Additively Damped AFAC Variants for High Order Discretisations

Charles Murray, Tobias Weinzierl
c.d.murray@durham.ac.uk, tobias.weinzierl@durham.ac.uk

Durham University

Stencil Integration

Construction of an accurate matrix equation can become increasingly expensive for complicated material parameters—this is compounded by the construction of coarse grid equations within Multigrid

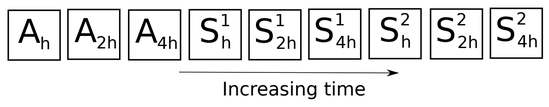


Figure 1: Breaking Multigrid operations into tasks. $A_{k^d h}$ corresponds to matrix construction, $S^i_{k^d h}$ corresponds to smoothing

Multigrid smoothing focuses on reducing the error in a solution approximate in each iteration
Therefore an exact fine grid equation may not be required—merely a "good enough" approximation
The overall matrix equation is dependent upon a finite element integration—this can readily be written as a quadrature
Can easily compose such a quadrature as an iterative process (start with low number of intervals, p, and increase to improve accuracy of quadrature)
This methodology should be effective for any matrix computation that can be written as an iterative process with subsequent iterations showing a lower error than earlier.

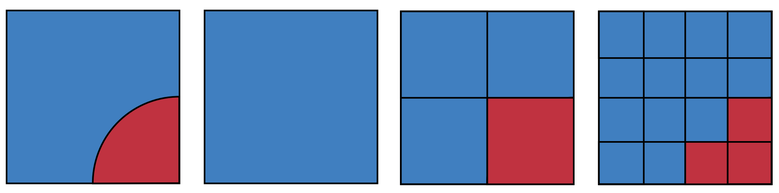


Figure 2: Accurate material data (left); Quadrature iterates to capture material data (rest)

Delayed Evaluation

Enforcing syncronicity between the multigrid algorithm and the equation construction can again delay time to solution—multigrid iterates must wait on integration iterates to complete before they can start
Instead store a local approximate which the multigrid algorithm can use if the local integration has not terminated yet

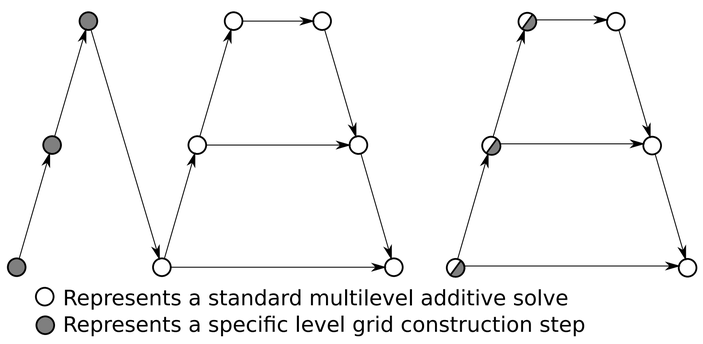


Figure 3: V-cycle with precomputed coarse grid construction (left); V-cycle with delayed coarse grid evaluation (right)

Can start smoothing on coarse grids once approximation of the equation exists on those grids
Ritz-Galerkin formulation requires a matrix triple product that is dependent upon the fine grid equation
With early approximations of the fine grid stencils start computing a coarse grid stencil—these can be used as soon as they are available
Implementation plugged into grid traversal
When finish handling of grid level compute an update to the equation held on this grid level—updates ripple up single grid level at a time

Multitasking

Rather than treating as single monolithic break down into a series of tasks with known dependencies

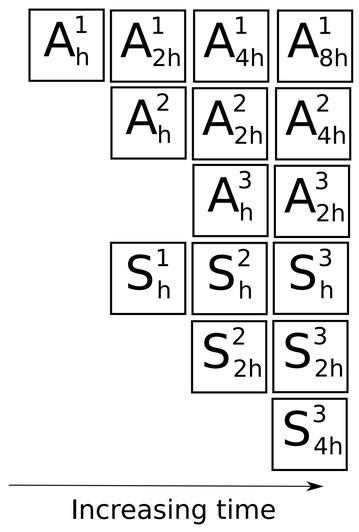


Figure 4: Representation of pipelined tasks from (Fig 1)

