18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

Solving Models with *Off-the-Shelf* Software: An Example of Potential Pitfalls Associated with the Use and Abuse of Default Parameter Settings

Ric D. Herbert[1] **and Peter J. Stemp**[2]

[1] School of Design, Communication and Information Technology, The University of Newcastle, Ourimbah, New South Wales, 2258, Australia. Email address: ric.herbert@newcastle.edu.au.

[2] Department of Economics, Monash University, P.O. Box 197, Clayton East, Victoria, 3145, Australia. Email address: peter.stemp@buseco.monash.edu.au

Abstract:

When working with large-scale models or numerous small models, there can be a temptation to rely on default settings in proprietary software to derive solutions to the model. In this paper we show that, for the solution of non-linear dynamic models, this approach can be inappropriate.

We consider a simple linear model with two stable eigenvalues (real-valued or complex-valued) and one unstable eigenvalue (real-valued). The chosen model is an extended version of a model previously described by Turnovsky (2000). It can be derived from the Sargent and Wallace (1973) extension of the Cagan (1956) model by adding a labour market defined by both employment and wages and by introducing sluggish adjustment for both these variables.

We choose parameters for the model. The configuration of eigenvalues can be changed by choosing alternative parameter values. It is possible to choose parameter configurations that generate complex-valued eigenvalues and real-valued eigenvalues. For this paper we focus on the real-valued eigenvalues. Alternative linear and non-linear specifications of the model are examined. One version of the model, expressed in levels, is highly non-linear. A second version of the model, expressed in logarithms, is linear. The dynamic solution of each model version has a combination of stable and unstable eigenvalues so that any dynamic solution requires the calculation of appropriate “jumps” in endogenous variables.

We can derive a closed-form solution of the model, which we use as our "true" benchmark, for comparison with computational solutions of both linear and non-linear models. Our approach is to compare the "goodness of fit" of reverse-shooting solutions for both the linear and non-linear model, by comparing the computational solutions with the benchmark solution. To make all three solutions comparable, each solution is expressed in levels, with the closed-form and computational solutions of the linear model being converted to levels by taking exp of endogenous variables.

Under the basic solution method with default settings and using the "eyeball" metric, we show that there is significant difference between the computational solution for the non-linear model and the benchmark closed-form solution. We show that this result can be substantially improved using modifications to the solver and to parameter settings. A Table documents the different approaches considered in determining the best method for solving the non-linear model. We discuss how we have used a variety of approaches before finding a solution methodology that did not fail.

In this paper, we have considered linear and non-linear versions of a dynamic macroeconomic model. We have shown that standard default settings in proprietary software give unsatisfactory results for solving the dynamic path of the non-linear model. Our results demonstrate how the solution of the model can be substantially improved by changes to parameter settings.

Keywords: *Solving Non-linear Models, Reverse-Shooting, Computational Economics , Computer Software.*

1. INTRODUCTION

When working with large-scale models or numerous small models, there can be a temptation to rely on default settings in proprietary software to derive solutions to the model. In this paper we show that, for the solution of non-linear dynamic models, this approach can be inappropriate.

We consider a simple linear model with two stable eigenvalues (real-valued or complex-valued) and one unstable eigenvalue (real-valued). Alternative linear and non-linear specifications of the model are examined. One version of the model, expressed in levels, is highly non-linear. A second version of the model, expressed in logarithms, is linear. The dynamic solution of each model version has a combination of stable and unstable eigenvalues so that any dynamic solution requires the calculation of appropriate “jumps” in endogenous variables.

We start by showing the results that can be derived using the default settings of a typical solver. We show that this gives unsatisfactory results for the non-linear model; and then, show how the default settings and choice of solver can be adjusted to give acceptable results. This paper builds on previous work by Stemp and Herbert (2006) and Herbert and Stemp (2008).

2. THE MODEL

2.1 The linear model (expressed in logarithms)

The following model is an extended version of a model previously described by Turnovsky (2000). It can be derived from the Sargent and Wallace (1973) extension of the Cagan (1956) model by adding a labour market defined by both employment and wages and by introducing sluggish adjustment for both these variables. The model is given by the following set of equations:

$$\bar{m} - p = \alpha_1 y - \alpha_2 \dot{p} \qquad (1a)$$

$$y = \beta + (1-\gamma)n, 0 < \gamma < 1 \qquad (1b)$$

$$\dot{n} = \theta(\delta - \gamma n - w + p) \qquad (1c)$$

$$\dot{w} = \eta(n - \bar{n}) \qquad (1d)$$

where Greek symbols denote parameters with a positive value, all variables are functions of time and lower-case letters denote logarithms:
y = output (expressed in logarithms);
n = employment (expressed in logarithms);
$\bar{n}$ = full employment (expressed in logarithms);
p = price level (expressed in logarithms);
$\bar{m}$ = nominal money stock (expressed in logarithms), assumed to be constant; and
w = wage rate (expressed in logarithms).

