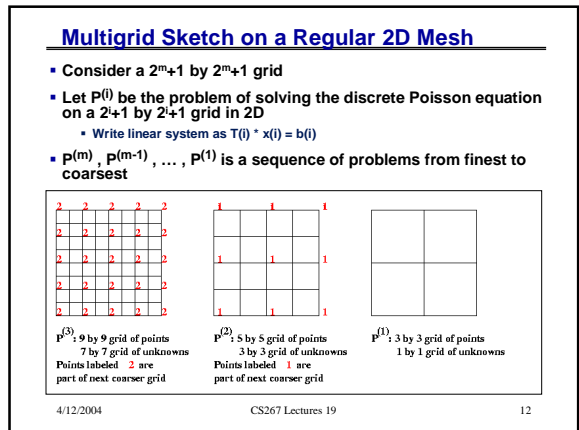
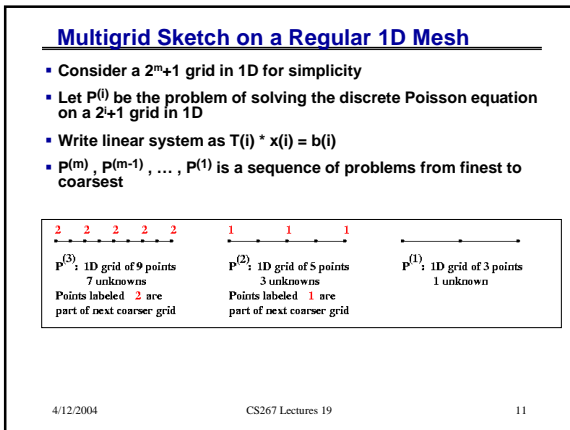
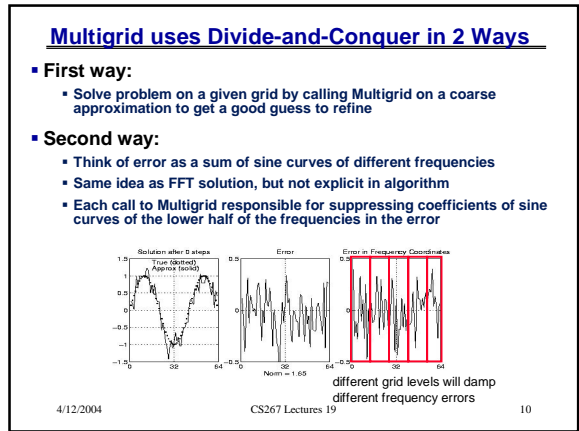


- ### Multigrid Overview
- Basic Algorithm:
 - Replace problem on fine grid by an approximation on a coarser grid
 - Solve the coarse grid problem approximately, and use the solution as a starting guess for the fine-grid problem, which is then iteratively updated
 - Solve the coarse grid problem **recursively**, i.e. by using a still coarser grid approximation, etc.
 - Success depends on coarse grid solution being a good approximation to the fine grid
- 4/12/2004 CS267 Lectures 19 8

- ### Same Big Idea Used in Other Problems
- Replace fine problem by coarse approximation, recursively
 - Multilevel Graph Partitioning (METIS):
 - Replace graph to be partitioned by a coarser graph, obtained via Maximal Independent Set
 - Given partitioning of coarse grid, refine using Kernighan-Lin
 - Barnes-Hut (and Fast Multipole Method):
 - Approximate leaf node of QuadTree containing many particles by Center-of-Mass and Total-Mass (or multipole expansion)
 - Approximate internal nodes of QuadTree by combining expansions of children
 - Force in any particular node approximated by combining expansions of a small set of other nodes
 - All examples depend on coarse approximation being accurate enough (at least if we are far enough away)
- 4/12/2004 CS267 Lectures 19 9



Multigrid Operators

- For problem $P^{(i)}$:
 - $b(i)$ is the RHS and
 - $x(i)$ is the current estimated solution
 - $T(i)$ is implicit in the operators below.)

