Lab 5: PETSc and Krylov Methods

1 PETSc Installation

For the PETSc installation, please refer to the *install.sh* script in the appendix. Unless noted, the installation, and *petsc* folder is in
PETSC_ROOT=*gpfs/accounts/ners570f20_class_root/ners570f30_class/$USER*.
Logging is used for each step, as per the install script.

1. To download PETSc with git:

   git clone -b release https://gitlab.com/petsc/petsc.git

2. To load modules:

   module load gcc openmpi openblas lapack python3.6-anaconda

3. To configure PETSc with release mode, in the class user directory, with openblas, and optimization and debugging compiler options:

   –with-debugging=0 –prefix=$PETSC_ROOT –with-openblas=1

   –with-openblas-dir=$OPENBLAS_ROOT

   -COPTFLAGS=-g -O3 -CXXOPTFLAGS=-g -O3 -FOPTFLAGS=-g -O3

4. To configure PETSc to have the examples and tutorials, the base configuration appears to be adequate.

5. To run the tests, in the *petsc* directory,

   make test

   The test results were as follows, and only a small number of tests failed, primarily seeming to do with a *window_sync-flavor* module that is likely not installed for this configuration:

```
  Summary
# -------------
# FAILED diff-vec_is_sf_tutorials-ex1_7+sf_window_sync-fence_sf_window_flavor-
dynamic
vec_is_sf_tutorials-ex1_7+sf_window_sync-active_sf_window_flavor-dynamic
vec_is_sf_tutorials-ex1_7+sf_window_sync-lock_sf_window_flavor-dynamic
diff-vec_is_sf_tutorials-ex1_2+sf_window_sync-fence_sf_window_flavor-create
diff-vec_is_sf_tutorials-ex1_2+sf_window_sync-fence_sf_window_flavor-dynamic
diff-vec_is_sf_tutorials-ex1_2+sf_window_sync-active_sf_window_flavor-create
vec_is_sf_tutorials-ex1_2+sf_window_sync-active_sf_window_flavor-dynamic
```

```
diff-vec_is_sf_tutorials-ex1_2+sf_window_sync-lock_sf_window_flavor-create
diff-vec_is_sf_tutorials-ex1_2+sf_window_sync-lock_sf_window_flavor-dynamic
diff-vec_is_sf_tests-ex4_2_window+sf_window_sync-fence_sf_window_flavor-dynamic
vec_is_sf_tests-ex4_2_window+sf_window_sync-active_sf_window_flavor-dynamic
vec_is_sf_tests-ex4_2_window+sf_window_sync-lock_sf_window_flavor-dynamic
diff-vec_is_sf_tutorials-ex1_3+sf_window_sync-fence_sf_window_flavor-dynamic
...
# success 7271/9161 tests (79.4%)
# failed 54/9161 tests (0.6%)
# todo 225/9161 tests (2.5%)
# skip 1611/9161 tests (17.6%)
#
# Wall clock time for tests: 2797 sec
# Approximate CPU time (not incl. build time): 9637.589999999878 sec
#
# To rerun failed tests:
#     /usr/bin/gmake -f gmakefile test test-fail=1
#
# Timing summary (actual test time / total CPU time):
#   ksp_ksp_tutorials-ex56_2: 926.77 sec / 1267.83 sec
#   ksp_ksp_tutorials-ex70_fetidp_lumped: 83.10 sec / 345.95 sec
#   ksp_ksp_tutorials-ex43_3: 70.63 sec / 91.58 sec
#   ksp_ksp_tutorials-ex70_fetidp_saddlepoint_lumped: 68.00 sec / 276.56 sec
#   mat_tests-ex33_2: 66.24 sec / 89.44 sec
```

6. To install the PETSc build, it is ensured that the environmental variables are first set, PETSC_DIR=$PETSC_ROOT, and PETSC_ARCH="". Then, in the the *make all* command is issued, based on the final output of the *configure* command

   make PETSC_DIR=$PETSC_ROOT/petsc PETSC_ARCH=arch-linux-c-opt all

7. To view the source code, the *ex15*documenation is viewed on the petsc website.

8. To find and compile the example tutorials, the *KSP* Krylov solver examples can be found in:

   $PETSC_DIR/petsc/src/ksp/ksp/tutorials

   The specific program of interest can be made as:

   make clean && make ex15.ext

   where ext is either *c* or *f90* for the C or Fortran languages.

2 Comparison of Krylov Methods

To generate the PETSc runtime statistics for the various Krylov methods, including the number of iterations, run time, and final residual error norm, the following process was followed, with scripts in the appendix. The various combinations of executable arguments, including the *ksp_type*, whether to

use *ksp_gmres_restart*, the matrix size *m,n*, and the use of no preconditioners *pc_type*, are generated using *run.py*, which generates a bash script of the commands *run.sh*. A job script *job.slurm* is then used to run the commands on compute nodes.

The following results of the Krylov statistics for *ex15* are shown in Table 1 and Figure 1.

Table 1: Summary of statistics of Krylov methods: iterations (runtime [s]) for various matrix sizes for *ex15*. The GMRES methods have either no restart (iterating up to $n = m$ iterations), or restart at 30 or 200 iterations.

