CSE380 Project 1

Shreyas Gaikwad
(shreyas.gaikwad@utexas.edu)

Instructor - Dr. Karl Schulz

Shout out to Afzal and Allison for some really fun discussions!

Language - C++

Eduonix

Roses are Red
Violets are Blue
Unexpected
'{'
on Line 32

Driver routine

```cpp
#include <iostream>
#include <fstream>
#include <cmath>
#include "solvers.h"
#include "matrix_assemble.h"
#include "q_assemble.h"
#include <masa.h>
#include <grvy.h>
#include "T_exact_assemble.h"
#include "global_variables.h"
#include "print.h"
//#include <boost/test/unit_test.hpp>
```

Header files

```cpp
using namespace std;
using namespace MASA;
using namespace GRVY;

GRVY_Timer_Class gt;
GRVY_Input_Class iparse;
```

Namespaces and objects of GRVY class

```cpp
int main(int argc, char *argv[]) {

    grvy_log_setlevel(GRVY_INFO);
    // Initialize timer function
    gt.Init("GRVY Performance timing");
    gt.BeginTimer(__func__);
```

GRVY timer begin

```cpp
    // Declare and read variables
    int n, dimension, accuracy, MAX_ITERS, order, nn, dim_system, verification_mode;
    double L, TOL, dx, k_0;
    double** A;
    double *q, *T_exact, *T_computed, *delta_T;
    std::string solver, mode;
```

Declare variables

```
if(! iparse.Open("./input.dat"))
        exit(1);

// BE WARNED - Post-processing scripts grep and awk for certain
// strings to parse std::out properly, change print
// statements after careful consideration

if( iparse.Read_Var("verification",&verification_mode) )
    printf("--> %-11s = %i\n","verification_mode",verification_mode);

if( iparse.Read_Var("mode",&mode) )
            printf("--> %-11s = %s\n","mode",mode.c_str());

// Set debug mode if the input specification is such
std::string str1 ("debug");
if(mode.compare(str1) == 0)
    grvy_log_setlevel(GRVY_DEBUG);

if( iparse.Read_Var("grid/grid_points",&n,0) )
        printf("--> %-11s = %i\n","n",n);
if( n < 2 ) cerr << "Invalid value for number of gridpoints, has to be >= 2";

if( iparse.Read_Var("grid/dimension",&dimension,0) )
        printf("--> %-11s = %i\n","dimension",dimension);
if( dimension != 1 && dimension != 2 ) cerr << "Invalid value for dimension of domain, choose 1 or 2";

if( iparse.Read_Var("grid/length",&L,0.) )
        printf("--> %-11s = %f\n","length",L);
if( L <= 0.) cerr << "Invalid length of domain, has to be > 0";

if( iparse.Read_Var("solver/order",&order,0) )
        printf("--> %-11s = %i\n","order",order);
if( order != 2 && order != 4 ) cerr << "Invalid order of solver accuracy. Choose 2 or 4.";

if( iparse.Read_Var("solver/error_TOL",&TOL,0.) )
        printf("--> %-11s = %.17g\n","error_TOL",TOL);

if( iparse.Read_Var("solver/thermal_conductivity",&k_0,2.) )
        printf("--> %-11s = %f\n","thermal_conductivity",k_0);
if ( k_0 < 0. ) cerr << "Invalid value for thermal conductivity. Needs to be > 0";

if( iparse.Read_Var("solver/max_iters",&MAX_ITERS,10000000) )
        printf("--> %-11s = %i\n","max_iters",MAX_ITERS);
if ( MAX_ITERS < 1 ) cerr << "Invalid number for maximum number of iterations. Needs to be >= 1";

if( iparse.Read_Var("solver/solver_name",&solver) )
        printf("--> %-11s = %s\n","solver",solver.c_str());

std::string str2 ("jacobi"), str3 ("gauss");
if ( solver.compare(str2) != 0 && solver.compare(str3) != 0 ) cerr << "Invalid solver name. Use either gauss or jacobi.";
```

