A new MATLAB interface to MueLu

Tobias Wiesner
Jonathan Hu
Brian Kelley
Chris Siefert

Sandia National Labs

October 27, 2015

SAND2015-9327 PE

Multigrid Framework

MueLu

MATLAB

MATLAB is registered trademark of The MathWorks, Inc.

MATLAB

MueLu
Multigrid Framework

MATLAB is registered trademark of The MathWorks, Inc.

MueMex

Multigrid Framework

Chris Siefert

Brian Kelley

MueMex = MATLAB interface for MueLu

What is MueLu?

Sandia National Laboratories

MueLu is . . .

. . . the next-generation multigrid framework package in Trilinos.

- provides AMG methods to solve large linear systems of equations
- can be understood as successor but not replacement for ML
- supports both Epetra and Tpetra as linear algebra framework
- is *the* typical Trilinos package

What is MueLu?

MueLu is . . .

. . . the next-generation multigrid framework package in Trilinos.

- provides AMG methods to solve large linear systems of equations
- can be understood as successor but not replacement for ML
- supports both Epetra and Tpetra as linear algebra framework
- is *the* typical Trilinos package
  - makes heavy use of other Trilinos packages
    (thanks to the Amesos2, Ifpack2, Zoltan2, Tpetra, Epetra, Kokkos, . . . developers)

What is MueLu?

MueLu is . . .

. . . the next-generation multigrid framework package in Trilinos.

- provides AMG methods to solve large linear systems of equations
- can be understood as successor but not replacement for ML
- supports both Epetra and Tpetra as linear algebra framework
- is *the* typical Trilinos package
  - makes heavy use of other Trilinos packages
    (thanks to the Amesos2, Ifpack2, Zoltan2, Tpetra, Epetra, Kokkos, . . . developers)
  - runs on usual laptops, super-computers and (soon) next-generation HPC systems

What is MueLu?

MueLu is . . .

. . . the next-generation multigrid framework package in Trilinos.

- provides **AMG methods** to solve **large linear systems** of equations
- can be understood as **successor** but not replacement for **ML**
- supports both **Epetra and Tpetra** as linear algebra framework
- is *the* typical **Trilinos package**
  - makes **heavy use** of **other Trilinos packages** (thanks to the Amesos2, Ifpack2, Zoltan2, Tpetra, Epetra, Kokkos, . . . developers)
  - runs on usual laptops, super-computers and (soon) next-generation HPC systems
- MueLu is *international:* we can give **support** in

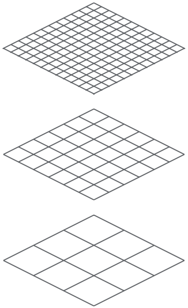
english (since 2010)

german (since 2011)

russian (since 2012)

french (discontinued in 2012)

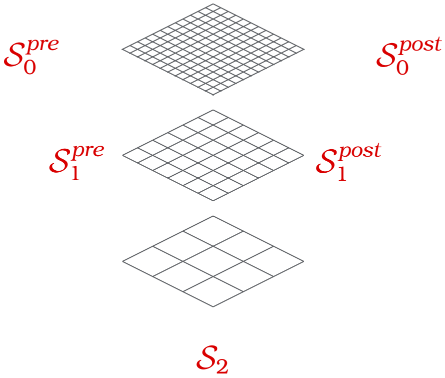
Basic concept of Algebraic Multigrid (AMG)



Main idea

Capture errors at multiple resolutions.

Basic concept of Algebraic Multigrid (AMG)



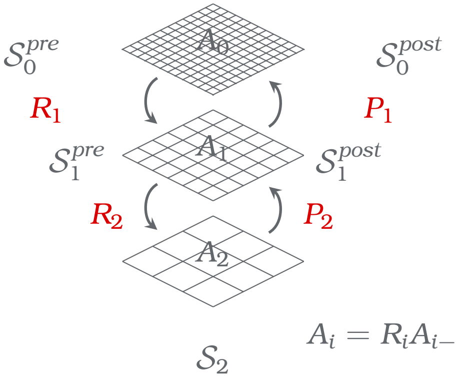
Two main components

- Smoothers
  - Approximate solve on each level
  - ``Cheap'' reduction of oscillatory error (high energy)
  - $\mathcal{S}_L \approx A_L^{-1}$ on the coarsest level L

Main idea

Capture errors at multiple resolutions.