Matrix Size	CG	BiCGSTAB	GMRES (0)	GMRES(30)	GMRES(200)
8	10(0.380)	10(0.282)	26(0.222)	10(0.218)	10(0.217)
32	58(0.224)	44(0.220)	118(0.218)	119(0.219)	58(0.217)
128	218(0.236)	169(0.241)	379(0.449)	1341(0.459)	216(0.428)
256	427(0.340)	313(0.399)	652(3.789)	4501(4.124)	753(3.611)
512	838(1.540)	602(2.171)	1164(50.240)	10000(41.419)	2489(47.586)
1024	1637(12.505)	1310(19.024)	2195(829.508)	1000(183.881)	8888(739.275)

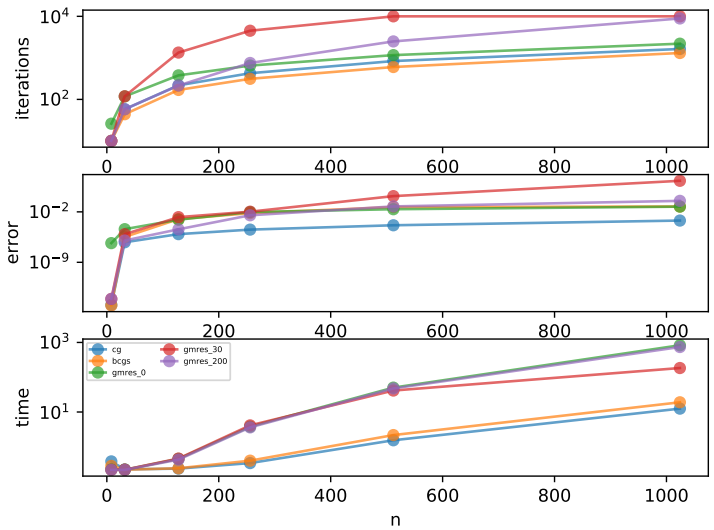

Figure 1: Curves of scaling of statistics of Krylov methods for various matrix sizes for *ex15*. The GMRES methods have either no restart (iterating up to $n = m$ iterations), or restart at 30 or 200 iterations.

From these results, it is evident that there are tradeoffs between time of iterations, and total iterations and the resulting error norm of the residual. The method that scales best with grid size is, in terms of runtime, is possibly the CG or BiCGSTAB methods, that require no restarts and take less time and less iterations.

When GMRES is run with restarts r of 30 and 200 iterations, for small matricies of size $n < r$, the number of iterations does not change drastically, as the Krylov subspace is built up within n iterations. However for larger matrices where $n > r$, the number of total iterations increases drastically, and even reaches the 10,000 iteration threshold tolerance for the largest matrices. This is due to the iterations restarting many times before building up the Krylov subspace to dimension n.

3 Preconditioners

The following results of the Krylov statistics for *ex15* are shown in Table **??** and Figure **??**. It should be noted that th GMRES methods have no restart, iterating up to $n = m$ iterations, and the ILU and ICC preconditioners are unable to be parallalized in PETSc , and so *mpiexec* was not used for runtimes.

3.1 GMRS

Table 2: Summary of scaling of statistics of GMRES method preconditoners for various matrix sizes for *ex15*. The GMRES methods have no restart (iterating up to $n = m$ iterations).

Matrix Size	None	Jacobi	SOR	ILU	ICC	GAMG
8	26(0.248)	26(0.253)	17(0.253)	10(0.231)	10(0.231)	7(0.251)
32	118(0.256)	118(0.254)	42(0.247)	27(0.233)	27(0.234)	7(0.257)
128	379(0.672)	379(0.676)	117(0.394)	85(0.361)	85(0.362)	8(0.301)
256	652(5.424)	652(5.425)	189(1.560)	142(1.499)	142(1.572)	9(0.446)
512	1164(65.640)	1164(72.788)	327(14.323)	247(15.052)	247(15.635)	9(1.139)
1024	2195(1289.856)	2195(1112.954)	630(176.528)	476(177.825)	476(188.280)	10(3.674)

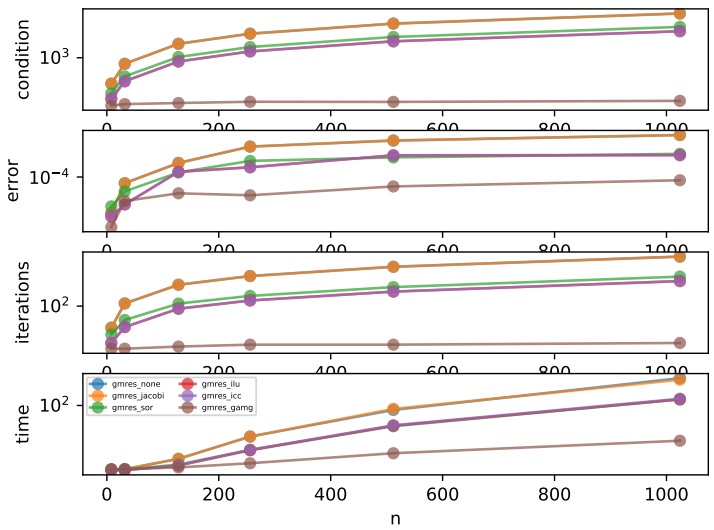

Figure 2: Curves of scaling of statistics of GMRES method preconditoners for various matrix sizes for *ex15*. The GMRES methods have no restart (iterating up to $n = m$ iterations).