Parse input file and initialize variables. Some defensive checks in place.

```
    // nn is useful for 2D
    nn = n*n;
    dx = L/(n-1);

    // Assemble the linear system
    A = assemble_A(n, order, dimension);
    q = assemble_q(n, order, dimension, L, k_0);
    T_exact = assemble_T_exact(n, order, dimension, L, k_0);

    // Defensive checks
    if(A == NULL) cerr << "The matrix A formation has an error, NULL pointer returned." << endl;
    if(q == NULL) cerr << "The vector q formation has an error, NULL pointer returned." << endl;
    if(T_exact == NULL) cerr << "The vector T_exact formation has an error, NULL pointer returned." << endl;
    // Since we don't know if the user chose 1D or 2D, the dimensions of the system could be n or nn

    if (dimension == 1) dim_system = n;
    else if (dimension == 2) dim_system = nn;
    else cout << "dimensions can only be 1 or 2." << endl;

    // Solve the linear system
    T_computed = solve(solver, dim_system, A, q, TOL, MAX_ITERS);
    if(T_computed == NULL) cerr << "The vector T_computed formation has an error, NULL pointer returned." << endl;
    //// VERIFICATION MODE

    print_verification_mode(T_exact, T_computed, delta_T, n, verification_mode, dimension);

    //// Write results to output.log file

    write_results_output_file(dx, T_exact, T_computed, n, dimension);

    //// DEBUG MODE

    print_matrix_A(A, dim_system);
    print_vector_q(q, dim_system);
    print_compare_q_Texact_Tcomputed(q, T_exact, T_computed, dim_system);

    //// DEALLOCATE MEMORY

    delete[] q;
    delete[] T_exact;
    delete[] T_computed;
    delete[] delta_T;
    for(int i = 0; i < dim_system; i++){
        delete[] A[i];
    }
    delete[] A;

    ////End timers

    gt.EndTimer  (__func__);
    gt.Finalize();
    gt.Summarize();

    return 0;
}
```

