

Final Project – HPC(8810)

Team Members:

Prashanth Reddy Kadire

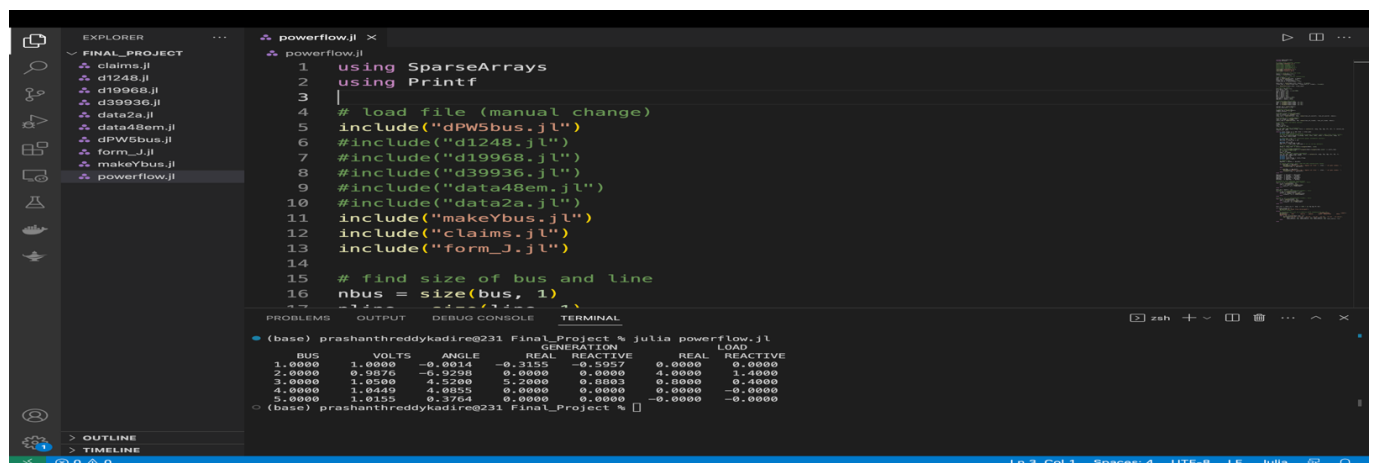
Tejasree Challagundla

We have chosen Julia language for converting Matlab code; there are several reasons to choose; some of them are Performance; Julia is designed to be fast, with Performance comparable to C and Fortran. It has a just-in-time (JIT) compiler that compiles code on-the-fly, optimizing it for the specific machine architecture. This makes Julia well-suited for numerical and scientific computing, where Performance is critical. The second one is easy to write code; thirdly, Julia was designed from the ground up to support parallelism and distributed computing. Its built-in support for parallelism makes writing code that runs on multiple cores or machines easy.

We have opted for the MPI API to parallelize the Julia code because the given code calculates power flow using the Newton-Raphson method. It updates the voltage magnitude and phase angle at each bus iteratively until convergence. Parallelization can be helpful when we have a large power system network with many buses and lines. As the power system network grows, the computational time also increases. To decrease the computational time, we can parallelize the code.

The MPI (Message Passing Interface) API is used to parallelize the Julia code because it is a widely used and standardized API for parallel programming. MPI allows for distributed memory parallelism and communication between processes running on different nodes. This is necessary for parallelizing the power flow calculation because the computation involves many matrix operations and requires communication between the different processes. MPI can be used to distribute the matrix operations and the communication between the different processes.

We have successfully converted Matlab code into Julia, and we got same results as sequential matlab code. **Sequential Julia Code Results:**



```
1 using SparseArrays
2 using Printf
3
4 # load file (manual change)
5 include("dPW5bus.jl")
6 #include("d1248.jl")
7 #include("d19968.jl")
8 #include("d39936.jl")
9 #include("data48em.jl")
10 #include("data2a.jl")
11 include("makeYbus.jl")
12 include("claims.jl")
13 include("form_J.jl")
14
15 # find size of bus and line
16 nbus = size(bus, 1)
```

