Estimation and inference in DSGE models using derivatives of the likelihood

Marco Ratto and Nikolay Iskrev

Preliminary and incomplete. Comments are welcome.
This draft: September 18, 2012

Keywords: DSGE models, maximum likelihood, score, Hessian, Kalman filter

JEL classification: C32, C51, C52, E32

Maximum likelihood is rarely the method of choice for estimating DSGE models, partly because it requires the numerical solution of a highly non-linear optimisation problems. Solving such problems is usually slow and is likely to encounter convergence difficulties. It is well-known that the performance of numerical optimization algorithms can be greatly improved if they use information contained in the derivatives of the function. In particular, methods using first and second order derivatives are much faster to converge than search algorithms which do not use derivatives. In this paper we show how to efficiently evaluate the derivatives of the log-likelihood function of linear Gaussian DSGE models. The availability of such derivatives is beneficial not only in terms of easier and more reliable likelihood-based estimation

---

Contact Information: *Marco Ratto*, Joint Research Centre, European Commission, marco.ratto@jrc.ec.europa.eu; *Nikolay Iskrev*, Banco de Portugal, Nikolay.Iskrev@bportugal.pt
Prepared for the DYNARE 2012 Conference, Zurich, September 20-21, 2012.

of DSGE models, but also in the analysis of parameter identification as well as for conducting inference in empirical applications.

The main challenge in computing the log-likelihood derivatives has to do with the transformation of the parameters from the structural model to the state space representation of the solution. While it has been shown that the derivative of the transformation with respect to the vector of deep parameters can be computed analytically (Iskrev, 2010), the currently available approach makes an extensive use of sparse Kronecker-product matrices that are computationally inefficient and require a large amount of memory allocation. As a result that method is not well suited for analysis of large-scale models. In this paper we apply an alternative approach which computes derivatives with respect to each deep parameter separately. This leads to a system of generalized Sylvester equations that can be solved efficiently and accurately using existing numerical algorithms. We show that this method leads to a dramatic increase in the speed of computations at virtually no cost in terms of accuracy.

Estimating DSGE models requires also a second transformation, from the state-space representation to the objective function of the estimation problem. The latter may be the likelihood or the posterior distribution function, or a function of the distance between model-implied and empirical quantities, such as moments or impulse response functions. In this paper we focus on the likelihood function, derivatives of which are computed trough the same Kalman filtering recursion used to construct the log-likelihood function. Extensions to other objective functions are straightforward.

We demonstrate the usefulness of our method using several DSGE models of different size and complexity. Specifically, we consider the use of first and second order derivatives for, (1) local identification analysis, (2) maximizing the likelihood function with Newton methods, (3) drawing from the posterior distribution of the parameters, and (4) computing asymptotic covariances of the parameter estimates. We compare our approach to numerical differentiation and the method of Iskrev (2010) for computing analytic derivatives. All computations have been integrated into DYNARE (Adjemian et al., 2011), and our experience so far shows that our approach has substantial advantages

in terms of speed, reliability and accuracy over the alternative methods.

1 DSGE models

We recall here the notation adopted in Iskrev (2010) for linearized DSGE models. A DSGE model can be expressed as a system $\boldsymbol{g}$ of m non-linear equations:

$$E_t\Big(\boldsymbol{g}(\hat{\boldsymbol{z}}_t, \hat{\boldsymbol{z}}_{t+1}, \hat{\boldsymbol{z}}_{t-1}, \boldsymbol{u}_t|\boldsymbol{\theta})\Big) = 0 \tag{1}$$

where $\hat{\boldsymbol{z}}_t$ is a $m-$dimensional vector of endogenous variables, $\boldsymbol{u}_t$ an n-dimensional random vector of structural shocks with $\mathrm{E}\,\boldsymbol{u}_t = \boldsymbol{0}$, $\mathrm{E}\,\boldsymbol{u}_t\boldsymbol{u}_t' = \boldsymbol{\Sigma}_u$ and $\boldsymbol{\theta}$ a $k-$dimensional vector of deep parameters. $\boldsymbol{\Sigma}_u$ is $n \times n$ is a symmetric semi-positive definite matrix. Here, $\boldsymbol{\theta}$ is a point in $\boldsymbol{\Theta} \subset \mathbb{R}^k$ and the parameter space $\boldsymbol{\Theta}$ is defined as the set of all theoretically admissible values of $\boldsymbol{\theta}$.

Most studies involving either simulation or estimation of DSGE models use linear approximations of the original models around the steady-state $\hat{\boldsymbol{z}}^*$, namely:

$$\boldsymbol{\Gamma}_0(\boldsymbol{\theta})\boldsymbol{z}_t = \boldsymbol{\Gamma}_1(\boldsymbol{\theta})\,\mathrm{E}_t\,\boldsymbol{z}_{t+1} + \boldsymbol{\Gamma}_2(\boldsymbol{\theta})\boldsymbol{z}_{t-1} + \boldsymbol{\Gamma}_3(\boldsymbol{\theta})\boldsymbol{u}_t \tag{2}$$

where $\boldsymbol{z}_t = \hat{\boldsymbol{z}}_t - \hat{\boldsymbol{z}}^*$. The elements of the matrices $\boldsymbol{\Gamma}_0$, $\boldsymbol{\Gamma}_1$, $\boldsymbol{\Gamma}_2$ and $\boldsymbol{\Gamma}_3$ are functions of $\boldsymbol{\theta}$.

