Manual

Meshless Multi–Physics Software MeMPhyS

Version 2.1
Date: 1 July 2021

Developed By

Dr. Shantanu Shahane
sshahan2@illinois.edu
Prof. Surya Pratap Vanka
spvanka@illinois.edu

University of Illinois at Urbana–Champaign, USA

Website: github.com/shahaneshantanu/memphys

Contents

1 Installation **3**

2 Setup **4**

3 Software Capabilities **5**
3.1 Multi–Physics Components 5
3.2 Parameters Setting 6
3.3 List of Current Examples 7

4 List of Publications **9**

5 Details of Current Version **10**
5.1 Modifications 10
5.2 Bugs fixed 10

6 Details of Previous Versions **11**
6.1 Version 2.0: 13 June 2021 11
6.1.1 Modifications 11
6.1.2 Bugs fixed 11
6.2 Version 1.4: 20 January 2021 11
6.2.1 Modifications 11
6.2.2 Bugs fixed 11
6.3 Version 1.3: 4 December 2020 11
6.3.1 Modifications 11
6.3.2 Bugs fixed 12
6.4 Version 1.2: 29 November 2020 12
6.4.1 Modifications 12
6.4.2 Bugs fixed 12
6.5 Version 1.1: 15 November 2020 12
6.5.1 Modifications 12
6.5.2 Bugs fixed 12
6.6 Version 1.0: 8 November 2020 13

Chapter 1. Installation

Install the following software (links to websites provided in blue colors):

1. Download the latest version of MeMPhyS from GitHub
2. Latest version of Ubuntu LTS:
   - Tested on Ubuntu 20.04.1 LTS
   - May work on other Linux distributions and versions
3. Install and test MPI: example
4. HYPRE Solver:
   - Tested on Version 2.11.1
   - Steps provided in 'INSTALL' file in HYPRE directory to 'make install' its code files
   - Edit the variable 'HYPRE_DIR' in 'Makefile' to specify location of the HYPRE directory
5. Eigen Library:
   - Tested on Version 3.3.7
   - No need to install; extract the source files to a directory
   - Edit the variable 'Eigen_directory' in 'Makefile' and point to above directory
6. Spectra Library:
   - Tested on Version 0.8.1
   - No need to install; extract the source files to a directory
   - Edit the variable 'spectra_directory' in 'Makefile' and point to above directory
7. Mesh files generated using GMSH.
   - Install latest version of GMSH with these instructions:
   - MSH file should be exported with *version 2* (not default setting of 4)
8. Visualization of contours using Tecplot. Download from UIUC webstore.
9. Install Visual Studio Code for code development/editing

All the above software other than Tecplot are free or open source.

Chapter 2. Setup

Steps for setting new problems:

1. Make a copy of any '.cpp' files in the examples folder with a new name say, 'filename.cpp'
2. Generate msh file from GMSH and link it in 'filename.cpp'
3. MSH file should be exported with *version 2* (not default setting of 4)
4. Set appropriate boundary conditions inside the code 'filename.cpp'
5. Add four lines for compiling 'filename.cpp' in the 'Makefile'; for example:
   filename.o : filename.cpp ../../header_files/class.hpp ../../header_files/general_functions.hpp
   $(CC) $(CFLAGS) $(LFLAGS) $(LIBS) -I $(INCLUDE) filename.cpp
   filename : $(OBJS) filename.o
   $(CC) $(PROFLAGS) -o out filename.o $(OBJS) $(LIBS)
6. Update 'parameters_file.csv' if needed
7. Compile with command: **time make filename**
8. Execute with command: **time ./out**
9. Use command **make clean** occasionally to clear all object files
10. Use command **make rm_sim_files** to clear all csv and plt files

Chapter 3. Software Capabilities

3.1 Multi–Physics Components

MeMPhyS solves partial differential equations arising from various engineering and physics problems. It uses the Polyharmonic spline radial basis function with appended polynomials to estimate the differential operators over point clouds. Current capabilities include:

- Interpolation over scattered points
- Gradient and Laplacian stencils for 2D and 3D geometries
- Heat conduction (Poisson's equation) with Dirichlet, Neumann and periodic boundary conditions
- Scalar transport equation
- Incompressible fluid flow problems in 2D (Navier–Stokes equations)
- Forced convection problems
- Types of boundary conditions for fluid flow problems:
    1. Wall boundary: no slip and no penetration
    2. Inlet boundary
    3. Symmetry boundary
    4. Outlet boundary: prescried uniform pressure
    5. Outlet boundary: normal derivatives of velocities set to zero
    6. Periodic condition
- Fluid flow problems can be solved by two algorithms:
    1. Fractional step algorithm: restrictive timestep with Courant number less than unity; useful for problems with fast transients; does not iterate at a particular timestep
    2. Semi-implicit algorithm: larger timesteps allowed with Courant number over unity (recommended value: 5 to 20); useful for problems with slow transients or steady state; iterations required at each timestep
- Solidification of metal alloys for 2D and 3D geometries

3.2 Parameters Setting

The accuracy and efficiency of MeMPhyS can be tuned by setting appropriate values in the 'parameters_file.csv':

- **poly_deg**: degree of appended polynomial: integer in range [2, 15]
  A degree of k is expected to give $\mathcal{O}(k+1)$, $\mathcal{O}(k)$ and $\mathcal{O}(k-1)$ convergence in the numerical interpolation, gradient and Laplacian operators respectively. However, condition number of the radial basis function matrix increases with higher degree. This also increases the bandwidth of the assembled solution matrix thus, making a higher order solution more expensive.

