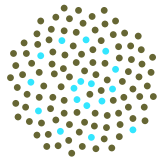


Tangent Linear Models by Operator Overloading No Pain: Some Gain

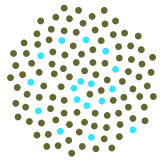
Formerly: Algorithmic Differentiation for Analysis of Global Change and the Carbon Cycle

Ian G. Enting

MASCOS

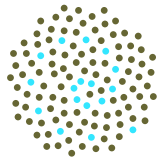
The University of Melbourne

Acknowledgments



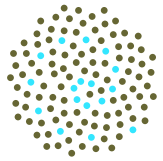
- The Center of Excellence for Mathematics and Statistics of Complex Systems (MASCOS) is funded by the Australian Research Council (ARC).
- My fellowship at MASCOS is supported by CSIRO through a sponsorship agreement.
- The Fortran-90 development is supported by the ARC Earth System Science Network (ARCNESS).
- Collaborators: Cathy Trudinger and YingPing Wang of CSIRO Marine and Atmospheric Research and members of the MATCH working group on the Brazilian Proposal.

Summary



- Algorithmic differentiation (AD) in analysing models
- Implementing tangent linear models using operator overloading in C++ and Fortran-90/95
- Applications of AD in:
  - Analysing the Brazilian proposal
  - Calibrating CASACNP terrestrial carbon model (with N and P)
    Progress report

Model Analysis



Most common model calculation is forward projection by (numerical) integration of DEs.

Much model analysis involves differentiation:

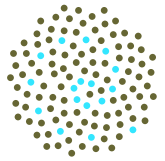
Initialisation Solving for zero rate of change

Sensitivity analysis — derivatives with respect to parameters

Calibration — techniques such as Maximum Likelihood imply optimisations, facilitated by use of derivatives

Data assimilation — real-time model adjustment — dynamic calibration

Tangent Linear Model (TLM)



For N DEs: $\frac{d}{dt}x_j = g_j(\{x_k\}, \alpha, t)$ for $j = 1, N$

we can define sensitivities as

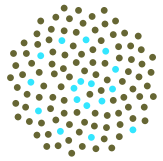
$y_j = \frac{\partial}{\partial \alpha} x_j$ for $j = 1, N$ or $y_{j,p} = \frac{\partial}{\partial \alpha_p} x_j$

to give 'tangent linear model(s)':

$$\frac{d}{dt}y_m = \frac{\partial}{\partial \alpha} g_m(\{x_k\}, \alpha, t) + \sum_n \frac{\partial}{\partial x_n} g_m(\{x_k\}, \alpha, t)\, y_n$$

$$\frac{d}{dt}y_{m,p} = \frac{\partial}{\partial \alpha_p} g_m(\{x_k\}, \alpha, t) + \sum_n \frac{\partial}{\partial x_n} g_m(\{x_k\}, \alpha, t)\, y_{n,p}$$

Using the TLM



The Tangent Linear Model (TLM) gives sensitivities with respect to parameters — useful for analysis of uncertainty.
Also provides a linearisation of a model relation defined in terms of DEs.
Useful for variational calibration

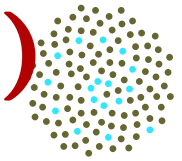
$$\Theta = [\underline{z}_j - \mathcal{H}(\underline{\alpha})]^{\mathsf{T}} \underline{R}^{-1} [\underline{z}_j - \mathcal{H}(\underline{\alpha})] - \ln[\mathrm{Pr}_{\mathrm{prior}}(\underline{\alpha})]$$

'model relation' $\mathcal{H}(\underline{\alpha})$ from solution of DEs.

$$\mathrm{d}\Theta = 2[\underline{z}_j - \mathcal{H}(\underline{\alpha})]^{\mathsf{T}} \underline{R}^{-1} \underline{H} \mathrm{d}\underline{\alpha} - \ldots$$

The TLM gives the linearisation $\underline{H}$

Algorithmic Differentiation (AD)



Differentiation by successive use of chain rule.
For binary operation $c = f(a, b)$,

$$\frac{\partial c}{\partial \alpha} = \frac{\partial f}{\partial a} * \frac{\partial a}{\partial \alpha} + \frac{\partial f}{\partial b} * \frac{\partial b}{\partial \alpha}$$

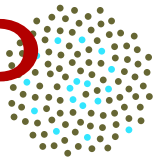
e.g.

$$c = a + b \quad \rightarrow \frac{\partial c}{\partial \alpha} = \frac{\partial a}{\partial \alpha} + \frac{\partial b}{\partial \alpha}$$

$$c = a * b \quad \rightarrow \frac{\partial c}{\partial \alpha} = b * \frac{\partial a}{\partial \alpha} + a * \frac{\partial b}{\partial \alpha}$$

Convert program to code for derivatives, one operation at a time.