} both live on grids of size 2^{i-1}
- Multigrid will be defined by a repeated sequence of operators
- The multigrid operators average values on neighboring grid points
 - Neighboring grid points on coarse problems are far away in fine problems, so information moves quickly on coarse problems
 - Levels will be constructed explicitly
 - The next slide gives an overview of the operators; details will come later

4/12/2004

CS267 Lectures 19

13

Multigrid Operators

- For problem $P^{(i)}$:
 - $b(i)$ is the RHS and
 - $x(i)$ is the current estimated solution

} both live on grids of size 2^{i-1}
- The restriction operator $R(i)$ maps $P^{(i)}$ to $P^{(i-1)}$
 - Restricts problem on fine grid $P^{(i)}$ to coarse grid $P^{(i-1)}$
 - Uses sampling or averaging
 - Right-hand sided is also restricted: $b(i-1) = R(i)(b(i))$
- The interpolation operator $In(i-1)$ maps solution $x(i-1)$ to $x(i)$
 - Maps one approximate solution to another
 - Interpolates solution on coarse grid $P^{(i-1)}$ to fine grid $P^{(i)}$
 - $x(i) = In(i-1)(x(i-1))$
- The solution operator $S(i)$ takes $P^{(i)}$ and improves solution $x(i)$
 - Uses "weighted" Jacobi or SOR on a single level of the grid
 - $x^{improved}(i) = S(i)(b(i), x(i))$

4/12/2004

CS267 Lectures 19

14

Multigrid V-Cycle Algorithm

Function MGV ($b(i), x(i)$)

... Solve $T(i)*x(i) = b(i)$ given $b(i)$ and an initial guess for $x(i)$

... return an improved $x(i)$

if ($i = 1$)

compute exact solution $x(1)$ of $P^{(1)}$ only 1 unknown

return $x(1)$

else

$x(i) = S(i)(b(i), x(i))$

improve solution by damping high frequency error

$r(i) = T(i)*x(i) - b(i)$

compute residual

$d(i) = In(i-1)(MGV(R(i)(r(i)), 0))$

solve $T(i)*d(i) = r(i)$ recursively

$x(i) = x(i) - d(i)$

correct fine grid solution

$x(i) = S(i)(b(i), x(i))$

improve solution again

return $x(i)$

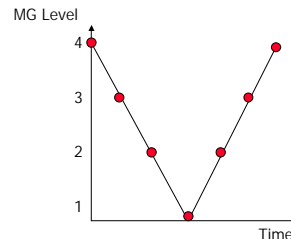
4/12/2004

CS267 Lectures 19

15

Why is this called a V-Cycle?

- Just a picture of the call graph
- In time a V-cycle looks like the following



4/12/2004

CS267 Lectures 19

16

Complexity of a V-Cycle on a 2D Grid

- On a serial machine
 - Work at each point in a V-cycle is $O(\text{the number of unknowns})$
 - Cost of Level i is $(2^{i-1})^2 = O(4^i)$
 - If finest grid level is m , total time is:

$$\sum_{i=1}^m O(4^i) = O(4^m) = O(\# \text{ unknowns})$$

- On a parallel machine (PRAM)
 - With one processor per grid point and free communication, each step in the V-cycle takes constant time
 - Total V-cycle time is $O(m) = O(\log \# \text{ unknowns})$

4/12/2004

CS267 Lectures 19

17

Full Multigrid (FMG)

- Intuition:
 - improve solution by doing multiple V-cycles
 - avoid expensive fine-grid (high frequency) cycles
 - analysis of why this works is beyond the scope of this class

Function FMG ($b(m), x(m)$)

... return improved $x(m)$ given initial guess

compute the exact solution $x(1)$ of $P(1)$

for $i=2$ to m

$x(i) = MGV(b(i), In(i-1)(x(i-1)))$

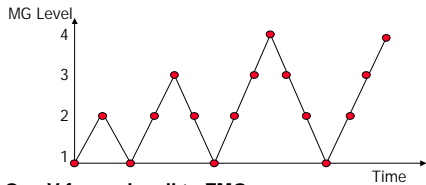
- In other words:
 - Solve the problem with 1 unknown
 - Given a solution to the coarser problem, $P^{(i-1)}$, map it to starting guess for $P^{(i)}$
 - Solve the finer problem using the Multigrid V-cycle

4/12/2004

CS267 Lectures 19

18

Full Multigrid Cost Analysis



- One V for each call to FMG
- Serial time: $\sum_{i=1}^m O(4^i) = O(4^m) = O(\# \text{ unknowns})$
- PRAM time: $\sum_{i=1}^m O(i) = O(m^2) = O(\log^2 \# \text{ unknowns})$

4/12/2004

CS267 Lectures 19

19

Complexity of Solving Poisson's Equation

- Theorem: error after one FMG call $\leq c^*$ error before, where $c < 1/2$, independent of # unknowns
- Corollary: We can make the error $<$ any fixed tolerance in a fixed number of steps, independent of size of finest grid
- This is the most important convergence property of MG, distinguishing it from all other methods, which converge more slowly for large grids
- Total complexity just proportional to cost of one FMG call

