Automatic differentiation in practice: An application to the Estimation of SDGE models

Luca Guerrieri

Federal Reserve Board

Perspective

I am a user of AD tools, not a developer

This presentation is a "lab report" on the turn of events following a presentation at the Society for Computational Economics

I hope some of my mistakes make for useful lessons

Mistake Number 1

It’s "automatic" isn’t it?

Well, you get to do some amazing stuff with AD, but "automatic" might be a stretch.

It seems obvious, but are you sure that the code for your function is differentiable?

This might be trickier to ascertain than you might think at first and might not be picked up by your AD tool

Mistake Number 1 (continued)

A composition of functions might be differentiable even if the underlying functions are not

As a simple example think of adding two functions with perfectly symmetric kinks. Their sum is differentiable but the original functions are not

This kind of problem seems pervasive when relying on some common matrix decompositions (SVD, Eigenvalue/Eigenvector decomposition)

Mistake Number 2

The simplest code to write is the faster to differentiate

Not when it involves, for example, imaginary numbers

You might be better off rewriting your code than struggling to understand what is stumping your AD tool

Mistake Number 3

Dual language programming makes your life easy

Well, at times, but you might be much better off sticking to one language only, as most AD tools handle gracefully only one language

Grandiose plans of differentiating separately the different parts of the code with a “Black Box” approach can slow you down more than you think

Most tools don’t have extensive documentation and reverse-engineering the calling structure of intermediate functions can be quite painful

Mistake Number 4

So what if my function calls all sorts of LAPACK and BLAS routines?

Surely someone has else differentiated them.

Well, not yet.

Which tool to pick?

Fortunately for users, several options exist for various programming languages

When choosing between alternatives, take a good look at the documentation that is shared

Sooner or later, the tool you are using will have trouble digesting some of the code you intend to differentiate

Chances are you’ll be scouring the manuals then

If you are starting off, it might pay to pick a well documented tool, even if it is not the most efficient

What’s next?

An example of the use of Automatic Differentiation

SDGE Estimation Made Easier

(preliminary joint work with Gary Anderson and Houtan Bastani)

What do we do?

We build the score and information matrix for the likelihood function of a dynamic general equilibrium model using automatic differentiation techniques.

As a byproduct, we compute the first and second derivative of reduced-form parameters in the solution of a SDGE model with respect to the fundamental parameters.

Our toolbox is applicable to linear and linearized models.

Through Monte Carlo experiments, we show that finding the maximum of the likelihood of several (two, at the moment) SDGE models is greatly facilitated by more precise computation of the information matrix.

Higher precision in the information matrix also facilitates sizing asymptotic standard errors for maximum likelihood estimates.

Finite Difference Derivatives

Analytical derivatives are only available for special cases of the models we are interested in. Limitation is on the number of state variables.

The macroeconometrics literature has relied on finite difference methods to obtain the Jacobian. The Broyden updating algorithm is typically used for the Hessian.

The error bounds for FD derivatives depend on the size of higher derivatives.

In practice, choice of step size can dramatically affect the results. (Will provide examples.)

Key Elements of Solving a Model Under Rational Expectations

Upon linearization, any SDGE model can be written as:

$$H(\theta)\begin{pmatrix} E_t X_{t+1} \\ X_t \\ X_{t-1} \end{pmatrix} = 0.$$

The model's solution takes the form:

$$X_t = S(H(\theta))X_{t-1},$$

Partitioning X_t such that $X_t = \begin{pmatrix} x_t \\ \epsilon_t \end{pmatrix}$ yields

$$x_t = A(H(\theta))x_{t-1} + B(H(\theta))\epsilon_t.$$

The Likelihood Function

Given a subset of the entries in x_t as observable, call these entries y_t , the state-space representation of the system takes the form:

$$\begin{aligned} x_t &= A(H(\theta))x_{t-1} + B(H(\theta))\epsilon_t \\ y_t &= Cx_t \end{aligned}$$

Using the Kalman Filter, we can express the likelihood function for the model as: $L = L(A(\theta), B(\theta), C, y_{t-h}, ..., y_t)$ where y_{t-h} and y_t are respectively the first and last observation points available.

Our routines produce $\frac{\partial L}{\partial \theta}$ and $\frac{\partial^2 L}{\partial \theta^2}$.

As an intermediate product, our routines yield first and second derivatives of A and B wrt θ.

