

# Automatic Differentiation for a Structural Optimization Solver

Presented to *AD2CompEng Meeting, Shrivenham, 14 May 2004*

Shaun Forth & Mohamed Tadjouddine

Andy Keane

Applied Maths. & OR Group

Computational Engineering Design Centre

Engineering Systems Department

School of Engineering Sciences

Cranfield University (Shrivenham Campus)

University of Southampton, Highfield

Swindon SN6 8LA, UK

Southampton SO17 1BJ, UK

Work funded by EPSRC under Grant GR/R85358/01 *AD2CompEng - Automatic Differentiation and Adjoints Applied to Computational Engineering*

# Introduction



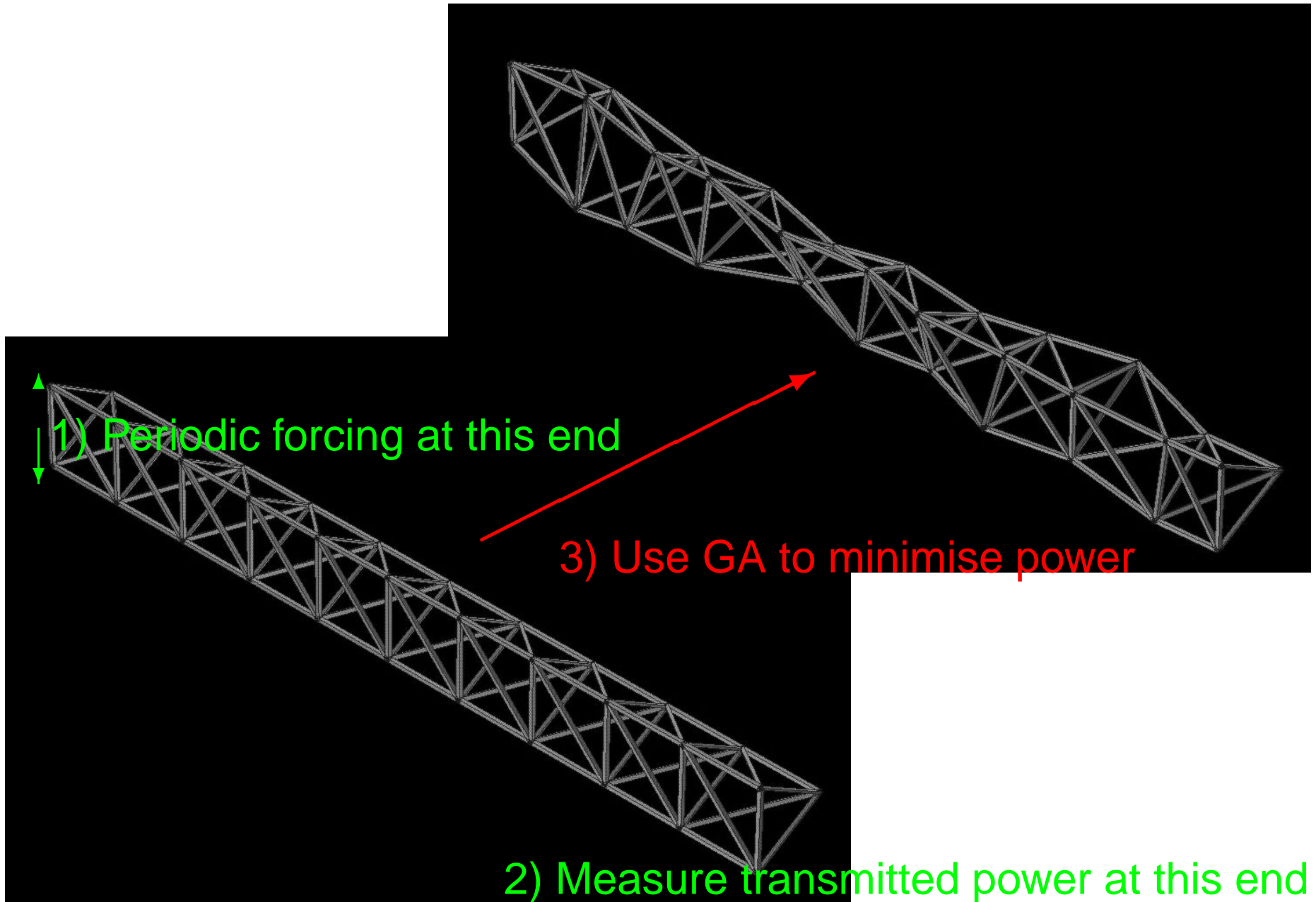
NASA Photo ID: STS61B-120-052

- Lightweight cantilever structure for suspending scientific instruments away from satellite.
- Wish to minimise transmission of vibration through structure from satellite to instrument