4/12/2004

CS267 Lectures 19

20

Details of Multigrid Operators

- For problem $P(i)$:
 - $b(i)$ is the RHS and
 - $x(i)$ is the current estimated solution
 - $(T(i))$ is implicit in the operators below.
 (both are $2^{i-1} \times 2^{i-1}$ grids)
- The **restriction operator $R(i)$** maps $P(i)$ to $P(i-1)$
 - $b(i-1) = R(i)(b(i))$
- The **interpolation operator $ln(i-1)$** maps an approximate solution $x(i-1)$ to an $x(i)$
 - $x(i) = ln(i-1)(x(i-1))$
- The **solution operator $S(i)$** takes $P(i)$ and computes improved $x(i)$
 - $x_{\text{improved}}(i) = S(i)(b(i), x(i))$

4/12/2004

CS267 Lectures 19

21

The Solution Operator $S(i)$ - Details

- The solution operator, $S(i)$, also called the *smoother* may be a *weighted Jacobi*
- Consider the 1D problem
- At level i , pure Jacobi replaces $x(j)$ by

$$x(j) := 1/2 (x(j-1) + x(j+1) + b(j))$$
- *Weighted Jacobi* uses:

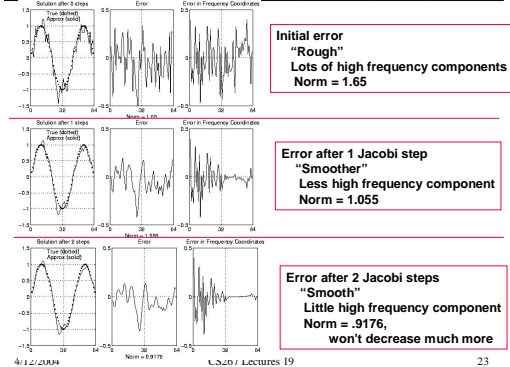
$$x(j) := 1/3 (x(j-1) + x(j) + x(j+1) + b(j))$$
- In 2D or 3D, similar average of nearest neighbors
- Other smoothers may be used

4/12/2004

CS267 Lectures 19

22

Weighted Jacobi Damps High Frequency Error



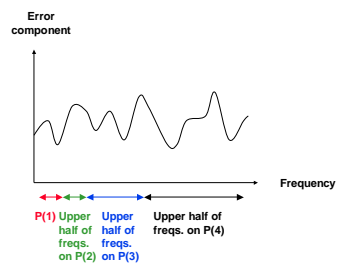
4/12/2004

CS267 Lectures 19

23

Multigrid as Divide and Conquer Algorithm

- Each level in a V-Cycle reduces the error in one part of the frequency domain



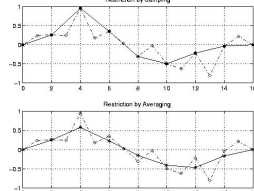
4/12/2004

CS267 Lectures 19

24

The Restriction Operator R(i) - Details

- The restriction operator, $R(i)$, takes
 - a problem $P^{(l)}$ with RHS $b^{(l)}$ and
 - maps it to a coarser problem $P^{(l-1)}$ with RHS $b^{(l-1)}$
- Simplest way: sampling
- Averaging values of neighbors is better; in 1D this is
 - $x_{\text{coarse}}(i) = 1/4 * x_{\text{fine}}(i-1) + 1/2 * x_{\text{fine}}(i) + 1/4 * x_{\text{fine}}(i+1)$



- In 2D, average with all 8 neighbors (N,S,E,W,NE,NW,SE,SW)

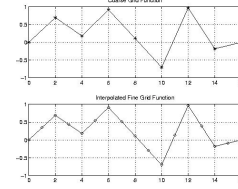
4/12/2004

CS267 Lectures 19

25

Interpolation Operator

- The interpolation operator $I_n(i-1)$, takes a function on a coarse grid $P^{(i-1)}$, and produces a function on a fine grid $P^{(i)}$
- In 1D, linearly interpolate nearest coarse neighbors
 - $x_{\text{fine}}(i) = x_{\text{coarse}}(i)$ if the fine grid point i is also a coarse one, else
 - $x_{\text{fine}}(i) = 1/2 * x_{\text{coarse}}(\text{left of } i) + 1/2 * x_{\text{coarse}}(\text{right of } i)$



- In 2D, interpolation requires averaging with 2 or 4 nearest neighbors (NW,SW,NE,SE)

