Turbulent Flow Simulation on HPC-Systems

Simulation of a Flow over a Backward Facing Step

Check-pointing and PETSc-Solvers

Group 3:

Mahyar Valizadeh, Jonas Buchmeier,

Stefan Gavranovic, Ajay Karthik

Overview and Team Management

- Assigning the each member's task
  - Verification and physical background
  - Checkpointing
  - Examining petsc solvers
- Group meetings and discussion
- Final conclusion and presentation

Backward Facing Step – Why is it Important test case?

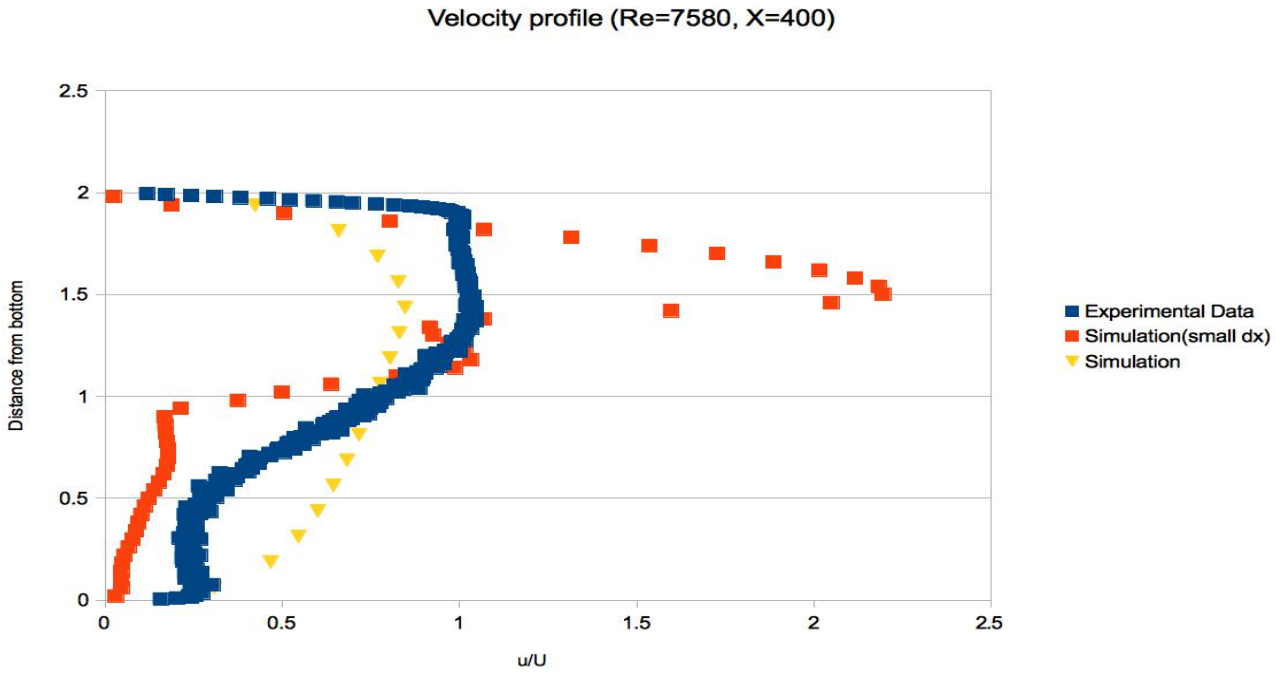
- Massively separated flow (defined separation point)
- Boundary Layer Flow
- Fully-detached mixing layer