Basic concept of Algebraic Multigrid (AMG)



$$A_i = R_i A_{i-1} P_i$$

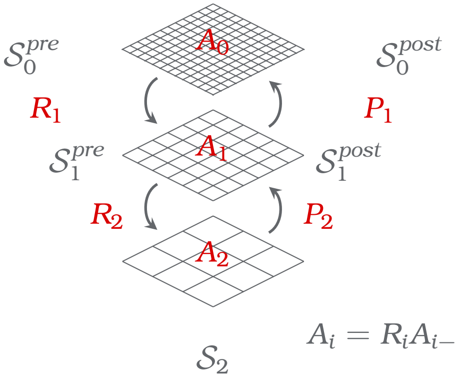
Two main components

- Smoothers
  - Approximate solve on each level
  - ''Cheap'' reduction of oscillatory error (high energy)
  - $\mathcal{S}_L \approx A_L^{-1}$ on the coarsest level L
- Grid transfers (prolongators and restrictors)
  - Data movement between levels
  - Definition of coarse level matrices.

Main idea

Capture errors at multiple resolutions.

Basic concept of Algebraic Multigrid (AMG)



$$A_i = R_i A_{i-1} P_i$$

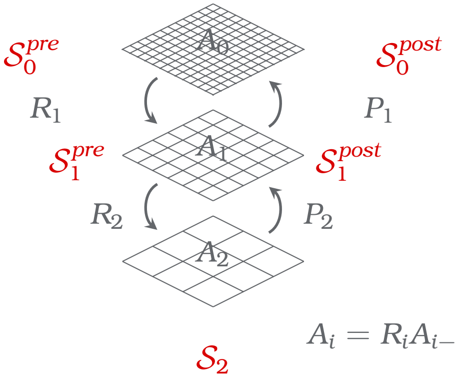
Algorithmic phases

- Setup phase
  - Build transfer operators to determine coarse level matrices
  - Initialize level smoothers

Main idea

Capture errors at multiple resolutions.

Basic concept of Algebraic Multigrid (AMG)



$$A_i = R_i A_{i-1} P_i$$

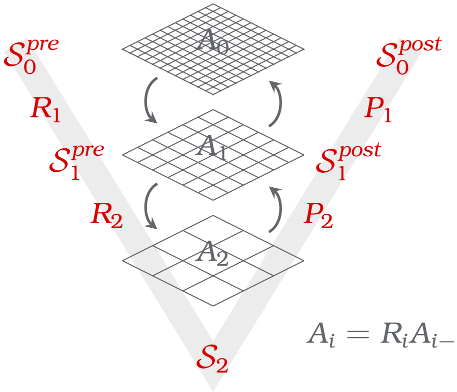
Algorithmic phases

- Setup phase
  - Build transfer operators to determine coarse level matrices
  - Initialize level smoothers

Main idea

Capture errors at multiple resolutions.

Basic concept of Algebraic Multigrid (AMG)



$$A_i = R_i A_{i-1} P_i$$

Algorithmic phases

- Setup phase
  - Build transfer operators to determine coarse level matrices
  - Initialize level smoothers
- Solving phase
  - Run through multigrid cycle (e.g. V-cycle)
  - Iteratively solve linear system or apply some sweeps with multigrid as preconditioner within an iterative linear solver

Main idea

Capture errors at multiple resolutions.

What is MueMex good for?

What can MueLu provide for MATLAB?

What is MueMex good for?

What can MueLu provide for MATLAB?

- MueMex runs MueLu as preconditioner in Belos
  - Access to all features of Belos (linear solvers, multiple RHS. . .).
  - Access to all preconditioners from Trilinos through MueLu

What is MueMex good for?

What can MueLu provide for MATLAB?

- MueMex runs **MueLu as preconditioner** in **Belos**
  - Access to all features of Belos (linear solvers, multiple RHS. . .).
  - Access to all preconditioners from Trilinos through MueLu
- **Efficient iterative solution** of very **large linear systems**
  - Run MueLu multigrid setup once
  - Use multigrid hierarchy for solving several (similar) linear systems with varying linear operators and/or right hand sides.

What is MueMex good for?

What can MueLu provide for MATLAB?

- MueMex runs **MueLu as preconditioner** in **Belos**
  - Access to all features of Belos (linear solvers, multiple RHS. . .).
  - Access to all preconditioners from Trilinos through MueLu
- **Efficient iterative solution** of very **large linear systems**
  - Run MueLu multigrid setup once
  - Use multigrid hierarchy for solving several (similar) linear systems with varying linear operators and/or right hand sides.

Why is MATLAB useful for MueLu?

What is MueMex good for?

What can MueLu provide for MATLAB?

- MueMex runs **MueLu as preconditioner** in **Belos**
  - Access to all features of Belos (linear solvers, multiple RHS. . .).
  - Access to all preconditioners from Trilinos through MueLu
- **Efficient iterative solution** of very **large linear systems**
  - Run MueLu multigrid setup once
  - Use multigrid hierarchy for solving several (similar) linear systems with varying linear operators and/or right hand sides.

Why is MATLAB useful for MueLu?

- **Analyze** and tweak **multigrid methods** using the full functionality of MATLAB.

What is MueMex good for?

What can MueLu provide for MATLAB?