For the GMRES method, for time scaling, the preconditioners of *none*, *jacobi* and *sor* all took similar time, and took longer much longer time than the *ilu*, *icc*, and *gamg* methods. The time appears to obey approximately an exponential relationship with the size of the matrix.

For the GMRES method, the preconditioners that performed best, in terms of the *gamg* algebraic multigrid method performed best, due to even at high matrix sizes, scaling the best. It used the least time and performed the least iterations, and still had the lowest residual error norm, and the lowest condition number estimate, compared to the other preconditioners.

When analysing the plots in 2 of run time and iterations versus matrix size, there is a direct, positive relationship between the scaling of the total runtime and number of iterations for each preconditioner. For each preconditioner, the total runtime appears to scale approximately exponentially with matrix size, and the number of iterations appears to scale polynomially with matrix size. The order of the preconditioners, in terms of slowest to fasest methods appears to be the same when looking at total runtime or number of iterations, with the *gamg* being fastest, then *ilu* and *icc*, and then *none jacobi*, and *sor* methods being slowest.

3.2 CG

Table 3: Summary of scaling of statistics of CG method preconditoners for various matrix sizes for *ex15*.

Matrix Size	None	Jacobi	SOR	ILU	ICC	GAMG
8	10(0.798)	10(0.236)	15(0.237)	9(0.228)	9(0.226)	7(0.355)
32	58(0.240)	58(0.241)	42(0.239)	9(0.223)	27(0.227)	8(0.242)
128	218(0.451)	218(0.446)	122(0.312)	27(0.278)	88(0.277)	9(0.285)
256	427(1.588)	427(1.595)	204(0.572)	88(0.502)	145(0.489)	9(0.420)
512	838(10.190)	838(10.166)	336(2.202)	145(2.071)	277(2.084)	10(1.035)
1024	1637(73.899)	1637(73.554)	651(17.286)	277(13.455)	489(13.844)	11(4.030)

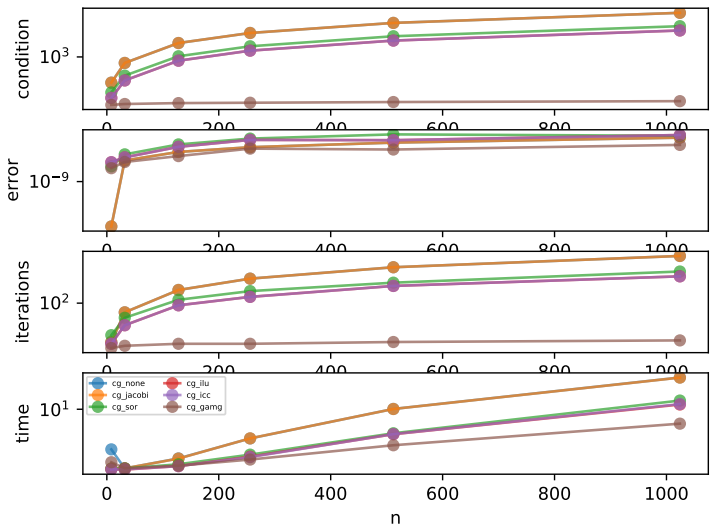

Figure 3: Curves of scaling of statistics of CG method preconditoners for various matrix sizes for *ex15*.

For each preconditioner, the options are as follows, for example

time mpiexec -n 2 ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type jacobi

-ksp_monitor_singular_value -ksp_gmres_restart 1024

where the *pc_type* passes in the preconditioner type.

4 GMRES Error Scaling and Condition Number

For each preconditioner, the options are as follows, for example

time mpiexec -n 2 ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type jacobi

-ksp_monitor_singular_value -ksp_gmres_restart 1024

where the *pc_type* passes in the preconditioner type, and the *ksp_monitor_singular_value* allows the condition number to be estimated as the ratio of the approximated maximum and minimum singular values.

Table 4: Summary of condition number estimates from GMRES method preconditoners for various matrix sizes for *ex15*.

Matrix Size	None	Jacobi	SOR	ILU	ICC	GAMG
8	28.8956	28.8956	7.53438	3.58455	3.58455	1.42788
32	433.996	433.996	77.4939	39.6772	39.6772	1.74397
128	6729.31	6729.31	1100.5	596.623	596.623	2.02631
256	26733.1	26733.1	4315.74	2366.39	2366.39	2.40286
512	106549	106549	17099.2	9427.55	9427.55	2.35848
1024	425414	425414	68075.4	37636.1	37636.1	2.72101

When looking at the condition numbers using the preconditioners, the most efficient reduction in condition number was algebraic multigrid *gamg* method, which is orders of magnitude lower than the other methods, and original condition number. This also directly corresponds to the number of iterations required, and the time taken to complete the algorithm, with this algebraic multigrid method being signicifantly faster than the other methods.

5 CG Error Scaling and Condition Number

For each preconditioner, the options are as follows, for example

time mpiexec -n 2 ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type jacobi

-ksp_monitor_singular_value

where the *pc_type* passes in the preconditioner type, and the *ksp_monitor_singular_value* allows the condition number to be estimated as the ratio of the approximated maximum and minimum singular values.