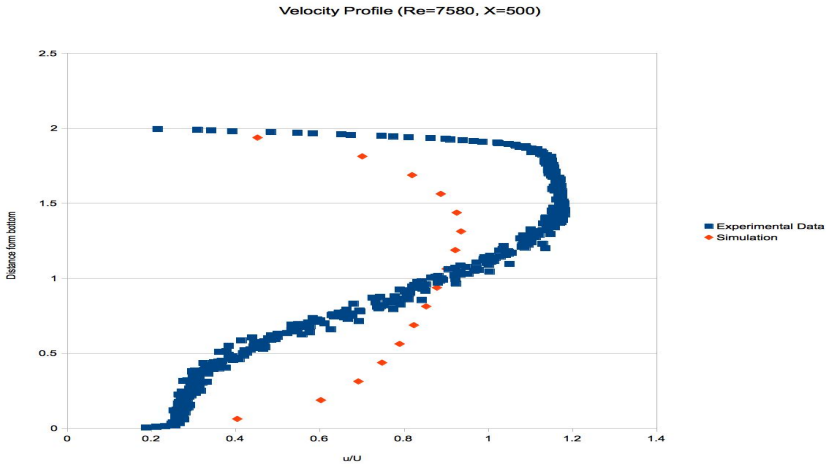
➔A need for a appropriate model which can handle a non-parallel flow with separation

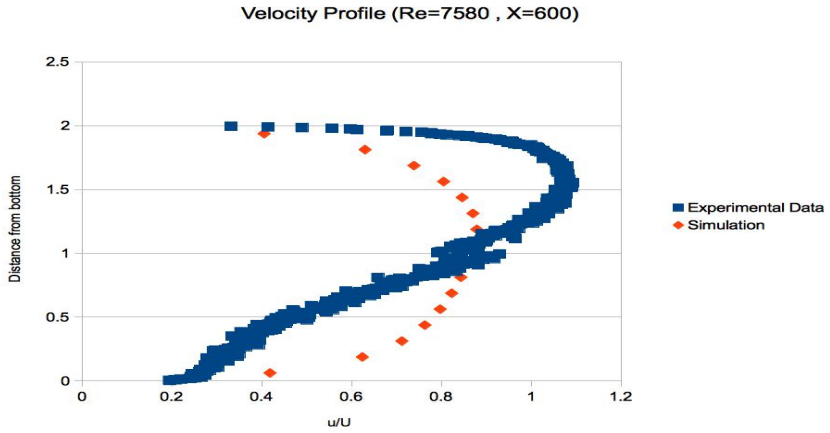
Verification

- Similarity
  - geometric
  - Kinematic
  - and Dynamic
- Dimensionless analysis
- Effective Dimensionless numbers

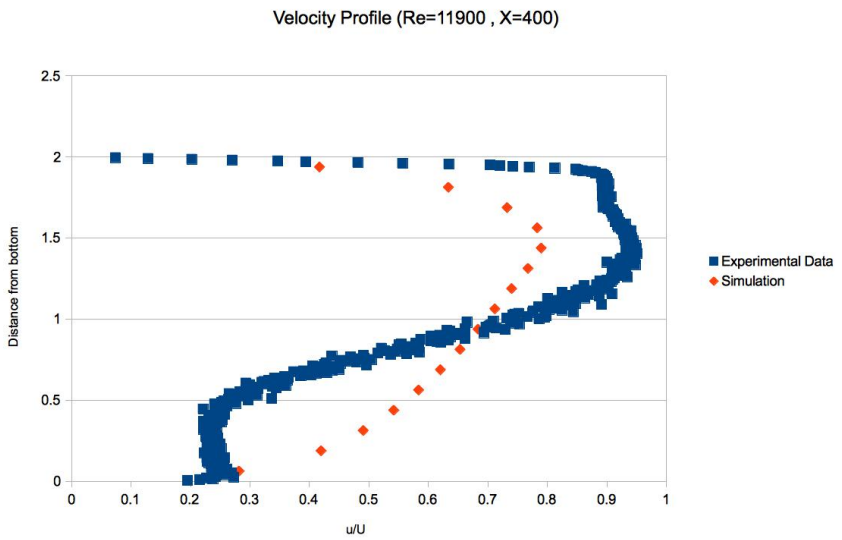
Verification (Velocity Profiles)