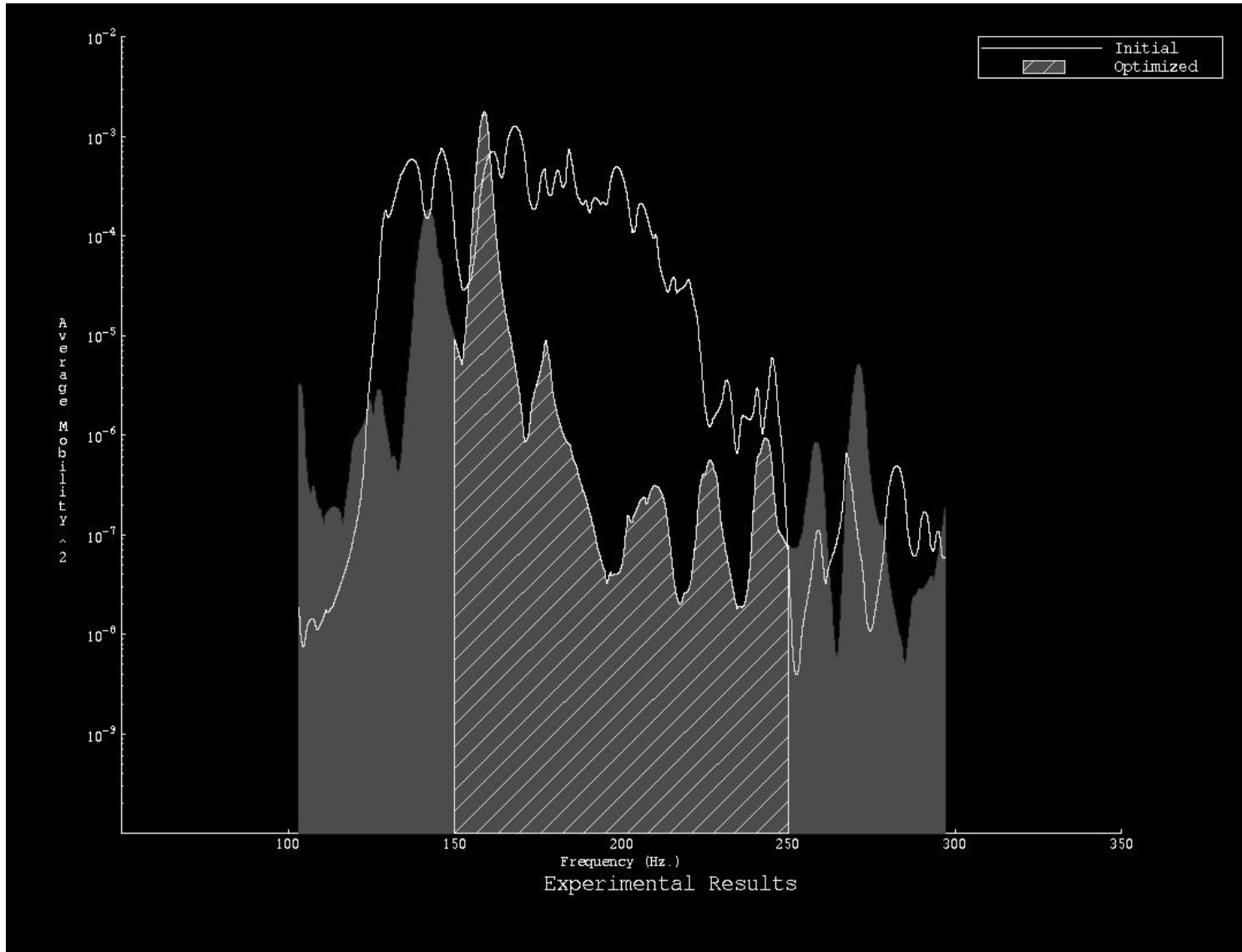
# Model Using Receptance Theory [SK95]