Computational choices

AIM algorithm, as detailed in Anderson (1987).

Anderson showed a roadmap to build the jacobian and hessian of the solution for a linear model.

Automatic differentiation facilitates the implementation of Gary's algorithm.

Kalman filter with training, as detailed in Hamilton, or Durbin and Koopman.

– Plan to add menu of choices for initialization

Implementation Details

We used Tapenade in vector mode.

Tapenade required limited manual intervention on our part. Remarkable, given code consisted of 80 subroutines for a total of over 17,000 lines (56 kb)

The derivative-augmented code produced by Tapenade covers approximately 25,000 lines (78 kb).

The original code was written in a mixture of C and Fortran 77 (Lapack and Blas routines).

Blas Functions					
daxpy.f	dcopy.f	ddot.f	dgemm.f	dgemv.f	dger.f
dnrm2.f	drot.f	dscal.f	dswap.f	dtrmm.f	dtrmv.f
dtrsm.f					

Lapack Functions					
dgebak.f	dgebal.f	dgeesx.f	dgehd2.f	dgehrd.f	dgeqp3.f
dgeqr2.f	dgeqrf.f	dgesv.f	dgetf2.f	dgetrf.f	dgetrs.f
dhseqr.f	dlacn2.f	dlacpy.f	dladiv.f	dlaexc.f	dlahqr.f
dlahr2.f	dlaln2.f	dlange.f	dlanv2.f	dlapy2.f	dlaqp2.f
dlaqps.f	dlaqr0.f	dlaqr1.f	dlaqr2.f	dlaqr3.f	dlaqr4.f
dlaqr5.f	dlarfb.f	dlarf.f	dlarfg.f	dlarft.f	dlarfx.f
dlartg.f	dlascl.f	dlaset.f	dlassq.f	dlaswp.f	dlasy2.f
dorg2r.f	dorghr.f	dorgqr.f	dorm2r.f	dormqr.f	dtrexc.f
dtrsen.f	dtrsyl.f	dtrtrs.f			

Testing Tapenade’s output

Two decompositions in the model solution, the real Schur decomposition and the singular-value decomposition, are not always unique.

Restrictions on model features could ensure uniqueness Schur and SVD decompositions. We verified that Tapenade derivatives satisfied some basic analytical properties but our test failed for models implying non-uniqueness of the Schur and SVD decompositions.

The Real Schur Decomposition

We relied on the Lapack routine DGEESX to implement the real Schur decomposition.

For a given real matrix E , this decomposition produces a unitary matrix X , such that $T = X^H E X$ is block triangular.

Since Xunitary, $X^H X = X X^H = 0$. Then $\frac{\partial X^H}{\partial \theta} X + X \frac{\partial X^H}{\partial \theta} = 0$.

This property failed to be met by our AD derivatives when our choice of E implied a non-unique Schur decomposition.

We substituted the AD derivative for the DGEESX routine with the analytical derivative of the Schur decomposition as outlined in Anderson 1987.

The Singular Value Decomposition

We relied on the DGESVD routine in the Lapack library to implement the singular value decomposition

Given a real matrix E , produces unitary matrices U and V and a diagonal matrix D , such that $E = UDV^T$.

Given $\frac{\partial E}{\partial \theta}$, then $U^T \frac{\partial E}{\partial \theta} V = U^T \frac{\partial U}{\partial \theta} D + \frac{\partial D}{\partial \theta} + D \frac{\partial V}{\partial \theta} V$, where $\frac{\partial D}{\theta}$ is diagonal and $U^T \frac{\partial U}{\partial \theta}$ and $\frac{\partial V}{\partial \theta} V$ are both antisymmetric.

Our AD derivative of the routine DGESVD failed to satisfy this property when the matrix E had repeated singular values (making the decomposition non-unique).

We substituted our AD derivative with the analytical derivative derived by Anderson 1987.

Testing the Tapenade Output

For special cases of our model that could be simplified we computed analytical derivatives and found them in agreement with our AD derivatives.

To test the derivatives for more complex models that we could not solve analytically, we relied on comparisons with centered FD derivatives.

Generally with a step size of 10^{-8} we found broad agreement between our AD derivatives and FD first derivatives. For second derivatives, matching is conditional on ad hoc choice of step size.

Do AD derivatives improve the quality of ML estimates?

Use two SDGE models to investigate this question.