- MueMex runs **MueLu as preconditioner** in **Belos**
  - Access to all features of Belos (linear solvers, multiple RHS. . .).
  - Access to all preconditioners from Trilinos through MueLu
- **Efficient iterative solution** of very **large linear systems**
  - Run MueLu multigrid setup once
  - Use multigrid hierarchy for solving several (similar) linear systems with varying linear operators and/or right hand sides.

Why is MATLAB useful for MueLu?

- **Analyze** and tweak **multigrid methods** using the full functionality of MATLAB.
- Perform basic research on multigrid methods for specific problems.

1. How to use MueMex

Basic Laplace example — Setup

Define problem:

```
>> [A,coords] = laplacianfun([50,50]);
```

Basic Laplace example — Setup

Define problem:

```
>> [A,coords] = laplacianfun([50,50]);
```

Multigrid setup:

Minimal setup: Use default parameters defined by MueLu

```
>> [problemID, oc] = muelu('setup', A);
```

Basic Laplace example — Setup

Define problem:

```
>> [A,coords] = laplacianfun([50,50]);
```

Multigrid setup:

Minimal setup: Use default parameters defined by MueLu

```
>> [problemID, oc] = muelu('setup', A);
```

Multigrid parameters: Provide user parameters (see MueLu user guide)

```
>> [problemID, oc] = muelu('setup', A, 'coarse: max
   size', 50);
```

Basic Laplace example — Setup

Define problem:

```
>> [A,coords] = laplacianfun([50,50]);
```

Multigrid setup:

Minimal setup: Use default parameters defined by MueLu

```
>> [problemID, oc] = muelu('setup', A);
```

Multigrid parameters: Provide user parameters (see MueLu user guide)

```
>> [problemID, oc] = muelu('setup', A, 'coarse: max
   size', 50);
```

XML parameter file: Provide user parameters through xml file

```
[problemID, oc] = muelu('setup', A, 'xml parameter
   file', 'myParams.xml');
```

MATLAB 7.11.0 (R2010b)

File Edit Debug Parallel Desktop Window Help

Current Folder: /home/tobias/promotion/trilinos-mex/packages/muelu/matlab/bin

Shortcuts How to Add What's New

Current Folder

« matlab ▸ bin ▸

Name

- CMakeFiles
- Tests
- cmake_install.cmake
- constructAggregates.m
- constructGraph.m
- CTestTestfile.cmake
- driver1b.xml
- Makefile
- matlab
- muelu.m
- muemex.mexa64
- mymatlab
- setup.m

Tests (File Folder)

No details available

Command Window

```
  Ptent   # rows per proc   : avg = 5.40e+01,  dev =   0.0%,  min =   +0.0%,  max =   +0.0%
  Ptent   # nnz per proc    : avg = 5.40e+01,  dev =   0.0%,  min =   +0.0%,  max =   +0.0%
 Eigenvalue estimate
  Calculating max eigenvalue estimate now (max iters = 10)
  Prolongator damping factor = 0.97 (1.33 / 1.38)
 Fused (I-omega*D^{-1} A)*Ptent
 P size =  54 x 8, nnz = 127
 P Load balancing info
 P   # active processes: 1/1
 P   # rows per proc   : avg = 5.40e+01,  dev =   0.0%,  min =   +0.0%,  max =   +0.0%
 P   # nnz per proc    : avg = 1.27e+02,  dev =   0.0%,  min =   +0.0%,  max =   +0.0%
Transpose P (MueLu::TransPFactory)
 R size =  8 x 54, nnz = 127
 R Load balancing info
 R   # active processes: 1/1
 R   # rows per proc   : avg = 8.00e+00,  dev =   0.0%,  min =   +0.0%,  max =   +0.0%
 R   # nnz per proc    : avg = 1.27e+02,  dev =   0.0%,  min =   +0.0%,  max =   +0.0%
Computing Ac (MueLu::RAPFactory)
 MxM: A x P
  Matrix product nnz per row estimate = 9
 MxM: R x (AP) (explicit)
  Matrix product nnz per row estimate = 30
 Ac size =  8 x 8, nnz = 44
 Ac Load balancing info
 Ac   # active processes: 1/1
 Ac   # rows per proc   : avg = 8.00e+00,  dev =   0.0%,  min =   +0.0%,  max =   +0.0%
 Ac   # nnz per proc    : avg = 4.40e+01,  dev =   0.0%,  min =   +0.0%,  max =   +0.0%
Max coarse size (<= 50) achieved
Setup Smoother (MueLu::Amesos2Smoother{type = Superlu})

--------------------------------------------------------------------------------
---                            Multigrid Summary                             ---
--------------------------------------------------------------------------------
Number of levels    = 4
Operator complexity = 1.33

level  rows    nnz  nnz/row  c ratio  procs
  0    2500  12300    4.92               1
  1     425   3591    8.45     5.88      1
  2      54    404    7.48     7.87      1
  3       8     44    5.50     6.75      1

Smoother (level 0) both : "Ifpack2::Relaxation": {Initialized: true, Computed: true, Type: Symmetric Gauss-

Smoother (level 1) both : "Ifpack2::Relaxation": {Initialized: true, Computed: true, Type: Symmetric Gauss-

Smoother (level 2) both : "Ifpack2::Relaxation": {Initialized: true, Computed: true, Type: Symmetric Gauss-

Smoother (level 3) pre  : SuperLU solver interface, direct solve
Smoother (level 3) post : no smoother

Set up problem #0
>> [problemID, oc] = muelu('setup', A, 'coarse: max size', 50);
```

