Adjoint and Hessian of a Satellite Code written in C

Michael Voßbeck
FastOpt

joint work with
Thomas Lavergne and Bernard Pinty (JRC),
Ralf Giering and Thomas Kaminski (FastOpt)

Copy of presentation at http://www.FastOpt.com

Outline

- Overview TAC++
- Twostreams Model
- Performance
- Demonstration
- Recent AD of Fortran CFD

TAC++

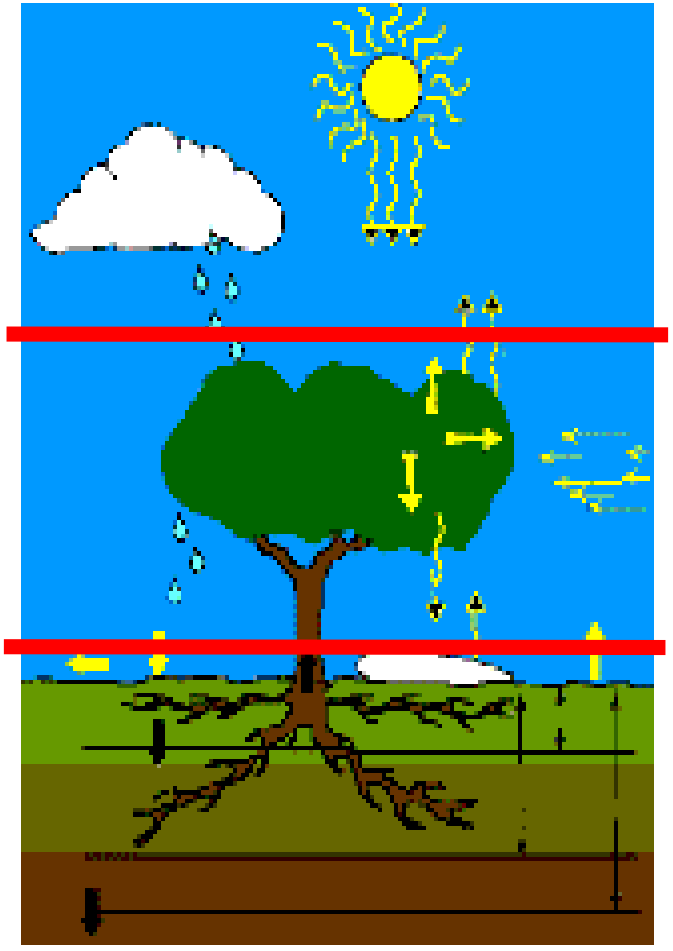
- TAC++ is designed to become FastOpt's AD tool for the C (and later C++) programming language
- TAC++ actually supports:
  - pointer types (with restrictions)
  - structured data types (with restrictions)
  - all intrinsic functions from ANSI C89
  - subroutine-like function calls
  - scalar and vector tangent linear mode
  - scalar reverse mode (also with pure option)
- Preprocess code before (single file only)
- extend TAC++ application by application (to customer demands)

Twostreams

Slide provided by Thomas Lavergne

Radiative transfer in a vegetation canopy

Observables



Observables are light fluxes.

R*, Reflectance (or Albedo) is the flux that returns to the atmosphere.

A*, Absorption is the flux that is captured inside the canopy and which never exits.

T*, Transmittance is the flux that reaches the ground below the vegetation.

For details see: *Pinty et al.* (JGR, 2005)

Figure taken from *Bonan, G.B.* (2002) Cambridge University Press

Twostreams and its inverse

- Twostreams: Simplified (one-dimensional) radiative transfer (RT) model for efficient retrieval of vegetation canopy properties from remote sensing data (Pinty et al., JGR, 2005)
- Twostream model is implemented in C and comprises about 330 lines of code
- Task: Implementation of an inverse model that uses observed radiances to estimate parameters *and* their uncertainties.
- Bayesian approach (see, e.g., *Tarantola,* 1987): Combining observations, d, and prior information on the parameters, m_{pr}, via the model yields posterior probability distribution for the unknowns.
- Assume Gaussian distribution on observations and priors, i.e. they are represented by their mean values and covariance matrices (C_d and C_{pr}, respectively)

Parameter Estimation

Optimal parameter m_{post} at minimum of misfit function J:

covariance of uncertainty in measurements + model

$$J(m) = \tfrac{1}{2} \left[(m-m_{pr})^T C_{pr}^{-1} (m-m_{pr}) + (M(m)-d)^T C_d^{-1} (M(m)-d) \right]$$