1) simple RBC model with log utility and fixed labor

2) variant of Smets-Wouters model

RBC model

$$\begin{aligned}\max_{c_t,k_{t+1},i_t} \quad & \sum_{t=0}^{\infty} \beta^t \log(c_t) \\ & +\beta^t \lambda_{ct} \left[e^{z_t} k_t^{\alpha} - c_t - i_t\right] \\ & +\beta^t \gamma_t \left[(1-\delta)k_t + i_t - k_{t+1}\right]\end{aligned}$$

where $\log(z_t) = \rho_z \log(z_{t-1}) + \sigma_z \epsilon_t$ and $\epsilon_t \sim NID$

This is incredibly simple, but deceptively tough to estimate with limited data.

Following Uhlig's toolbox paper, we can use simple symbolic differentiation to check the AD derivatives.

Monte Carlo Experiment

Calibrate model in a standard way.

Generate 100 repetitions of 200 observations for quarterly data for one observed series: c_t.

For each repetition of the data, estimate 4 parameters of the model: α, δ, ρ_z, ϵ_z.

Optimization Algorithm

Our maximum likelihood estimates were constructed using the MATLAB optimization routine FMINUNC. Boundaries are imposed with a simple penalty function.

Very similar results obtained using unconstrained optimization algorithm in NAG library

When the optional argument “LargeScale” is set to “ON” a variant of a Newton-Raphson algorithm that takes as inputs the jacobian and Hessian

When “LargeScale” is set to “OFF”, FMINUNC imputes the Jacobian with FD methods and the Hessian with Broyden’s algorithm.

Algorithm requires initial point to conduct search.

Initialization Points

Parameter values used in the data-generating process:

α	δ	ρ_z	σ_z
0.3	0.025	0.95	0.4

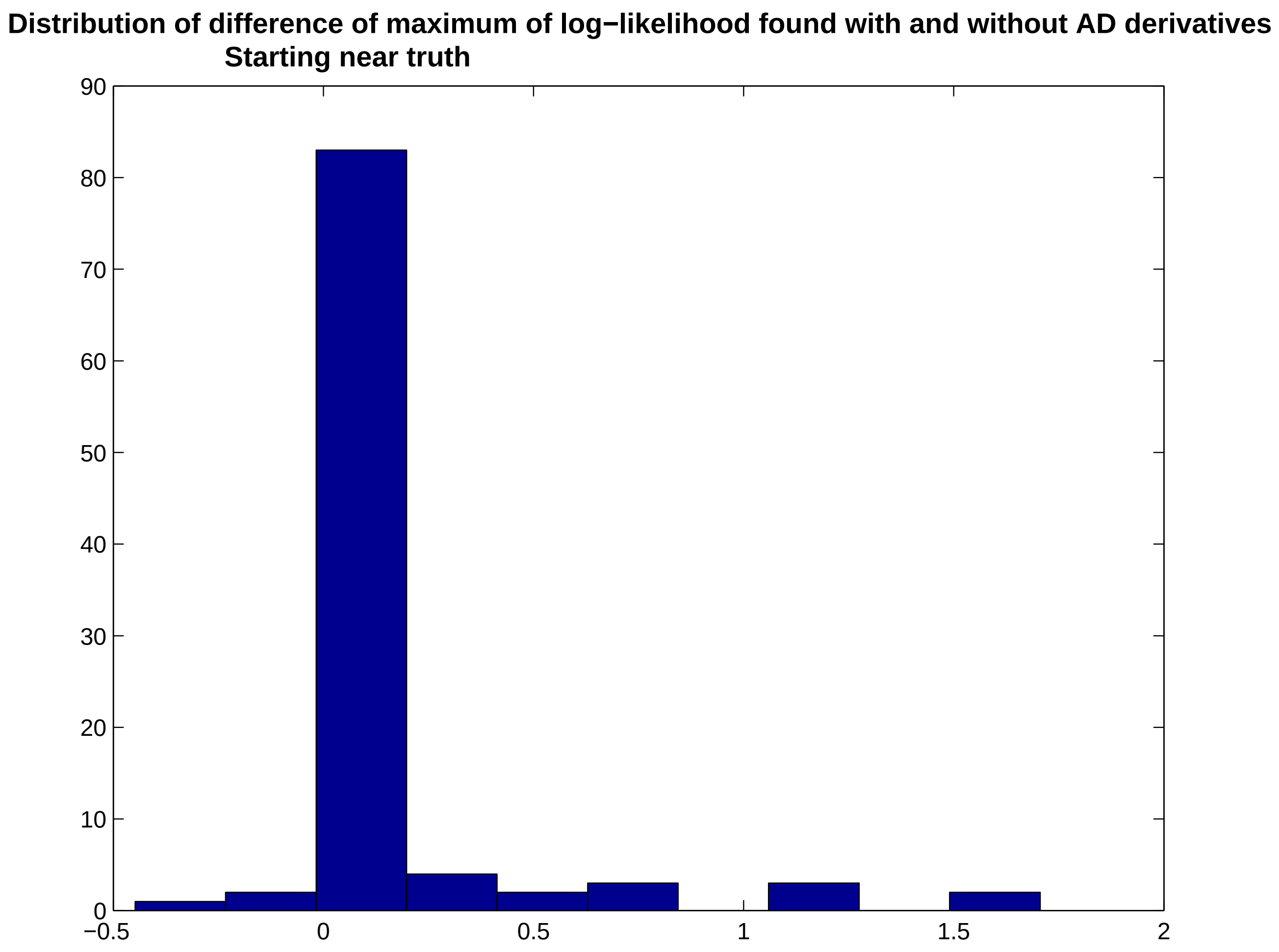
Consider 2 alternative choices for initial point for search:

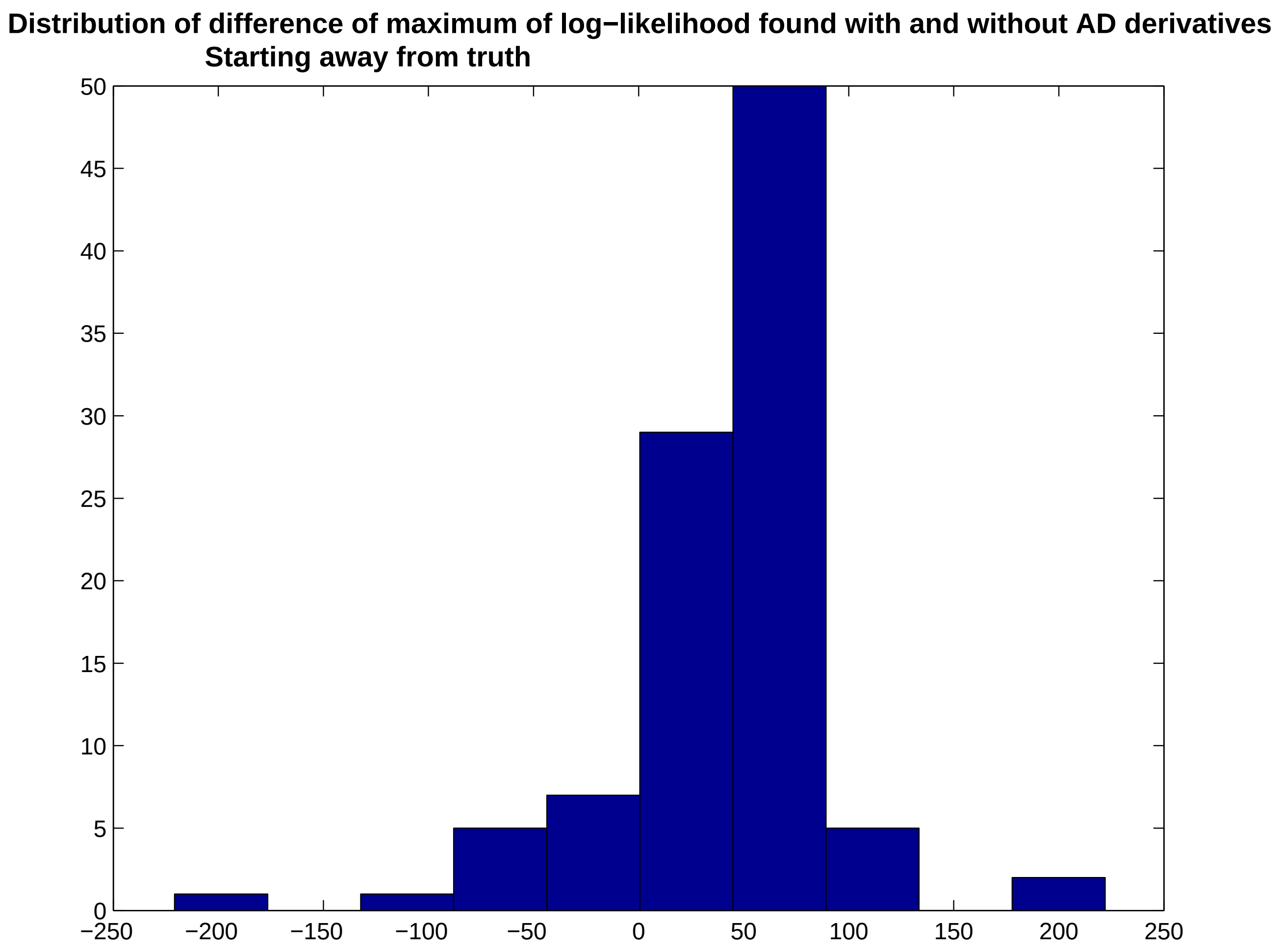
1) start close to true value for the parameters to be estimated