Computational Complexity of AD



For $u_k \longrightarrow x_j(1) \longrightarrow \ldots x_j(m) \cdots \longrightarrow x_j(M) \longrightarrow v_j$,

m is operation count, Jacobian is: $J_{jk} = \dfrac{\partial v_j}{\partial u_k}$

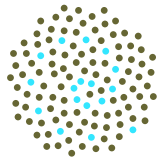
$$J_{jk} = \sum \frac{\partial v_j}{\partial x_n(M)} \cdots \frac{\partial x_{n'}(m)}{\partial x_{n''}(m-1)} \cdots \frac{\partial x_{p'}(1)}{\partial x_p(0)} \frac{\partial x_p(0)}{\partial u_k}$$

Tangent: $$\frac{\partial v_j}{\partial \alpha} = \sum_k J_{jk} \frac{\partial u_k}{\partial \alpha}$$ vector $\times$ sparse matrix product

Gradient: $$\frac{\partial \phi}{\partial u_k} = \sum_j J_{jk} \frac{\partial \phi}{\partial y_j}$$

Adjoint model achieves efficiency of vector $\times$ sparse matrix by using chain rule backwards in time.

Adjoints (of TLM)



$$\frac{d}{dt}y_{m,p} = \frac{\partial}{\partial \alpha_p} g_m(\{x_k\}, \alpha, t) + \sum_n \frac{\partial}{\partial x_n} g_m(\{x_k\}, \alpha, t)\, y_{n,p}$$

$$\frac{d}{dt}y_{m,p} = f_{m,p}(t) + \sum_n g_{m,n}(t)\, y_{n,p}$$

TLM is **linear** and therefore has a (linear)

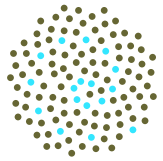
Green's function operator which has an adjoint.

Jacobian is projection of operator onto matrix.

$J_{k,1}$ gives $\frac{\partial v_k}{\partial u_1}$: carry forward single set of $\frac{\partial x_m}{\partial u_1}$

$J_{1,j}$ gives $\frac{\partial v_1}{\partial u_j}$: carry backward single set of $\frac{\partial v_1}{\partial x_m}$

Sparse Matrix Factorisation



Convert matrix $\times$ vector to successive multiplications of vector by sparse matrices.

e.g. FFT: $\tilde{f}(k) = \sum_n \exp(2\pi i n k/N) f(n)$ for $N = 2^M$

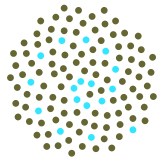
$$\tilde{\mathbf{f}} = \mathbf{M}_N \mathbf{f} = \prod_{j=1}^{M} \mathbf{L}_j \mathbf{f}$$

Jacobian for AD is product of sparse matrices:
$\mathbf{J}_m$ for $c = a$ *.op.* b is identity, except for column c, rows a,b,c.

Applications in lattice statistical mechanics:

.. conventional algorithm would .. run for time comparable to lifetime of the universe on .. fastest supercomputer to achieve .. same results .. Guttmann and Enting,1988, J Phys A, **21** L165

Fast Fourier Transform



FFT expresses transform of size $N = 2^M$ as sparse matrix product

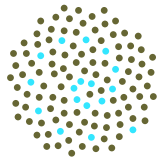
$$\tilde{\mathbf{f}} = \mathbf{M}_N \mathbf{f} = \prod_{j=1}^{M} \mathbf{L}_j \mathbf{f}$$

with $\mathbf{f}$ in 'bit-reversed' order,

$$\mathbf{M}_N = \begin{bmatrix} \mathbf{I} & \mathbf{W}_1 \\ \mathbf{W}_2 & \mathbf{I} \end{bmatrix} \times \begin{bmatrix} \mathbf{M}_{N/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{N/2} \end{bmatrix}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are diagonal, $(\exp(2\pi ik/N))$
Thus $N^2 = 2^{2M}$ multiplications become $2MN = 2^{M+1}M$

Approaches to AD



Hand-code program to calculate derivatives — laborious, error-prone and must be repeated each time the model changes.