Assuming that a unique solution exists, it can be written as:

$$\boldsymbol{z}_t = \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{z}_{t-1} + \boldsymbol{B}(\boldsymbol{\theta})\boldsymbol{u}_t \tag{3}$$

where the $m \times m$ matrix $\boldsymbol{A}$ and the $m \times n$ matrix $\boldsymbol{B}$ are functions of $\boldsymbol{\theta}$.

For a given value of $\boldsymbol{\theta}$, the matrices $\boldsymbol{A}$, $\boldsymbol{\Omega} := \boldsymbol{B}\boldsymbol{\Sigma}_u\boldsymbol{B}'$, and $\hat{\boldsymbol{z}}^*$ completely characterize the equilibrium dynamics and steady state properties of all endogenous variables in the linearized model. Typically, some elements of these matrices are constant, i.e. independent of $\boldsymbol{\theta}$ and it is useful to separate the solution parameters that depend on $\boldsymbol{\theta}$ from those that do not. Iskrev (2010) uses $\boldsymbol{\tau}$ to denote the vector collecting the non-constant elements of $\hat{\boldsymbol{z}}^*$, $\boldsymbol{A}$, and $\boldsymbol{\Omega}$, i.e. $\boldsymbol{\tau} := [\boldsymbol{\tau}_z',\ \boldsymbol{\tau}_A',\ \boldsymbol{\tau}_\Omega']'$, where $\boldsymbol{\tau}_z$, $\boldsymbol{\tau}_A$, and $\boldsymbol{\tau}_\Omega$ denote the elements of $\hat{\boldsymbol{z}}^*$, $\mathrm{vec}(\boldsymbol{A})$ and $\mathrm{vech}(\boldsymbol{\Omega})$ that depend on $\boldsymbol{\theta}$.

In most applications the model in (3) cannot be taken to the data directly since some of the variables in $\boldsymbol{z}_t$ are not observed. Instead, the solution of the DSGE model is expressed in a state space form, with transition equation given by (3), and a measurement equation

$$\boldsymbol{x}_t = \boldsymbol{C}\boldsymbol{z}_t + \boldsymbol{D}\boldsymbol{u}_t + \boldsymbol{\nu}_t \tag{4}$$

where $\boldsymbol{x}_t$ is a l-dimensional vector of observed variables and $\boldsymbol{\nu}_t$ is a l-dimensional random vector with $\mathrm{E}\,\boldsymbol{\nu}_t = \boldsymbol{0}$, $\mathrm{E}\,\boldsymbol{\nu}_t\boldsymbol{\nu}_t' = \boldsymbol{Q}$, where $\boldsymbol{Q}$ is $l \times l$ symmetric semi-positive definite matrix [1].

2 Computing the first order derivatives of linearized DSGE models

In order to compute first derivatives of DSGE models, the key element is to take the derivatives of the transformation from $\boldsymbol{\theta}$ to $\boldsymbol{\tau}$, i.e. the Jacobian of the mapping between the deep parameters and the state space form of the linearized DSGE model. In fact, any other derivative (theoretical moments, likelihood, etc.) is directly obtained, by some form of the chain rule, from the direct derivation w.r.t. the state space form $\boldsymbol{\tau}$ (3)-(4) combined with the derivatives of $\boldsymbol{\tau}$ w.r.t. $\boldsymbol{\theta}$. The derivatives of the transformation from $\boldsymbol{\theta}$ to $\boldsymbol{\tau}$ can be divided into three groups corresponding to the three blocks of $\boldsymbol{\tau}$: $\boldsymbol{\tau}_z$, $\boldsymbol{\tau}_A$ and $\boldsymbol{\tau}_\Omega$.

2.1 The derivatives of the steady state

In Iskrev (2010) it is assumed that $\hat{\boldsymbol{z}}^*$ is a known function of $\boldsymbol{\theta}$, implied by the steady state of the model, so that the derivative of $\boldsymbol{\tau}_z$ can be computed by direct differentiation. This is in general not true, since one can implement a non-linear DGSE model in packages like DYNARE, which provide the steady state computation and linearization even when the former is not available

[1] In the DYNARE framework, the state-space and measurement equations are always formulated such that $\boldsymbol{D} = 0$

explicitly. Here we provide the extension to this case, by first noting that the 'static' model $\boldsymbol{g}^* = \boldsymbol{g}(\hat{\boldsymbol{z}}^*, \hat{\boldsymbol{z}}^*, \hat{\boldsymbol{z}}^*, 0|\boldsymbol{\theta}) = 0$ provides and implicit function between $\hat{\boldsymbol{z}}^*$ and $\boldsymbol{\theta}$. Therefore, $\frac{\partial \hat{\boldsymbol{z}}^*}{\partial \boldsymbol{\theta}'}$ can be computed exploiting the analytic derivatives of g^* with respect to $\hat{\boldsymbol{z}}^*$ and $\boldsymbol{\theta}$, provided by the symbolic preprocessor of DYNARE:

$$\frac{\partial \hat{\boldsymbol{z}}^*}{\partial \boldsymbol{\theta}'} = -\Big(\frac{\partial \boldsymbol{g}^*}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big)^{-1} \cdot \frac{\partial \boldsymbol{g}^*}{\partial \boldsymbol{\theta}'} \tag{5}$$

and finally $\frac{\partial \boldsymbol{\tau}_z}{\partial \boldsymbol{\theta}'}$ is obtained by removing the zeros corresponding to the constant elements of $\hat{\boldsymbol{z}}^*$.