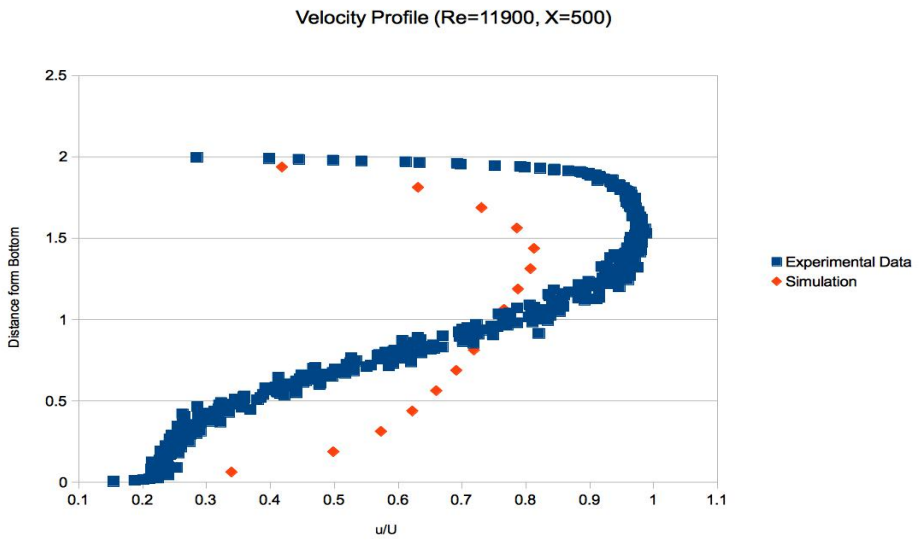


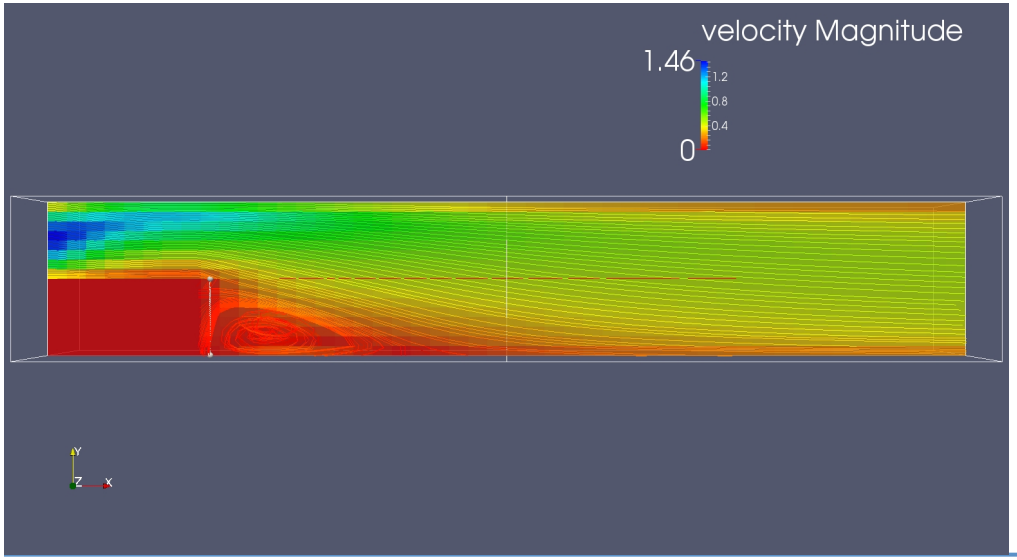


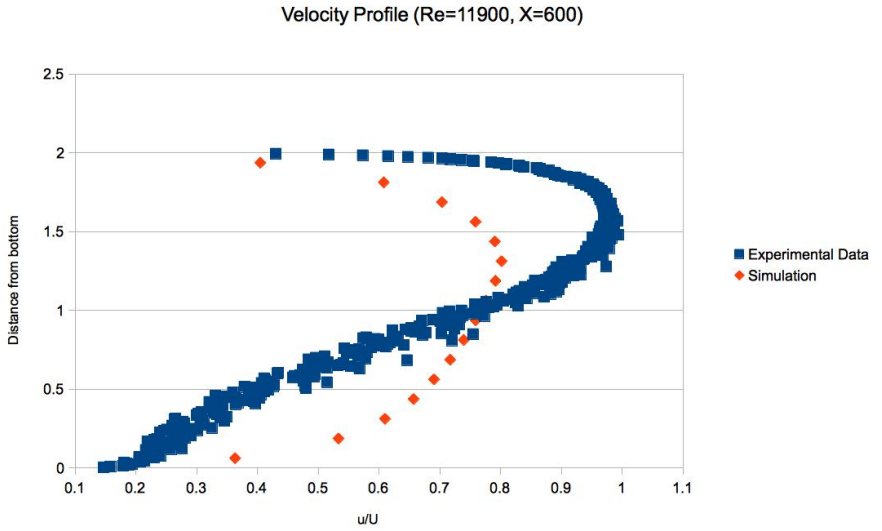


Verification (Continued)