BUS	VOLTS	ANGLE	GENERATION		LOAD	
			REAL	REACTIVE	REAL	REACTIVE
1.0000	1.0000	-0.0014	-0.3155	-0.5557	0.0000	0.0000
2.0000	0.9876	-6.5290	0.0000	0.0000	4.0000	1.4000
3.0000	1.0500	4.5200	0.2000	0.0000	0.0000	0.4000
4.0000	1.0449	4.0855	0.0000	0.0000	0.0000	-0.0000
5.0000	1.0152	0.3764	0.0000	0.0000	-0.0000	-0.0000

```
EXPLORER
FINAL_PROJECT
claims.jl
d1248.jl
d19968.jl
d39936.jl
data2a.jl
data48em.jl
dPW5bus.jl
form_3.jl
makeYbus.jl
powerflow.jl

powerflow.jl
1 using SparseArrays
2 using Printf
3
4 # load file (manual change)
5 #include("dPW5bus.jl")
6 #include("d1248.jl")
7 include("data2a.jl")
8 #include("data48em.jl")
9 #include("d19968.jl")
10 #include("d39936.jl")
11
12 include("makeYbus.jl")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(base) prashanthreddykadireg231 Final_Project % julia powerflow.jl
0.5 constant current load, svc at bus 101

BUS VOLTS ANGLE GENERATION REAL REACTIVE LOAD REAL REACTIVE
1.0000 1.0380 0.0000 7.1563 2.9211 0.0000 0.0000
2.0000 1.0180 -10.4470 7.0000 4.8232 0.0000 0.0000
3.0000 0.8889 -20.9258 0.0000 0.0000 0.0000 0.0000
4.0000 0.8882 -31.6625 0.0000 0.0000 12.7690 1.0000
20.0000 0.9895 -6.7342 0.0000 0.0000 -0.0000 -0.0000
101.0000 1.0380 -7.7005 7.1000 3.1011 0.0000 0.0000
110.0000 1.0100 -10.2812 7.0000 5.2527 0.0000 0.0000
120.0000 0.8764 -34.9869 0.0000 0.0000 -0.0000 0.0000
14.0000 0.8665 -40.5039 0.0000 0.0000 14.7400 1.0000
20.0000 0.9374 -17.5403 0.0000 0.0000 -0.0000 -0.0000
101.0000 0.8889 -30.8090 0.0000 0.0000 0.0000 -0.0000
110.0000 0.9866 -14.5261 0.0000 0.0000 0.0000 0.0000
120.0000 0.9384 -25.4275 0.0000 0.0000 0.0000 0.0000
(base) prashanthreddykadireg231 Final_Project %
```

```
EXPLORER
FINAL_PROJECT
claims.jl
d1248.jl
d19968.jl
d39936.jl
data2a.jl
data48em.jl
dPW5bus.jl
form_3.jl
makeYbus.jl
powerflow.jl

powerflow.jl
2 using Printf
3
4 # load file (manual change)
5 #include("dPW5bus.jl")
6 #include("d1248.jl")
7 #include("data2a.jl")
8 include("data48em.jl")
9 #include("d19968.jl")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(base) prashanthreddykadireg231 Final_Project % julia powerflow.jl
There are Qg > Qgmax at iter 1 at gen index: [78, 95, 96, 124, 127, 128, 129, 131, 132]
There are Qg > Qgmax at iter 2 at gen index: [78, 95, 96, 124, 127, 128, 129, 131, 132]
There are Qg > Qgmax at iter 3 at gen index: [78, 95, 96, 124, 127, 128, 129, 131, 132]
There are Qg > Qgmax at iter 4 at gen index: [78, 95, 96, 124, 127, 128, 129, 131, 132]
There are Qg > Qgmax at iter 5 at gen index: [78, 95, 96, 124, 127, 128, 129, 131, 132]

BUS VOLTS ANGLE GENERATION REAL REACTIVE LOAD REAL REACTIVE
1.0000 1.0111 4.9591 0.0000 0.0000 -0.0000 0.0000
2.0000 1.0067 4.1701 0.0000 0.0000 -0.0000 -0.0000
3.0000 1.0470 4.1249 0.0000 0.0000 0.0000 0.8800
4.0000 1.0076 4.2181 0.0000 0.0000 0.0000 0.0000
5.0000 1.0038 2.4535 0.0000 0.0000 -0.0000 -0.0000
6.0000 1.0070 1.9431 0.0000 0.0000 3.2000 1.5300
7.0000 1.0233 3.3347 0.0000 0.0000 3.2000 0.3200
8.0000 1.0225 2.0794 0.0000 0.0000 0.0000 0.0000
9.0000 1.0182 1.1657 0.0000 0.0000 1.5000 0.3000
10.0000 1.0420 7.9779 0.0000 0.0000 -0.0000 0.0000
11.0000 0.9841 6.5027 0.0000 0.0000 0.0000 1.0300
12.0000 1.0251 6.0000 0.0000 0.0000 2.7400 1.1500
13.0000 1.0450 10.7331 0.0000 0.0000 -0.0000 0.0000
14.0000 1.0400 10.4400 0.0000 0.0000 2.4000 0.0000
15.0000 1.0292 3.5000 0.0000 0.0000 3.0900 -0.9200
16.0000 1.0394 4.5690 0.0000 0.0000 2.2400 0.4700
17.0000 1.0390 3.3620 0.0000 0.0000 1.3000 0.1700
18.0000 1.0260 1.5910 0.0000 0.0000 2.8100 0.7600
19.0000 1.0400 6.5557 0.0000 0.0000 2.0600 0.2000
20.0000 1.0416 0.2361 0.0000 0.0000 2.0400 0.2700
21.0000 0.9800 13.0392 6.5000 2.1770 0.0000 0.0000