Assemble matrix, RHS and exact solution

Compute temperature vector

Verification mode output

Write output to output.log

Debug mode verbose output

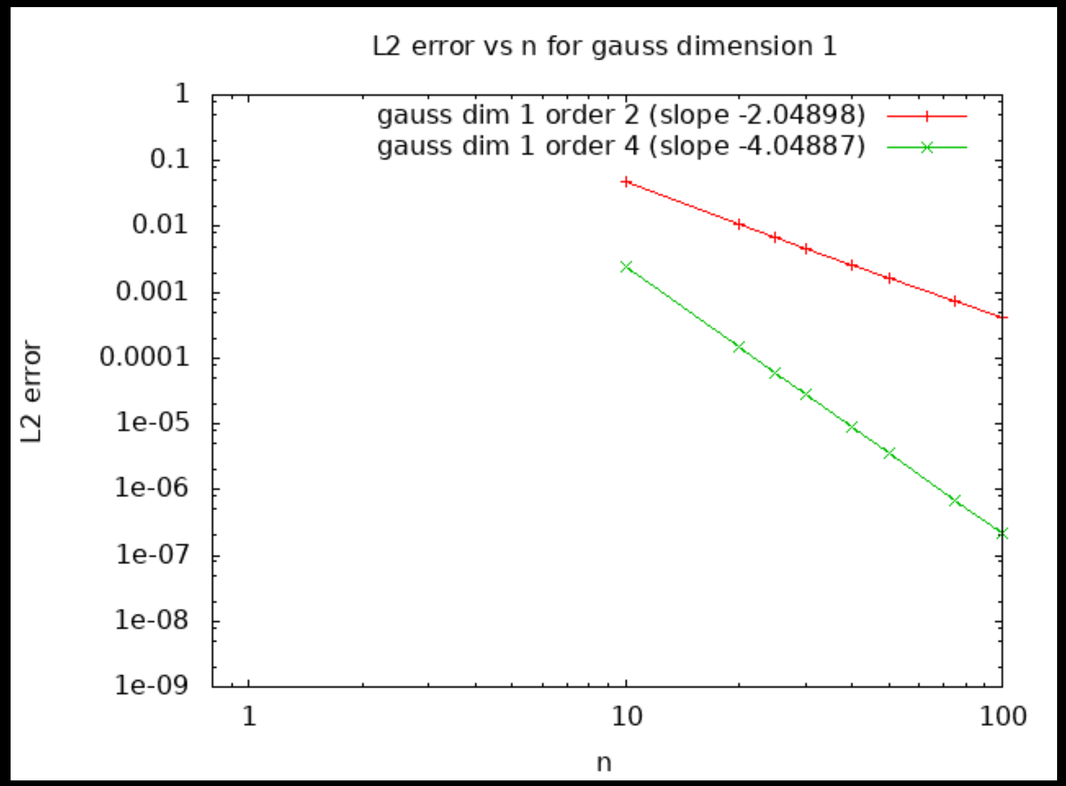
Deallocate memory

End GRVY timer

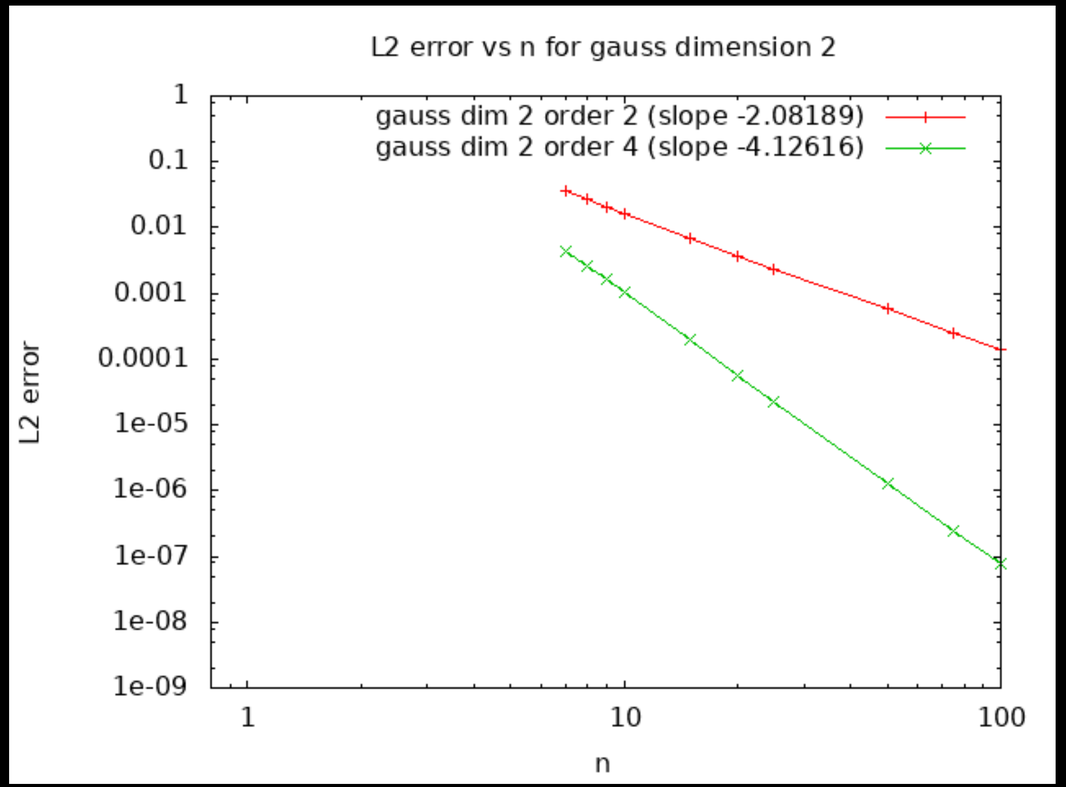
Input file

```
verification = 1    #Comparison with MASA solution? 1 for yes; no otherwise; Always keep as 1.
Otherwise a couple of regression tests might fail.
mode = debug        #To enable debug mode, use 'debug', anything else is normal mode.

[grid]

length          =   1.0     # Length of domain in each direction
dimension       =   2       # dimension of domain
grid_points     =   10      # Number of points in one direction


[solver]

thermal_conductivity    =   1.0                 # Thermal conductivity k_0
solver_name             = jacobi                # Use either jacobi or gauss
order                   = 2                     # Order of accuracy of stencil, use 2 or 4
error_TOL               = 0.00000000000000001   # Tolerance
max_iters               = 1000000               # Maximum number of iterations
```