When looking at the condition number κ of the bare congugate gradient method (without preconditioning), when viewed as a function of matrix size n, and viewing the residual error norm e as a function of matrix size, these data can be used to find a bound on the error. As shown in Figure 4, the error is shown to be bounded at least by the equation

$$e(n) \leq 2\left(\frac{\sqrt{\kappa(n)}-1}{\sqrt{\kappa(n)}+1}\right)^{k(n)}.$$

Table 5: Summary of condition number estimates from CG method preconditoners for various matrix sizes for *ex15*.

Matrix Size	None	Jacobi	SOR	ILU	ICC	GAMG
8	29.2841	29.2841	7.66582	3.60845	3.60845	1.38219
32	437.698	437.698	76.852	39.665	39.665	1.56475
128	6740.68	6740.68	1097.26	596.647	596.647	1.77385
256	26765	26765	4310.33	2366.41	2366.41	1.86193
512	106549	106549	17099.2	9427.55	9427.55	2.35848
1024	425799	425799	68050.5	37636.2	37636.2	2.31255

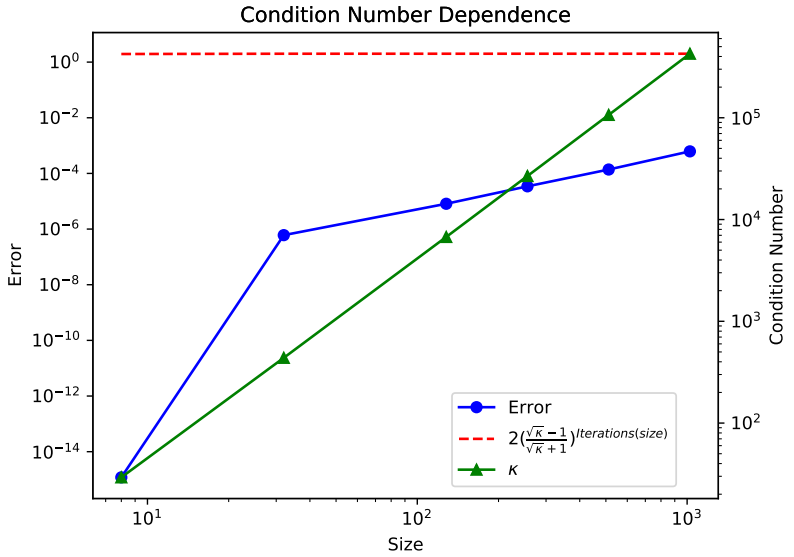

Figure 4: Curves of scaling of error and condition number for the CG method as a function of matrix size, for *ex15*.

6 Appendix

6.1 *install.sh*

```
#!/bin/bash

# Paths
cwd=$(pwd)
build_dir=$(pwd)


# Builds
build_download=0
build_load=1
build_configure=0
build_make=1
build_install=1
```

```
buid_test=0

# Download PETSc
echo Downloading PETSc ...
url=https://gitlab.com/petsc/petsc.git
folder=petsc
flags=("-b release")

if [[ ${build_download} -eq 1 ]]
then
if [[ ! -d ${folder} ]]
then
git clone ${flags[@]} ${url} ${folder}
fi
fi
cd ${folder}
git pull


# Load Modules
echo Loading modules ...
collection=petsc-build
cc=gcc
modules=(${cc} openmpi openblas lapack python3.6-anaconda)
module purge


if [[ ${build_load} -eq 1 ]]
then
if [[ -z $(module savelist 2>&1 | grep ${collection}) ]]
then
module disable ${collection}
for m in ${modules[@]}
do
echo $m
module load ${m}
done

module save ${collection}
else
module restore ${collection}
fi
fi
module list


# Install PETSc
```

```
config=configure
log=config.log

flags=("--with-debugging=0" "--prefix=${build_dir}" "--with-openblas=1" "--with-openblas-di
#mkdir -p build


if [[ ${build_configure} -eq 1 ]]
then
./${config} "${flags[@]}" 2>&1 | tee ${log}
fi



# Make PETSc
makelog=make.log

pattern="Configure stage complete. Now build PETSc libraries with:"
makecmd=$(grep "${pattern}" -A1 ${log} | sed "s%${pattern}\(.*\)%\1%")
if [[ ${build_make} -eq 1 ]]
then
$makecmd 2>&1 | tee ${makelog}
fi

pattern="Now to install the libraries do:"
makecmd=$(grep "${pattern}" -A1 ${makelog} | sed "s%${pattern}\(.*\)%\1%")

if [[ ${build_install} -eq 1 ]]
then
$makecmd 2>&1 | tee ${makelog}
fi

# Cleanup
cd ${cwd}
```