Basic Laplace example — Solve

Solve problem:

```
>> b = ones(2500,1);
>> [x, numIters] = muelu(problemID, b);
```

MATLAB output:

```
>> [x, numIters] = muelu(problemID, b);
*******************************************************
***** Belos Iterative Solver:  Pseudo Block Gmres
***** Maximum Iterations: 1000
***** Block Size: 1
***** Residual Test:
*****   Test 1 : Belos::StatusTestImpResNorm<>: (2-Norm Res Vec) / (2-Norm Prec Res0), tol = 1e-08
*******************************************************
Iter    0, [ 1] :    1.000000e+00
Iter    1, [ 1] :    5.608530e-01
Iter    2, [ 1] :    1.953063e-02
Iter    3, [ 1] :    1.303521e-03
Iter    4, [ 1] :    7.361411e-05
Iter    5, [ 1] :    4.277384e-06
Iter    6, [ 1] :    2.581678e-07
Iter    7, [ 1] :    1.169750e-08
Iter    8, [ 1] :    7.920252e-10
Success, Belos converged!
```

Basic Laplace example — Analysis

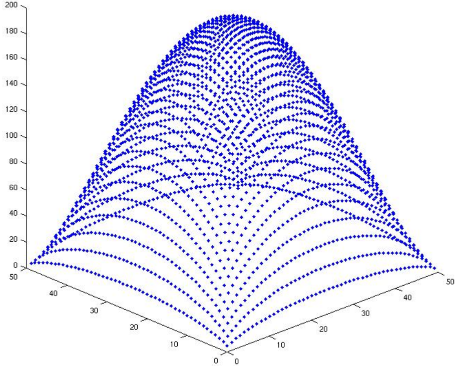
Visualize solution and error:

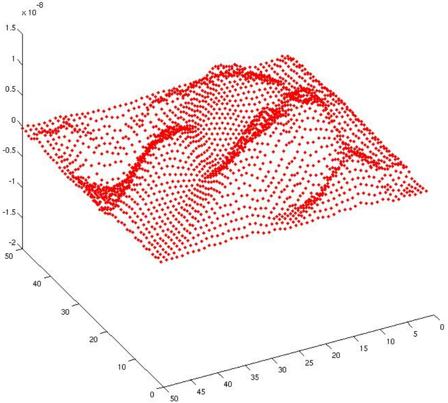
```
>> plot3(coords(:,1),coords(:,2),x,'.')
>> plot3(coords(:,1),coords(:,2),x-A\b,'r.')
```

Basic Laplace example — Analysis

Visualize solution and error:

```
>> plot3(coords(:,1),coords(:,2),x,'.')
>> plot3(coords(:,1),coords(:,2),x-A\b,'r.')
```





Complex scalars

MueLu also works with complex scalars:

```
>> A = gallery(`tridiag`,2500,-1.0,2.0,-1.0) + gallery(`tridiag
    `,2500,-100.0,200.0,-100.0) * i;
>> [p,oc] = muelu('setup',A);
```

MATLAB output:

```
 --------------------------------------------------------------------------------
---                            Multigrid Summary                             ---
--------------------------------------------------------------------------------
Number of levels    = 2
Operator complexity = 1.33
level  rows   nnz  nnz/row  c ratio  procs
  0    2500  7498     3.00               1
  1     834  2500     3.00     3.00      1
Smoother (level 0) both : "Ifpack2::Relaxation":
        {Initialized: true, Computed: true,
         Type: Symmetric Gauss-Seidel,
         sweeps: 1, damping factor: (1,0),
         Global matrix dimensions: [2500, 2500],
         Global nnz: 7498}
Smoother (level 1) pre  : SuperLU solver interface, direct solve
Smoother (level 1) post : no smoother
Set up problem #0
```

Multiple right-hand sides

MueLu can solve multiple right-hand sides:

```
>> b = ones(2500,2);
>> for j = 1:2500, b(j,2) = 1/2500*j + 1/(2500*2500)*j.*j*i; end
>> [x,numIters] = muelu(p,b);
```