2.2 The derivatives of the state space matrices

In order to properly compute the derivatives of $\boldsymbol{\tau}_A$ and $\boldsymbol{\tau}_\Omega$, the structural form (2) has to be re-written explicitly accounting for the dependency to $\hat{\boldsymbol{z}}^*$:

$$\boldsymbol{\Gamma}_0(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)\boldsymbol{z}_t = \boldsymbol{\Gamma}_1(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)\,\mathrm{E}_t\,\boldsymbol{z}_{t+1} + \boldsymbol{\Gamma}_2(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)\boldsymbol{z}_{t-1} + \boldsymbol{\Gamma}_3(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)\boldsymbol{u}_t \tag{6}$$

Also in this case, one can take advantage of the DYNARE symbolic preprocessor. The latter provides derivatives $\frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \boldsymbol{\theta}'}$ consistent with the form (6). However, since the dependence of $\hat{\boldsymbol{z}}^*$ to $\boldsymbol{\theta}$ is not known explicitly to the preprocessor, these derivatives miss the contribution of the steady state. Therefore, one has to exploit the computation of the Hessian, provided by DYNARE for the second order approximation of non-linear DSGE models. The Hessian gives the missing derivatives $\frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \hat{\boldsymbol{z}}^{*\prime}}$, allowing one to perform the correct derivation as:

$$\frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}'} = \frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}'} = \frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \boldsymbol{\theta}'} + \frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \hat{\boldsymbol{z}}^{*\prime}} \cdot \frac{\partial \hat{\boldsymbol{z}}^*}{\partial \boldsymbol{\theta}'} \tag{7}$$

The derivatives of $\boldsymbol{\tau}_A$ and $\boldsymbol{\tau}_\Omega$ can be obtained from the derivatives of $\mathrm{vec}(\boldsymbol{A})$ and $\mathrm{vech}(\boldsymbol{\Omega})$, by removing the zeros corresponding to the constant elements of $\boldsymbol{A}$ and $\boldsymbol{\Omega}$. In Iskrev (2010) the derivative of $\mathrm{vec}(\boldsymbol{A})$ is computed using the implicit function theorem. An implicit function of $\boldsymbol{\theta}$ and $\mathrm{vec}(\boldsymbol{A})$ is provided by the restrictions the structural model (2) imposes on the reduced

form (3). In particular, from (3) we have $\mathrm{E}_t\, \boldsymbol{z}_{t+1} = \boldsymbol{A}\boldsymbol{z}_t$, and substituting in (2) yields

$$(\boldsymbol{\Gamma}_0 - \boldsymbol{\Gamma}_1\boldsymbol{A})\boldsymbol{z}_t = \boldsymbol{\Gamma}_2\boldsymbol{z}_{t-1} + \boldsymbol{\Gamma}_3\boldsymbol{u}_t \tag{8}$$

Combining the last equation with equation (3) gives to the following matrix equation

$$\boldsymbol{F}(\boldsymbol{\theta}, \mathrm{vec}(\boldsymbol{A})) := \Big(\boldsymbol{\Gamma}_0(\boldsymbol{\theta}) - \boldsymbol{\Gamma}_1(\boldsymbol{\theta})\boldsymbol{A}\Big)\boldsymbol{A} - \boldsymbol{\Gamma}_2(\boldsymbol{\theta}) = \mathbf{O} \tag{9}$$

Vectorizing (9) and applying the implicit function theorem gives

$$\frac{\partial \mathrm{vec}(\boldsymbol{A})}{\partial \boldsymbol{\theta}'} = -\left(\frac{\partial \mathrm{vec}(\boldsymbol{F})}{\partial \mathrm{vec}(\boldsymbol{A})'}\right)^{-1} \frac{\partial \mathrm{vec}(\boldsymbol{F})}{\partial \boldsymbol{\theta}'} \tag{10}$$

Closed-form expressions for computing the derivatives in (10) are provided in Iskrev (2010). Such a derivation requires the use of Kronecker products, implying a dramatic growth in memory allocation requirements and in computational time as the size of the model increases. The typical size of matrices to be handled in Iskrev (2010) is of $m^2 \times m^2$, which grows very rapidly with m. Here we propose an alternative method to compute derivatives, allowing to reduce both memory requirements and the computational time. Taking the derivative of (9) with respect to each θ_j, for $j = 1, \ldots, k$, one gets a set of k equations in the unknowns $\frac{\partial \boldsymbol{A}}{\partial \theta_j}$ of the form:

$$\boldsymbol{M}(\boldsymbol{\theta})\frac{\partial \boldsymbol{A}}{\partial \theta_j} + \boldsymbol{N}(\boldsymbol{\theta})\frac{\partial \boldsymbol{A}}{\partial \theta_j}\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{Q}_j(\boldsymbol{\theta}) \tag{11}$$

where

$$\begin{aligned}
\boldsymbol{M}(\boldsymbol{\theta}) &= \Big(\boldsymbol{\Gamma}_0(\boldsymbol{\theta}) - \boldsymbol{\Gamma}_1(\boldsymbol{\theta})\boldsymbol{A}(\boldsymbol{\theta})\Big) \\
\boldsymbol{N}(\boldsymbol{\theta}) &= -\boldsymbol{\Gamma}_1(\boldsymbol{\theta}) \\
\boldsymbol{P}(\boldsymbol{\theta}) &= \boldsymbol{A}(\boldsymbol{\theta}) \\
\boldsymbol{Q}_j(\boldsymbol{\theta}) &= \frac{\partial \boldsymbol{\Gamma}_2}{\partial \theta_j} - \Big(\frac{\partial \boldsymbol{\Gamma}_0}{\partial \theta_j} - \frac{\partial \boldsymbol{\Gamma}_1}{\partial \theta_j}\boldsymbol{A}(\boldsymbol{\theta})\Big)\boldsymbol{A}(\boldsymbol{\theta})
\end{aligned}$$

