TAF and TAC++: Source-to-source transformation in Fortran and C

Ralf Giering, Thomas Kaminski, and Michael Voßbeck

***Fast*Opt**

Web: http://www.FastOpt.com

DMV, Rostock, September 2003

Outline

- FastOpt tools: TAF
- FastOpt projects
- FastOpt tools: TAC++
- Conclusions

TAF: Transformation of Algorithms in Fortran

- Source-to-source translator for Fortran-77/95
- Commercial successor of TAMC
- Forward and reverse mode (1st derivatives):
  Tangent linear and adjoint models
- Scalar and vector mode
- Efficient Hessian (2nd derivative) code
  by applying TAF twice (e.g. forward over reverse)
- Command line program with many options
- TAF-Directives are Fortran comments
- Extensive and complex code analyses
  (similar to optimising compilers)
- Generated code is structured and well readable

TAF
More features

- Generation of flexible storing/reading scheme for required variables triggered by TAF init and store directives
- Generation of checkpointing scheme triggered by combination of TAF init and store directives
- Generation of efficient storing/reading scheme (Christianson, 1996, 1998) for adjoints of converging iterations triggered by TAF loop directive
- TAF flow directives for black-box routines, or to include user provided derivative code (exploit self-adjointness, MPI wrappers, etc...)
- Automatic Sparsity Detection
- Basic support for MPI and OpenMP

TAF
Ongoing Development

- **TAF is constantly being adapted to new Fortran standards**
  - **Fortran 2000**
  - **OpenMP 2**
- **TAF code analyses are constantly being extended**
- **TAF algorithms are constantly being improved and adapted to the needs of the users**
- **FastOpt is giving support for TAF users**
- **FastOpt is offering consulting for AD and further projects**

some larger TAF Derivatives

Model (Who)	Lines	Lang	TLM	ADM	Ckp	HES
NASA/NCAR (w. Todling & Lin)	87'000	F90	2.7	6.8	2 lev	-
MOM3 (Galanti & Tziperman)	50'000	F77	Yes	4.6	2 lev	-
MITGCM (ECCO Consortium)	100'000	F77	1.8	5.5	3 lev	11.0/1
BETHY (w. Knorr, Rayner, Scholze)	5'400	F90	1.5	3.6	2 lev	12.5/5
Nav.-Stokes-Solver (Hinze, Slawig)	450	F77	-	2.0	steady	-
NSC2KE	2'500	F77	2.4	3.4	steady	9.8/1
HB_AIRFOIL (Thomas & Hall)	8'000	F90	-	3.0		-

- Lines: total number of Fortran lines without comments
- Numbers for TLM and ADM give CPU time for (function + gradient) relative to forward model
- HES format: CPU time for Hessian * n vectors rel. t. forw. model/ n
- 2 (3) level checkpointing costs 1 (2) additional model run(s)

Performance compared to hand coded adjoints

Code	Hand	TAF/TAMC	relativ
EPT (MINPACK-2)	1.5	1.9	26%
GL1 (MINPACK-2)	1.5	1.7	13%
GL2 (MINPACK-2)	2.1	1.3	-40%
MSA (MINPACK-2)	1.75	1.6	-9%
PJB (MINPACK-2)	1.75	2.2	+25%
SSC (MINPACK-2)	1.18	1.13	-5%
Nav.-Stokes (H & S)	1.9	1.3	-30%

=> Performance of TAF generated adjoints is comparable to that of hand written adjoints

Outline

- FastOpt tools: TAF
- FastOpt projects
- FastOpt tools: TAC++
- Conclusions

Ocean Data Assimilation
ECCO Consortium

- MIT-GCM: Primitive Equation Model of General Oceanic Circulation
- Various Configurations ~ 100'000 lines of Fortran 77 (without comments)
- Uses MPI and OpenMP for parallelisation
- Tangent Linear, Adjoint, Hessian code generated by TAF
  Only hand written code for communication wrappers
- Adjoint uses 2 or 3 level checkpointing
- Is used for Variational Data Assimilation, Uncertainty Analysis, Kalman Filter and Sensitivity Studies
- AD for further compoments in progress: biogeochemistry and atmosphere

