Diffpack® Multilevel Toolbox

This toolbox is a complete and flexible lab for constructing fast multigrid solvers.

Whether you work with pure multigrid research or you simply want to cut CPU time, this tool will help you reach your goals.

Linear Solvers at Theoretically Optimal Speed

When multilevel solvers were discovered in the mid 70'ties, many experts would simply not believe the work count to be O(N). Today, thousands of papers have established a rich theory and the optimality of these solvers is well understood.

Being super-algorithms, i.e. compositions of advanced sub-algorithms, multilevel solvers are not only the fastest but also the most complicated and configurable of all types of linear solvers. For large problem classes the solution is reached in const $\times$ N flops, but the constant can be large and it is strongly configuration and problem dependent.

Optimal problem-specific configurations are seldom known from analysis. The design of fast multilevel solvers therefore relies on experiments, something that puts high demands on the functionality and flexibility of multilevel software.

A Complete Framework for Multigrid Solvers

The Diffpack Multilevel Toolbox is a complete and flexible laboratory for building fast problem-dependent multigrid solvers. The open design also allows extensions to other multilevel solvers, e.g. based on domain decomposition techniques.

The example below shows the number of iterations and CPU time required for solving the Poisson equation ($-\nabla^2 u = f$) on the unit square using structured meshes of different sizes.

	CG/RILU		CG/MG		MG	
Size	# it	CPU	# it	CPU	# it	CPU
4 225	44	0.47	5	0.14	7	0.14
16 641	81	3.69	5	0.64	7	0.67
66 049	146	27.20	5	2.69	7	2.77
263 169	265	194.60	5	11.00	7	11.02

From left to right, the different solvers are conjugate gradient (CG) with a RILU preconditioner, CG with a multigrid (MG) preconditioner and pure multigrid. The configuration of multigrid was optimized by using the multiple loop feature(1).

Simple to Use

About 10 lines of code are all it takes to make the full gallery of multigrid options available to an existing Diffpack application. When converting your program to a multigrid code it is equipped with:

- ✓ a coarse mesh and coarse mesh linear system and solver
- ✓ a hierarchy of structured or unstructured meshes
- ✓ iterative linear solvers for pre- and post-smoothing
- ✓ transfer operators between mesh levels (projections)

These items can all be initialized from input files and/or run time menu choices.

(turn page)

inuTech GmbH
Fürther Str. 212
90429 Nürnberg
Germany

Phn: +49 911 32 38 43 - 0
Fax: +49 911 32 38 43 - 43
Eml: diffpack@inutech.de

www.diffpack.com
www.inutech.de

Easy to Configure

You can select structured or unstructured meshes in 1, 2 or 3 space dimensions, and the mesh hierarchy can support both nested and non-nested multigrid. You can select the pre- and post-smoothers from Diffpack's iterative solver library or you can link in your favorite smoothers from other sources.

You have the same great flexibility when dealing with other parameters such as, e.g. projections, residuals, smoother sweeps, V- and W cycles, etc. Using the menu system you could for example set up multiple runs over different sweeps, V- and W-cycles to optimize these parameters for some given problem.

You can use the multigrid solvers directly or as preconditioners for other iterative solvers such as, e.g. the Krylov solvers available from the Diffpack Kernel.

For Industrial Production Runs or Pure Research

Obviously, the Multilevel Toolbox can give great savings in connection to large industrial calculations. Due to its run time flexibility, the Multilevel Toolbox is also the perfect "lab" for those who are mainly interested in pure multilevel research.

---

(1) For this particular example Diffpack's multiple loop feature, provided by the menu system, was used to investigate

- candidates for pre- and post smoothers
- number of sweeps for pre- and post smoothers
- V- or W-cycles
- coarse grid solver

Convergence showed up to be particularly sensitive to the selection of smoothers. The optimized configuration selected was SSOR with relaxation parameter 0.8 as pre- and post-smoother, one sweep before and after, v-cycle and gaussian elimination as solver on the coarse grid.