- **phs_deg**: exponent of polyharmonic spline: integer with options: 3, 5, 7, 9, 11
  A value of '3' is recommended. This has minimal impact on the condition number and stability of the system.

- **cloud_size_multiplier**: fraction in the range [1.5, 2.5]
  It is recommended to set this around '2.0'. Higher value increases the bandwidth and slows down convergence.
  Usage: cloud_size = cloud_size_multiplier*(No. of appended polynomials)

- **nt**: No. of timesteps: positive integer

- **Courant**: courant number (CFL criterion)
  Set in range [0,1] for explicit time marching schemes. Implicit Euler is unconditionally stable
  Set in range [5,20] for semi-implicit fluid flow algorithm
  Usage: [dt=2*Courant/lambda] where, lambda: sum of largest magnitude convection and diffusion eigenvalues

- **solver_type**: options: hypre_ilu_gmres, eigen_direct, eigen_ilu_bicgstab
  hypre_ilu_gmres: ILU preconditioned GMRES from the library HYPRE
  eigen_direct direct solver from the library Eigen
  eigen_ilu_bicgstab: ILU preconditioned BiCGSTAB from the library Eigen
  eigen_direct is not recommended for large size problems

- **steady_tolerance**: solved to steady state till absolute difference between consecutive timesteps for all variables is less than this tolerance. Useful only for problems having steady state solution.

- **solver_tolerance**: used in iterative solvers for absolute and relative residuals (recommended: less than steady_tolerance)

- **gmres_kdim**: dimension of Krylov subspace in GMRES solver (refer the HYPRE manual for details)

- **precond_droptol**: Drop tolerance for ILU preconditioner (refer the HYPRE manual or Eigen manual for details)

- **n_iter**: no. of iterations of the GMRES or BiCGSTAB solvers

3.3 List of Current Examples

The 'examples' folder has following problems setup under various sub–folders:

1. Problems involving single scalar transport under sub–folder 'advection_diffusion':
   (a) 'heat_conduction_manuf_soln.cpp': heat conduction using implicit (Euler) time–marching with manufactured solution in 2D and 3D
   (b) 'heat_conduction_manuf_soln_periodic.cpp': heat conduction using implicit (Euler) time–marching with a periodic manufactured solution in 2D and 3D
   (c) 'heat_conduction.cpp': heat conduction using implicit (Euler) time–marching on general geometries in 2D and 3D
   (d) 'heat_conduction_periodic.cpp': heat conduction using implicit (Euler) time–marching on general geometries in 2D and 3D with a periodic boundary condition
   (e) 'euler_steady_heat_cond_manuf_sol.cpp': heat conduction using explicit (Euler) time–marching with steady manufactured solution in 2D and 3D
   (f) 'euler_unsteady_heat_cond_manuf_sol.cpp': heat conduction using explicit (Euler) time–marching with unsteady manufactured solution in 2D and 3D
   (g) 'ab2_unsteady_heat_cond_manuf_sol.cpp': heat conduction using explicit (second order Adam–Bashforth) time–marching with unsteady manufactured solution in 2D and 3D
   (h) 'scalar_transport_implicit.cpp': scalar transport using implicit (Euler) time–marching on general geometries in 2D and 3D
2. Two–dimensional fluid flow problems under sub–folder 'navier_stokes_2D':
   (a) 'channel_flow.cpp': developing flow in a channel
   (b) 'channel_flow_periodic.cpp': fuly developed flow in a channel with periodicity
   (c) 'couette_flow.cpp': Couette flow to analyze spatial errors in solution
   (d) 'eccentric_couette_flow.cpp': Couette flow with eccentric inner cylinder
   (e) 'elliptic_couette_flow.cpp': Couette flow with outer cylinder as ellipse
   (f) 'kovasznay_flow.cpp': Kovasznay flow to analyze spatial errors in solution
   (g) 'flow_over_cylinder.cpp': steady and unsteady flow over a circular cylinder
   (h) 'couette_2_cylinders_ellipse.cpp': Couette flow in the gap between two rotating cylinders in an elliptical enclosure
   (i) 'taylor_green_steady.cpp': steady version of Taylor–Green flow to analyze spatial errors in solution

(j) ‘taylor_green_modified.cpp’: modified Taylor–Green flow to analyze temporal errors in solution (second order Adam–Bashforth)

(k) ‘bell_colella_JCP.cpp’: Initial velocity decaying to zero in time

(l) ‘bell_colella_JCP_periodic.cpp’: solution of Euler equations in a double periodic square geometry resulting into multiple rolling vortices

(m) ‘bellow_periodic.cpp’: Flow in a sinusoidal bellow with periodicity in the flow direction