Equation (11) is a generalized Sylvester equation and can be solved using

available algebraic solvers. For example, in DYNARE, this kind of equations is solved applying a QZ factorization for generalized eigenvalues of the matrices $\boldsymbol{M}(\boldsymbol{\theta})$ and $\boldsymbol{N}(\boldsymbol{\theta})$ and solving recursively the factorized problem. It is also interesting to note that the problems to be solved for different θ_j only differ in the right-hand side $\boldsymbol{Q}_j(\boldsymbol{\theta})$, allowing to perform the QZ factorization only once for all parameters in $\boldsymbol{\theta}$. In practice we replace here the single big algebraic problem of dimension $m^2 \times m^2$ of Iskrev (2010) with a set of k problems of dimension $m \times m$.

Using $\boldsymbol{\Omega} = \boldsymbol{B}\boldsymbol{B}'$, the differential of $\boldsymbol{\Omega}$ is given by

$$\mathrm{d}\boldsymbol{\Omega} = \mathrm{d}\boldsymbol{B}\boldsymbol{B}' + \boldsymbol{B}\,\mathrm{d}\boldsymbol{B}' \tag{12}$$

Having $\mathrm{d}\boldsymbol{\Omega}$ in terms of $\mathrm{d}\boldsymbol{B}$ is convenient since it shows how to obtain the derivative of $\boldsymbol{\Omega}$ from that of $\boldsymbol{B}$. Note that from equations (8) and (3) we have

$$\Big(\boldsymbol{\Gamma}_0 - \boldsymbol{\Gamma}_1\boldsymbol{A}\Big)\boldsymbol{B} = \boldsymbol{\Gamma}_3 \tag{13}$$

and therefore

$$\mathrm{d}\boldsymbol{B} = \Big(\boldsymbol{\Gamma}_0 - \boldsymbol{\Gamma}_1\boldsymbol{A}\Big)^{-1}\Big(\mathrm{d}\boldsymbol{\Gamma}_3 - (\mathrm{d}\boldsymbol{\Gamma}_0 - \mathrm{d}\boldsymbol{\Gamma}_1\boldsymbol{A} - \boldsymbol{\Gamma}_1\,\mathrm{d}\boldsymbol{A})\Big) \tag{14}$$

Thus, once $\frac{\partial \mathrm{vec}(\boldsymbol{A})}{\partial \boldsymbol{\theta}'}$ is available, it is straightforward to compute, first $\frac{\partial \mathrm{vec}(\boldsymbol{B})}{\partial \boldsymbol{\theta}'}$ and $\frac{\partial \mathrm{vech}(\boldsymbol{\Omega})}{\partial \boldsymbol{\theta}'}$, and then $\frac{\partial \boldsymbol{\tau}_A}{\partial \boldsymbol{\theta}'}$ and $\frac{\partial \boldsymbol{\tau}_\Omega}{\partial \boldsymbol{\theta}'}$.

3 Extension to second order derivatives

3.1 The derivatives of the steady state

In order to compute $\frac{\partial^2 \hat{\boldsymbol{z}}^*}{\partial\theta_j\partial\theta_l}$, we need the implicit second order derivative from the implicit function $\boldsymbol{g}^* = \boldsymbol{g}(\hat{\boldsymbol{z}}^*, \hat{\boldsymbol{z}}^*, \hat{\boldsymbol{z}}^*, 0|\boldsymbol{\theta}) = 0$:

$$\frac{\partial^2 \hat{\boldsymbol{z}}^*}{\partial\theta_j\partial\theta_l} = -\Big(\frac{\partial \boldsymbol{g}^*}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big)^{-1} \cdot \Big(\frac{\partial^2 \boldsymbol{g}^*}{\partial\theta_j\partial\theta_l} + \boldsymbol{\gamma}^*\Big) \tag{15}$$

where each element γ_h^*, $h = 1, \ldots, m$, of the vector $\boldsymbol{\gamma}^*$ is given by:

$$\gamma_h^* = \Big(\frac{\partial}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big(\frac{\partial g_h^*}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big)' \cdot \frac{\partial \hat{\boldsymbol{z}}^*}{\partial \theta_j}\Big)' \cdot \frac{\partial \hat{\boldsymbol{z}}^*}{\partial \theta_l} + \frac{\partial}{\partial \theta_j}\Big(\frac{\partial g_h^*}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big) \cdot \frac{\partial \hat{\boldsymbol{z}}^*}{\partial \theta_l} + \frac{\partial}{\partial \theta_l}\Big(\frac{\partial g_h^*}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big) \cdot \frac{\partial \hat{\boldsymbol{z}}^*}{\partial \theta_j}$$

Here we need from the DYNARE preprocessor the second order derivatives of $\boldsymbol{g}^*$ with respect to $\boldsymbol{\theta}$ and $\hat{\boldsymbol{z}}^*$ and the first order derivative of the Jacobian $\frac{\partial \boldsymbol{g}^*}{\partial \hat{\boldsymbol{z}}^{*\prime}}$ with respect to $\boldsymbol{\theta}$.