Three classes of task:
1- Construction of the fine grid matrix equation
2- Subsequent construction of coarse grid matrix equations
3- Multigrid smoothing
In order to parallelise the tasks their dependencies must be known: Fine grid approximation relies on earlier fine grid approximations; Coarse grid approximation is dependent upon a fine grid equation; Smoothing is dependent upon a grid existing on that level in the multigrid hierarchy
Start process (in parallel) as soon as data is available
Can compose these as tasks—that is create background tasks that are deployed to an idle part of the system as soon as it can be done
Fine grid computation is expensive computation and time consuming so benefits the most from this tasking

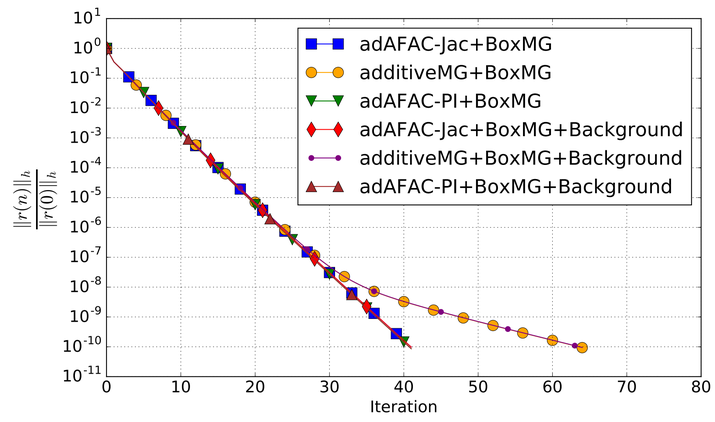
Greedy Algorithm

Algorithm 1 This function is called when a fine grid cell is encountererd for the first time during a traversal of the grid.

```
function GREEDY-INTEGRATION
  if p = ⊥ then
    p ← 1
    IntegrateStencil(p)
    Wait for initial integration
    LocalStencil ← NewStencil
    p ← 2
    IntegrateStencil(p)
  else if p = ⊤ then
    Take no action
  else
    if IntegrationTerminated then
      if NewStencil ≈ LocalStencil then
        p ← ⊤
      else
        p ← p + 1
        IntegrateStencil(p)
      end if
      LocalStencil ← NewStencil
      IntegrationTerminated ← False
    end if
  end if
end function
```

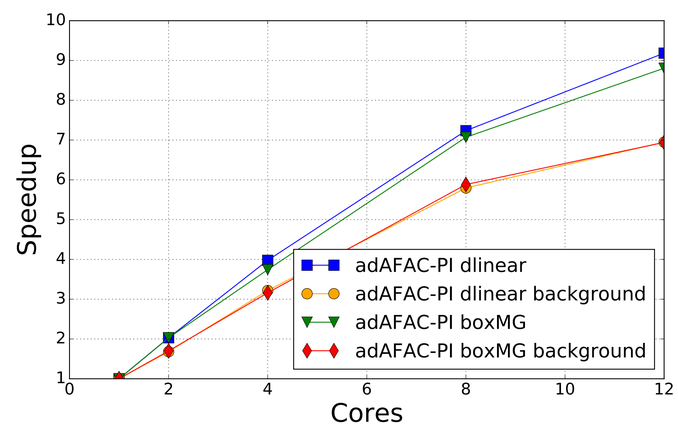
Results 3

Comparison of computing all stencils exactly before start with our method of computing in parallel with solve
Coarse grids computed algebraically as results ripple up through the coarse grids
No difference in number of iterations required to reach convergence between the two methods



Results 4

Initial single node scalability studies
Scalability is shown to be similar to that of precomputed–however this hides the true advantange a reduction in time to solution of a third



Takeaway

AdaFAC-Jac has proved to be an effective method of damping additive multigrid to improve the rate of convergence.
The improvement is similar to the stability benefits from BoxMG.
Delayed construction of the matrix equations by deployment of background tasks reduces the time to solution by about a third and shows similar scalability.

References

[1] J. M. Smith and A. B. Jones. *Book Title*. Publisher, 7th edition, 2012.

[2] A. B. Jones and J. M. Smith. Article Title. *Journal title*, 13(52):123–456, March 2013.