

UNIVERSITY OF CALIFORNIA AT SAN DIEGO
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

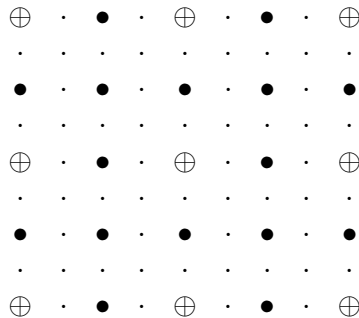
CSE 262 (Winter 2015)
System Support for Applications of Parallel Computation

Multigrid

Multigrid, or MG for short, is a fast algorithm for solving Poisson's Equation. The running time of MG, in terms of the number of iterations, is independent of N . In other words, instead of taking more iterations for larger N , multigrid takes the same number of iterations. The price we pay will be that one iteration of MG takes $\log_2 N$ steps. In addition, we will not only need to communicate among nearest neighbors, but with neighbors 2 away, 4 away, 8 away, ... , $N/2$ away. This more distant communication is the reason for the quicker convergence, because information is "spread around" the mesh more quickly than is possible with nearest neighbor communication. This discussion follows William Briggs, *A Multigrid Tutorial*, SIAM 1987. Also see the web site, which includes presentation slides:

<https://computation.llnl.gov/casc/people/henson/mgtut/slides.html>

We assume for convenience that $N = 2^n - 1$. We will need a sequence of meshes, each with about one fourth as many points as its predecessor. The finest mesh will be denoted u_0 , with values $u_{0,i,j}$, and corresponds to the same mesh as for Jacobi and SOR. The next finer mesh will consist of every other point (in both directions) in $u_{0,i,j}$ and be denoted u_1 , with entries $u_{1,i,j}$, $0 \leq i, j \leq 2^{n-1}$. u_2 will have entries $u_{2,i,j}$, $0 \leq i, j \leq 2^{n-2}$. This continues for u_k , k up to $n - 1$. $u_{k,2i,2j}$ occupies the same mesh point as $u_{k+1,i,j}$. This is shown for $N = 7 = 2^3 - 1$ below. The u_0 are the finest mesh points, the u_1 the heavy dots, and the u_2 the $\oplus$.



The MG algorithm works by using an approximate solution for Poisson's equation from a coarse grid u_{k+1} as an initial guess for starting an iterative scheme for Poisson's equation on the finer grid u_k . The approximate solution on u_{k+1} is obtained by applying this algorithm recursively. We could continue down to the coarsest possible coarsest grid, u_{n-1} , which has just 1 unknown, which solve exactly. However, we usually terminate the recursion at a finer level. It is easiest to describe the algorithm in terms of various simple building blocks.

$GSRB(k, d, u_k, b_k)$ - Gauss-Seidel Red-Black: apply d iterations of a Red-Black Gauss-Seidel's method on the grid with starting values u_k and with right hand side b_k . (For simplicity we omit superscripts indicating iteration number.)

```

for  $m = 1 : d$ 
  forall  $(1 \leq i, j \leq 2^{n-k} - 1)$ 
     $u_{k,i,j} = (u_{k,i-1,j} + u_{k,i+1,j} + u_{k,i,j-1} + u_{k,i,j+1} + b_{k,i,j}) / 4$ 
  endfor
endfor

```

In order to use the solution from a coarse grid to help solve a problem on a fine grid, we need to be able to transfer a problem from the fine to coarse grid and back again. Going from fine to coarse is called *restriction* (a form of *downsampling*, and from coarse to fine is called *prolongation* (a form of *up-sampling*, *interpolation*).

$R(k, u_k, u_{k+1})$ - Restrict the fine grid data in u_k to the coarse grid data u_{k+1} . Do this by taking 1/16 of the sum of the NW (northwest), NE, SW, and SE fine grid neighbors, 1/8 of the N, E, W, and S fine grid neighbors, and 1/4 of the fine grid value at the same location.

```

forall  $(1 \leq i, j \leq 2^{n-k-1} - 1)$ 
   $u_{k+1,i,j} = (u_{k,2i-1,2j-1} + u_{k,2i+1,2j-1} + u_{k,2i-1,2j+1} + u_{k,2i+1,2j+1}) / 16 +$ 
     $(u_{k,2i,2j-1} + u_{k,2i,2j+1} + u_{k,2i-1,2j} + u_{k,2i+1,2j}) / 8 + u_{k,2i,2j} / 4$ 
endfor
forall boundary points, let  $u_{k+1,i,j} = u_{k,2i,2j}$ 