3. Miscellaneous problems under sub–folder ‘miscellaneous’:

   (a) ‘interpolation.cpp’: analyze error of interpolation stencil for 2D and 3D problems

4. Solidification problem under sub–folder ‘solidification’:

   (a) ‘solidification_problem_1.cpp’: 2D and 3D solidification of Aluminum alloy

Chapter 4. List of Publications

If you use MeMPhyS, please cite the following research papers:

1. N Bartwal, S Shahane, S Roy, SP Vanka (2021). Application of a High Order Accurate Meshless Method to Solution of Heat Conduction in Complex Geometries: arXiv

2. S Shahane, SP Vanka (2021). A Semi-Implicit Meshless Method for Incompressible Flows in Complex Geometries: arXiv

3. A Radhakrishnan, M Xu, S Shahane, SP Vanka (2021). A Non-Nested Multilevel Method for Meshless Solution of the Poisson Equation in Heat Transfer and Fluid Flow: arXiv

4. S Shahane, A Radhakrishnan, SP Vanka (2020). A High-Order Accurate Meshless Method for Solution of Incompressible Fluid Flow Problems: arXiv

Chapter 5. Details of Current Version

5.1 Modifications

- Added Python scripts for generation of gradient and Laplacian coefficients
- Added sample GMSH geometry and mesh files for examples

5.2 Bugs fixed

- No bugs reported/fixed

Chapter 6. Details of Previous Versions

6.1 Version 2.0: 13 June 2021

6.1.1 Modifications

- Semi-implicit iterative algorithm developed for solution of Navier-Stokes equations
- Implicit solver for scalar transport equation
- Periodic boundary condition implemented
- Euler equations solved with hyper-viscosity source terms
- Generalized Tecplot writing functions

6.1.2 Bugs fixed

- No bugs reported/fixed

6.2 Version 1.4: 20 January 2021

6.2.1 Modifications

- All main files of various problems categorized inside the 'examples' directory
- Setup of solidification problem
- For Navier–stokes problems: option for 2 types of outlet BCs:
  - Fixed uniform pressure
  - $\frac{\partial u}{\partial \hat{\boldsymbol{n}}} = 0$ and $\frac{\partial v}{\partial \hat{\boldsymbol{n}}} = 0$

6.2.2 Bugs fixed

- Corrected the nearest neighbor function to get sufficient neighbors for boundary points

6.3 Version 1.3: 4 December 2020

6.3.1 Modifications

- Fractional step algorithm with Adam–Bashforth giving second order temporal accuracy

- Outlet boundary implemented with Dirichlet condition for pressure
- Efficient cloud computation by coupling with open source library Nanoflann

6.3.2 Bugs fixed

- No bugs reported/fixed

6.4 Version 1.2: 29 November 2020

6.4.1 Modifications

- Function for sparse matrix to compute spectrally accurate interpolation
- Modularized fractional step function into class
- Added optional body force terms in the fractional step algorithm
- Main files for steady and unsteady Taylor–Green vortices
- Explicit Euler and 2^{nd} Order Adam-Bashforth giving 1^{st} and 2^{nd} order temporal accuracies respectively for 2D unsteady heat conduction problems

6.4.2 Bugs fixed

- No bugs reported/fixed

6.5 Version 1.1: 15 November 2020

6.5.1 Modifications

- Added an option to choose ILU preconditioned BiCGSTAB solver implemented in the Eigen library
- For 2D problems till 10000 points, performance similar to ILU+GMRES from HYPRE
- For refined grids in 3D Eigen's solver is computationally efficient

6.5.2 Bugs fixed

- No bugs reported/fixed

6.6 Version 1.0: 8 November 2020

- Works on 2D (triangular elements) and 3D (tetrahedral elements) from GMSH grids
- Exponential convergence based on the degree of appended polynomial
- Direct solver from Eigen used to calculate gradient and Laplacian coefficients
- Preconditioned GMRES from HYPRE or direct solver from Eigen are available for solution of the assembled linearized system
- Tested with manufactured solutions of heat conduction and scalar transport
- Fractional step algorithm with explicit convection and diffusion terms used to solve the Navier–Stokes equations
- As of now, following 2D fluid flow problems are setup:
    1. Kovasznay flow
    2. Couette flow (concentric rotating circle inside circle)
    3. Eccentric Couette flow (eccentric rotating circle inside circle)
    4. Elliptical Couette flow (concentric rotating circle inside ellipse)
    5. Steady and unsteady flow over circular cylinder
- Steps for setting new problems:
    1. Make a copy of the file 'flow_over_cylinder.cpp' with a new name say, 'filename.cpp'
    2. Generate msh file from GMSH and link it in 'filename.cpp'
    3. MSH file should be exported with *version 2* (not default setting of 4)
    4. Set appropriate boundary conditions inside the code 'filename.cpp'
    5. Add two lines for compiling 'filename.cpp' in the 'Makefile' (similar to the lines for 'flow_over_cylinder.cpp')
    6. Update 'parameters_file.csv' if needed
    7. Compile with command: **time make filename**
    8. Execute with command: **time ./out**
    9. Use command **make clean** occasionally to clear all object files