MATLAB output:

```
*******************************************************
***** Belos Iterative Solver:  Pseudo Block Gmres
***** Maximum Iterations: 1000
***** Block Size: 1
***** Residual Test:
*****   Test 1 : Belos::StatusTestImpResNorm<>: (2-Norm Res Vec) / (2-Norm Prec Res0), tol = 1e-08
*******************************************************
Iter    0, [  1] :    1.000000e+00  Iter    0, [  2] :    1.000000e+00
Iter    2, [  1] :    1.592409e-01  Iter    2, [  2] :    1.414063e-01
Iter    4, [  1] :    8.602037e-05  Iter    4, [  2] :    7.553543e-05
Iter    6, [  1] :    4.425200e-08  Iter    6, [  2] :    3.866241e-08
Iter    7, [  1] :    1.070092e-09  Iter    7, [  2] :    9.321384e-10
Success, Belos converged!
```

Multiple right-hand sides

MueLu can solve multiple right-hand sides:

```
>> b = ones(2500,2);
>> for j = 1:2500, b(j,2) = 1/2500*j + 1/(2500*2500)*j.*j*i; end
>> [x,numIters] = muelu(p,b);
```

MATLAB output:

```
*******************************************************
***** Belos Iterative Solver:  Pseudo Block Gmres
***** Maximum Iterations: 1000
***** Block Size: 1
***** Residual Test:
*****   Test 1 : Belos::StatusTestImpResNorm<>: (2-Norm Res Vec) / (2-Norm Prec Res0), tol = 1e-08
*******************************************************
Iter    0, [  1] :    1.000000e+00  Iter    0, [  2] :    1.000000e+00
Iter    2, [  1] :    1.592409e-01  Iter    2, [  2] :    1.414063e-01
Iter    4, [  1] :    8.602037e-05  Iter    4, [  2] :    7.553543e-05
Iter    6, [  1] :    4.425200e-08  Iter    6, [  2] :    3.866241e-08
Iter    7, [  1] :    1.070092e-09  Iter    7, [  2] :    9.321384e-10
Success, Belos converged!
```

Attention:

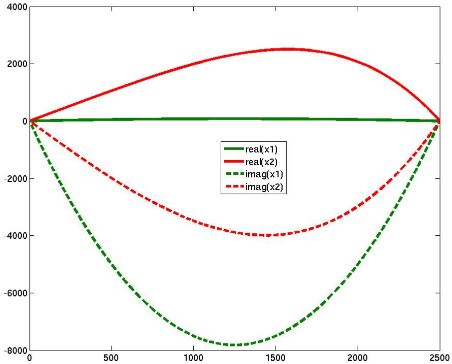
If the hierarchy is built with a complex operator A, the RHS vector has to contain at least one imaginary value!

Results

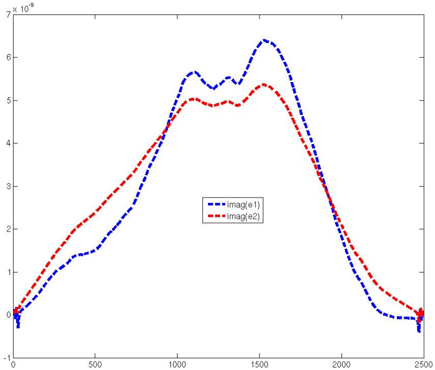
Visualization of results in MATLAB:

```
>> plot(real(x(:,1))); hold on; plot(real(x(:,2)));
>> plot(imag(x(:,1))); plot(imag(x(:,2))); hold off;
>> plot(imag(A\b-x));
```

Plot of solution vector:



Plot of imaginary part of error:



2. How to access MueLu data

Study multigrid methods

Example: Study the multigrid effect on a 1d example:

```
>> A = gallery('tridiag',600,-1,2,-1);
>> b=ones(600,1);
>> [problemID,oc] = muelu('setup',A,'coarse: max size',50,'
   multigrid algorithm','unsmoothed');
```

MATLAB output:

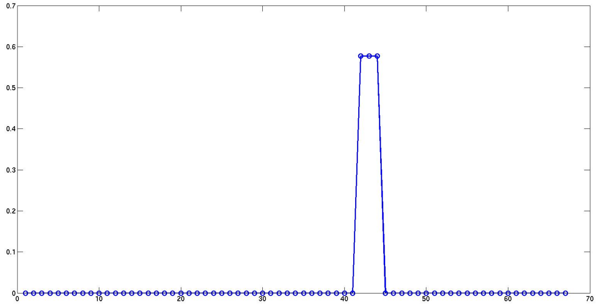
```
--------------------------------------------------------------------------------
---                          Multigrid Summary
---
--------------------------------------------------------------------------------
Number of levels     = 4
Operator complexity = 1.48
level  rows   nnz  nnz/row  c ratio  procs
  0     600  1798    3.00              1
  1     200   598    2.99     3.00     1
  2      67   199    2.97     2.99     1
  3      23    67    2.91     2.91     1
```

How to access transfer operators?

