Exercise 1

I set the timeron flag by creating a timer.flag file.
With that I saw benchmark times of about 6s and that the functions *psinv* and *resid* took a lot of time.
Also I saw that mg3P.constprop.4, the optimized version of mg3p spent quite a lot of time in the children calls of those two.

Output with timeron set:

```
 Benchmark

 No input file. Using compiled defaults
 Size:  256x 256x 256  (class B)
 Iterations:                    20

  iter   1
  iter   5
  iter  10
  iter  15
  iter  20

 Benchmark completed
 VERIFICATION SUCCESSFUL
 L2 Norm is  1.8005644013551E-06
 Error is    6.6330115975290E-14


 Benchmark Completed.
 Class           =                        B
 Size            =             256x 256x 256
 Iterations      =                       20
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                    3.3.1
  SECTION   Time (secs)
  benchmk :    5.307  (100.00%)
  mg3P    :    3.984  ( 75.07%)
  psinv   :    1.331  ( 25.09%)
  resid   :    2.649  ( 49.91%)
    --> mg-resid:    1.377  ( 25.95%)
  rprj3   :    0.635  ( 11.96%)
  interp  :    0.493  (  9.28%)
  norm2   :    0.051  (  0.97%)
  comm3   :    0.100  (  1.88%)
Execution finished. Running gprof...
Appending gprof analysis to log file...
Flat profile:

Each sample counts as 0.01 seconds.
```

```
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 42.72      2.84     2.84      147    19.33    22.24  resid
 20.16      4.18     1.34      168     7.98    10.49  psinv
 14.44      5.14     0.96   131072     0.01     0.01  vranlc
 10.83      5.86     0.72      147     4.90     7.42  rprj3
  7.67      6.37     0.51      147     3.47     3.47  interp
  3.91      6.63     0.26      485     0.54     2.52  norm2u3
  0.00      6.63     0.00   131642     0.00     0.00  randlc
  0.00      6.63     0.00     1123     0.00     0.00  timer_start
  0.00      6.63     0.00     1119     0.00     0.00  timer_stop

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
           function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
           the function in the gprof listing. If the index is
           in parenthesis it shows where it would appear in
           the gprof listing if it were to be printed.

                     Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) for 0.15% of 6.63 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    0.00    6.63                 mg3P.constprop.2 [1]
                2.84    0.43     147/147         resid [2]
                1.34    0.42     168/168         psinv [3]
                0.72    0.37     147/147         rprj3 [5]
                0.51    0.00     147/147         interp [7]
-----------------------------------------------
```

```
              2.84    0.43     147/147         mg3P.constprop.2 [1]
[2]     49.3    2.84    0.43     147         resid [2]
              0.09    0.34     170/485         norm2u3 [4]
              0.00    0.00     340/1119        timer_stop [10]
              0.00    0.00     170/1123        timer_start [9]
-----------------------------------------------
              1.34    0.42     168/168         mg3P.constprop.2 [1]
[3]     26.6    1.34    0.42     168         psinv [3]
              0.09    0.33     168/485         norm2u3 [4]
              0.00    0.00     336/1119        timer_stop [10]
              0.00    0.00     168/1123        timer_start [9]
-----------------------------------------------
                                   8             norm2u3 [4]
              0.08    0.29     147/485         rprj3 [5]
              0.09    0.33     168/485         psinv [3]
              0.09    0.34     170/485         resid [2]
[4]     18.4    0.26    0.96     485+8       norm2u3 [4]
              0.96    0.00  131072/131072      vranlc [6]
              0.00    0.00  131642/131642      randlc [8]
              0.00    0.00     491/1123        timer_start [9]
              0.00    0.00       2/1119        timer_stop [10]
                                   8             norm2u3 [4]
-----------------------------------------------
              0.72    0.37     147/147         mg3P.constprop.2 [1]
[5]     16.4    0.72    0.37     147         rprj3 [5]
              0.08    0.29     147/485         norm2u3 [4]
              0.00    0.00     294/1119        timer_stop [10]
              0.00    0.00     147/1123        timer_start [9]
-----------------------------------------------
              0.96    0.00  131072/131072      norm2u3 [4]
[6]     14.5    0.96    0.00  131072         vranlc [6]
-----------------------------------------------
              0.51    0.00     147/147         mg3P.constprop.2 [1]
[7]      7.7    0.51    0.00     147         interp [7]
              0.00    0.00     147/1119        timer_stop [10]
              0.00    0.00     147/1123        timer_start [9]
-----------------------------------------------
              0.00    0.00  131642/131642      norm2u3 [4]
[8]      0.0    0.00    0.00  131642         randlc [8]
-----------------------------------------------
              0.00    0.00     147/1123        rprj3 [5]
              0.00    0.00     147/1123        interp [7]
              0.00    0.00     168/1123        psinv [3]
              0.00    0.00     170/1123        resid [2]
              0.00    0.00     491/1123        norm2u3 [4]
[9]      0.0    0.00    0.00    1123         timer_start [9]
-----------------------------------------------
              0.00    0.00       2/1119        norm2u3 [4]
              0.00    0.00     147/1119        interp [7]
              0.00    0.00     294/1119        rprj3 [5]
              0.00    0.00     336/1119        psinv [3]
              0.00    0.00     340/1119        resid [2]
```

```
[10]     0.0    0.00    0.00    1119         timer_stop [10]
-----------------------------------------------

 This table describes the call tree of the program, and was sorted by
 the total amount of time spent in each function and its children.

 Each entry in this table consists of several lines.  The line with the
 index number at the left hand margin lists the current function.
 The lines above it list the functions that called this function,
 and the lines below it list the functions this one called.
 This line lists:
     index      A unique number given to each element of the table.
                Index numbers are sorted numerically.
                The index number is printed next to every function name so
                it is easier to look up where the function is in the table.

     % time     This is the percentage of the `total' time that was spent
                in this function and its children.  Note that due to
                different viewpoints, functions excluded by options, etc,
                these numbers will NOT add up to 100%.

     self       This is the total amount of time spent in this function.

     children   This is the total amount of time propagated into this
                function by its children.

     called     This is the number of times the function was called.
                If the function called itself recursively, the number
                only includes non-recursive calls, and is followed by
                a `+' and the number of recursive calls.

     name       The name of the current function.  The index number is
                printed after it.  If the function is a member of a
                cycle, the cycle number is printed between the
                function's name and the index number.


 For the function's parents, the fields have the following meanings:

     self       This is the amount of time that was propagated directly
                from the function into this parent.

     children   This is the amount of time that was propagated from
                the function's children into this parent.

     called     This is the number of times this parent called the
                function `/' the total number of times the function
                was called.  Recursive calls to the function are not
                included in the number after the `/'.

     name       This is the name of the parent.  The parent's index
                number is printed after it.  If the parent is a
```