3.2 The derivatives of the state space matrices

Computing second order derivatives of the model with respect to structural parameters can be performed recursively, starting from knowing second order derivatives of $\boldsymbol{\Gamma}_i$:

$$\begin{aligned}\frac{\partial^2 \boldsymbol{\Gamma}_i(\boldsymbol{\theta})}{\partial \theta_j \partial \theta_l} = \frac{\partial^2 \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*(\boldsymbol{\theta}))}{\partial \theta_j \partial \theta_l} = \frac{\partial^2 \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \theta_j \partial \theta_l} \\ + \Big(\frac{\partial}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big(\frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big)' \cdot \frac{\partial \hat{\boldsymbol{z}}^*}{\partial \theta_j}\Big)' \cdot \frac{\partial \hat{\boldsymbol{z}}^*}{\partial \theta_l} + \frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \hat{\boldsymbol{z}}^{*\prime}} \cdot \frac{\partial^2 \hat{\boldsymbol{z}}^*}{\partial \theta_j \partial \theta_l} \\ + \frac{\partial}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big(\frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \theta_l}\Big) \cdot \frac{\partial \hat{\boldsymbol{z}}^*}{\partial \theta_j} + \frac{\partial}{\partial \theta_j}\Big(\frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big) \cdot \frac{\partial \hat{\boldsymbol{z}}^*}{\partial \theta_l} \quad (16)\end{aligned}$$

Here we can use the second order derivatives of the steady state with respect to $\boldsymbol{\theta}$ and we also need from the DYNARE preprocessor the second order derivatives of $\boldsymbol{\Gamma}_i$ with respect to $\boldsymbol{\theta}$ and $\hat{\boldsymbol{z}}^*$ ($\frac{\partial^2 \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \theta_j \partial \theta_l}$ and $\frac{\partial}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big(\frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big)$), where the latter can be obtained from DYNARE third order approximation of non-linear DSGE models. We also need the first order derivative of the Hessian $\Big(\frac{\partial \boldsymbol{\Gamma}_i(\boldsymbol{\theta}, \hat{\boldsymbol{z}}^*)}{\partial \hat{\boldsymbol{z}}^{*\prime}}\Big)$ with respect to $\boldsymbol{\theta}$.

Having obtained the second order derivatives of $\boldsymbol{\Gamma}_i$, we can work out the second order derivatives of (9) with respect to θ_j and θ_l, for $j, l = 1, \ldots, k$, getting a set of $k \cdot (k+1)/2$ equations in the unknowns $\frac{\partial^2 \boldsymbol{A}}{\partial \theta_l \partial \theta_j}$ again of the form of a generalized Sylvester equation:

$$\boldsymbol{M}(\boldsymbol{\theta})\frac{\partial^2 \boldsymbol{A}}{\partial\theta_l\partial\theta_j} + \boldsymbol{N}(\boldsymbol{\theta})\frac{\partial^2 \boldsymbol{A}}{\partial\theta_l\partial\theta_j}\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{Q}_{l,j}(\boldsymbol{\theta}) \tag{17}$$

where

$$\begin{aligned}\boldsymbol{Q}_{l,j}(\boldsymbol{\theta}) = &\frac{\partial \boldsymbol{Q}_j}{\partial\theta_l} \\ &- \Big(\frac{\partial \boldsymbol{M}(\boldsymbol{\theta})}{\partial\theta_l}\frac{\partial \boldsymbol{A}}{\partial\theta_j} + \frac{\partial \boldsymbol{N}(\boldsymbol{\theta})}{\partial\theta_l}\frac{\partial \boldsymbol{A}}{\partial\theta_j}\boldsymbol{P}(\boldsymbol{\theta}) + \boldsymbol{N}(\boldsymbol{\theta})\frac{\partial \boldsymbol{A}}{\partial\theta_j}\frac{\partial \boldsymbol{P}(\boldsymbol{\theta})}{\partial\theta_l}\Big)\end{aligned} \tag{18}$$