Extract (non-smooth) prolongation operators:

```
>> Ptent1 = muelu('get', problemID, 1, 'P');
>> Ptent2 = muelu('get', problemID, 2, 'P');
>> Ptent3 = muelu('get', problemID, 3, 'P');
>> plot(Ptent3(:,15)); hold on; plot(Ptent3(:,15),'o');
```

Plot of non-smooth basis function #15 of `Ptent3`:



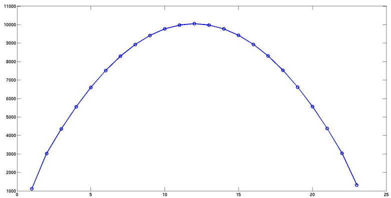
How to access coarse level operators?

Extract coarse level operator on level 3 and solve coarse level problem using MATLAB:

```
>> b1=Ptent1'*b;
>> b2=Ptent2'*b1;
>> b3=Ptent3'*b2;
>>
>> A3 = muelu('get',
   problemID, 3, 'A');
>>
>> x3 = A3 \ b3;
```

Plot of coarse level solution (level 3):

```
>> plot(x3); hold on;
>> plot(x3,'o'); hold off;
```

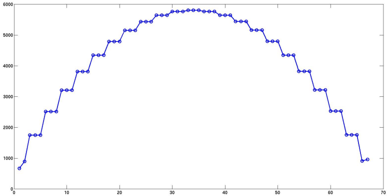


Exact coarse level solution of fine level problem.

Fine level solution and level smoothing

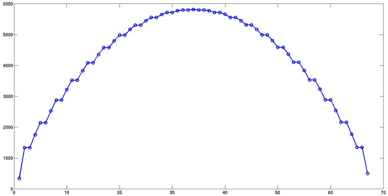
Plot prolongated solution on level 2:

```
>> x2p = Ptent3 * x3;
>> plot(x2p); hold on;
>> plot(x2p,'o'); hold off;
```



Apply one sweep with Jacobi to prolongated solution:

```
>> A2 = muelu('get',
   problemID, 2, 'A');
>> T = inv(D) * (tril(-A2
   ,-1)+triu(-A2,1));
>> x2s = T * x2p + inv(D) *
   b2;
>> plot(x2s); hold on; plot(
   xs2,'o'); hold off;
```

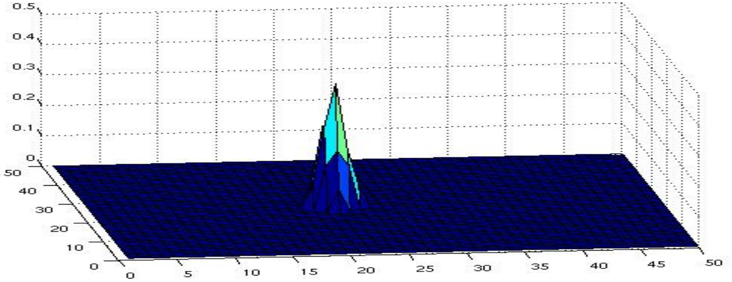


2D example — default parameters

Visualize **transfer operator basis functions** of 2D Laplace problem:

```
>> [A,coords] = laplacianfun([50 50]);
>> [problemID] = muelu('setup', A);
>> P1 = muelu('get', problemID, 1, 'P');
>> [X,Y]=meshgrid(coords(:,1),coords(:,2));
>> Z = griddata(coords(:,1),coords(:,2),full(P1(:,212)),X, Y);
>> surf(X,Y,Z);
```

Plot of *smooth* prolongator basis function (associated with aggregate 212):

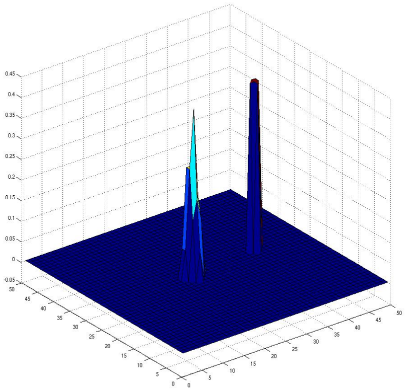


2D example — non-smooth transfers

Sandia National Laboratories

Comparison of *smooth* prolongator basis function (associated with aggregate 212) and non-smooth basis function (associated with aggregate 234):

```
>> [problem2, oc] = muelu('setup
   ',A,'multigrid algorithm','
   unsmoothed');
>> Ptent1 = muelu('get',
   problem2, 1, 'P');
>> Z2 = griddata(coords(:,1),
   coords(:,2),full(Ptent1
   (:,234)),X,Y);
>> surf(X,Y,Z); hold on; surf(X,
   Y,Z2); hold off;
```