Since this model is linear, it has a unique equilibrium which satisfies the following equations:

$$\bar{m} - p = \alpha_1 y \qquad (2a)$$

$$y = \beta + (1-\gamma)n \qquad (2b)$$

$$\delta - \gamma n = w - p \qquad (2c)$$

$$n = \bar{n} \qquad (2d)$$

We will refer to this specification of the model as "the linear model".

2.2 Closed-form solution of the linear model

We choose parameters for the model as follows: $\alpha_1 = 1$, $\alpha_2 = 0.5$, $\beta = 0$, $\gamma = 0.5$, $\delta = \log(0.5)$ and $\theta = 1$. The exogenous variables are $\bar{n} = 1$, $\bar{m} = 0.1$ and $\bar{m}_0 = 0$.

As we have shown previously (Stemp and Herbert, 2006) the configuration of eigenvalues can be changed by choosing alternative values for the parameter η, with η chosen from the values: 0.2, 0.5, 10.0 and 100.0; the associated eigenvalues are given by $\lambda_1, \lambda_2, \lambda_3$ in Table 1.

Table 1. Values of η and resulting eigenvalues

η	λ_1	λ_2, λ_3
0.2	2.3	-0.50, -0.34
0.5	2.3	$-0.41 \pm 0.51i$
10.0	2.1	$-0.32 \pm 3.0i$
100.0	2.0	$-0.26 \pm 9.9i$

It will be observed that as η grows increasingly positive, two of the eigenvalues first become complex-valued and then as η increases further the imaginary part of the complex-valued eigenvalues become increasingly large. Eigenvalues with large imaginary parts are associated with increasingly oscillating dynamics.

The dynamic solutions assume that initially the model is in a steady-state associated with $m = \bar{m}_0$. Following a monetary shock to $m = \bar{m}$, the price level "jumps" so that the economy moves onto the stable trajectory and evolves to a new post-shock steady-state (Blanchard and Kahn, 1980)

A closed-form solution for the linear model can be derived using standard matrix techniques. Figure 1 shows dynamic paths for each of the endogenous variables: p, n and w, as the parameter η is allowed to vary across the four values given in Table 1. The dynamic solution path becomes increasingly oscillatory as η is allowed to increase.

2.3 The non-linear model (expressed in levels)

An equivalent model specification can be rewritten in levels as follows:

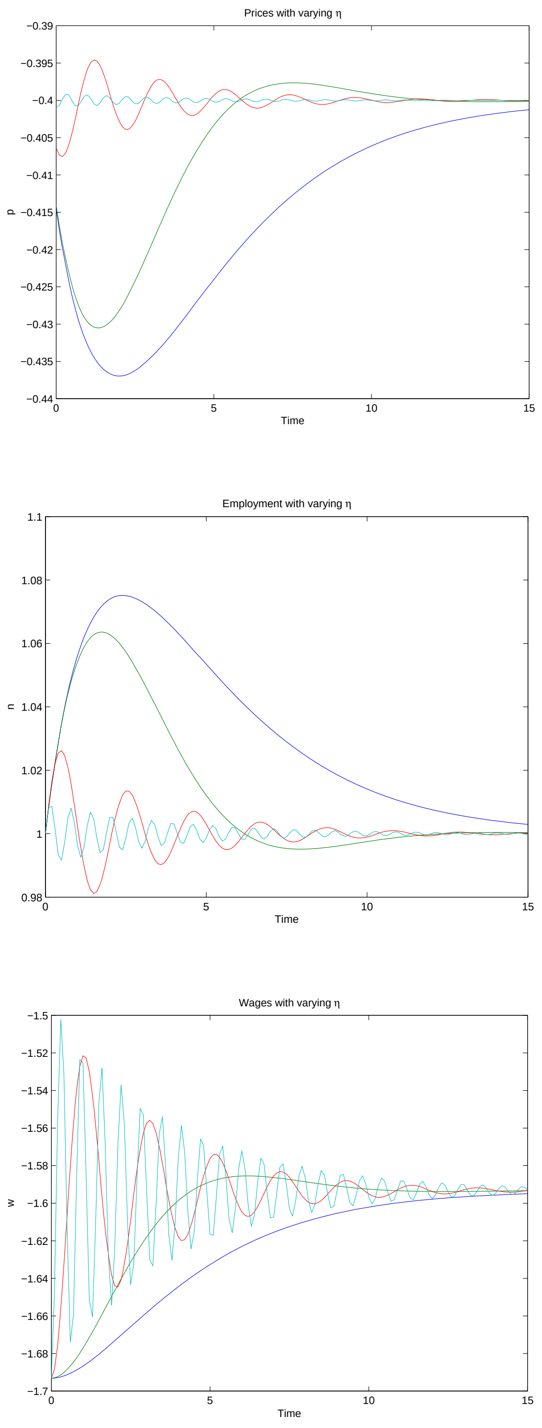

Figure 1. Plots of p, n and w with varying values of η derived from analytic solution of linear model.