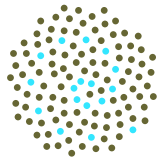
Symbolic algebra (e.g. Mathematica) — problematic for adjoints.

Tangent/adjoint compilers — transform source into code for tangent or adjoint models.

Operator overloading to produce a 'script' that is analysed to give code for the derivatives.

Use operator overloading capabilities directly — straightforward for tangent-linear-model, but restricted applicability to adjoint models.

Operator Overloading (C++)



Replace real variable x, (type `double`), with composite variable $\tilde{x}$ (type `Xvar`), representing both value x and its derivatives with respect to K model quantities, α_k as:

$$\tilde{x}_0 = x \quad \text{and} \quad \tilde{x}_k = \frac{\partial}{\partial \alpha_k} x \quad \text{for } k = 1, K$$

Operator overloading implements $\tilde{c} = \tilde{a} * \tilde{b}$, representing:

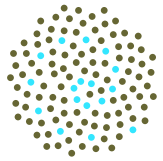
$$\tilde{c}_0 = \tilde{a}_0 * \tilde{b}_0 \quad \text{and} \quad \tilde{c}_k = \tilde{a}_0 * \tilde{b}_k + \tilde{a}_k * \tilde{b}_0$$

Overloaded functions, $\tilde{c} = f(\tilde{a})$, represent:

$$\tilde{c}_0 = f(\tilde{a}_0) \quad \text{and} \quad \tilde{c}_k = f'(\tilde{a}_0) * \tilde{a}_k$$

where $f'(.)$ denotes the derivative of $f(.)$

Class Definitions

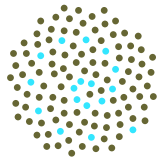


Fragment of C++ class definition to implement operator overloading:

```
class Xvar{
public :
static const int ns = _NUMDERIVS+1;
double xs[_NUMDERIVS+1];
Xvar operator*(Xvar);
...
};

Xvar Xvar::operator*(Xvar b){ Xvar c;
for (int i=1; i < ns; i++)
    c.xs[i] = xs[i]*b.xs[0]+xs[0]*b.xs[i];
c.xs[0] = xs[0]*b.xs[0];
return c;} ;
...
```

Usage



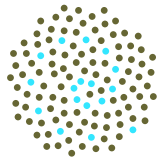
Original

```
double F_co2(double c){
double a;
a = log(c/280.0)*5.35;
return a;
};
...
double cc;
...
ff = F_CO2(cc)
```

Transformed

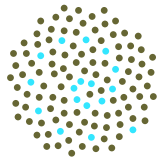
```
Xvar F_co2(Xvar c){
Xvar a;
a = log(c/280.0)*5.35;
return a;
};
...
Xvar cc;
// Derivatives wrt
// initial value of cc
cc.set(280,1);
...
ff = F_CO2(cc)
```

Definitions (Fortran-90/95)



```
type x_var
real, dimension(0:max_d) ::  v
end type
...
interface operator (*)
module procedure xv_mul, xv_mulr, xv_rmul, xv_muli, xv_imul
end interface
...
type (x_var) function xv_mul(xb,xc) !
type (x_var), intent (in) ::  xb,xc
do 1 j= 1,n_der
1 xv_mul%v(j) = xc%v(j)*xb%v(0) + xc%v(0)*xb%v(j)
xv_mul%v(0) = xc%v(0)*xb%v(0)
return
end function
```

Putting AD into models

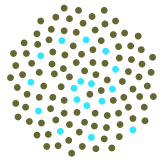


Need to modify model by changing:

- type declarations
- output
- initialisation (and input)
- other surprises ????

Use syntax checking of compiler to help ensure validity.
Real = *x_var* should be undefined, detection by compiler implies failure to declare all neccessary variables.