6.2 *run.py*

```
#!/usr/bin/env python

# Import modules
import os,sys,stat,itertools,copy

# Delimeters
delimeters={}
delimeters["flag"]="-"
delimeters["set"]=" "
delimeters["IFS"]=" "
delimeters["file"]=","


def set_value(key,value,array):
a = array
if isinstance(key,list):
for k in key[:-1]:
if not isinstance(a.get(k),dict):
a[k] = {}
a = a[k]
else:
key = [key]
a[key[-1]] = value
return


def set_flag(key,value,array):
value=delimeters["IFS"].join(["%s%s"%(delimeters["flag"],v) for v in value.split(delimeters
set_value(key,value,array)
return

def set_parameter(key,value,array):
if not isinstance(value,(tuple,list)):
value=[value]
value = [str(v) if not callable(v) else v for v in value]
set_value(key,value,array)
return


def set_argument(key,value,array):
if value in [delimeters["IFS"]]:
return
for k in key.split(delimeters["IFS"]):
if isinstance(value,str):
val="%s%s%s"%(k,delimeters['set'],value)
```

```
else:
val= lambda *args,key=k,_value=value,**kwargs: _value(key,*args,**kwargs)
array.append(val)
return

def get_argument(key,array):
return array[key]

def get_iterations(flag,args):
keys=['%s%s%s'%(delimeters['flag'],s,delimeters['set']) for s in ['n','m']]
values=[]
for a in args:
for k in keys:
if k in a:
values.append(a.replace(k,''))
if values == []:
return ''


values=list(set(values))
value = str(min(values))
arg ='%s%s%s'%(flag,delimeters['set'],value)
return arg


# Simulation Type
_args = ['simulation','presource','source','exe']
_defaults = ['release','time mpiexec -n 2 ','./','ex15']
args = dict(zip(_args,_defaults))
args.update(dict(zip(_args,sys.argv[1:])))

# Program
files={}
_file = '.'.join(__file__.split('.')[:-1])
files['run'] = '%s.%s'%(_file,'sh')
files['output'] = '%s.%s'%(_file,'log')


# Parameters
flags={}
subflags={}
parameters={}
subparameters={}
```

```
_simulations_keys = ['debug','release','condition']
_simulations_defaults = {'ksp_type':['cg'],'n m':[8],
                     'ksp_gmres_restart':[get_iterations],'pc_type':['none'],
}
_subsimulations_defaults = {'ksp_type':{'gmres':['ksp_gmres_restart']},
    # 'pc_type': {'ilu':['mat_type'],'icc':['mat_type']}
   }

_keys = {k: ["ksp_type","n m",'pc_type'] for k in _simulations_keys}
_keys['condition'].extend(['ksp_monitor_singular_value'])

_simulations = {k: copy.deepcopy(_simulations_defaults) for k in _simulations_keys}
_simulations['release'].update({
       'ksp_type':['cg','bcgs','gmres'],
       'n m':[8,32,128,256,512,1024],
       'ksp_gmres_restart':[get_iterations,30,200],
                'ksp_monitor_singular_value':[''],
})

_simulations['condition'].update({
       'ksp_type':['cg','gmres'],
       'n m':[8,32,128,256,512,1024],
       'pc_type':['none','jacobi','sor','ilu','icc','gamg'],
       'ksp_gmres_restart':[get_iterations],
                'ksp_monitor_singular_value':[''],
       'mat_type':[],
})
_subsimulations={k:copy.deepcopy(_subsimulations_defaults) for k in _simulations_keys}

keys = _keys[args['simulation']]
simulations = _simulations[args['simulation']]
subsimulations = _subsimulations[args['simulation']]

#print(keys)
for key in keys:

flag=key
values=simulations[flag]
set_flag(key,flag,flags)
set_parameter(key,values,parameters)
for value in subsimulations.get(key,[]):
for flag in subsimulations[key][value]:
values=simulations[flag]
set_flag([key,value],flag,subflags)
set_parameter([key,value],values,subparameters)

#print(subflags)
```

```
#print(subparameters)
# Arguments
arguments = []
values = [list(values) for values in itertools.product(*[parameters[key] for key in keys],r
for value in values:
arguments.append([])
for key,val in zip(keys,value):
set_argument(flags[key],val,arguments[-1])

subarguments={}
for key in subparameters:
subarguments[key]={}
for value in subparameters[key]:
_value = []
set_argument(flags[key],value,_value)
_value = _value[0]
subarguments[key][_value] = []
for s in subparameters[key][value]:
set_argument(subflags[key][value],s,subarguments[key][_value])

#print(arguments)
for argument in arguments[:]:
_arguments = [[s(argument) if callable(s) else s for s in subarguments[key][value]]
for key in subarguments
for value in subarguments[key]
if ((argument[keys.index(key)+sum([k.count(delimeters['IFS']) for k in keys[:keys.index(key
if _arguments == []:
continue
i = arguments.index(argument)
_argument = arguments.pop(i)
i -= 1
for _arg in itertools.product(*_arguments,repeat=1):
i += 1
arguments.insert(i,_argument+list(_arg))



# Run Commands
writer = lambda f,s: f.write("%s\n"%(s))
shebang='#!/bin/bash'
logging=["{","} 2>&1 | tee %s"%(files['output'])]
commands = [*[(lambda *arguments,s=s:' '.join(['%s%s%s%s'%(s,args['presource'] if not any([
            *[(lambda *arguments,s=s: '%s'%(s)) for s in ['sleep 2','echo ','']]
           ]
with open(files['run'],'w') as f:
writer(f,shebang)
writer(f,logging[0])
```

```
for argument in arguments:
for command in commands:
writer(f,command(*argument))
writer(f,logging[1])



st = os.stat(files['run'])
os.chmod(files['run'], st.st_mode | stat.S_IXUSR)
```

6.3 *job.slurm*

```
#!/bin/bash

#SBATCH --account=ners570f20_class
#SBATCH --job-name=ex15
#SBATCH --partition=standard
#SBATCH --mail-user=mduschen@umich.edu
#SBATCH --mail-type=END
#SBATCH --nodes=1
#SBATCH --mem-per-cpu=4000m
#SBATCH --time=01:00:00
#SBATCH --ntasks-per-node=3

# Setup
module purge
module restore petsc-build

./run.sh
```