Convergence plots for Gauss - Seidel solver

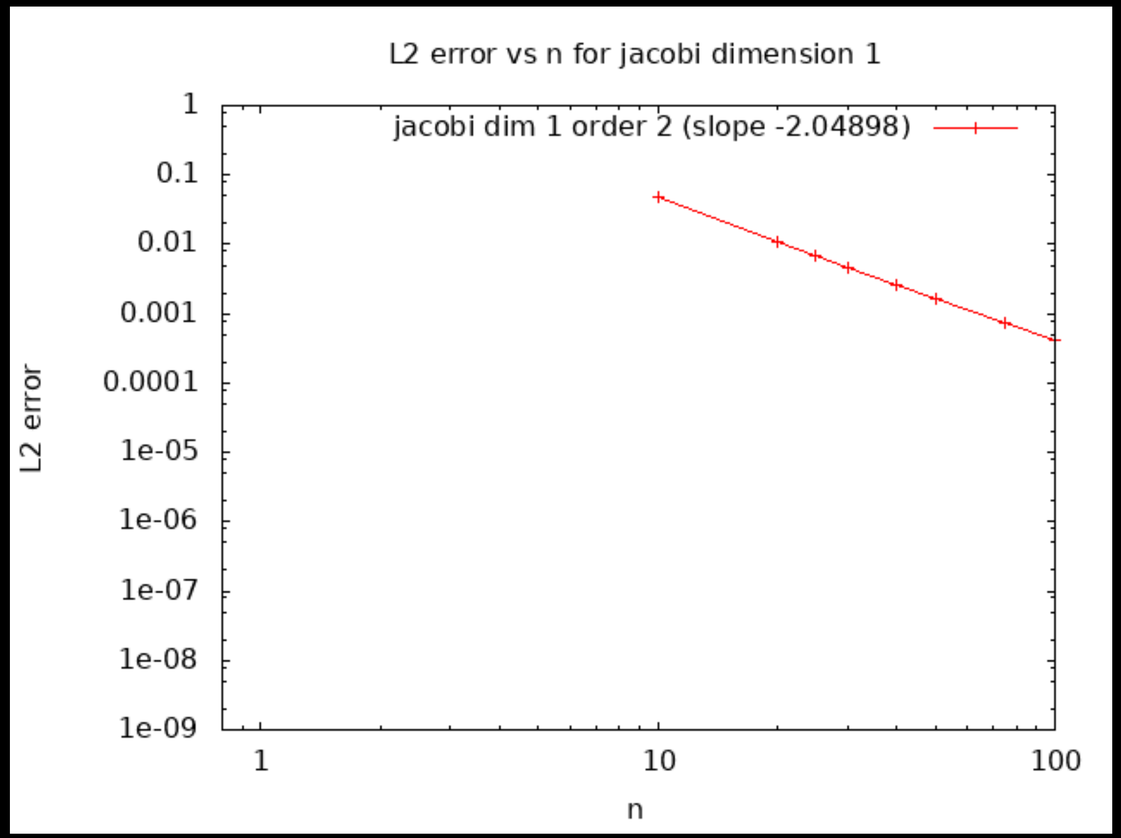


1D

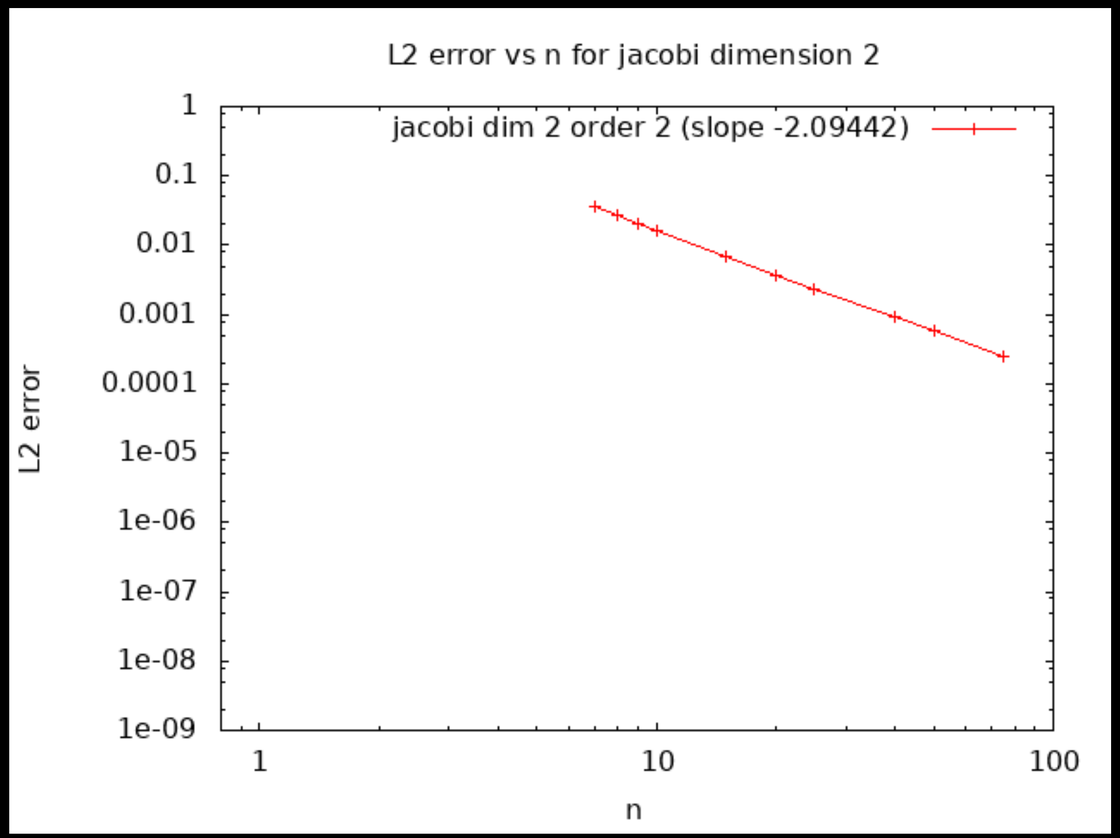


2D

Convergence plots for Jacobi solver



1D



2D

Timer results for Gauss 2D 100x100 4th order simulation

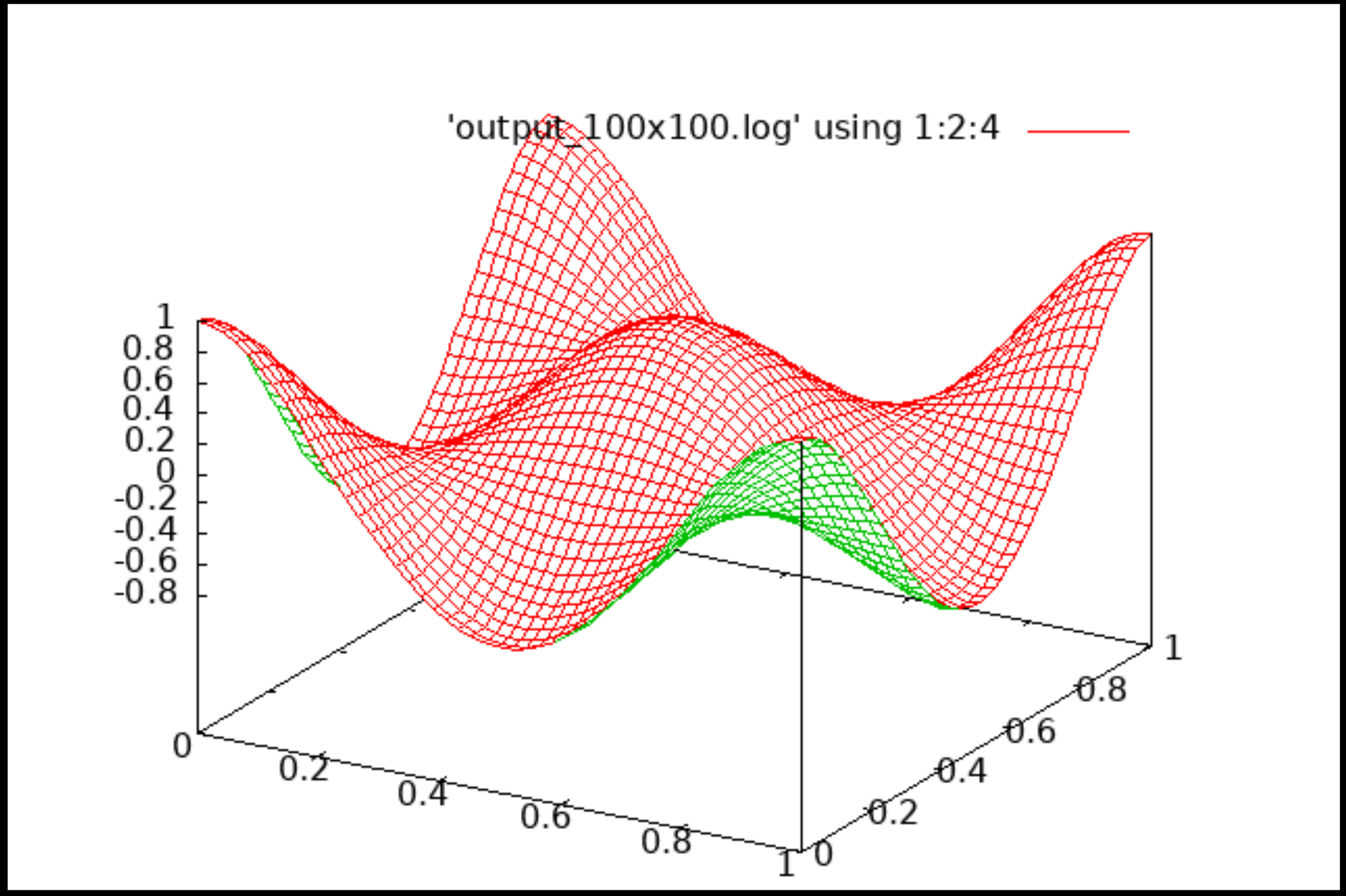
Total time taken = 4547 seconds *

```
-------------------------------------------------------------------------------------------
GRVY Performance timing - Performance Timings:                    |      Mean      Variance       Count
--> gauss                              : 4.54536e+03 secs ( 99.9662 %) | [4.54536e+03  0.00000e+00          1]
--> l2_norm                            : 5.21321e-01 secs (  0.0115 %) | [1.49162e-05  2.77484e-08      34950]
--> matrix_order4_dim2                 : 4.56124e-01 secs (  0.0100 %) | [4.56124e-01  0.00000e+00          1]