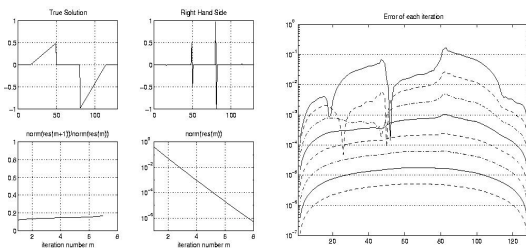


4/12/2004

CS267 Lectures 19

26

Convergence Picture of Multigrid in 1D



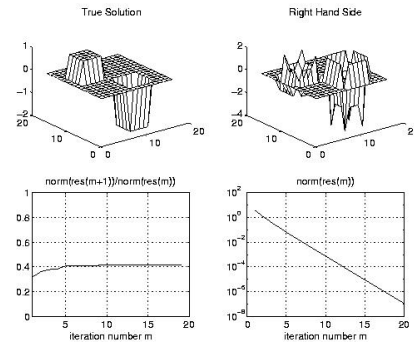
- Full Multigrid on each iteration
- Error decreases by a factor >5 on each iteration

4/12/2004

CS267 Lectures 19

27

Convergence Picture of Multigrid in 2D

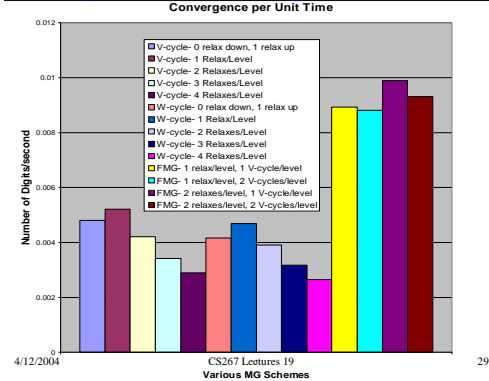


4/12/2004

CS267 Lectures 19

28

Convergence Rate of Multigrid Variations



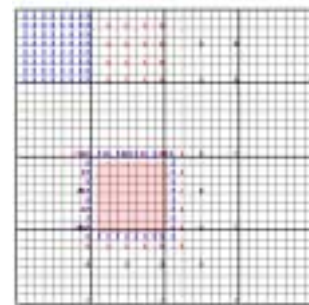
4/12/2004

CS267 Lectures 19

29

Parallel 2D Multigrid

- Multigrid on 2D requires nearest neighbor (up to 8) computation at each level of the grid
- Start with $n=2^m+1$ by 2^m+1 grid (here $m=5$)
- Use an s by s processor grid (here $s=4$)



4/12/2004

CS:

Performance Model of parallel 2D Multigrid

- Assume 2^{m+1} by 2^{m+1} grid of unknowns, $n = 2^{m+1}$, $N = n^2$
- Assume $p = 4^k$ processors, arranged in 2^k by 2^k grid
 - Each processor starts with 2^{m-k} by 2^{m-k} subgrid of unknowns
- Consider V-cycle starting at level m
 - At levels m through k of V-cycle, each processor does some work
 - At levels $k-1$ through 1 , some processors are idle, because a 2^{k-1} by 2^{k-1} grid of unknowns cannot occupy each processor
- Cost of one level in V-cycle
 - If level $j \geq k$, then cost =
 - $O(4^{j-k})$... Flops, proportional to the number of grid points/processor
 - $+ O(1) \alpha$... Send a constant # messages to neighbors
 - $+ O(2^{2j-k}) \beta$... Number of words sent
 - If level $j < k$, then cost =
 - $O(1)$... Flops, proportional to the number of grid points/processor
 - $+ O(1) \alpha$... Send a constant # messages to neighbors
 - $+ O(1) \beta$... Number of words sent
- Sum over all levels in all V-cycles in FMG to get complexity

4/12/2004

CS267 Lectures 19

31

Comparison of Methods

	# Flops	# Messages	# Words sent
MG	$N/p + \log p * \log N$	$(\log N)^2$	$(N/p)^{1/2} + \log p * \log N$
FFT (transpose)	$N \log N / p$	p	$N^*(p-1)/p^2$
SOR	$N^{3/2} / p$	$N^{1/2}$	N/p

- SOR is slower than others on all counts
- Flops for MG and FFT depends on accuracy of MG
- MG communicates less total data (bandwidth)
- Total messages (latency) depends ...
 - This analysis can't say whether MG or FFT is better when $\alpha \gg \beta$