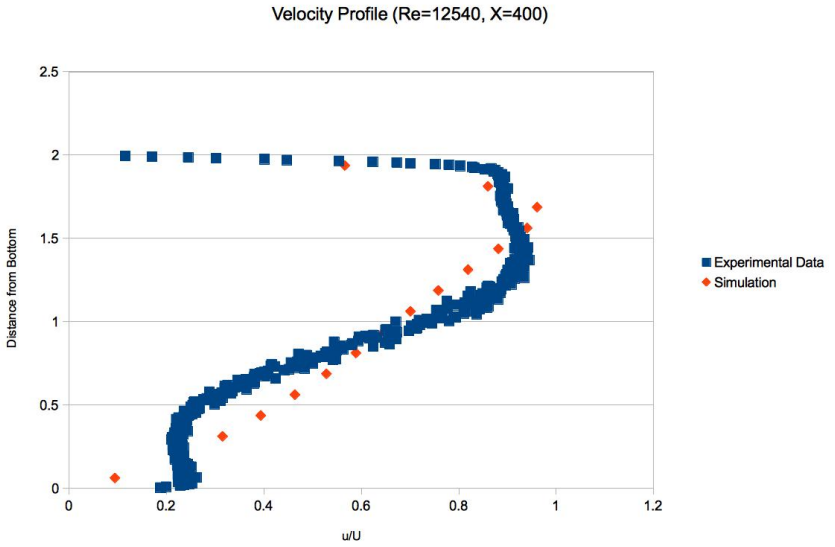


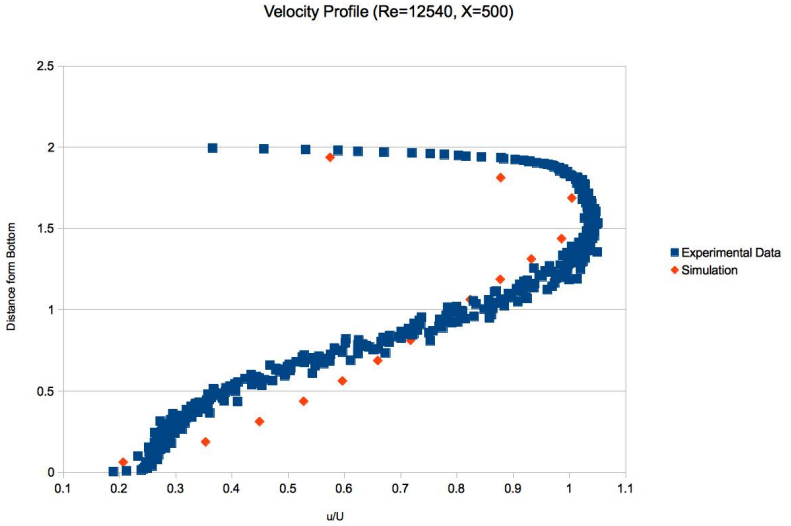