Basic Operator Requirements



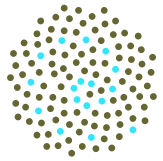
A basic set:

	Binary				
Op	X.op.X	R.op.X	X.op.R	X.op.I	I.op.X
$=$	Y	N/A	Y	Y	N/A
$+$	Y	Y	Y	Y	Y
$-$	Y	Y	Y	Y	Y
$*$	Y	Y	Y	Y	Y
$/$	Y	Y	Y	Y	Y
$**$	–	–	–	Y	–

Unary operations (and intrinsics)

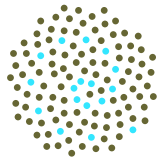
Op	–	sqrt	cos	sin	log	exp
	Y	Y	Y	Y	Y	Y

Other Requirements



- Output routines for variables with derivatives — (Fortran *write* accepts *type(x_var)* as Fortran array)
- Ability to identify variables for differentiation
- Deal with non-differentiable operations: *max*, *.GT.* etc.
- Miscellaneous: error trapping, version identifier, initialise number of derivatives, match variable names to derivatives.

Extensions of overloading



Differentiation —

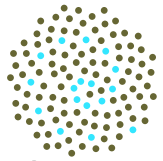
- Higher derivatives
- Taylor's series

Other —

- Multi-region (or multi-category in general)
- Isotopes?

C++ implementation of second derivatives used in analysis of Brazilian Proposal.

Brazilian Proposal



Tabled by Brazil during negotiations leading to Kyoto Protocol — Flicked-passed to Subsidiary Body for Scientific and Technical Advice (SBSTA).

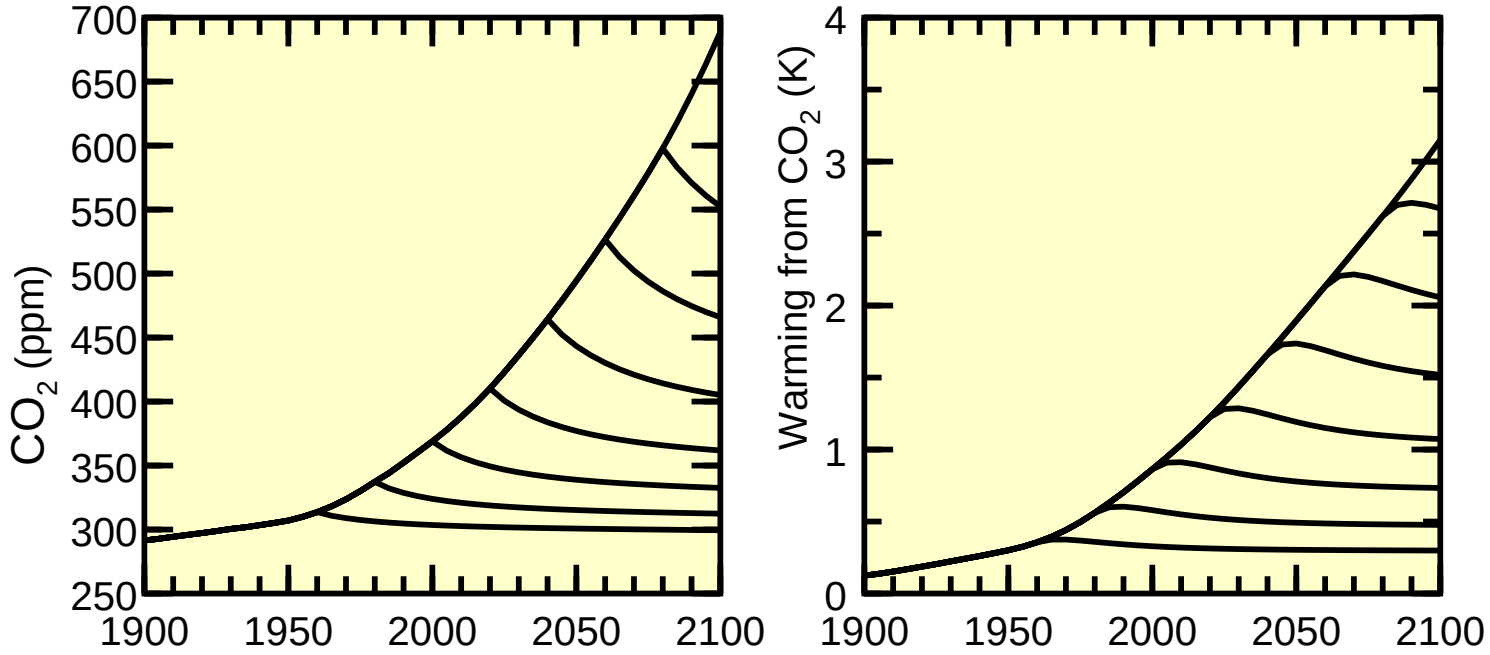
Proposes that emission reduction targets should be proportional to nation's relative responsibility for the greenhouse effect.