3. Where can i learn more about MueLu?

MueLu resources

- The **MueLu user guide**
  - can be found here: https://trilinos.org/packages/muelu/muelu-documentation/
  - serves as reference handbook
  - provides an **overview of** all available **user parameters** and basic examples
- **Examples** come with the MueLu sources in the `examples` folder
- Doxygen

The MueLu tutorial

- can be found here: www.trilinos.org/packages/muelu/muelu-tutorial
- comes with an interactive GUI for individual experiments
- no Trilinos installation necessary: we provide a VirtualBox image and a docker container
- The MueLu tutorial is divided into
  - The beginners tutorial: Chapters 1-5 meant for absolute multigrid beginners. No programming skills necessary. Explains basic usage for standard problems.
  - The advanced tutorial: Chapters 6-11 meant for intermediate users of MueLu. Explains design concepts of MueLu and focuses on advanced use concepts.
  - Expert topics: Chapters 12-end cover expert topics primarily for developers.

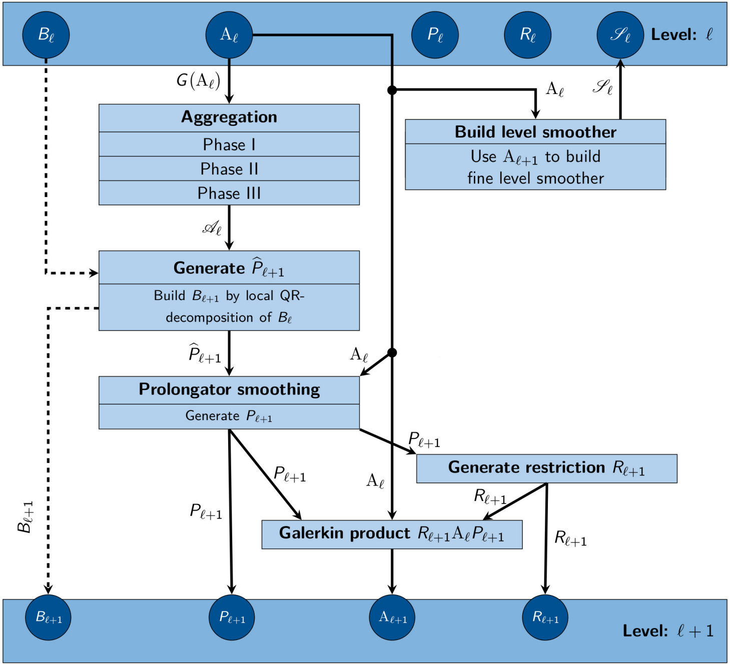
Conclusion

- MueMex: MATLAB interface for MueLu
- allows to use MueLu as solver within MATLAB
- provides easy access to MueLu internals for further analysis
- works also for complex problems and multiple RHS
- ideal tool for research in context of multigrid
  - no C++ knowledge necessary
  - rapid development (no compilation necessary)
  - perfect tool for quick experiments and parameter studies
- MueMex is still under heavy development

One more thing. . .