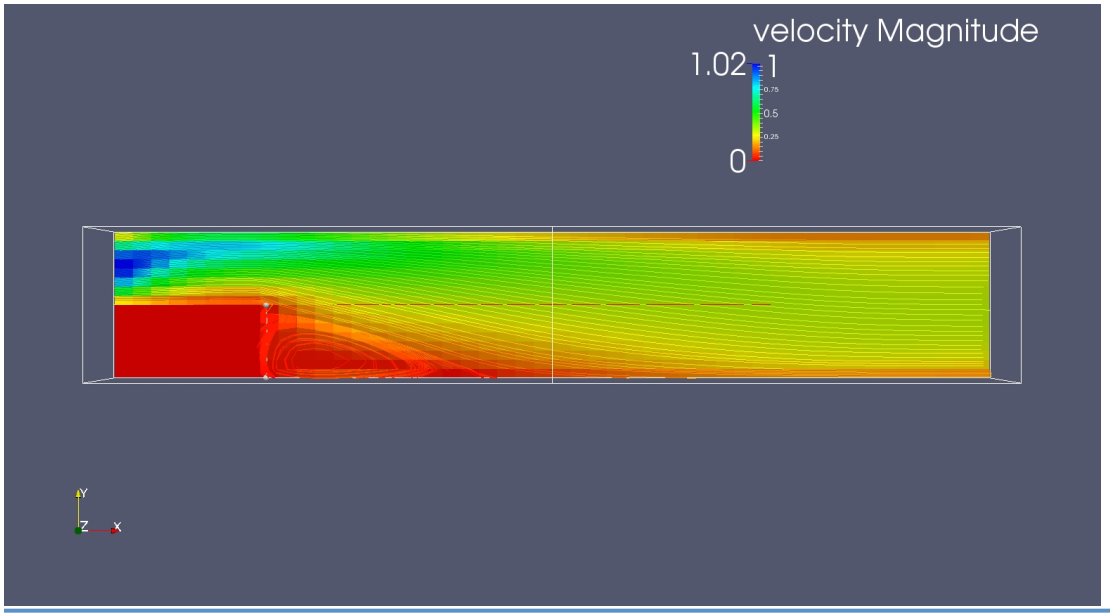


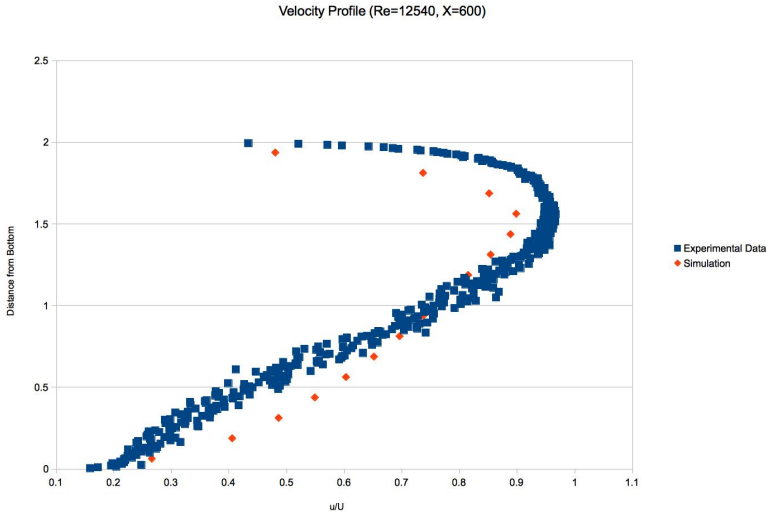


Verification (Continued)



