Towards a tool for forward and reverse mode source to source transformation in C++

*Fast*Opt http//FastOpt.com

M. Voßbeck, R. Giering, and T. Kaminski

*Fast*Opt

- Founded in February 2000 at Hamburg
- By Ralf Giering and Thomas Kaminski
- One more colleague as of July 2003: Michael Voßbeck
- Two kinds of business:
  -> Develop and provide tools for Automatic Differentiation (AD)/ Adjoint coding
  -> Consulting projects with focus on AD, Inverse Modelling, Data Assimilation
- 15 years of Experience in AD

Motivation

- Only available Source to Source tool for C(++):
  ADIC (Bischof et al., 1997;Hovland et al, 2002): restricted to forward mode
- Reverse mode AD for C(++) only implemented as operator overloading,
  e.g. ADOL-C (Griewank et al., 1996)
- Goal of this study:
  Demonstrate feasibility of reverse mode source transformation for ANSI-C
  by differentiating a test-code

Test-code

- Roe Solver (1997) of CFD-code EULSOLDO (Cusdin and Mueller, 2003)
- C code generated by f2c from original Fortran 77 version (129 lines without comments)
- yields simple C code

AD Tool

- Applies same philosophy as TAF (Giering and Kaminski, 1998)
- Uses (simplified) implementations of a subset of TAF algorithms
  (e.g. ERA, Giering and Kaminski, 2002)
- Handles only subset of language elements that are relevant for test-code:
  -> Selected datatypes: `int` and `double` in scalar, array, and typedef form
  -> Basic arithmetics: addition, subtraction, multiplication, division
  -> mathematical intrinsics of ANSI-C89:
  `sqrt, fabs, exp, pow, log, sin, tan, asin, atan, sinh, tanh,...`
  -> A few control flow structures: `for, if,` conditional expression
- Function code needs to be pre-processed (by cpp)
- Generated adjoint operates in pure mode, i.e. evaluates gradient but not function
- Adjoint of test-code comprises 650 lines (including comments)
  of well-readable C-code

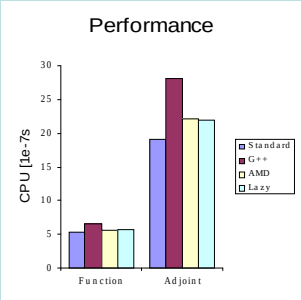
Performance

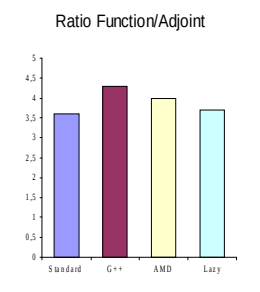
- CPU time for generated adjoint code measured in multiples
  of CPU time for function code
- Performance depends on factors such as:
  -> CPU
  -> Compiler
  -> Compiler Flags
- Tested four configurations:

Configuration	CPU	Compiler	Flags	Ratio
standard	PIV 3MHz	icc (8.0)	-O2 -ip -tpp7 -xN	3,6
g++	PIV 3MHz	g++	-O3	4,3
AMD	AMD XP1900+	icc (8.0)	-O3 -ip -tpp6	4,0
lazy	PIV 3MHz	icc (8.0)	-O2	3,7

configuration *standard* is supposed to mimic situation of a typical user with interest in high performance

Performance continued





- Using ADOL-C with the same configurations yields ratios between 16.9 and 18.9 for function plus gradient evaluation
- Forcing ADOL-C to tape on disk increases ratio to 40.4

Comparison to TAF

- TAF generates fastest adjoint code of the original Fortran version of test-code with command line options `-replaceintr -split` (Cusdin and Mueller, 2003)
- TAF achieves a performance ratio of 2.9 (in a configuration comparable to standard)
- Applying f2c to this TAF-generated Fortran adjoint yields a C adjoint that still achieves a performance ratio of 3.1 (in standard configuration)
- This indicates scope for further improvements

Conclusions

- Have demonstrated feasibility of reverse mode source transformation in C by building first reverse mode source transformation tool
- Generated code is efficient, appears faster than operator overloading
- New tool serves as starting point for design of TAC++,
  the TAF equivalent for C/C++
- New tool is valuable already as it can support hand coders of C-adjoints
- Extension of functionality will be demand-driven,
  i.e. from application to application
- TAF experience was very helpful
- Further development will benefit from well proved TAF concepts

Acknowledgements

We thank Paul Cusdin and Jens-Dominik Mueller for providing EULSOLDO in its original Fortran 77 version as well as Andrea Walther, Olaf Vogel, and Andreas Griewank for providing ADOL-C.

References

- Bischof, Christian H., Lucas Roh, and Andrew Mauer.ADIC --- An extensible automatic differentiation tool for ANSI-C. *Software--Practice and Experience*, 27(12):1427--1456, 1997.
- Cusdin, P., and J.-D. Müller. Improving the performance of code generated by automatic differentiation. *Technical Report* QUB-SAE-03-04, QUB School of Aeronautical Engineering, 2003. *Submitted to Optimization Methods and Software*.
- Cusdin, P., and J.-D. Mueller. Improving the performance of code generated by automatic differentiation. submitted to Optimization Methods and Software, 2003.
- Giering, R., and T. Kaminski, Recipes for Adjoint Code Construction, *ACM Trans. Math. Software, 24(4)*: 437-474, 1998.
- Giering, R., and T. Kaminski. Generating recomputations in reverse mode AD. In George Corliss, Andreas Griewank, Christele Faure, Laurent Hascoet, and Uwe Naumann, editors, *Automatic Differentiation of Algorithms: From Simulation to Optimization*, chapter--33, pages 283--291. Springer Verlag, Heidelberg,2002.
- Griewank, A., David Juedes, and Jean Utke. ADOL-C, a package for the automatic differentiation of algorithms written in C/C++. *ACM Trans. Math. Software*, 22(2):131--167, 1996.
- Griewank, A., David Juedes, and Jean Utke. A package for the automatic differentiation of algorithms written in C/C++. User manual. Technical report, Institute of Scientific Computing, Technical University of Dresden, Dresden, Germany, 1996. See www.math.tu-dresden.de/wir/project/wwwadolc/index.html.
- Hovland, P., B. Norris, and B. Smith. Making automatic differentiation truly automatic: Coupling PETSc with ADIC. In P. M. A. Sloot, C. J. K. Tan, J. J. Dongarra, and A. G. Hoekstra, editors, *Computational Science -- ICCS 2002, Proceedings of the International Conference on Computational Science, Amsterdam, The Netherlands, April 21--24, 2002. Part II, volume 2330 of Lecture Notes in Computer Science*, pages 1087--1096, Berlin, 2002. Springer.