Courtesy: Patrick Heimbach, MIT

ECCO state estimation: problem size

- Dimensionality:
  - grid @ 1°×1° resolution: $n_x \cdot n_y \cdot n_z$ = 360 · 160 · 23 — $1,324,800$
  - model state: 17 3D + 2 2D fields — $\sim 2 \cdot 10^7$
  - timesteps: 10 years @ 1-hour time step — $87,600$
  - control vector — $\sim 1 \cdot 10^8$
    - initial temperature (T), salinity (S)
    - time-dependent surface forcing (every 2 days)
  - cost function: observational elements: — $\sim 1 \cdot 10^8$
- Computational size:
  - 60 processors (15 nodes) @ 512MB per proc.
  - I/O: 10 GB input, 35GB output
  - time: 59 hours per iteration @ 60 processors
- What we would ideally want:
  - 1/10°×1/10° resol., 1000 years, full model error covariance ...

Aerodynamics: NSC2KE
with T. Slawig, TU-Berlin

- Model: NSC2KE by Bijan Mohammadi (1994)
- Mixed finite volume-finite element Galerkin CFD model
- 2 dimensional on unstructured grid
- Euler Part with Roe-, Osher-, or Kinematic solver
- K-epsilon turbulence model
- 4^{th} order Runge Kutta
- 2500 lines of Fortran 77 code without comments
- Previous AD applications of this code by:
  - Mohammadi et al. (1994) w/Odyssée (Rostaing et al, 1993)
  - Hovland et al. (1997), Slawig (2001) w/ADIFOR (Bischof et al. '96)
  - S. Ulbrich (2001) w/TAMC (Giering, 1997)

Aerodynamics
Automatic Differentiation of NSC2KE

Adjoint:
- **16 store directives inserted**
- **Used TAF iteration directive to trigger efficient write/read scheme (Christianson, 1996, 1998): stores only steady state values of required variables**
- **Required variables kept in memory: 1.5 MB**
- **CPU(gradient+function)/CPU(function) = 3.4**

Tangent linear:
- **CPU(derivative+function)/CPU(function) = 2.4**

Hessian:
- **Forward over Reverse mode**
- **CPU(Hessian*1 vector)/CPU(function) = 9.8**

Outline

- FastOpt tools: TAF
- FastOpt projects
- FastOpt tools: TAC++
- Conclusions

Reverse mode

- Required variables must be provided in reverse order of computation
- Required variables can be taped (stored/read) or recomputed
- Efficient code uses a combination of taping and recomputation
- TAF, TAC++ do recomputations by default, Taping is triggered by directives in a very flexible way

TAC++, Transformation of Algorithms in C++

- **Source-to-source Automatic Differentiation (AD) tool for C++**
- **First reverse mode source-to-source C-tool**
- **Uses same algorithm as our Fortran tool TAF**
  - **Can achieve comparable performance**
  - **will be as flexible (directives, options)**
- **First prototype for tool design experiments**

Source-to-source AD

- Components
  - Front end (TAF,TAC++)
    - Scanner, parser, semantic analysis
  - Normalization (TAF,TAC++)
  - Data dependence (TAF)
  - Data flow (global) (TAF)
  - Transformation (TAF,TAC++)
  - Back end (TAF,TAC++)
    - Source code writer

TAC++ feasibilty test

- **2D Euler model code EULSOLDO by Jens-Domenic Müller (1991), thanks to Paul Cusdin!**
  - **Roe flux solver**
  - **Fortran-77 code converted with f2c**
  - **cpp preprocessed to be handled by TAC++**
- **Adjoint C code generated by TAC++**
  - **Adjoint code verified**
  - **Performance of code almost comparable with TAF generated code**

TAC++ demo

- **tac++ roeflux.c**

Challenges in C++/Fortran-90

- Dynamic memory (TAF)
- Structured types (TAF)
- Accessing private variables
- Pointers
  (static:TAF)
- Generic functions
- Operator overloading

Additional challenges in C++

- Classes
  - Constructures, destructures
  - Heritage
  - Virtual methods
- Templates
- STL, Standard Template Library
  - Class complex
  - Class valarray
- Implicit type casts
- Exception handling

Conclusions

- TAF has generated efficient derivative code for large models
- FastOpt has started to develop a source-to-source AD-tool for C++ programs: TAC++
- prototype already useful for real applications
- Reverse mode was main challenge, forward mode will be easier
- Handles subset of C: further development driven by applications
- Need for code preparations will gradually decrease
- TAC++ and TAF (Fortran-2000) development will go hand in hand