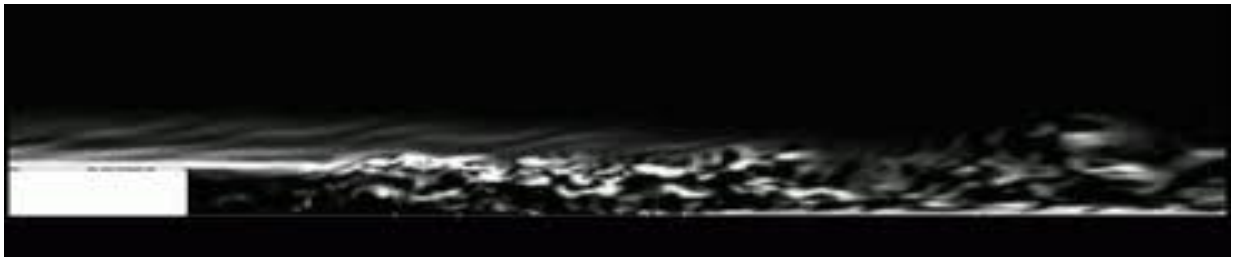






Conclusion on the Mixing length Model

- Questionable in separated flow
- Prescription of the mixing length becomes problematic in flows that are not approximately parallel, thin shear layers
- BFS: 2 shear layers at any x within the separated region. Detached mixing layer, bottom-wall boundary layer



Turbulent Flow Simulation on HPC-Systems

- MPI I/O -

GROUP 3

TU Munich

Common Way of Implementing I/O in Parallel Programs

- Sequential way illustrated:
  - All processes send data to one master process which writes it to the file.
  - Lack of parallelism limits scalability and performance

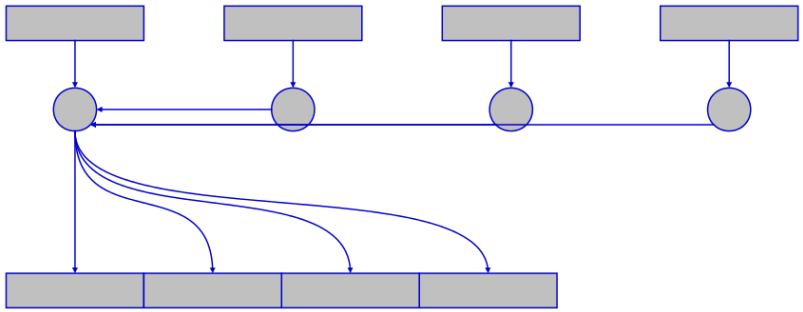


Figure: Master Process

Shared file

- Shared File
  - Each process performs I/O to a single file which is shared.
- Performance
  - Data layout within the shared file is very important.
  - At large process counts contention can build for file system resources.

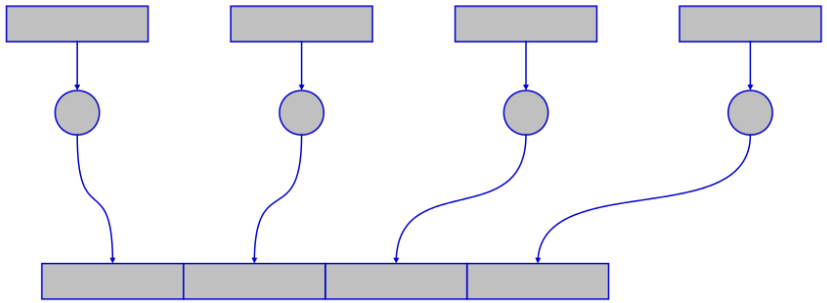


Figure: Shared file

MPI Calls

- MPI call for opening one single shared file.
- With Arguments for creating file if it is nor already created.
- Returns pointer to the created file.

```
MPI_File_open( MPI_COMM_WORLD, "testfile",
MPI_MODE_CREATE | MPI_MODE_WRONLY,
MPI_INFO_NULL, &thefile);
```

MPI Calls

- MPI call for setting the offsets in the file.
- Assigns regions of the file to the processors with respect to the desired offset.

```
MPI_File_set_view( thefile, myrank * offset *
sizeof(int),
MPI_INT, MPI_INT, "native",
MPI_INFO_NULL);
```

MPI Calls

- MPI call reading from the file.
- If call **MPI_File_set_view** precedes then every process is able to read from its own assigned part of the file.

```
MPI_File_read(fh, buffer, count, MPI_INT, &status);
```

MPI Calls

- MPI call writing into the file.
- If call **MPI_File_set_view** precedes then every process is able to write into its own part of the file.

```
MPI_File_read(fh, buffer, count, MPI_INT, &status);
```

Petsc Introduction

- Petsc as a package, provides us with adapting our application with data structures, routines and also suitable scalable options.
- Features several linear solvers
  - Direct solvers: LU, Cholesky, QR...
  - Krylov methods: CG, BiCG, GMRES, BCGS..
  - Preconditioners: ILU, Jacobi, additive Schwarz, ...

