solution.

Successive Over-Relaxation (SOR):

Compute Gauss-Seidel approximation and then go further:

$$U_i^{GS} = \frac{1}{2}(U_{i-1}^{[k+1]} + U_{i+1}^{[k]} + \Delta x^2 f(x_i))$$
$$U_i^{[k+1]} = U_i^{[k]} + \omega(U_i^{GS} - U_i^{[k]})$$

where $1 < \omega < 2$.

SOR method

Gauss-Seidel move solution in right direction but not far enough in general.

Iterates "relax" towards solution.

Successive Over-Relaxation (SOR):

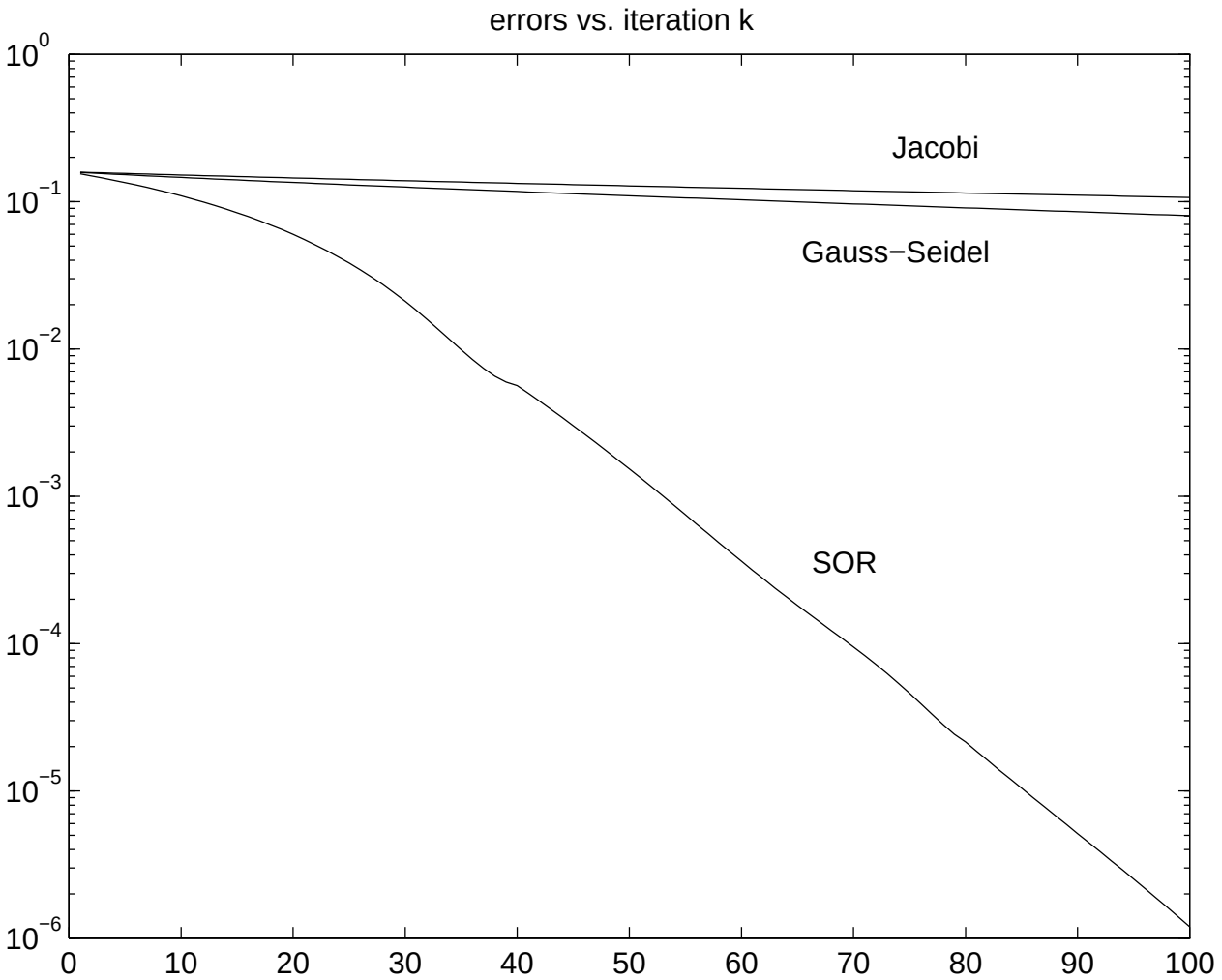
Compute Gauss-Seidel approximation and then go further:

$$U_i^{GS} = \frac{1}{2}(U_{i-1}^{[k+1]} + U_{i+1}^{[k]} + \Delta x^2 f(x_i))$$
$$U_i^{[k+1]} = U_i^{[k]} + \omega(U_i^{GS} - U_i^{[k]})$$

where $1 < \omega < 2$.

Optimal omega (For this problem): $\omega = 2 - 2\pi\Delta x$.

Convergence rates



Red-Black SOR in 1D

```
do iter=1,maxiter
    dumax = 0.d0

    ! UPDATE ODD INDEX POINTS:
    !$omp parallel do reduction(max : dumax) &
    !$omp private(uold, ugs)
    do i=1,n,2
        uold = u(i)
        ugs = 0.5d0*(u(i-1) + u(i+1) + dx**2*f(i))
        u(i) = uold + omega*(ugs-uold)
        dumax = max(dumax, abs(u(i)-uold))
        enddo

    ! UPDATE EVEN INDEX POINTS:
    !$omp parallel do reduction(max : dumax) &
    !$omp private(uold, ugs)
    do i=2,n,2
        uold = u(i)
        ugs = 0.5d0*(u(i-1) + u(i+1) + dx**2*f(i))
        u(i) = uold + omega*(ugs-uold)
        dumax = max(dumax, abs(u(i)-uold))
        enddo

    ! check for convergence...
```

Note that `uold, ugs` must be private!

Totalview debugger

`http://www.roguewave.com/products/totalview-family`

Student versions available — see email to class.