- Structure of long, thin beams executing transverse & purely longitudinal Euler-Bernoulli modes.
- $N = 90$  Beams coupled at their ends or ends are hinged/clamped.
- Formulate beam end displacements in terms of applied forces via Green functions.
- Leads to large linear system whose solution gives forces at end of each beam.
- Vibrations-
  - Consider forcing  $\mathbf{x}_{1,1} = \Delta \mathbf{x}_{1,1} e^{i\omega t}$  at first end of beam 1
  - Solve complex linear system for amplitude and phase of structure response  $\mathbf{x}_{i,j}$  at end  $j$  of beam  $i$ .
  - From force  $\mathbf{f}_{i,j}$  at end  $j$  of node  $i$  calculate power
$$P = \frac{1}{2} \text{Re}(\mathbf{f}_{i,j} \cdot \mathbf{x}_{i,j})$$

# Optimisation by Genetic Algorithms



# Reduction in Transmitted Power (2D)



# Optimisation by Genetic Algorithm

- Not a huge problem, the number of independents  
 $n = 3 \times 30 = 90$  (30 free joint locations)
- Use of GA for 10 generations with population size of 100 gives slow convergence
- Run times of **20 days** using parallel processing [KB96].
- Wish to speed convergence using **Meta-Lamarckian** approach [OK04]
  - Lamarckian evolution - *Inheritance of acquired characteristics*
  - Couple gradient descent with the GA
  - But gradient of transmitted power expensive to approximate with FD
- **Need an Adjoint Solver**

# The BEAM3D code

Read in:

coordinates of beam ends  $x_{i,j}$

frequency range  $[\omega_{min}, \omega_{max}]$

No. of frequencies  $N_\omega$

Initialise  $I = 0$  (integral of power)

$$\Delta\omega = (\omega_{max} - \omega_{min}) / (N_\omega - 1)$$

For  $k = 1, N_\omega$

$$\omega = \omega_{min} + (k - 1) * \Delta\omega$$

Assemble:

Green Function  $N \times N$  Matrix  $A(x, i\omega)$ , ( $N = 1044$ )

r.h.s. forcing  $b(x, i\omega)$

**Solve  $A(x, i\omega)f = b(x, i\omega)$  (LAPACK) - 50% of CPU time**

Obtain displacement  $x_{i,j}$  at ends of beam  $i$

Obtain power  $P = \frac{1}{2} Re(f_{i,j} \cdot x_{i,j})$

Update integral  $I = I + P^2 * \Delta\omega$

end

# Differentiation with ADIFOR 3.0

- We need  $\frac{\partial I}{\partial x_{i,j}}$
- Initial choice of ADIFOR 3.0 [CF00] (good complex arithmetic support, free license)
- N.B. 50% of CPU time spent in LAPACK linear solve `zgesv`
- Some code clean-up performed
  - Non-standard Fortran - `sed` scripts
  - Restructure so reading of `x` performed outside of differentiated subroutines
- 2 issues associated with non-differentiability.

# Differentiability (1) - Exceptions

- Initial inconsistency between AD and FD
- ADIFOR's exception handling for

$$\frac{d \cos^{-1}(x = 1)}{dx} = \infty$$

led us to following code:

```
L2=DATAN2(XDIFF,ZDIFF)
M2= DSQRT(XDIFF*XDIFF + ZDIFF*ZDIFF)/BEAM_LEN(I)
SGN1 = -1.0D0
SGN2 = -1.0D0
SGN3 = -1.0D0
IF(XDIFF.LT.0.0D0) SGN1 = -SGN1
IF(YDIFF.GT.0.0D0) SGN2 = -SGN2
IF(ZDIFF.LT.0.0D0) SGN3 = -SGN3
YOR(1,I) = SGN1*DABS(DSIN(DACOS(M2))*DSIN(L2))
YOR(2,I) = SGN2*DABS(M2)
YOR(3,I) = SGN3*DABS(DSIN(DACOS(M2))*DCOS(L2))
```

# Differentiability (1) ctd

- Use elementary trigonometry to replace with

```
BEAM_LEN(I) = DSQRT(XDIFF*XDIFF +  
& YDIFF*YDIFF ZDIFF*ZDIFF)  
MYT1 = YDIFF/BEAM_LEN(I)  
MYT2 = DSQRT(XDIFF*XDIFF + ZDIFF*ZDIFF)  
YOR(1,I) = MYT1*(-XDIFF/MYT2)  
YOR(2,I) = MYT2/BEAM_LEN(I)  
YOR(3,I) = MYT1*(-ZDIFF/MYT2)
```

which is almost equivalent !