α	δ	ρ_z	σ_z
0.3	0.05	0.8	0.3

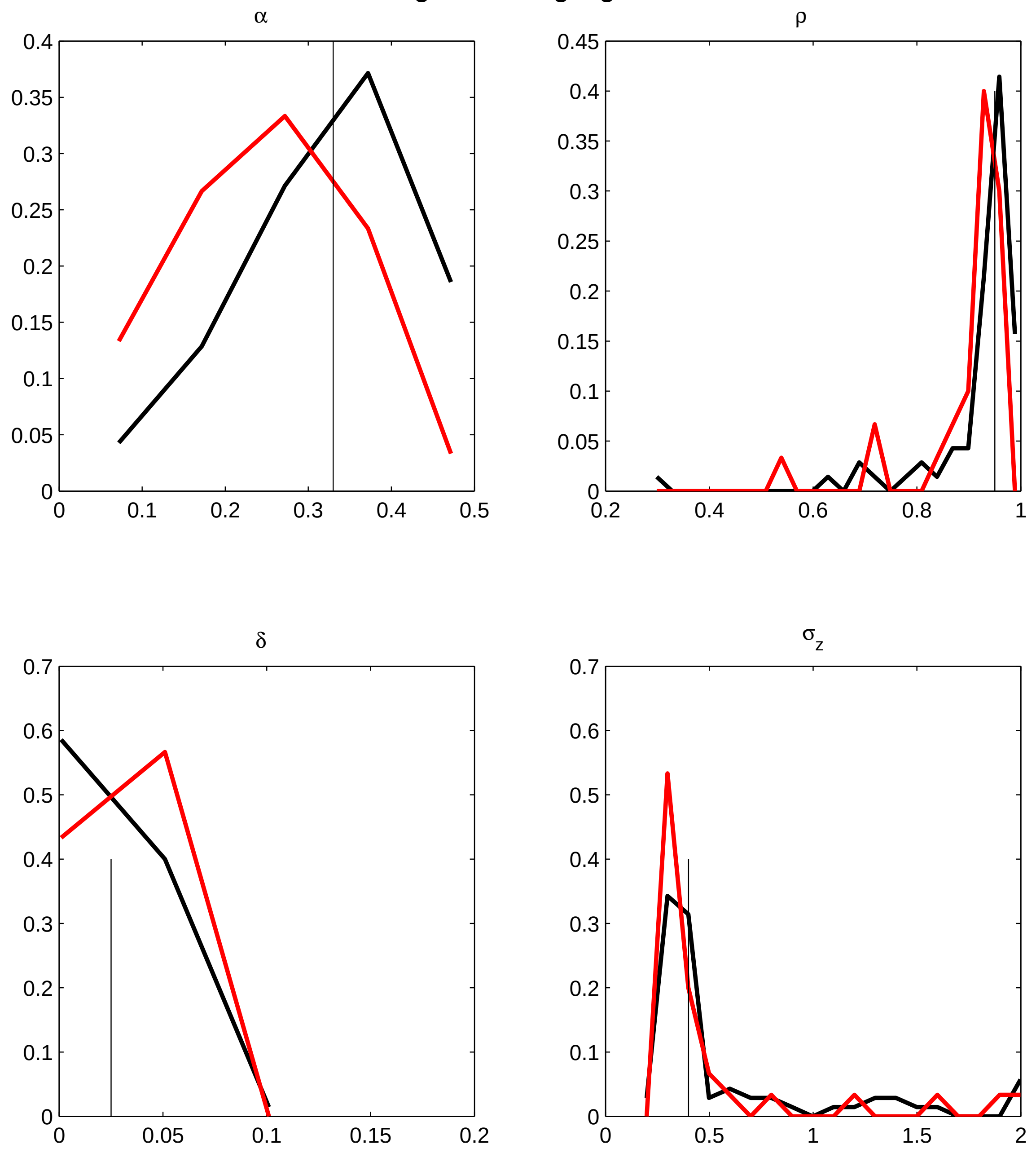
2) start away from the true values

α	δ	ρ_z	σ_z
0.5	0.3	0.1	0.2


Distribution of difference of maximum of log−likelihood found with and without AD derivatives
Starting near truth
90
80
70
60
50
40
30
20
10
0
−0.5
0
0.5
1
1.5
2


Distribution of difference of maximum of log−likelihood found with and without AD derivatives
Starting away from truth
50
45
40
35
30
25
20
15
10
5
0
−250
−200
−150
−100
−50
0
50
100
150
200
250

Sampling Distributions of Parameter Estimates, Conditioning on Finding higher Likelihood



A much larger model

The Households in the model solve:

$$
\begin{aligned}
&\max_{[C_t(h),W_t(h),I_t(h),K_{t+1}(h),B_{t+1}(h)]} E_t \sum_{j=0}^{\infty} \beta^j \Big(U(C_{t+j}(h), C_{t+j-1}(h)) \\
&+V(L_{t+j}(h))\Big) + \beta^j \lambda_{t+j}(h) \Big[\Pi_t(h) + T_{t+j}(h) + (1-\tau_{Lt})W_{t+j}(h)L_{t+j}(\\
&+(1-\tau_{Kt})R_{kt+j}K_{t+j}(h) - \frac{1}{2}\psi_I P_{t+j}\frac{\Big(I_{t+j}(h) - I_{t+j-1}(h)\Big)^2}{I_{t+j-1}(h)} \\
&-P_{t+j}C_{t+j}(h) - P_{t+j}I_{t+j}(h) - \int_s \psi_{t+j+1,t+j}B_{t+j+1}(h) + B_{t+j}(h)\Big] \\