member of a cycle, the cycle number is printed between
the name and the index number.

If the parents of the function cannot be determined, the word
`<spontaneous>' is printed in the `name' field, and all the other
fields are blank.

For the function's children, the fields have the following meanings:

self This is the amount of time that was propagated directly
from the child into the function.

children This is the amount of time that was propagated from the
child's children to the function.

called This is the number of times the function called
this child `/' the total number of times the child
was called. Recursive calls by the child are not
listed in the number after the `/'.

name This is the name of the child. The child's index
number is printed after it. If the child is a
member of a cycle, the cycle number is printed
between the name and the index number.

If there are any cycles (circles) in the call graph, there is an
entry for the cycle-as-a-whole. This entry shows who called the
cycle (as parents) and the members of the cycle (as children.)
The `+' recursive calls entry shows the number of function calls that
were internal to the cycle, and the calls entry for each member shows,
for that member, how many times it was called from other members of
the cycle.

Index by function name

```
   [7] interp (real.c)        [8] randlc                [9] timer_start
   [4] norm2u3 (real.c)       [2] resid (real.c)       [10] timer_stop
   [3] psinv (real.c)         [5] rprj3 (real.c)        [6] vranlc
```
Job complete.

```
int i3, i2, i1;
if (timeron) timer_start(T_resid);
#pragma omp parallel for private(i3, i2, i1) collapse(2) schedule(dynamic)
for (i3 = 1; i3 < n3-1; i3++) {
        for (i2 = 1; i2 < n2-1; i2++) {
                double u1[M], u2[M]; //chat gpt proposed this, but seemed to slow down the
program a bit compared to declaration outside + private copies
                for (i1 = 0; i1 < n1; i1++) {
                        u1[i1] = u[i3][i2-1][i1] + u[i3][i2+1][i1]
                        + u[i3-1][i2][i1] + u[i3+1][i2][i1];
                        u2[i1] = u[i3-1][i2-1][i1] + u[i3-1][i2+1][i1]
                        + u[i3+1][i2-1][i1] + u[i3+1][i2+1][i1];
                }
                for (i1 = 1; i1 < n1-1; i1++) {
                        r[i3][i2][i1] = v[i3][i2][i1]
                        - a[0] * u[i3][i2][i1]
                        //-------------------------------------------------------------------
                        /               / Assume a[1] = 0 (Enable 2 lines below if a[1] not= 0)
                        //-------------------------------------------------------------------
                        // - a[1] * ( u[i3][i2][i1-1] + u[i3][i2][i1+1]
                        // + u1[i1] )
                        //-------------------------------------------------------------------
                        - a[2] * ( u2[i1] + u1[i1-1] + u1[i1+1] )
                        - a[3] * ( u2[i1-1] + u2[i1+1] );
                }
        }
}
```