Issues:

- Indicator? What quantity is used as a measure of the greenhouse effect?
- For what period of emissions is responsibility attributed?
- How are non-linear responses attributed?

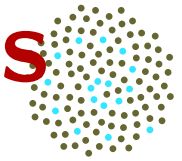
Timescales

CO_2 concentrations and consequent warming, partitioned according to time of emission.



Lowest bands are from pre-1960 emissions, next from 1960 to 1980 emissions, etc. Increase in contribution to warming after time of emissions from 'committed warming' effect.

Brazilian Proposal as Derivatives



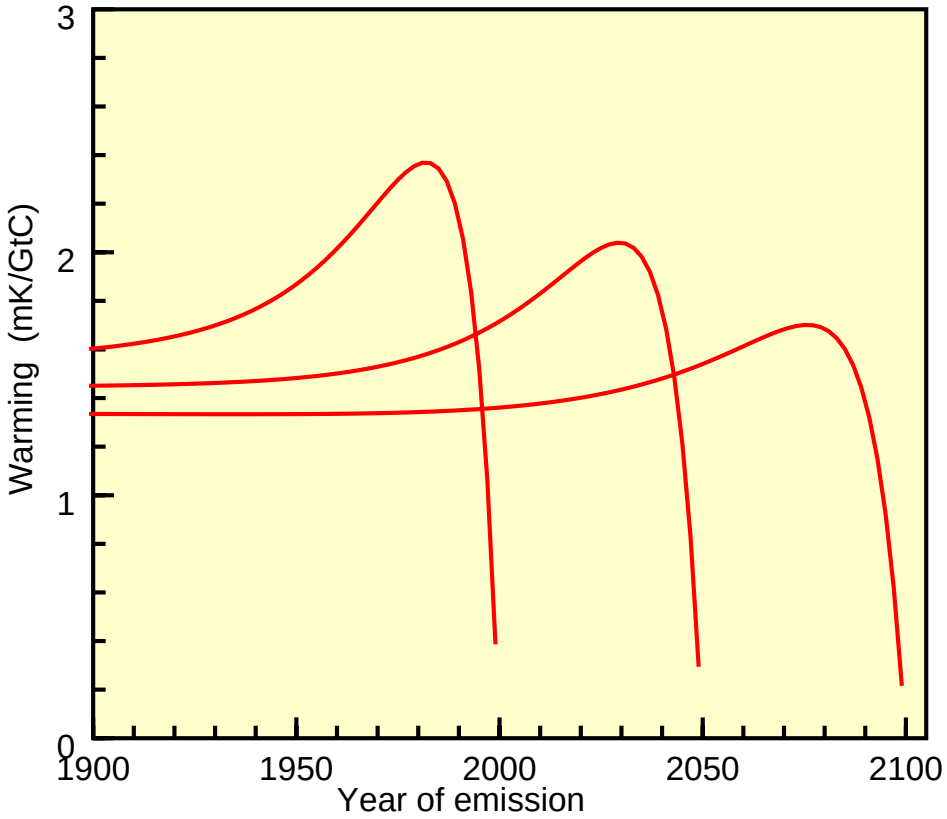
As example, use indicator $T^* = T_{CO2}(2100)$= warming in 2100 from CO_2 emissions.
T^* is to be attributed to emissions $E_j(t)$ from country j with $E(t) = \sum_j E_j(t)$.

Differential attribution to country j of emissions at time t is

$$\frac{\partial T^*}{\partial E_j(t)} E_j(t) = \frac{\partial T^*}{\partial E(t)} E_j(t) = S(t) E_j(t)$$

where $S(t)$ is a Frechet derivative.
Cumulated attribution: $T_j^* = \int S(t) E_j(t)\, dt$

Results: Frechet Derivatives



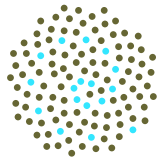
Assumes IS92a emissions.