and

$$\begin{aligned}\frac{\partial \boldsymbol{M}(\boldsymbol{\theta})}{\partial\theta_l} &= \Big(\frac{\partial \boldsymbol{\Gamma}_0(\boldsymbol{\theta})}{\partial\theta_l} - \frac{\partial \boldsymbol{\Gamma}_1(\boldsymbol{\theta})}{\partial\theta_l}\boldsymbol{A}(\boldsymbol{\theta}) - \boldsymbol{\Gamma}_1(\boldsymbol{\theta})\frac{\partial \boldsymbol{A}(\boldsymbol{\theta})}{\partial\theta_l}\Big) \\ \frac{\partial \boldsymbol{N}(\boldsymbol{\theta})}{\partial\theta_l} &= -\frac{\partial \boldsymbol{\Gamma}_1(\boldsymbol{\theta})}{\partial\theta_l} \\ \frac{\partial \boldsymbol{P}(\boldsymbol{\theta})}{\partial\theta_l} &= \frac{\partial \boldsymbol{A}(\boldsymbol{\theta})}{\partial\theta_l} \\ \frac{\partial \boldsymbol{Q}_j(\boldsymbol{\theta})}{\partial\theta_l} &= \frac{\partial^2 \boldsymbol{\Gamma}_2}{\partial\theta_l\partial\theta_j} - \Big(\frac{\partial^2 \boldsymbol{\Gamma}_0}{\partial\theta_l\partial\theta_j} - \frac{\partial^2 \boldsymbol{\Gamma}_1}{\partial\theta_l\partial\theta_j}\boldsymbol{A}(\boldsymbol{\theta})\Big)\boldsymbol{A}(\boldsymbol{\theta}) \\ &\quad - \Big(\frac{\partial \boldsymbol{\Gamma}_0}{\partial\theta_j} - \frac{\partial \boldsymbol{\Gamma}_1}{\partial\theta_j}\boldsymbol{A}(\boldsymbol{\theta})\Big)\frac{\partial \boldsymbol{A}(\boldsymbol{\theta})}{\partial\theta_l} \\ &\quad + \frac{\partial \boldsymbol{\Gamma}_1}{\partial\theta_j}\frac{\partial \boldsymbol{A}(\boldsymbol{\theta})}{\partial\theta_l}\boldsymbol{A}(\boldsymbol{\theta})\end{aligned}$$

The problem (17) can be solved exactly in the same way as for first order derivatives, still keeping the same QZ decomposition for matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ for all $j, l = 1, \ldots, k$ and only changing the right hand side term $\boldsymbol{Q}_{l,j}$.

3.3 Using analytic derivatives for inference: DYNARE Implementation

Having solved the main problem of computing the derivatives of the state space matrices of DSGE models with respect to parameters, it is possible to compute the derivatives of theoretical moments (useful for identification analysis and moment matching procedures) and the likelihood (useful for identification and estimation).

The engine for computing first and second order analytic derivatives has been implemented in DYNARE. The identification toolbox benefits from analytic derivation in computing the Jacobian of the theoretical moments and the asymptotic Hessian of the likelihood with respect to model parameters. The likelihood computation also has the possibility to activate the option `analytic_derivation` which allows to use the analytic gradient during optimization and compute the analytic Hessian at the posterior mode. This feature in estimation is so far only available for stationary models.

4 Tests

We performed a number of tests for assessing the properties of analytic derivatives in DSGE models. First, we evaluate the accuracy and the computational time required to compute the first and second order derivatives of the vector $\boldsymbol{\tau}$ with respect to the estimated parameters. Second, we compare the use of numerical and analytic derivatives for assessing identification of DSGE models. Third, we compare the use of numerical and analytic derivatives for likelihood based (ML or Bayesian) optimization and estimation.

4.1 Computing first order derivatives

We first summarize here the results and performance of the DYNARE implementation of the computation of first derivatives of DSGE models. The performed two types of checks: (i) consistency between the two analytical approaches and the numerical one (by perturbation); (ii) gain in computational time of the Sylvester equation solution with respect to the approach in Iskrev (2010). We considered a set of models of different size and complexity: Kim (2003), An and Schorfheide (2007), Levine et al. (2008), Smets and Wouters (2007), QUEST III (Ratto et al., 2009, 2010). The models of An and Schorfheide (2007) and Smets and Wouters (2007) are linearized DSGE models, and as such their DYNARE implementation already contains explicitly the steady state dependence on $\boldsymbol{\theta}$, thus not requiring the generalized form discussed in (7). On the other hand, the models of Kim (2003), Levine

model	Computing time (s)		model size (m)
	Sylvester	Iskrev (2010)	
Kim (2003)	0.0062	0.0447	4
An and Schorfheide (2007)	0.0075	0.054	5
Levine et al. (2008)	0.016	0.109	13
Smets and Wouters (2007)	0.183	5.9	40
Ratto et al. (2009)	1.6	907.6	107
Ratto et al. (2010)	11.1	∞	210

Table 1: Computational time required for the evaluation of first order analytic derivatives of models of growing size.

et al. (2008) and QUEST III (Ratto et al., 2009, 2010) are fed to DYNARE in their full original non-linear form, thus allowing to test all elements of the proposed computational procedure.

The consistency of all different methods for computing derivatives is fulfilled in all models: in particular the maximum absolute relative difference between numerical derivatives and analytic ones was in the range $(10^{-6} - 10^{-9})$ across the different models, while the two analytic approaches are practically identical, in terms of numerical accuracy (maximum absolute relative difference in the range $(10^{-11} - 10^{-14})$). Concerning computational time, the gain of the approach proposed in this paper is evident looking at Table 1. The computational cost for the Iskrev (2010) approach becomes unsustainable for Ratto et al. (2009) and Ratto et al. (2010). Also note that we performed the tests with a 64-bit version of MATLAB, on a powerful HP ProLiant machine with 4 dual core processors (8 processors as a whole). This has a significant effect on the speed of the algorithm based on Kronecker products, linked to the multi-thread architecture of recent versions of MATLAB. Using only one single dual core processor for Smets and Wouters (2007), the computational cost doubles (11.24 s), while for Ratto et al. (2009) the computation of all derivatives lasted 47.5 minutes!