&+\beta^j Q_{t+j}(h)\Big[(1-\delta)K_{t+j}(h) + I_{t+j}(h) - K_{t+j+1}(h)\Big].
\end{aligned}
$$

Model Description (continued)

Final production

$$Y_t = \left[\int_0^1 Y_t(f)^{\frac{1}{1+\theta_p}}\right]^{1+\theta_p} \tag{1}$$

Intermediate prodution

$$Y_t(f) = e^{Z_t} K_t(f)^{\alpha} L_t^d(f)^{1-\alpha}. \tag{2}$$

where technology is given by

$$\log(Z_t) = \rho_z \log(Z_{t-1}) + \sigma_z \epsilon_{zt}, \tag{3}$$

Finally, the government sector sets a nominal risk-free interest rate according to the reaction function:

$$i_t = \frac{\pi}{\beta} - 1 + \gamma_\pi(\pi_t - \pi) + \gamma_y(log(Y_t) - log(Y_{t-1}) + \epsilon_{it}, \tag{4}$$

The government budget constraint is

$$\tau_{Lt} W_t L_t + \tau_{Kt} R_{Kt} K_t = G_t + T_t. \tag{5}$$

we introduce shocks to labor taxes, capital taxes and government spending.

Monte Carlo Experiments

Generate 100 repetitions of 200 observations for quarterly data for four observed series: gdp growth, inflation, wage inflation, policy interest rate

Estimate 6 parameters of the model keeping all other parameters at their value in the data-generating process.