--> print_matrix_A                     : 3.68663e-01 secs (  0.0081 %) | [3.68663e-01  0.00000e+00          1]
--> write_results_output_file          : 1.22107e-01 secs (  0.0027 %) | [1.22107e-01  0.00000e+00          1]
--> main                               : 3.95012e-02 secs (  0.0009 %) | [3.95012e-02  0.00000e+00          1]
--> print_verification_mode            : 1.77879e-02 secs (  0.0004 %) | [1.77879e-02  0.00000e+00          1]
--> q_order4_dim2                      : 3.11208e-03 secs (  0.0001 %) | [3.11208e-03  0.00000e+00          1]
--> T_exact_order4_dim2                : 2.11692e-03 secs (  0.0000 %) | [2.11692e-03  0.00000e+00          1]
--> print_compare_q_Texact_Tcomputed   : 3.91006e-05 secs (  0.0000 %) | [3.91006e-05  0.00000e+00          1]
--> print_vector_q                     : 3.81470e-05 secs (  0.0000 %) | [3.81470e-05  0.00000e+00          1]
--> assemble_A                         : 5.96046e-06 secs (  0.0000 %) | [5.96046e-06  0.00000e+00          1]
--> solve                              : 3.81470e-06 secs (  0.0000 %) | [3.81470e-06  0.00000e+00          1]
--> assemble_q                         : 9.53674e-07 secs (  0.0000 %) | [9.53674e-07  0.00000e+00          1]
--> GRVY_Unassigned                    : 3.84521e-03 secs (  0.0001 %)

                 Total Measured Time = 4.54689e+03 secs (100.0000 %)
-------------------------------------------------------------------------------------------
```

GRVY summarized output of time taken by various subroutines

* This is very slightly different from what was committed to shared git repo, since this is a different run.