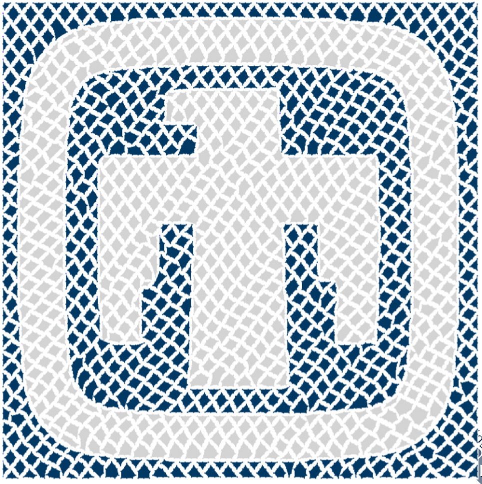
MueMex — MATLAB extensions for MueLu

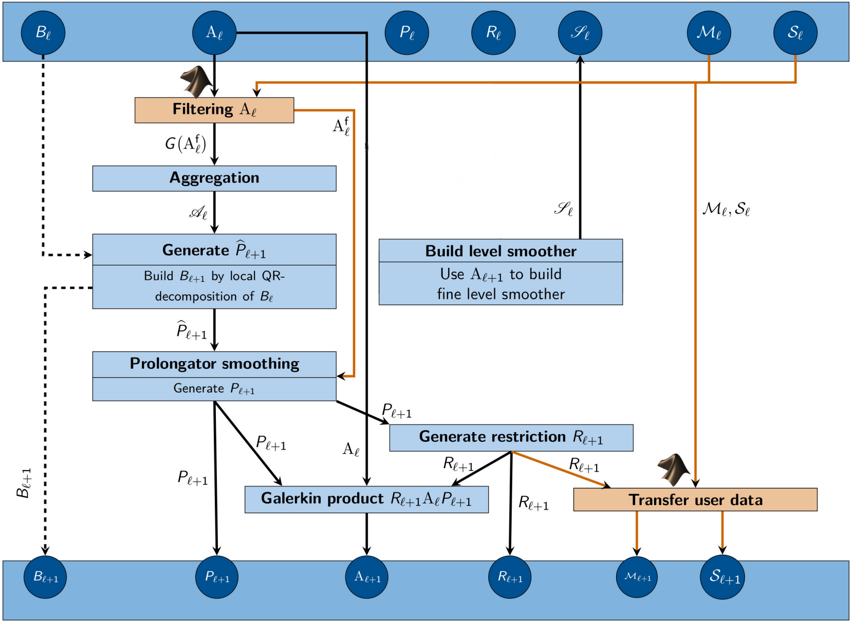
- Advanced software design principles of MueLu
  - Flexibility through modularity: multigrid framework
  - Strict splitting of algorithms and data
  - Algorithms are implemented in *factories* which use some input data to calculate/generate some other output data
- MueMex fully integrates in MueLu framework
  - Use callback mechanism to allow to plug in new factories in MueLu written in MATLAB
  - MueMex factories have full access to MueLu framework
  - MueMex factories can use full power of MATLAB


B_ℓ
A_ℓ
P_ℓ
R_ℓ
$\mathscr{S}_\ell$
Level: ℓ
$G(\mathrm{A}_\ell)$
A_ℓ
$\mathscr{S}_\ell$
Aggregation
Phase I
Phase II
Phase III
Build level smoother
Use $\mathrm{A}_{\ell+1}$ to build
fine level smoother
$\mathscr{A}_\ell$
Generate $\widehat{P}_{\ell+1}$
Build $B_{\ell+1}$ by local QR-
decomposition of B_ℓ
$\widehat{P}_{\ell+1}$
A_ℓ
Prolongator smoothing
Generate $P_{\ell+1}$
$P_{\ell+1}$
Generate restriction $R_{\ell+1}$
$P_{\ell+1}$
A_ℓ
$R_{\ell+1}$
$B_{\ell+1}$
$P_{\ell+1}$
Galerkin product $R_{\ell+1}\mathrm{A}_\ell P_{\ell+1}$
$R_{\ell+1}$
$B_{\ell+1}$
$P_{\ell+1}$
$\mathrm{A}_{\ell+1}$
$R_{\ell+1}$
Level: $\ell+1$

Demonstration

Use monochrome picture data to drop entries in input graph of aggregation factory to enforce user-specified aggregates:





B_ℓ
A_ℓ
P_ℓ
R_ℓ
$\mathscr{S}_\ell$
$\mathcal{M}_\ell$
$\mathcal{S}_\ell$
Filtering A_ℓ
A_ℓ^f
$G(A_\ell^f)$
Aggregation
$\mathscr{A}_\ell$
Generate $\widehat{P}_{\ell+1}$
Build $B_{\ell+1}$ by local QR-decomposition of B_ℓ
$\widehat{P}_{\ell+1}$
Prolongator smoothing
Generate $P_{\ell+1}$
Build level smoother
Use $A_{\ell+1}$ to build fine level smoother
$\mathscr{S}_\ell$
$\mathcal{M}_\ell, \mathcal{S}_\ell$
$P_{\ell+1}$
Generate restriction $R_{\ell+1}$
$P_{\ell+1}$
A_ℓ
$R_{\ell+1}$
$R_{\ell+1}$
$P_{\ell+1}$
$B_{\ell+1}$
Galerkin product $R_{\ell+1}A_\ell P_{\ell+1}$
$R_{\ell+1}$
Transfer user data
$B_{\ell+1}$
$P_{\ell+1}$
$A_{\ell+1}$
$R_{\ell+1}$
$\mathcal{M}_{\ell+1}$
$\mathcal{S}_{\ell+1}$

Demonstration

- Write a factory in MATLAB which uses
  - the non-filtered matrix A as input
  - drops all off-diagonal entries in A which represent connections between color 1 (e.g. blue) and color 2 (e.g. white).
  - stores the filtered matrix A for being used in aggregation algorithm
- transfer monochrome picture data accordingly to coarse level (this is optional if only two-level method is used)
- XML file controls the interconnection and dependencies of factories
  - MATLAB factories fully integrate in existing MueLu framework with all factories written in C++/MATLAB
  - optimal flexibility
  - recombination and reuse of factories without recompilation of source code
  - perfect tool to design new application-specific preconditioning strategies

One last thing. . .

Thanks

Thank you for your attention

Special thanks go to

- the MueMex developers, especially
  - Brian Kelley
  - Chris Siefert
- the MueLu developers, especially
  - Jonathan Hu
  - Andrey Prokopenko
  - Jeremie Gaidamour
- the developers of
  - Amesos and Amesos2
  - Ifpack and Ifpack2
  - Zoltan and Zoltan2
  - Epetra and Tpetra
- all Trilinos developers in general