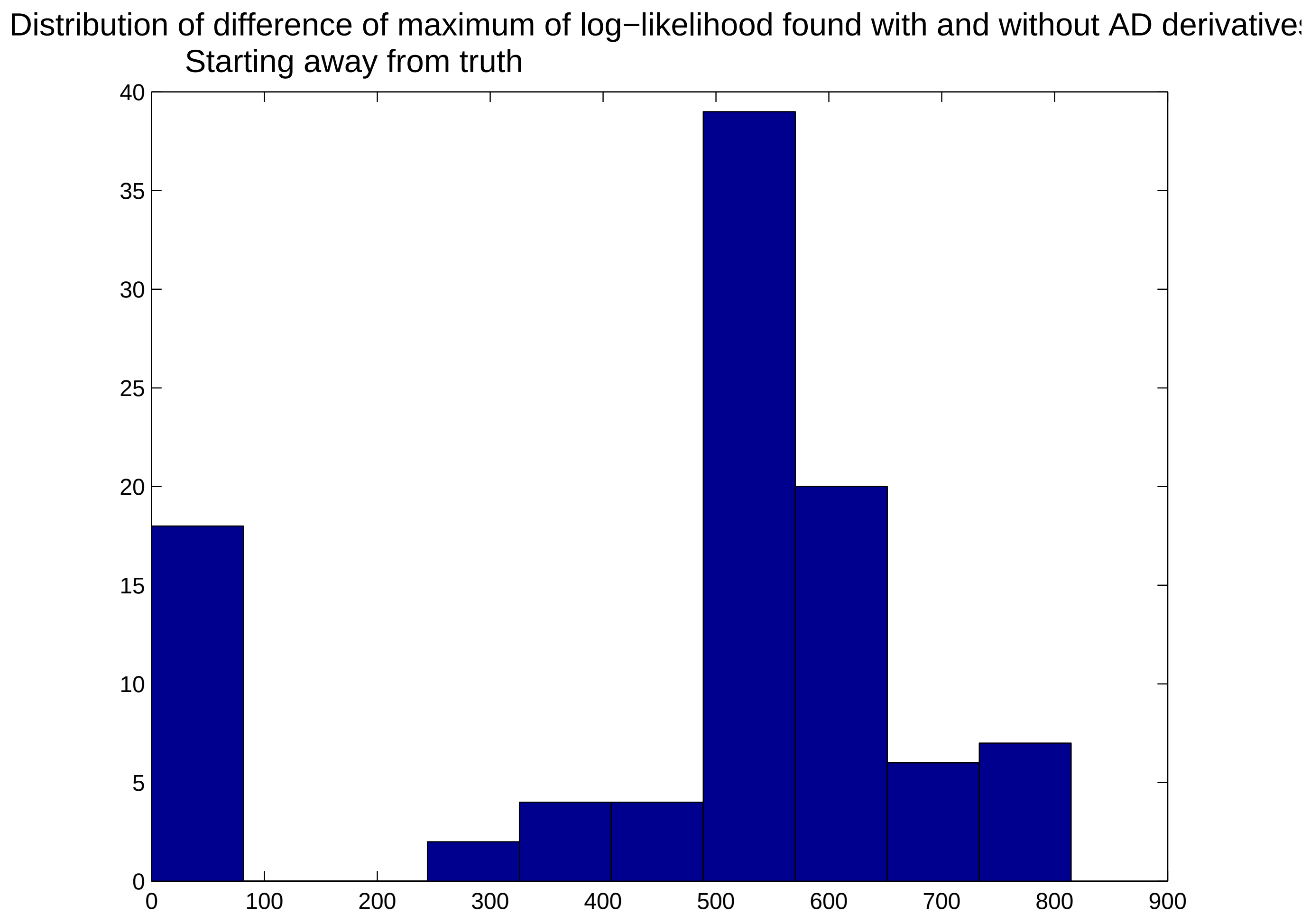
- Autoregressive coefficient on technology shock, and standard deviation of innovation.

- Weights on inflation and output growth in the interest rate reaction function.