The present results show that, with the algorithms proposed in this paper, the evaluation of analytic derivatives is affordable also for DSGE models of medium/large scale, enabling to perform detailed inference analysis for such kind of models.

model	Computing time (s)		model size (m)
	Sylvester	numerical	
Kim (2003)	0.22	0.15	4
An and Schorfheide (2007)	0.13	0.85	5
Levine et al. (2008)	0.95	1.02	13
Smets and Wouters (2007)	3.5	26.3	40
Ratto et al. (2009)	151.79	179.3	107

Table 2: Computational time required for the evaluation of second order analytic derivatives of models of growing size.

model	Maximum error		model size (m)
	absolute	relative	
Kim (2003)	17	9.5e3	4
An and Schorfheide (2007)	0.64	6.4e-4	5
Levine et al. (2008)	47	5.2e7	13
Smets and Wouters (2007)	0.0013	0.3121	40
Ratto et al. (2009)	0.12	370	107

Table 3: Error in the evaluation of second order analytic derivatives of models of growing size.

4.2 Computing second order derivatives

When computing second order derivatives, we compare the computational time to obtain the derivatives of $\boldsymbol{\tau}$ with numerical differentiation and with analytic derivatives (Table 2). Moreover, we compute the maximum absolute and relative error in second order derivatives computed numerically with respect to the analytic values (Table 3). As we can see, the numerical errors in computing second order derivatives tend to grow quite significantly as the model size or the non-linearity grows. It is also worth anticipating here that such errors, which sometimes happen to be quite large for individual second order derivatives of the model solution, happen to be much smaller in terms of the likelihood Hessian, which is perhaps the most important quantity that is evaluated for inference.

4.3 Identification analysis

In the case of identification analysis, one key issue is the detection of perfect collinearity among derivatives of theoretical moments of the observables with respect to different deep parameters. In that case we have noted that there is a clear advantage in the analytic derivation. The key issue is, in fact, to be able to distinguish possible weak indentification, that is, near linear dependence, form true perfect collinearity. Errors related to numerical differentiations introduce a non systematic component in the computation of the Jacobian matrices. As a result, the rank test for singularity is much more sensitive to the significance threshold set the by user in checking the rank when numerical derivatives are used. One example of this is the An and Schorfheide (2007) model: the identification of a specification of this model where the output gap is targeted in the Taylor rule is analyzed in Komunjer and Ng (2011), who study its identification failure. Repeating the identification analysis for An and Schorfheide (2007) with analytic derivatives there are several noticeable differences with respect to Komunjer and Ng (2011). First, they associate the lack of identification with linear dependence among the three Taylor rule parameters, which, as can be seen in the output from Dynare, are highly but not perfectly collinear (exact collinearity occurs only when the variance of the monetary shock is included). Second, they have to set much larger level of tolerance for Matlab's rank in order to correctly detect lack of identification (1.e-3). In particular, they report that at the default level their condition wrongly indicates identifiability. This may be due to the use of numerical derivatives although, on the computer we have tested it, DYNARE (using numerical derivatives) gives correct answer at TOL = 1e-8 and larger (the default is larger) and fails at TOL = 1e-9 or smaller (normalizing Jacobians with numerical derivatives, tolerance can be tightened at 1.e-11). Using analytic derivatives, DYNARE provides the right answer with TOL=1.e-13 (and larger) which is also the default in DYNARE, while using normalization (which is the default in DYNARE Identification) TOL can be tightened at 1.e-17. This example shows that, when non-identification is due to collinearity, numerical derivatives may imply a sensitivity with respect to

numerical tolerances in determining the rank in Jacobians. This can have strong implications in the understanding of the sources of non-identification.

4.4 Likelihood inference

One interesting aspect of using analytical derivatives is to see whether there are advantages in the estimation context. We report here some preliminary results. We have compared the estimation results of a number of models, both using analytic and numerical derivatives. We first considered the DYNARE implementations of Lubik and Schorfheide (2005, 2007) and Schorfheide (2000). In both cases, using numerical and analytic derivatives did not result in significant differences in terms of convergence of the optimization and the Metropolis (results not reported here). We then tried to move to larger models, where we report results in more detail: Smets and Wouters (2007) and Ratto et al. (2009).

In the case of Smets and Wouters (2007) the MATLAB `fmincon` with analytic scores provides the best speed of convergence to the posterior mode. Using numerical derivatives with the same optimizers takes about 10% more time. The computation of analytic scores is also more time consuming than the numeric ones. Concerning accuracy, numerical gradient (scores) is rather precise, while errors in the Hessian can be very large. This however does not imply big errors in the estimation of standard errors of parameter estimates, with relative errors smaller than 1%. Considering that the Hessian is just the starting point of a Metropolis algorithm for full posterior estimation, the use of analytic Hessian (much more time consuming than the numeric one, at least in the current DYNARE implementation) does not provide special advantages over the numeric ones.

In the case of Ratto et al. (2009) the portrait is more interesting. First, for this model all optimizers except `mode_compute=5` fail to converge to the posterior mode with numerical derivatives. One reason for this is presumably linked to scaling issues, where the values of estimated parameters may vary in order of magnitude (e.g. large adjustment cost parameters a la Rothemberg vis à vis small standard deviations). The optimizer `mode_compute=5`

	analytic deriv. `mode_compute=1`	numeric deriv. `mode_compute=1`
iterations	56	59
optimization	112.2s	124.7s
scores	1.652s	1.834s
Hessian	273.6s	61.5s

Table 4: Speed in the estimation and in computing scores and Hessian for Smets and Wouters (2007).