```

$P(k, u_k, u_{k+1})$ - Prolong the coarse grid data in u_{k+1} to the fine grid data u_k . We do this by averaging the nearest neighbors in the grid.

```

forall  $(1 \leq i, j \leq 2^{n-k-1} - 1)$ 
   $u_{k,2i,2j} = u_{k+1,i,j}$ 
endfor
forall  $(0 \leq i \leq 2^{n-k-1} - 1, 1 \leq j \leq 2^{n-k-1} - 1)$ 
   $u_{k,2i+1,2j} = (u_{k+1,i,j} + u_{k+1,i+1,j}) / 2$ 
endfor
forall  $(1 \leq i \leq 2^{n-k-1} - 1, 0 \leq j \leq 2^{n-k-1} - 1)$ 
   $u_{k,2i,2j+1} = (u_{k+1,i,j} + u_{k+1,i,j+1}) / 2$ 
endfor
forall  $(0 \leq i \leq 2^{n-k-1} - 1, 1 \leq j \leq 2^{n-k-1} - 1)$ 
   $u_{k,2i+1,2j+1} = (u_{k+1,i,j} + u_{k+1,i,j+1} + u_{k+1,i+1,j} + u_{k+1,i+1,j+1}) / 4$ 
endfor

```

Using these building blocks, we build a so-called Multigrid V-cycle: $MGV(k, u_k, b_k, \omega, d)$ - Per-

form multigrid V-cycle to solve Poisson's equation with right hand side b_k .

```

if  $k = n - 1$  then                                     //  $u_k$  has just one unknown
     $u_{k,1,1} = (u_{k,0,0} + u_{k,0,2} + u_{k,2,0} + u_{k,2,2} + b_{k,1,1})/4$ 
else
     $GSRB(k, d, u_k, b_k)$                                // Perform Red-Black Gauss Seidel
    forall ( $1 \leq i, j \leq 2^{n-k} - 1$ )                 // Compute the residual
         $r_{k,i,j} = (-4u_{k,i,j} + u_{k,i-1,j} + u_{k,i+1,j} + u_{k,i,j-1} + u_{k,i,j+1} + b_{k,i,j})/4$ 
    endfor
     $R(k, r_k, r_{k-1})$                                    // Restrict residual to coarser grid
    forall ( $1 \leq i, j \leq 2^{n-k-1} - 1$ )               // Initialize starting guess to coarse grid problem
         $u_{k+1,i,j} = 0$ 
    endfor
     $MGV(k+1, u_{k+1}, r_{k+1}, \omega, d)$                  // Solve coarse grid problem recursively
     $P(k, t_k, u_{k+1})$                                    // Prolong coarse grid correction back to fine grid
    forall ( $1 \leq i, j \leq 2^{n-k} - 1$ )                 // Update fine grid solution
         $u_{k,i,j} = u_{k,i,j} + t_{k,i,j}$ 
    endfor
     $GSRB(k, d, u_k, b_k)$                                // perform red-black Gauss-Seidel
endif

```

Finally, full Multigrid consists a sequence of V-cycles on ever finer grids, starting with MGV applied to the coarsest grid, interpolating this coarse solution as a starting guess to the next finer level. This proceeds recursively until we do a V-cycle on the finest grid.

$FMG(k, u_k, b_k, \omega, d_1, d_2)$ - Full multigrid to solve Poisson's equation with right hand side b_k .

```

if  $k = n - 1$  then                                     // coarsest grid, just one unknown
     $u_{k,1,1} = (u_{k,0,1} + u_{k,2,1} + u_{k,1,0} + u_{k,1,2} + b_{k,1,1})/4$ 
else
    forall ( $1 \leq i, j \leq 2^{n-k} - 1$ )                 // compute residual
         $r_{k,i,j} = (-4u_{k,i,j} + u_{k,i-1,j} + u_{k,i+1,j} + u_{k,i,j-1} + u_{k,i,j+1} + b_{k,i,j})/4$ 
    endfor
     $R(k, r_k, r_{k-1})$                                    // restrict residual to coarser grid
    forall ( $1 \leq i, j \leq 2^{n-k-1} - 1$ )               // initialize starting guess to coarse grid problem
         $u_{k+1,i,j} = 0$ 
    endfor
     $FMG(k+1, u_{k+1}, r_{k+1}, \omega, d_1, d_2)$            // solve coarse grid problem recursively
     $P(k, t_k, u_{k+1})$                                    // prolong coarse grid correction back to fine grid
    forall ( $1 \leq i, j \leq 2^{n-k} - 1$ )                 // update fine grid solution
         $u_{k,i,j} = u_{k,i,j} + t_{k,i,j}$ 
    endfor
    for  $i = 1$  to  $d_1$ 
         $MGV(k, u_k, b_k, \omega, d_2)$ 
    endfor
endif

```