Represents temperature by response function. Linear responses for ocean and biotic carbon, coupled non-linearly to atmospheric CO_2 (as in CSIRO study).

$\frac{\partial}{\partial E(t)} T(\tau)$ for $\tau = 2000, 2050, 2100$.

Decrease as $t \to \tau$ shows 'committed warming'.
At any time, warming from the most recent releases is yet to happen.

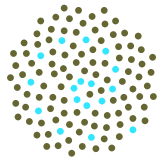
Implications



- For a given indicator, T^*, calculation of $S(t)$ allows attribution to any nation.
- $S(t)$ most efficiently calculated from adjoint model, but for multiple indicator times, tangent linear model not too inefficient.
- Sensitivity of T_j^* to model uncertainties can be obtained as second derivatives.
- Sensitivity of T_j^* to uncertainties in emissions can be obtained as

$$\mathrm{Var}[T_j^*] = \int\int S(t)\mathrm{Cov}[E_j(t), E_j(t')]S(t')\,dt'\,dt$$

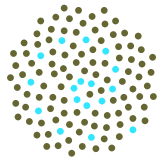
CASACNP AD Project



- Develop Fortran-90/95 implementation of AD by operator overloading;
- Use CASACNP as test case of AD in existing model;
- Explore use of CASACNP derivatives for calibration;
- Document procedure for ARC Network for Earth System Science.

Development funding obtained from ARCNESS for CASACNP and extension to CABLE.

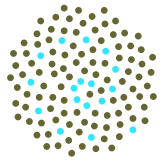
CASACNP



Extend CASA terrestrial carbon model to include nitrogen and phosphorus. (Wang, Field and Houlton).

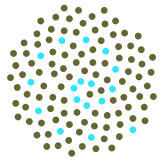
- Initially single region, 8 C pools, 9 N pools, 12 P pools.
- Models response to addition of P, N and competition between N-fixers and non-fixers.
- Validation transect (of soil age) in Hawaii.
- Initial conversion to use AD successful: May 2006 — development on-going.

Putting AD in CASACNP



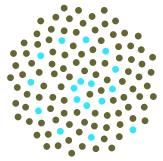
- Added *max* and *min* to initial AD definitions
- Redeclare types of variables
  — global edit — not efficient code
- Convert input to write to value – initialise derivatives to zero using compiler option
- Identify variables for differentiation
- Explicit re-writes of .GT. tests using values
- Explicit loops for *array* = *real* — explicit *1/delt*

Progress on CASACNP



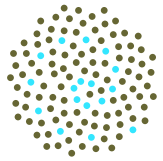
- C++ AD project on Brazilian Proposal: Presented at MODSIM 2005. Proof-of concept for AD by operator overloading.
- Successful proof-of concept of AD by operator overloading in Fortran-95.
- Successful runs using 'brute-force' conversion of CASACNP (May 2006)
- *Development on-going, supported by ARCNESS*

Future directions



- Incorporate AD into calibration tools.
- Second derivatives in Fortran.
- Explore overloading for Fortran vector operations.
- Develop calibration strategy for CASACNP and other carbon components of Australian Community Climate and Earth System Simulator (ACCESS).

Conclusions



Algorithmic differentiation —

Operator overloading is a straightforward way of developing tangent linear models (and obtaining higher derivatives if needed).

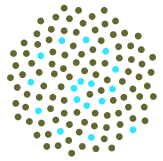
Brazilian Proposal —

Attribution in terms of derivatives is readily calculated using algorithmic differentiation. Higher derivatives give sensitivities.

Global change —

Potential should extend to other analyses of uncertainties in global change.

Further Information



Andreas Griewank, 2000, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, (SIAM, Philadelphia).

MATCH website (Brazilian Proposal):
http://www/match-info.net

I.G. Enting, 2005, *Automatic differentiation in the analysis of strategies for mitigation of global change*, International Congress on Modelling and Simulation, Melbourne, 2005. Ed. A. Zerger and R. M. Argent, 7pp
http://www.mssanz.org.au/modsim05/papers/enting.pdf