(base) prashanthreddykadireg231 Final_Project %
```

```
EXPLORER
FINAL_PROJECT
claims.jl
d1248.jl
d19968.jl
d39936.jl
data2a.jl
data48em.jl
dPW5bus.jl
form_3.jl
makeYbus.jl
powerflow.jl

powerflow.jl
3
4 # load file (manual change)
5 #include("dPW5bus.jl")
6 include("d1248.jl")
7 #include("data2a.jl")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1215.0000 0.9972 -8.6146 0.0000 0.0000 0.0000 0.0000
1216.0000 0.9415 -10.7562 0.0000 0.0000 2.3380 8.4000
1217.0000 0.9953 -11.3664 0.0000 0.0000 5.2200 1.7600
1218.0000 1.0114 -11.1760 0.0000 0.0000 0.0000 0.0000
1219.0000 1.0092 -6.2043 0.0000 0.0000 -0.0000 -0.0000
1220.0000 1.0039 -7.0250 0.0000 0.0000 -0.0000 -0.0000
1221.0000 0.9913 -7.0432 0.0000 0.0000 0.0000 0.0000
1222.0000 1.0058 -6.9289 0.0000 0.0000 -0.0000 -0.0000
1223.0000 1.0020 -8.6263 0.0000 0.0000 -0.0000 -0.0000
1224.0000 0.9953 -9.0135 0.0000 0.0000 3.2000 1.5300
1225.0000 1.0076 -7.5247 0.0000 0.0000 3.2940 3.2300
1226.0000 1.0160 -8.5928 0.0000 0.0000 0.0000 0.0000
1227.0000 1.0170 -9.4726 0.0000 0.0000 1.5800 0.3000
1228.0000 1.0409 -2.8203 0.0000 0.0000 0.0000 -0.0000
1229.0000 0.9860 -4.2626 0.0000 0.0000 6.8000 1.0300
1230.0000 1.0147 -5.0446 0.0000 0.0000 2.7400 1.1500
1231.0000 1.0407 -0.5070 0.0000 0.0000 0.0000 -0.0000
1232.0000 1.0353 -0.7082 0.0000 0.0000 2.4750 0.8460
1233.0000 1.0152 -7.4056 0.0000 0.0000 3.0860 -0.9220
1234.0000 1.0540 -5.5071 0.0000 0.0000 2.2400 0.4720
1235.0000 1.0450 -6.7494 0.0000 0.0000 1.3000 0.1700
1236.0000 1.0257 -8.7687 0.0000 0.0000 2.8100 0.7550
1237.0000 1.0488 -3.2346 0.0000 0.0000 2.0600 0.2760
1238.0000 1.0505 -0.4817 0.0000 1.2902 2.8350 1.2600
1239.0000 1.0470 -4.4304 2.5000 1.6707 0.0000 0.0000
1240.0000 1.0400 -1.5893 5.7293 7.7131 0.0920 0.0460
1241.0000 0.9831 1.8549 6.5000 2.4184 0.0000 0.0000
1242.0000 0.9972 -9.4177 6.3200 1.6851 0.0000 0.0000
1243.0000 1.0123 0.9393 5.0000 1.9459 0.0000 0.0000
1244.0000 1.0493 4.4984 6.5000 2.7801 0.0000 0.0000
1245.0000 1.0035 7.2070 5.0000 1.3038 0.0000 0.0000
1246.0000 1.0278 -1.2956 5.4000 0.1013 0.0000 0.0000
1247.0000 1.0265 6.5795 8.3000 0.1926 0.0000 0.0000
1248.0000 1.0300 -10.9600 10.0529 1.6253 11.0400 2.5000
(base) prashanthreddykadireg231 Final_Project %
```

We tried to parallelize the sequential julia code using MPI. The steps followed are

Intialize MPI environment by calling MPI.Init().

- Get the total number of processes and rank of the current process using MPI.Comm_size() and MPI.Comm_rank() respectively.
- Load the power system data files only in process 0.
- Use MPI.Barrier() to synchronize all processes.
- Broadcast data (bus and line data) to all processes using MPI.Bcast().
- Convert external bus indices to internal bus indices using a sparse vector.
- Calculate the Y bus matrix.
- Read bus data.
- Set up arrays for non-reference and load buses.
- Set up the angle and voltage matrices.
- Set some variable values (iteration number, convergence flag, etc.).
- Calculate the power mismatch of the system.
- Enter a while loop to iterate until convergence or maximum iteration number is reached.
- Form the Jacobian matrix.
- Scatter the reduced power mismatch vectors using MPI.Scatterv().
- Calculate the local solution using the reduced power mismatch vectors.
- Gather the solutions using MPI.Gatherv().
- Update the angle and voltage values of the system.
- Recalculate the power mismatch of the system.

We wanted to let you know that we have been working hard to parallelize the code and run it, but we have been struggling with errors in the parallelized code. Despite this setback, we are pleased to report that we have still managed to complete much of the work within the time frame given.