a priori covariance matrix of parameter uncertainty

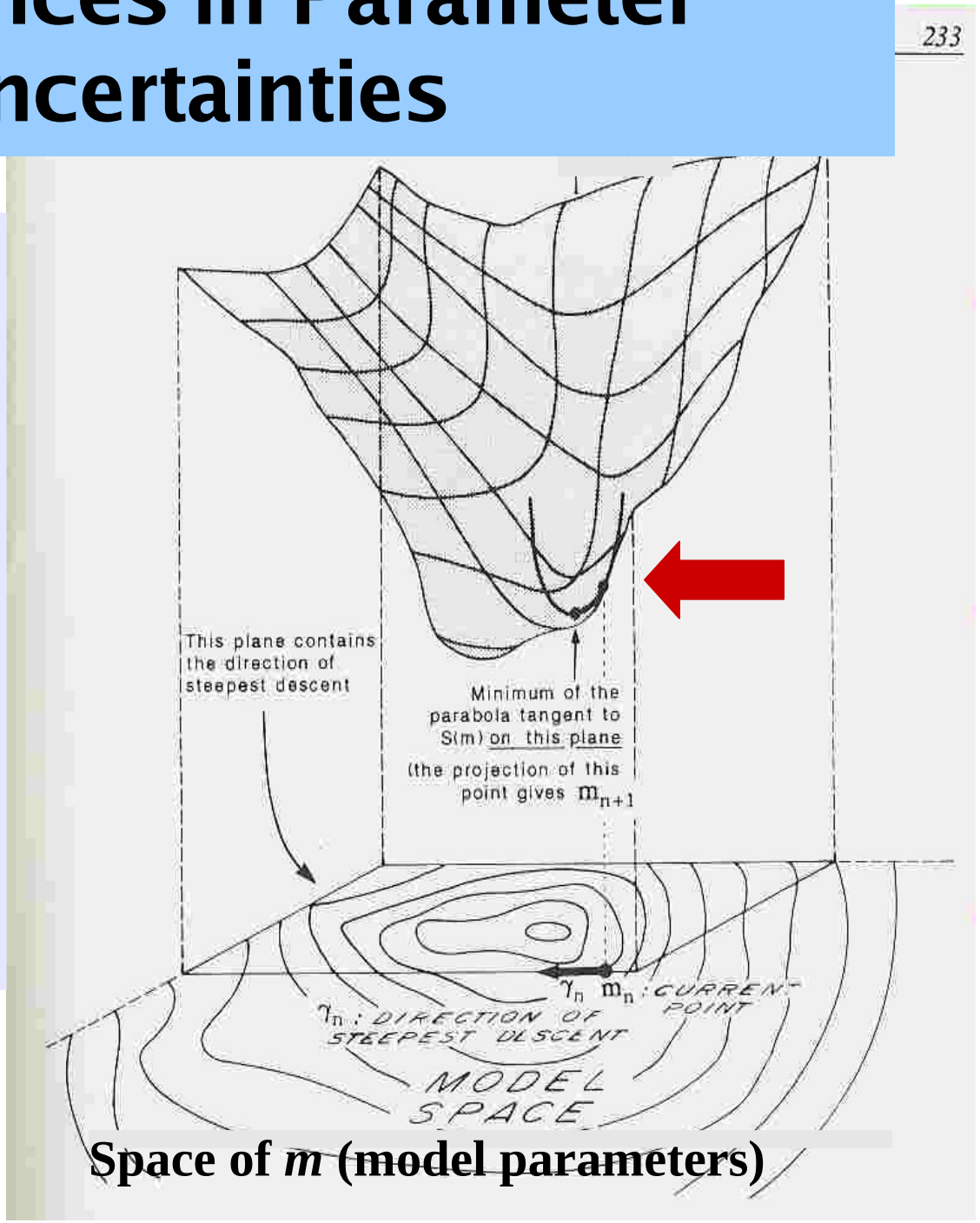
- Use gradient algorithm for minimisation, which iteratively evaluates both J and its gradient $\partial J(m)/\partial m$ (requires efficient gradient computation)
- Generate gradient by applying TAC++ to J in reverse mode (=> yields adjoint code of J)
- Build interface to minimisation algorithm from numerical recipes

Covariances in Parameter Uncertainties

Second Derivative (Hessian) of $J(m)$:

$$\partial^2 J(m)/\partial m^2$$

at m_{post} yields curvature of J, provides estimated uncertainty in m_{post}



Space of m (model parameters)

Figure taken from **Tarantola** '87

Parameter Uncertainties

- Uncertainty for m_{post} is quantified by its covariance matrix C_{post}
- Covariance matrix is approximated by the inverse of the Hessian of J evaluated at $m=m_{post}$:

$$C^{-1}_{post} \sim \partial^2 J(m) / \partial m^2 \Big|_{m=mpost}$$

- Compute Hessian in forward-over-reverse mode, i.e. apply TAC++ in forward mode to previously generated adjoint code
- Vector mode is favorable, because complete Hessian is needed
- TAC++ had to be extended to be able to operate in vector forward mode

Performance

- Test environment such that function code runs as fast as possible
  (here: icc with options `-O3 -ip -tpp7`)

Model	#lines of code	FUNC [s]	TLM / FUNC	ADM/ FUNC	HES/ FUNC
Roeflux (Cusdin, Mueller)	140	6.2E-7	3.3	3.9	-
2streams (Pinty et al.)	~330	3.9E-6	2.0	4.4	31/ 7 cols
TAU-ij (Gauger et al.)	~130	3.0E-3	--	2.6	-
LIBOR (Giles, Glasserman)	~210	2.0E-1	1.5	3.3	
GasNetOpt (Steinbach)	25	2.4E-7	2.4	2.7	-
Roeflux; F77, TAF	105	5.1E-7	--	2.9	

- Generated code is efficient
- Comparison to TAF (Roeflux) indicates some scope for improvement in terms of performance of the generated code

Recent AD of Fortan CFD

with Volkswagen:

- **CFD code based on Griebel et al. (1998)**
- **First 2D version then 3D**
- **Generated two adjoint code versions with TAF steady (TAF iteration directive) and standard version**
- **Performance ratio: (primal+adjoint)/primal = 1.7**

with DLR:

- **Industrial Fortran CFD solver FLOWer**
- **Adjoint running for a few time steps**
- **Performance ratio currently at (primal+adjoint)/primal = 7**

Thanks for your attention

Find more information:

- **FastOpt**
  http://www.fastopt.com
- **M. Voßbeck, R. Giering, and T. Kaminski. „Towards a tool for forward and reverse mode source to source transformation in C++“**
  http://www.fastopt.com/papers/vossbeckal04b.pdf
- **M. Voßbeck, R. Giering, and T. Kaminski. “Automatically generated tangent and adjoint C codes”**
  http://www.fastopt.com/papers/vossbeckal05-nice.pdf