Finally I optimized interp and rprj3 too to get some more speedup, basically identical procedure. The random generator vranlc also took a lot of time, but there were some dependencies and simple multithreading would have made the result non-deterministic though potentially still a valid generator.

Normal output with timeron set:

```
 Benchmark

 No input file. Using compiled defaults
 Size:  256x 256x 256  (class B)
 Iterations:                     20

  iter    1
  iter    5
  iter   10
  iter   15
  iter   20

 Benchmark completed
 VERIFICATION SUCCESSFUL
 L2 Norm is  1.8005644013551E-06
 Error is    6.6330115975290E-14


 Benchmark Completed.
 Class           =                        B
 Size            =            256x 256x 256
 Iterations      =                       20
 Operation type  =           floating point
 Verification    =               SUCCESSFUL
 Version         =                    3.3.1
  SECTION   Time (secs)
  benchmk :    2.712  (100.00%)
  mg3P    :    1.929  ( 71.12%)
  psinv   :    0.603  ( 22.23%)
  resid   :    1.415  ( 52.16%)
    --> mg-resid:    0.686  ( 25.29%)
  rprj3   :    0.146  (  5.38%)
  interp  :    0.321  ( 11.83%)
  norm2   :    0.054  (  2.01%)
  comm3   :    0.123  (  4.54%)
Execution finished. Running gprof...
Appending gprof analysis to log file...
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 95.66     25.16    25.16                             setup
  3.65     26.12     0.96   131072     0.01     0.01  vranlc
  0.72     26.31     0.19      485     0.39     2.37  norm2u3
  0.00     26.31     0.00   131642     0.00     0.00  randlc
  0.00     26.31     0.00     1123     0.00     0.00  timer_start
  0.00     26.31     0.00     1119     0.00     0.00  timer_stop
  0.00     26.31     0.00      168     0.00     2.37  psinv
  0.00     26.31     0.00      147     0.00     0.00  interp
  0.00     26.31     0.00      147     0.00     2.74  resid

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.
 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
           function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
           the function in the gprof listing. If the index is
           in parenthesis it shows where it would appear in
           the gprof listing if it were to be printed.


                     Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) for 0.04% of 26.31 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]     95.6   25.16    0.00                 setup [1]
-----------------------------------------------
                                                 <spontaneous>
[2]      4.4    0.00    1.15                 mg3P.constprop.8 [2]
                0.00    0.40     147/147         resid [5]
                0.00    0.40     168/168         psinv [6]
                0.06    0.29     147/485         norm2u3 [3]
                0.00    0.00     294/1119        timer_stop [9]
                0.00    0.00     147/1123        timer_start [8]
                0.00    0.00     147/147         interp [10]
-----------------------------------------------
                                   8             norm2u3 [3]
                0.06    0.29     147/485         mg3P.constprop.8 [2]
                0.07    0.33     168/485         psinv [6]
                0.07    0.34     170/485         resid [5]
[3]      4.4    0.19    0.96     485+8       norm2u3 [3]
                0.96    0.00  131072/131072      vranlc [4]
                0.00    0.00  131642/131642      randlc [7]
                0.00    0.00     491/1123        timer_start [8]
                0.00    0.00       2/1119        timer_stop [9]
                                   8             norm2u3 [3]
-----------------------------------------------
                0.96    0.00  131072/131072      norm2u3 [3]
[4]      3.7    0.96    0.00  131072         vranlc [4]
-----------------------------------------------
                0.00    0.40     147/147         mg3P.constprop.8 [2]
[5]      1.5    0.00    0.40     147         resid [5]
                0.07    0.34     170/485         norm2u3 [3]
                0.00    0.00     340/1119        timer_stop [9]
                0.00    0.00     170/1123        timer_start [8]
```

```
-----------------------------------------------
                0.00    0.40     168/168         mg3P.constprop.8 [2]
[6]      1.5    0.00    0.40     168         psinv [6]
                0.07    0.33     168/485         norm2u3 [3]
                0.00    0.00     336/1119        timer_stop [9]
                0.00    0.00     168/1123        timer_start [8]
-----------------------------------------------
                0.00    0.00  131642/131642      norm2u3 [3]
[7]      0.0    0.00    0.00  131642         randlc [7]
-----------------------------------------------
                0.00    0.00     147/1123        interp [10]
                0.00    0.00     147/1123        mg3P.constprop.8 [2]
                0.00    0.00     168/1123        psinv [6]
                0.00    0.00     170/1123        resid [5]
                0.00    0.00     491/1123        norm2u3 [3]
[8]      0.0    0.00    0.00    1123         timer_start [8]
-----------------------------------------------
                0.00    0.00       2/1119        norm2u3 [3]
                0.00    0.00     147/1119        interp [10]
                0.00    0.00     294/1119        mg3P.constprop.8 [2]
                0.00    0.00     336/1119        psinv [6]
                0.00    0.00     340/1119        resid [5]
[9]      0.0    0.00    0.00    1119         timer_stop [9]
-----------------------------------------------
                0.00    0.00     147/147         mg3P.constprop.8 [2]
[10]     0.0    0.00    0.00     147         interp [10]
                0.00    0.00     147/1119        timer_stop [9]
                0.00    0.00     147/1123        timer_start [8]
-----------------------------------------------

 This table describes the call tree of the program, and was sorted by
 the total amount of time spent in each function and its children.

 Each entry in this table consists of several lines.  The line with the
 index number at the left hand margin lists the current function.
 The lines above it list the functions that called this function,
 and the lines below it list the functions this one called.
 This line lists:
     index      A unique number given to each element of the table.
                Index numbers are sorted numerically.
                The index number is printed next to every function name so
                it is easier to look up where the function is in the table.

     % time     This is the percentage of the `total' time that was spent
                in this function and its children.  Note that due to
                different viewpoints, functions excluded by options, etc,
                these numbers will NOT add up to 100%.

     self       This is the total amount of time spent in this function.

     children   This is the total amount of time propagated into this
                function by its children.
```