2D Gauss-Seidel 100x100 mesh 4th order solution
Tools used - gnuplot splot



Gauss - Seidel 100x100 mesh 4th order solution

Valgrind result

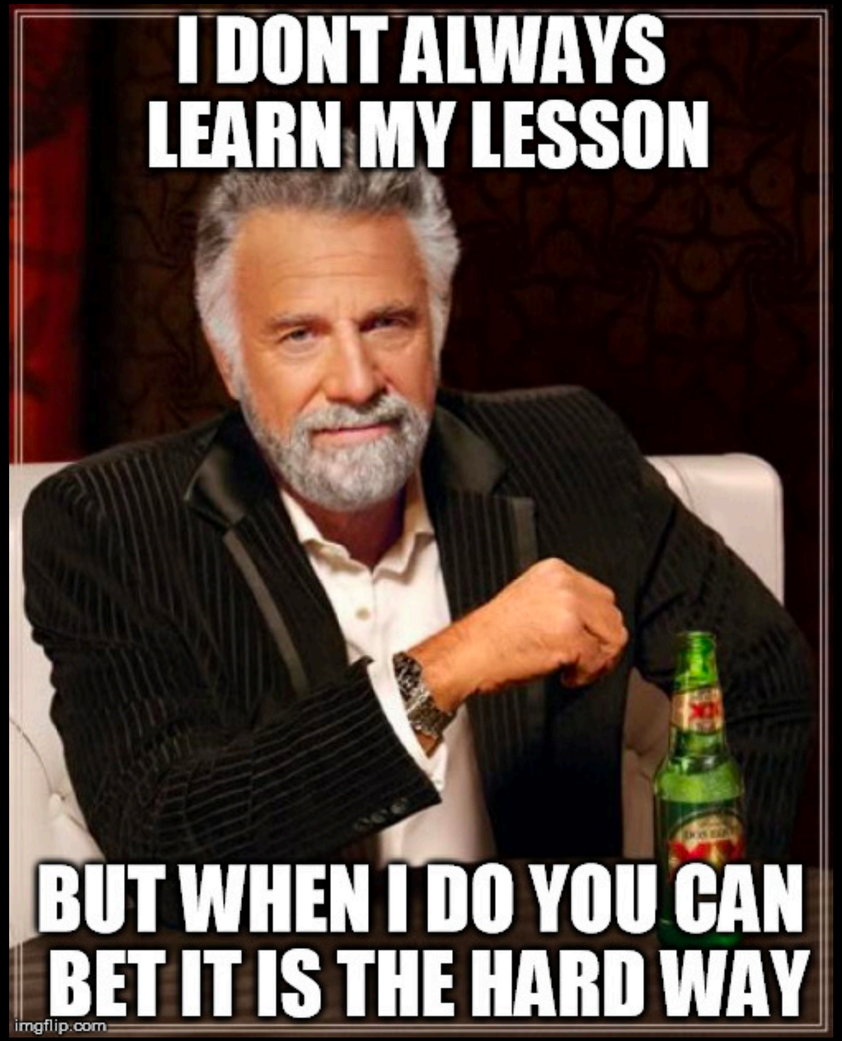
Gauss-Seidel 2D 25x25 mesh 4th order simulation

```
==219781==
==219781== HEAP SUMMARY:
==219781==     in use at exit: 116,137 bytes in 1,218 blocks
==219781==   total heap usage: 3,509 allocs, 2,291 frees, 3,473,027 bytes allocated
==219781==
==219781== LEAK SUMMARY:
==219781==    definitely lost: 24,064 bytes in 45 blocks
==219781==    indirectly lost: 88,209 bytes in 1,164 blocks
==219781==      possibly lost: 0 bytes in 0 blocks
==219781==    still reachable: 3,864 bytes in 9 blocks
==219781==         suppressed: 0 bytes in 0 blocks
==219781== Rerun with --leak-check=full to see details of leaked memory
==219781==
==219781== For lists of detected and suppressed errors, rerun with: -s
==219781== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Valgrind output to std::out - The last line means that everything’s alright!

Lessons learned

- How did you start out?
    - With solvers and matrix assembly, and compared results with MATLAB.
- What worked, what didn’t?
    - Struggled a bit with pointers initially, also struggled a lot with GNUplot. Python seems way better for plotting.
    - Used memcpy without understanding fully, which caused many issues.
- What would you do differently if starting over?
    - Use valgrind sooner and implement regression tests much earlier.
    - Exit with error status if defensive checks failed.
- Other wisdom
    - new()/delete() and malloc()/free() are non-interchangeable entities.
    - new()/delete() are more appropriate for C++.
    - git add -p allows you to commit individual lines or snippets.



Thank you!

Feel free to ask any questions!