4/12/2004

CS267 Lectures 19

32

Practicalities

- In practice, we don't go all the way to $P^{(1)}$
- In sequential code, the coarsest grids are negligibly cheap, but on a parallel machine they may not be.
 - Consider 1000 points per processor
 - In 2D, the surface to communicate is $4 \times \text{sqrt}(1000) \approx 128$, or 13%
 - In 3D, the surface is $1000 \cdot 8^3 \approx 500$, or 50%
- See Tuminaro and Womble, SIAM J. Sci. Comp., v14, n5, 1993 for analysis of MG on 1024 nCUBE2
 - on 64×64 grid of unknowns, only 4 per processor
 - efficiency of 1 V-cycle was .02, and on FMG .008
 - on 1024×1024 grid
 - efficiencies were .7 (MG Vcycle) and .42 (FMG)
 - although worse parallel efficiency, FMG is 2.6 times faster than V-cycles alone
 - nCUBE had fast communication, slow processors

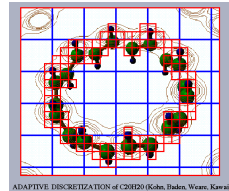
4/12/2004

CS267 Lectures 19

33

Multigrid on an Adaptive Mesh

- For problems with very large dynamic range, another level of refinement is needed
- Build adaptive mesh and solve multigrid (typically) at each level



- Can't afford to use finest mesh everywhere

4/12/2004

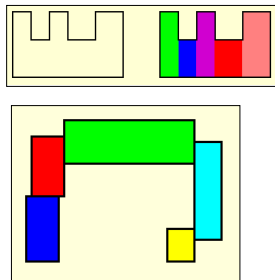
CS267 Lectures 19

34

Multiblock Applications

- Solve system of equations on a union of rectangles
 - subproblem of AMR

E.g.,



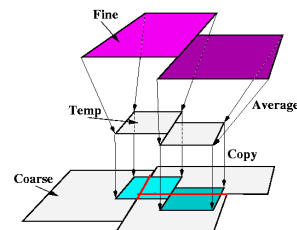
4/12/2004

CS267 Lectures 19

35

Adaptive Mesh Refinement

- Data structures in AMR
- Usual parallelism is to deal grids on each level to processors
- Load balancing is a problem

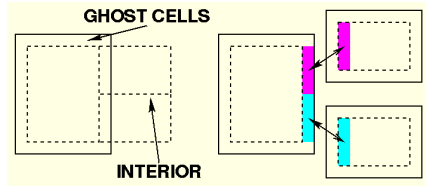


4/12/2004

36

Support for AMR

- Domains in Titanium designed for this problem
- Kelp, Boxlib, and AMR++ are libraries for this
- Primitives to help with boundary value updates, etc.



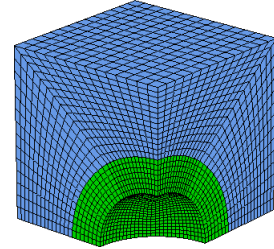
4/12/2004

CS267 Lectures 19

37

Multigrid on an Unstructured Mesh

- Another approach to variable activity is to use an unstructured mesh that is more refined in areas of interest
- Controversy over adaptive rectangular vs. unstructured
 - Numerics easier on rectangular mesh
 - Supposedly easier to implement (arrays without indirection) but boundary cases tend to dominate code for adaptive mesh



4/12/2004

CS267 Lectures 19

38

Multigrid on an Unstructured Mesh

- Need to partition graph for parallelism
- What does it mean to do Multigrid anyway?
- Need to be able to coarsen grid (hard problem)
 - Can't just pick "every other grid point" anymore
 - Use "maximal independent sets"
 - How to make coarse graph approximate fine one
- Need to define $R()$ and $In()$
 - How do we convert from coarse to fine mesh and back?
- Need to define $S()$
 - How do we define coarse matrix (no longer formula, like Poisson)
- Dealing with coarse meshes efficiently
 - Should we switch to using fewer processors on coarse meshes?
 - Should we switch to another solver on coarse meshes?
- See Prometheus by Mark Adams for unstructured Multigrid
 - Solved up to 39M unknowns on 960 processors with 50% efficiency
 - www.cs.berkeley.edu/~madams

4/12/2004

CS267 Lectures 19

39

Summary

- Multigrid is used to solve elliptic problems, such as the Poisson equation
- Multigrid is closely related to SOR, Jacobi and other nearest neighbor (sparse matrix-vector multiply) solvers
- The key idea is to speed up convergence by using coarser versions of the mesh
 - Information flows across the problem domain faster on the nearest neighbors of a coarse mesh
 - Different mesh levels damp different error frequencies
- Details of algorithm vary
 - Different "smoothers"
 - Different call graphs (V, W, FMG)
 - FMG better experimentally, but may depend on problem and machine

4/12/2004

CS267 Lectures 19

40