6.4 *process.sh*

```
#!/bin/bash

#file=${1:-"run.log"}
file=${1:-"run_nocondition.log"}
results=${2:-"results.log"}
length=${3:-6}
flags="-A${length} -E"

# indices=(8 32 128 256 512 1024)

names=(cg bcgs gmres_0 gmres_30 gmres_200)
indices=("ksp_type cg " "ksp_type bcgs " "ksp_type gmres .* \-ksp_gmres_restart (^30|^200|8
patterns_header=('\-m' '\-m' '\-m' '\-m' '\-m')

{
for i in ${!indices[@]}
do
name="${names[$i]}"
initial="${indices[$i]}"
pattern_header="${patterns_header[$i]}"
patterns=("error" "iterations" "real")
labels=("error" "iterations" "time")

header="$(grep "${initial}" ${file} | sed -e "s%.*${pattern_header}[ ]*\([^ ]*\).*%\1%" | t

# echo ${name}
for j in "${!patterns[@]}"
do
pattern="${patterns[$j]}"
label="${labels[$j]}"
# echo ${label}
_file=$(echo ${results} | sed "s%\(.*\)\.\([^.]*$\)%${label}.csv%")
if [[ ${i} == 0 ]]
then
echo "label,${header}" > ${_file}
fi
result="$(grep ${flags} "${initial}" ${file} | grep ${pattern} | sed -e "s%.*${pattern}[ ]*
echo "${name},${result}" >> ${_file}

done
# echo
done
} #2>&1 | tee ${results}
```

6.5 *plot.py*

```
#!/usr/bin/env python

import os,sys,glob
import numpy as np
import scipy as sp
import pandas as pd
import scipy.optimize
import matplotlib.pyplot as plt

def plot(x,y,fig,ax,props):
ax.plot(x,y,**{k: props.pop(k) for k in ['label','marker','linestyle','color','alpha'] if k
for k in props:
getattr(ax,'%s'%(k))(props[k])
return

def fit(x,y,f,params):
params[:],_ = sp.optimize.curve_fit(f,x,y,params)
return

def model(x,*params):
return params[0]*(x**params[1])

def importer(path):
df = pd.read_csv(path,header=None).T
df.columns = df.iloc[0]
df.drop(0,inplace=True)
index=df.columns[0]
df.set_index(index,drop=True,append=False, inplace=True)
return df

def preprocess(df,functions):
for func in functions:
df = df.applymap(func)
df.index = df.index.map(func)
print(df.index)
return df

# Arguments
paths=['*.csv']
if len(sys.argv[1:]):
paths = sys.argv[1:]
files=[f for path in paths for f in glob.glob(path)]
keys = ['.'.join(f.split('.')[:-1]).split('_')[0] for f in files]

functions = {'time':[lambda x: ((60*int(str(x).split(':')[0])+float(str(x).split(':')[1]))
```

```
N = len(keys)

data = {}
for k,f in zip(keys,files):
data[k] = importer(f)
data[k] = preprocess(data[k],functions.get(k,[]))




# Figure
labels = ['cg','gmres']
get_labels = lambda df,string,boolean=lambda string,label: string in label: [label for labe

# labels = ['nocondition']
# get_labels = lambda df,string: df.columns

for l in labels:
for k in keys:
print('key',k)
print(data[k][get_labels(data[key],l)])
print()

for l in labels:
fig,axes = plt.subplots(N,1)

_props = lambda **kwargs: {'set_xlabel':'n','set_ylabel':'%s'%(kwargs['set_ylabel']),'set_t
 'set_xscale':'linear','set_yscale':'log', **kwargs}
colors = ['r','g','b','m','y']

# Process

for i in range(N):
key = keys[i]
df = data[key]
ax = axes[i]

x = df.index.values

print('LABLES',get_labels(df,l))
for label in get_labels(df,l):
y = df[label].values
_slice = slice(0,None)
x,y = x[_slice],y[_slice]

# Fit
```

```
# _params = ['a','b']
# params = [1,2]
# fit(x,y,model,params)

# Plot
props=_props(**{'set_ylabel':key})
props['label'] = '%s'%(label)
props['marker'] = 'o'
props['linestyle'] = '-'
props['alpha'] = 0.7
#props['color'] = colors[i]
# print(key,x)
plot(x,y,fig,ax,props)

if i==(N-1):
ax.legend(ncol=2,fontsize=4.5)


# props['label'] = '%s'%(
#  '\n'.join(['%s: %0.3e'%(l,p)
#   for l,p in zip(_params,params)]))
# props['marker'] = '*'
# props['linestyle'] = '--'
# props['color'] = colors[i]
# plot(x,model(x,*params),fig,ax,props)

#fig.legend(ncol=2,loc=(0.48,0.15),fontsize=8)
fig.savefig('results_%s.%s'%(l,'pdf'))
```