called	This is the number of times the function was called. If the function called itself recursively, the number only includes non-recursive calls, and is followed by a `+' and the number of recursive calls.
name	The name of the current function. The index number is printed after it. If the function is a member of a cycle, the cycle number is printed between the function's name and the index number.

For the function's parents, the fields have the following meanings:

self	This is the amount of time that was propagated directly from the function into this parent.
children	This is the amount of time that was propagated from the function's children into this parent.
called	This is the number of times this parent called the function `/' the total number of times the function was called. Recursive calls to the function are not included in the number after the `/'.
name	This is the name of the parent. The parent's index number is printed after it. If the parent is a member of a cycle, the cycle number is printed between the name and the index number.

If the parents of the function cannot be determined, the word
`<spontaneous>' is printed in the `name' field, and all the other
fields are blank.

For the function's children, the fields have the following meanings:

self	This is the amount of time that was propagated directly from the child into the function.
children	This is the amount of time that was propagated from the child's children to the function.
called	This is the number of times the function called this child `/' the total number of times the child was called. Recursive calls by the child are not listed in the number after the `/'.
name	This is the name of the child. The child's index number is printed after it. If the child is a member of a cycle, the cycle number is printed between the name and the index number.

If there are any cycles (circles) in the call graph, there is an
entry for the cycle-as-a-whole. This entry shows who called the
cycle (as parents) and the members of the cycle (as children.)
The `+' recursive calls entry shows the number of function calls that
were internal to the cycle, and the calls entry for each member shows,
for that member, how many times it was called from other members of
the cycle.

```
Index by function name

  [10] interp (real.c)         [7] randlc                  [8] timer_start
   [3] norm2u3 (real.c)        [5] resid (real.c)          [9] timer_stop
   [6] psinv (real.c)          [1] setup (real.c)          [4] vranlc
Job complete.
```