- Differences in sign not significant
- Second axis of right-handed local coordinate system - just needs to be perpendicular to beam orientation.

# Differentiability (2) - Branching

```
IF (XDIFF.EQ.0.0 .AND. YDIFF.EQ.0.0) THEN
  YOR(1,I) = ZDIFF/BEAM_LEN(I)
  YOR(2,I) = 0.0
  YOR(3,I) = 0.0
ENDIF
```

- Above branch hit for initial configuration
  - Gives **point-valued** derivatives different to limit of Differentiability (1).
  - Discontinuous derivatives - prevents AD/FD validation
  - Similar to  $y = x$  example coded as

```
if (x.eq.0.0) then
  y=0.0
else
  y=x
endif
```

for which AD yields  $dy/dx = 0$  at  $x = 0.0$ .

- Removed such branches (constraints on geometry).

# Forward Diffn. of LAPACK zgesv

- May be the black-box differentiation of LAPACK routine gives us non-differentiable code?
- Forward mode differentiation of linear solve for forces  $\mathbf{A}\mathbf{f} = \mathbf{b}$

$$\begin{aligned}(\mathbf{A} + \dot{\mathbf{A}})(\mathbf{f} + \dot{\mathbf{f}}) &= \mathbf{b} + \dot{\mathbf{b}} \\ \mathbf{A}\mathbf{f} + \mathbf{A}\dot{\mathbf{f}} + \dot{\mathbf{A}}\mathbf{f} + \dot{\mathbf{A}}\dot{\mathbf{f}} &= \mathbf{b} + \dot{\mathbf{b}} \\ \mathbf{A}\dot{\mathbf{f}} &= \dot{\mathbf{b}} - \dot{\mathbf{A}}\mathbf{f}.\end{aligned}$$

Well defined derivative  $\dot{\mathbf{f}}$  for  $\mathbf{A}$  non-singular.

- Write differentiated LAPACK routine
  1. Perform an LU decomposition of the matrix  $\mathbf{A}$
  2. Solve  $\mathbf{A}\mathbf{f} = \mathbf{b}$
  3. Form  $\mathbf{b}_{new} = \dot{\mathbf{b}} - \dot{\mathbf{A}}\mathbf{f}$
  4. Re-use LU-decomposition to solve  $\mathbf{A}\dot{\mathbf{f}} = \mathbf{b}_{new}$

# Adjoint Diffn. of LAPACK zgesv

In the reverse mode, the following procedure is used [Ver98]:

- In the forward sweep
  1. Perform an LU decomposition of the matrix  $A$
  2. Store  $L$  and  $U$  and the pivot sequence  $IPIV$
  3. Solve  $Af = b$
- In the reverse sweep,
  1. Load  $L$  and  $U$  and the pivot sequence  $IPIV$
  2. Solve  $A^T \bar{b} = \bar{f}$
  3. Update  $\bar{A} = \bar{A} - \bar{b}\bar{b}^T$
- Only stores  $N^2$  complex coefficients + pivot sequence
- Black box adjoint tapes  $O(N^3)$  intermediates

# First Results

- Validate - check single directional derivative  $\nabla I(\mathbf{x}) \cdot \dot{\mathbf{x}}$ .

Method	$\nabla I(\mathbf{x}) \cdot \dot{\mathbf{x}}$	CPU( $\nabla I(\mathbf{x}) \dot{\mathbf{x}}$ )(s)
FD (1-sided)	0.124578003587	48.7
ADIFOR(fwd)	0.124571139127	54.0
ADIFOR(rev)	0.124571139130	311.5

- SUN Blade 1000, 1.2 GHz UltraSparcIII Processor
- FD for  $\nabla I(\mathbf{x})$ : 100 function evaluations - **37 MINS** CPU.
- Adjoint mode:  $\nabla I(\mathbf{x})$  in one adjoint - **5 MINS** CPU.
- Taping requires 6 GB of file-space or RAM.
- NB: Timings are obtained by running the code on local disk.