6.6 *run.sh*

```
#!/bin/bash
{
echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 8 -m 8 -pc_type none -ksp_monitor_singular_va
time mpiexec -n 2 ./ex15 -ksp_type cg -n 8 -m 8 -pc_type none -ksp_monitor_singular_value
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 8 -m 8 -pc_type jacobi -ksp_monitor_singular_
time mpiexec -n 2 ./ex15 -ksp_type cg -n 8 -m 8 -pc_type jacobi -ksp_monitor_singular_value
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 8 -m 8 -pc_type sor -ksp_monitor_singular_val
time mpiexec -n 2 ./ex15 -ksp_type cg -n 8 -m 8 -pc_type sor -ksp_monitor_singular_value
sleep 2
echo

echo time ./ex15 -ksp_type cg -n 8 -m 8 -pc_type ilu -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 8 -m 8 -pc_type ilu -ksp_monitor_singular_value
sleep 2
echo

echo time ./ex15 -ksp_type cg -n 8 -m 8 -pc_type icc -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 8 -m 8 -pc_type icc -ksp_monitor_singular_value
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 8 -m 8 -pc_type gamg -ksp_monitor_singular_va
time mpiexec -n 2 ./ex15 -ksp_type cg -n 8 -m 8 -pc_type gamg -ksp_monitor_singular_value
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 32 -m 32 -pc_type none -ksp_monitor_singular_
time mpiexec -n 2 ./ex15 -ksp_type cg -n 32 -m 32 -pc_type none -ksp_monitor_singular_value
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 32 -m 32 -pc_type jacobi -ksp_monitor_singula
time mpiexec -n 2 ./ex15 -ksp_type cg -n 32 -m 32 -pc_type jacobi -ksp_monitor_singular_val
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 32 -m 32 -pc_type sor -ksp_monitor_singular_v
time mpiexec -n 2 ./ex15 -ksp_type cg -n 32 -m 32 -pc_type sor -ksp_monitor_singular_value
sleep 2
```

```
echo

echo time ./ex15 -ksp_type cg -n 32 -m 32 -pc_type ilu -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 32 -m 32 -pc_type ilu -ksp_monitor_singular_value
sleep 2
echo

echo time ./ex15 -ksp_type cg -n 32 -m 32 -pc_type icc -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 32 -m 32 -pc_type icc -ksp_monitor_singular_value
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 32 -m 32 -pc_type gamg -ksp_monitor_singular_
time mpiexec -n 2 ./ex15 -ksp_type cg -n 32 -m 32 -pc_type gamg -ksp_monitor_singular_value
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 128 -m 128 -pc_type none -ksp_monitor_singula
time mpiexec -n 2 ./ex15 -ksp_type cg -n 128 -m 128 -pc_type none -ksp_monitor_singular_val
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 128 -m 128 -pc_type jacobi -ksp_monitor_singu
time mpiexec -n 2 ./ex15 -ksp_type cg -n 128 -m 128 -pc_type jacobi -ksp_monitor_singular_v
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 128 -m 128 -pc_type sor -ksp_monitor_singular
time mpiexec -n 2 ./ex15 -ksp_type cg -n 128 -m 128 -pc_type sor -ksp_monitor_singular_valu
sleep 2
echo

echo time ./ex15 -ksp_type cg -n 128 -m 128 -pc_type ilu -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 128 -m 128 -pc_type ilu -ksp_monitor_singular_value
sleep 2
echo

echo time ./ex15 -ksp_type cg -n 128 -m 128 -pc_type icc -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 128 -m 128 -pc_type icc -ksp_monitor_singular_value
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 128 -m 128 -pc_type gamg -ksp_monitor_singula
time mpiexec -n 2 ./ex15 -ksp_type cg -n 128 -m 128 -pc_type gamg -ksp_monitor_singular_val
sleep 2
echo
```

```
echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 256 -m 256 -pc_type none -ksp_monitor_singula
time mpiexec -n 2 ./ex15 -ksp_type cg -n 256 -m 256 -pc_type none -ksp_monitor_singular_val
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 256 -m 256 -pc_type jacobi -ksp_monitor_singu
time mpiexec -n 2 ./ex15 -ksp_type cg -n 256 -m 256 -pc_type jacobi -ksp_monitor_singular_v
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 256 -m 256 -pc_type sor -ksp_monitor_singular
time mpiexec -n 2 ./ex15 -ksp_type cg -n 256 -m 256 -pc_type sor -ksp_monitor_singular_valu
sleep 2
echo

echo time ./ex15 -ksp_type cg -n 256 -m 256 -pc_type ilu -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 256 -m 256 -pc_type ilu -ksp_monitor_singular_value
sleep 2
echo

echo time ./ex15 -ksp_type cg -n 256 -m 256 -pc_type icc -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 256 -m 256 -pc_type icc -ksp_monitor_singular_value
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 256 -m 256 -pc_type gamg -ksp_monitor_singula
time mpiexec -n 2 ./ex15 -ksp_type cg -n 256 -m 256 -pc_type gamg -ksp_monitor_singular_val
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 512 -m 512 -pc_type none -ksp_monitor_singula
time mpiexec -n 2 ./ex15 -ksp_type cg -n 512 -m 512 -pc_type none -ksp_monitor_singular_val
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 512 -m 512 -pc_type jacobi -ksp_monitor_singu
time mpiexec -n 2 ./ex15 -ksp_type cg -n 512 -m 512 -pc_type jacobi -ksp_monitor_singular_v
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 512 -m 512 -pc_type sor -ksp_monitor_singular
time mpiexec -n 2 ./ex15 -ksp_type cg -n 512 -m 512 -pc_type sor -ksp_monitor_singular_valu
sleep 2
echo

echo time ./ex15 -ksp_type cg -n 512 -m 512 -pc_type ilu -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 512 -m 512 -pc_type ilu -ksp_monitor_singular_value
```

```
sleep 2
echo

echo time ./ex15 -ksp_type cg -n 512 -m 512 -pc_type icc -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 512 -m 512 -pc_type icc -ksp_monitor_singular_value
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 512 -m 512 -pc_type gamg -ksp_monitor_singula
time mpiexec -n 2 ./ex15 -ksp_type cg -n 512 -m 512 -pc_type gamg -ksp_monitor_singular_val
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type none -ksp_monitor_singu
time mpiexec -n 2 ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type none -ksp_monitor_singular_v
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type jacobi -ksp_monitor_sin
time mpiexec -n 2 ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type jacobi -ksp_monitor_singular
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type sor -ksp_monitor_singul
time mpiexec -n 2 ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type sor -ksp_monitor_singular_va
sleep 2
echo

echo time ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type ilu -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type ilu -ksp_monitor_singular_value
sleep 2
echo

echo time ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type icc -ksp_monitor_singular_value
time ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type icc -ksp_monitor_singular_value
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type gamg -ksp_monitor_singu