Petsc Usage

- *DataStructures : KSP Object, PC Object.*
- Matrix structures, Preconditioner Matrix, re-using matrix structures.
- Best thing is to just use command line options and run time controls.

Current Application

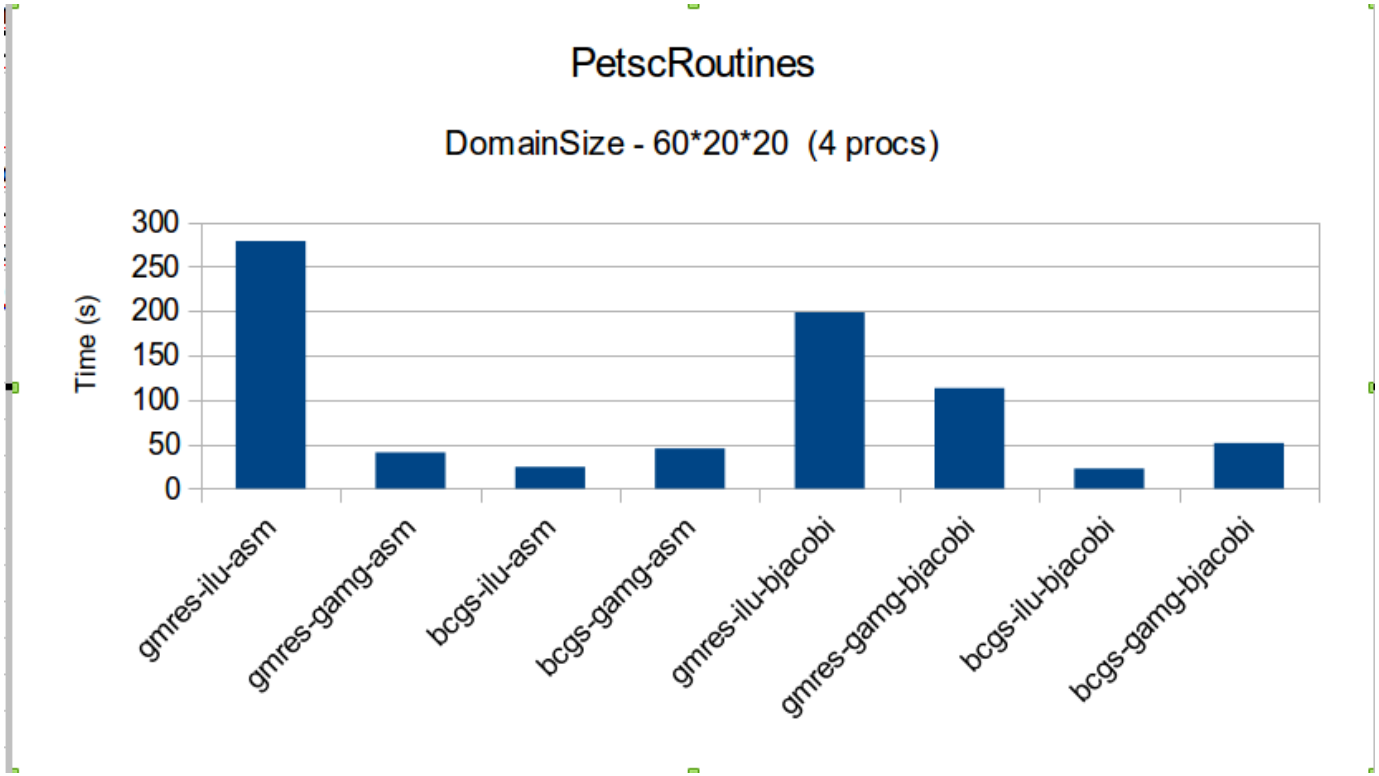


Figure: Petsc Solver Parameters

Thank you for the attention

Questions ?