	Maximum error	
	absolute	relative
scores	0.0672	0.0467%
Hessian	152.9	93.9%
st.err	3.437e-4	0.317%

Table 5: Accuracy of numerical derivatives in computing scores, Hessian and standard errors for Smets and Wouters (2007).

is a sort of 'brute force' algorithm, where at each iteration the increment of each parameter to compute gradients is tuned according to its effect on the objective function. This makes gradient computations longer but more adaptive to scaling. Moreover, this optimizer also allows to perform sequences of univariate optimizations, one for each estimated parameter: this is very useful when the direction for the line search has a bad angle. Using analytic derivatives, the MATLAB `fmincon` algorithm works very efficiently, reducing the time to reach the posterior mode by about 66% (1 hour instead of three hours). In terms of speed of computing scores, the analytical score take about twice the time for the numerical gradient. However, the efficiency of the optimizer combined with analytic scores results in a large improvement in the speed of convergence. The Hessian is very time consuming to compute (about 7 times longer).

In terms of accuracy, errors of numerical gradient can be larger than 100%, thus explaining the difficulties in getting to the posterior mode with all types of optimizers. The errors in the numerical Hessian can also be as large as 110% relative to the analytical ones. However, the resulting estimates of standard errors of estimated parameters are quite close, with

	analytic deriv. `mode_compute=1`	numeric deriv. `mode_compute=5`
optimization	1h07m	3h02m
scores	9.34s	5.75s
Hessian	2780s	387s

Table 6: Speed in the estimation and in computing scores and Hessian for Ratto et al. (2009).

	Maximum error	
	absolute	relative
scores	149.5	113%
Hessian	1.23e7	112%
st.err	0.269	4.6%

Table 7: Accuracy of numerical derivatives in computing scores, Hessian and standard errors for Ratto et al. (2009).

the difference between analytical and numerical ones being most 4.5%. As a result, Metropolis convergence is very similar when using numeric or analytic Hessian to define the covariance of the proposal distribution.

5 Conclusions

We have discussed an efficient method to compute analytical derivatives of linearized DSGE models with respect to estimated parameters. The proposed method (based on solving a set of algebraic Sylvester equations) improves with respect to previous methods that make large use oh Kronecker products and are therefore unsuitable for medium-large scale models. The use of analytic derivatives provides clear advantages for analysing identification of DSGE model, in which numerical derivations are prone to make weak- and non-identification undistinguishable. The accuracy in detecting perfect collinearity using analytic derivations, allows a more robust and cleaner interpretation and diagnostics of model properties. In the case of estimation, our preliminary tests showed that, for medium-large scale models, there can be a clear speed-up in converging to the posterior mode when using MATLAB `fmincon` coupled with analytic derivatives. So far, however, we did not

detect a clear advantage in using analytic Hessian (which is more time consuming) with respect to the numeric one both in determining the standard errors of parameter estimates and in defining the covariance of the proposal distribution for Metropolis algorithms.

References

Adjemian, S., H. Bastani, F. Karame, M. Juillard, J. Maih, F. Mihoubi, G. Perendia, M. Ratto, and S. Villemot (2011). Dynare: Reference manual, version 4. *Dynare Working Papers Series* (1). http://ideas.repec.org/p/cpm/dynare/001.html.

An, S. and F. Schorfheide (2007). Bayesian analysis of DSGE models. *Econometric Reviews 26*(2-4), 113–172. DOI:10.1080/07474930701220071.

Iskrev, N. (2010). Local identification in DSGE models. *Journal of Monetary Economics 57*, 189–202.

Kim, J. (2003, February). Functional equivalence between intertemporal and multisectoral investment adjustment costs. *Journal of Economic Dynamics and Control 27*(4), 533–549.

Komunjer, I. and S. Ng (2011, November). Dynamic identification of DSGE models. *Econometrica 79*(6), 1995–2032–1423.

Levine, P., J. Pearlman, and R. Pierse (2008). Linear-quadratic approximation, external habit and targeting rules. *Journal of Economic Dynamics and Control 32*(10), 3315 – 3349.

Lubik, T. and F. Schorfheide (2005, May). A bayesian look at new open economy macroeconomics. Economics Working Paper Archive 521, The Johns Hopkins University,Department of Economics. available at http://ideas.repec.org/p/jhu/papers/521.html.

Lubik, T. A. and F. Schorfheide (2007, May). Do central banks respond to exchange rate movements? a structural investigation. *Journal of Monetary Economics 54*(4), 1069–1087.

Ratto, M., W. Roeger, and J. in 't Veld (2009). QUEST III: An estimated open-economy DSGE model of the euro area with fiscal and monetary policy. *Economic Modelling 26*(1), 222 – 233.

Ratto, M., W. Roeger, and J. in 't Veld (2010, January). Using a DSGE model to look at the recent boom-bust cycle in the US. European Economy. Economic Papers 397, European Commission, Brussels.

Schorfheide, F. (2000). Loss function-based evaluation of DSGE models. *Journal of Applied Econometrics 15*(6), 645–670. available at http://ideas.repec.org/a/jae/japmet/v15y2000i6p645-670.html.

Smets, F. and R. Wouters (2007, June). Shocks and frictions in US business cycles: A Bayesian DSGE approach. *The American Economic Review 97*(3), 586–606.