# Parallel Loop

- Main Loop is parallelizable

**For**  $k = 1, N_\omega$

$$\omega = \omega_{min} + (k - 1) * \Delta\omega$$

⋮

**Obtain power**  $P = \frac{1}{2} Re(\mathbf{f}_{i,j} \cdot \mathbf{x}_{i,j})$

**Update integral**  $I = I + P^2 * \Delta\omega$

**end**

- By default AD tools generate adjoint code with adjoint loop in reverse order.
- But since loop parallelizable (independent of loop order) so is adjoint [HFH01]
- Hand-code adjoint of loop so for each  $k$  we execute the loop body code with taping, then immediately adjoin the loop body.
- Only need tape variables for one pass of the loop

# Improved Performance

Method	CPU( $\nabla f$ )(s)	CPU( $\nabla f$ )/CPU(f)
ADIFOR(rev.)	311.5	13.3
ADIFOR(rev.,par.)	192.0	8.2
FD (1-sided)	2215.9	93.1

- Gradient obtained now for cost equivalent to 8.2 function evaluations
- 11 times faster than 1-sided FD (without truncation error)
- Memory requirement of just 0.3 GB

# Differentiation with TAF

- Start with the same input code as for ADIFOR 3.0
- But, the code needed be rewritten for TAF
- *Irreducible* controlflow graph due to multiple exit points from loop (use of GOTOs)

```
DO 701,ILOOP=1,MAXLOOP
  IF (JOINT_ELEM(1,I).EQ.-1) GOTO 702
  IF(J.EQ.JOINT_ELEM(2,I)) THEN
    CIN=6*I-5
    RETURN
  ENDIF
  I=I+1
701 CONTINUE
702 CONTINUE
```

for the Adjoint Calculation

# Differentiation with TAF (continued)

- Previous Loop rewritten as

```
DO WHILE (JOINT_ELEM(1,I).NE.-1
        .AND. J.NE.JOINT_ELEM(2,I))
    I=I+1
ENDDO
IF(J.EQ.JOINT_ELEM(2,I)) THEN
    CIN=6*I-5
ELSE
    . . .
```

- Controlflow *normalisation* (Use of IF-constructs)

- Assumed size arrays

```
DOUBLE PRECISION A(*)
```

need be declared with explicit dimensions

- TAF Adjoint code is at the debugging stage.

# Improvements in AD Tools

- Language coverage?
- Exception handling, should

```
if (x.eq.0.0) then
  y=0.0
else
  y=x
endif
```

raise an exception?

- Support for LAPACK libraries?
- Automated and improved treatment of parallel loops  
n.b. TAF `parallel` directive would recalculate LU decomposition.

# Conclusions

- Applied the AD Tool ADIFOR 3.0 to a structural dynamics problem.
- Expert AD intervention - objective function gradient obtained for
  - 8.2 times the CPU time of the objective
  - 0.3 GB tape storage.
- Differentiation with TAF [TAF01] in progress.
- Tuning the TAF adjoint code (taping/recalculation, directives).
- Suggesting improvements to AD Tools - liaise with developers.

# References

- [CF00] Alan Carle and Mike Fagan. ADIFOR 3.0 overview. Technical Report CAAM-TR-00-02, Rice University, 2000.
- [CFG<sup>+</sup>01] George Corliss, Christèle Faure, Andreas Griewank, Laurent Hascoët, and Uwe Naumann, editors. *Automatic Differentiation: From Simulation to Optimization*. Computer and Information Science. Springer, New York, 2001.
- [HFH01] Laurent Hascoët, Stefka Fidanova, and Christophe Held. Adjoining independent computations. In Corliss et al. [CFG<sup>+</sup>01], chapter 35, pages 285–290.
- [KB96] A.J. Keane and S.M. Brown. The design of a satellite boom with enhanced vibration performance using genetic algorithm techniques. In I. C. Parmee, editor, *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control 96*, pages 107–113, Plymouth, 1996.
- [OK04] Y. Ong and A.J. Keane. Meta-lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computing*, 8(2), 2004.
- [SK95] K. Shankar and A.J. Keane. Energy flow predictions in a structure of rigidly joined beams using receptance theory. *Journal of Sound and Vibration*, 185(5):867–890, 1995.
- [TAF01] Transformation of Algorithms in Fortran, manual, draft version, TAF version 1.3. Manual, FastOpt, Dec. 2001.
- [Ver98] Arun Verma. *Structured Automatic Differentiation*. PhD thesis, Cornell University Department of Computer Science, Ithaca, NY, 1998.