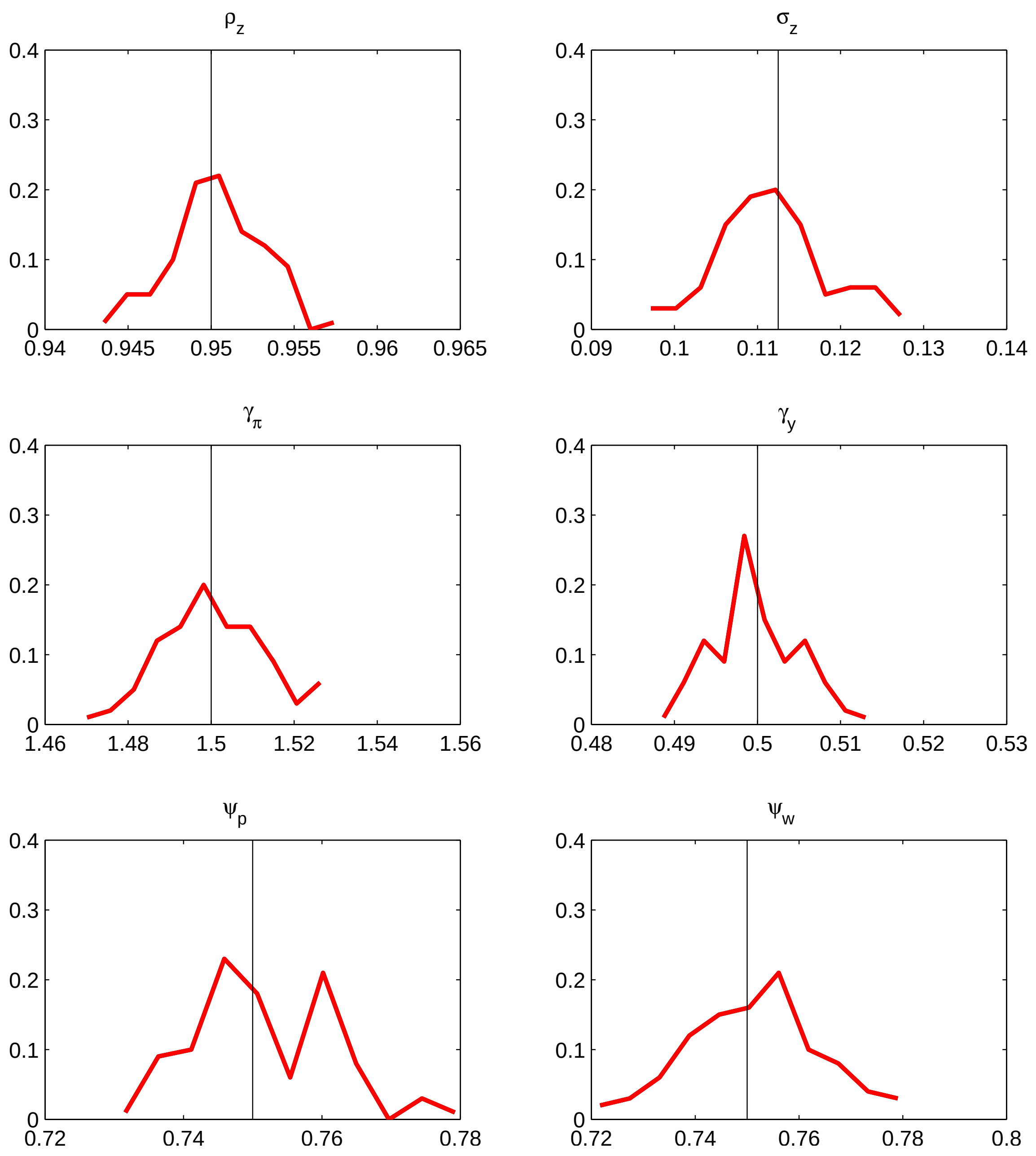
- Calvo parameters for wages and prices.

Initialization point

	ρ_z	σ_z	γ_π	γ_y	ψ_p	ψ_w
Truth	0.95	0.1125	1.5	0.5	0.75	0.75
Initial Pt	0.6	0.4	3	0.15	0.5	0.5


Distribution of difference of maximum of log−likelihood found with and without AD derivative
Starting away from truth
0
5
10
15
20
25
30
35
40
0
100
200
300
400
500
600
700
800
900

Sampling Distribution of Parameter Estimates



Related Papers

Anderson G (1987) sketches an algorithm to obtain the solution of the linear approximation of a SDGE model and its first and second derivatives.

Anderson E, Hansen, McGrattan, and Sargent (1996) computes analytical first derivative of the solution of a SDGE model.

Bastani Guerrieri (2008) builds the first derivative for the likelihood of a SDGE model with AD tools and documents some advantages for numerical optimization of the likelihood.

Iskrev (2008) focuses on the use of the information matrix to study identification.

Conclusion

We showed that the use of AD derivatives facilitates the estimation of a DGE model

Apart from speed gains, the accuracy gains over FD derivatives lead to higher convergence rates of commonly used optimization algorithms and more reliable measures of the asymptotic standard errors

Other applications include:

– checks on local identification

– gauging the importance of priors in Bayesian estimation

– facilitating the implementation of the Metropolis Hastings algorithm