time mpiexec -n 2 ./ex15 -ksp_type cg -n 1024 -m 1024 -pc_type gamg -ksp_monitor_singular_v
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type none -ksp_monitor_singular
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type none -ksp_monitor_singular_valu
sleep 2
echo
```

```
echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type jacobi -ksp_monitor_singul
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type jacobi -ksp_monitor_singular_va
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type sor -ksp_monitor_singular_
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type sor -ksp_monitor_singular_value
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type ilu -ksp_monitor_singular_value  -ksp_g
time ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type ilu -ksp_monitor_singular_value  -ksp_gmres_
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type icc -ksp_monitor_singular_value  -ksp_g
time ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type icc -ksp_monitor_singular_value  -ksp_gmres_
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type gamg -ksp_monitor_singular
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 8 -m 8 -pc_type gamg -ksp_monitor_singular_valu
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type none -ksp_monitor_singul
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type none -ksp_monitor_singular_va
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type jacobi -ksp_monitor_sing
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type jacobi -ksp_monitor_singular_
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type sor -ksp_monitor_singula
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type sor -ksp_monitor_singular_val
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type ilu -ksp_monitor_singular_value  -ksp
time ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type ilu -ksp_monitor_singular_value  -ksp_gmre
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type icc -ksp_monitor_singular_value  -ksp
```

```
time ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type icc -ksp_monitor_singular_value  -ksp_gmre
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type gamg -ksp_monitor_singul
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 32 -m 32 -pc_type gamg -ksp_monitor_singular_va
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type none -ksp_monitor_sing
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type none -ksp_monitor_singular_
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type jacobi -ksp_monitor_si
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type jacobi -ksp_monitor_singula
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type sor -ksp_monitor_singu
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type sor -ksp_monitor_singular_v
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type ilu -ksp_monitor_singular_value  -k
time ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type ilu -ksp_monitor_singular_value  -ksp_gm
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type icc -ksp_monitor_singular_value  -k
time ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type icc -ksp_monitor_singular_value  -ksp_gm
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type gamg -ksp_monitor_sing
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 128 -m 128 -pc_type gamg -ksp_monitor_singular_
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type none -ksp_monitor_sing
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type none -ksp_monitor_singular_
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type jacobi -ksp_monitor_si
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type jacobi -ksp_monitor_singula
sleep 2
```

```
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type sor -ksp_monitor_singu
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type sor -ksp_monitor_singular_v
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type ilu -ksp_monitor_singular_value  -k
time ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type ilu -ksp_monitor_singular_value  -ksp_gm
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type icc -ksp_monitor_singular_value  -k
time ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type icc -ksp_monitor_singular_value  -ksp_gm
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type gamg -ksp_monitor_sing
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 256 -m 256 -pc_type gamg -ksp_monitor_singular_
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type none -ksp_monitor_sing
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type none -ksp_monitor_singular_
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type jacobi -ksp_monitor_si
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type jacobi -ksp_monitor_singula
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type sor -ksp_monitor_singu
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type sor -ksp_monitor_singular_v
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type ilu -ksp_monitor_singular_value  -k
time ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type ilu -ksp_monitor_singular_value  -ksp_gm
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type icc -ksp_monitor_singular_value  -k
time ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type icc -ksp_monitor_singular_value  -ksp_gm
sleep 2
echo
```

```
echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type gamg -ksp_monitor_sing
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 512 -m 512 -pc_type gamg -ksp_monitor_singular_
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type none -ksp_monitor_si
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type none -ksp_monitor_singula
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type jacobi -ksp_monitor_
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type jacobi -ksp_monitor_singu
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type sor -ksp_monitor_sin
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type sor -ksp_monitor_singular
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type ilu -ksp_monitor_singular_value
time ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type ilu -ksp_monitor_singular_value  -ksp_
sleep 2
echo

echo time ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type icc -ksp_monitor_singular_value
time ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type icc -ksp_monitor_singular_value  -ksp_
sleep 2
echo

echo time mpiexec -n 2 ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type gamg -ksp_monitor_si
time mpiexec -n 2 ./ex15 -ksp_type gmres -n 1024 -m 1024 -pc_type gamg -ksp_monitor_singula
sleep 2
echo

} 2>&1 | tee ./run.log
```