Technische Universität München
Institut für Informatik
Dr. Tobias Neckel
Atanas Atanasov
Kristof Unterweger

WT 2011/2012

Lab Course
Scientific Computing

Worksheet 5

distributed: 21.12.2011
due to: 16.01.2012, 6:00 pm (per email to `unterweg@in.tum.de` and `atanasoa@in.tum.de`
personal presentation: 17.01.2012 (exact slots will be announced)

As we have seen in worksheet 3 and 4, we need solvers for large sparse systems of linear equations both to solve stationary equations and to apply implicit time discretizations. In this worksheet, we will examine the performance of the simple Gauss-Seidel relaxation and try to find a better method.

a) Solve the stationary boundary value problem

$$T_{xx} + T_{yy} = -2\pi^2 \sin(\pi x)\sin(\pi y) \text{ forall } (x,y) \text{ in }]0;1[^2 \quad (1)$$

$$T(x,y) = 0 \text{ forall } (x,y) \text{ in } \delta]0;1[^2 \quad (2)$$

from worksheet 3 with the Gauss-Seidel solver implemented in worksheet 3. Use $T(x,y) = 1$ for all (x,y) in $]0;1[^2$ as an initial guess for the solution. Record the number of iterations, the runtime and the storage requirements to achieve an accuracy of 10^{-4} for different grid resolutions:

$N_x = N_y$	3	7	15	31	63	127	255
# iterations							
factor	—						
runtime (sec)							
memory (floats)							

c) The exact solution of our system of equations is

$$F = \sin(\pi x)\sin(\pi y).$$

Plot the error

$$E = T - F$$

for $N_x = N_y = 255$ after 10, 20, 30 Gauss-Seidel iterations.

c) Modify the Gauss-Seidel solver to an SOR solver using the resolution dependend overrelaxation factor

$$\omega = \frac{2}{1 + \sin(\pi * h)}$$

with the mesh width $h = \frac{1}{N_x+1} = \frac{1}{N_y+1}$.

d) Solve the stationary boundary value problem from **a)** with the SOR solver implemented in **b)**. Use $T(x, y) = 1$ for all (x, y) in $]0; 1[^2$ as an initial guess for the solution. Record the number of iterations, the runtime and the storage requirements to achieve an accuracy of 10^{-4} for different grid resolutions:

$N_x = N_y$	3	7	15	31	63	127	255
# iterations							
factor	—						
runtime (sec)							
memory (floats)							

By which factor could we reduce the runtime for the highest resolution ($N_x = N_y = 255$) in comparison to the Gauss-Seidel solver?

Plot the solution for the finest grid.

e) Implement the components of a simple multigrid solver:

1) Implement a Gauss-Seidel smoother performing two Gauss-Seidel iterations as a function of the current approximation of the solution, the right hand side, and N_x and N_y on the respective grid level.
2) Implement a function computing the residual as a function of the current approximation of the solution, the right hand side, N_x and N_y.
3) Implement the restriction (injection) as a function of the fine grid function and the grid resolutions N_x and N_y of the coarse grid.
4) Implement the interpolation (bilinear) as a function of the coarse grid function and the grid resolutions N_x and N_y of the fine grid.

f) Implement a function performing one iteration of a multigrid solver (v-cycle) as a function of the current approximation of the solution, the right hand side, and the grid resolutions N_x and N_y. The output of the function is the new approximation of the solution and the residual norm after the iteration.

Hint: recursivity!!!

g) Implement a multigrid solver as a function of the right hand side and the grid resolutions N_x and N_y. The output of the function is the approximate solution. Iterate up to an accuracy of 10^{-4} (measured by the residual norm).

h) Solve the stationary boundary value problem from **a)** with the multigrid solver. Use $T(x, y) = 1$ for all (x, y) in $]0; 1[^2$ as an initial guess for the solution. Record the number of iterations, the runtime and the storage requirements to achieve an accuracy of 10^{-4} for different grid resolutions:

$N_x = N_y$	3	7	15	31	63	127	255
# iterations							
runtime (sec)							
memory (floats)							

By which factor could we reduce the runtime for the highest resolution ($N_x = N_y = 255$) in comparison to the Gauss-Seidel and to the SOR solver?

Again plot the solution for the finest grid.

Questions:

1) Examine the computational costs for the Gauss-Seidel and the SOR method: By which factor are the costs multiplied when you double the number of grid points in each coordinate direction? Is this an optimal behaviour for a solver? Which factor would be optimal?

2) Can you give upper bounds for the memory requirements of the multigrid solver implemented (if N is the number of unknowns on the finest grid)?

3) Can you give an upper bound for the computational costs (in floating point operations) of one multigrid v-cycle (as implemented above) in dependence on the number N of unknowns on the finest grid?

4) Is the multigrid method an optimal solver in the sense of question 1)?