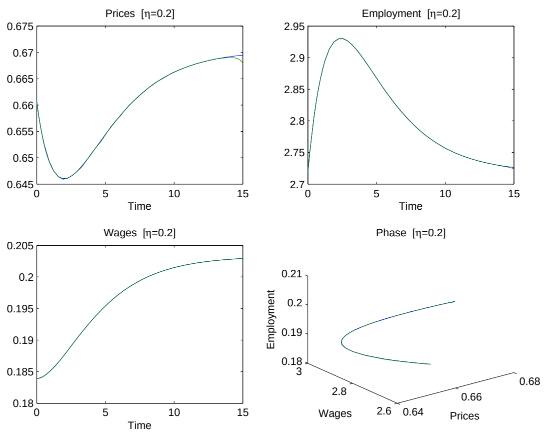

Figure 2. Solution to linear model using Runge-Kutta solver with default settings: $\eta = 0.2$

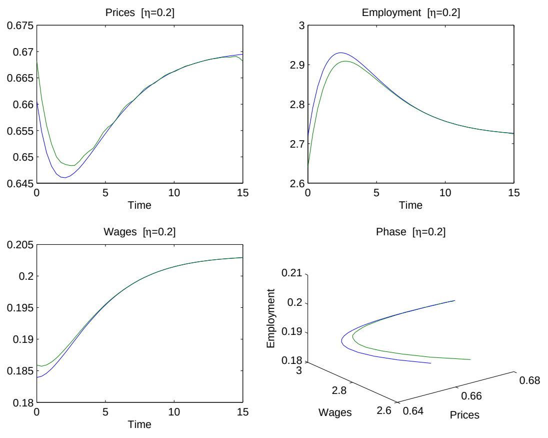

Figure 3. Solution to non-linear model using Runge-Kutta solver with default settings: $\eta = 0.2$

$$\dot{P} = P.\log\left[\frac{P.N^{(1-\gamma)\alpha_1}}{\overline{M}}\right]^{\frac{1}{\alpha_2}} \qquad (3a)$$

$$\dot{N} = N.\log\left[\frac{P.(1-\gamma)N^{-\gamma}}{W}\right]^{\theta} \qquad (3b)$$

$$\dot{W} = W.\log\left(\frac{N}{\overline{N}}\right)^{\eta} \qquad (3c)$$

The variables are as for the model above except that upper case letters denote levels, so that:
Y = output;
N = employment;
$\overline{N}$ = full employment;
P = price level;
$\overline{M}$ = nominal money stock, assumed to be constant; and
W = wage rate.

In particular, some (but not all) solutions to the non-linear model can be derived from solutions to the linear model using the following transformation:

$$P = \exp(p) \qquad (4a)$$

$$N = \exp(n) \qquad (4b)$$

$$W = \exp(w) \qquad (4c)$$

This non-linear specification of the model has multiple equilibria. There is an equilibrium that corresponds to the equilibrium for the linear specification of the model

.

But there is another equilibrium as well. For example, there is an additional equilibrium given by:

$$P = 0 \qquad (5a)$$

$$N = \overline{N} \qquad (5b)$$

$$W = 0 \qquad (5c)$$

We will focus on the dynamic solution that converges to the same equilibrium as the linear model and will refer to this specification of the model as "the non-linear model".

3. SOLVING THE MODEL

3.1 Previous work

Whilst it is possible to provide solutions to the linear model in closed-form, the non-linear model can only be solved using computational techniques. Two standard computational approaches to dynamic models with "jump" variables, like that considered in this paper, are the forward-shooting and reverse-shooting algorithms. We have previously considered these approaches in conjunction with the model considered in this paper (Stemp and Herbert, 2006; Herbert and Stemp, 2008).

In Stemp and Herbert (2006), we focused on the case of the linear model with two stable complex-valued eigenvalues and one unstable real-valued eigenvalue. We employed the closed-form solution of the linear model as a benchmark to compare the properties of model solutions derived using reverse-shooting and forward-shooting. Because the model has complex-valued eigenvalues, it will have cyclic dynamics and we argued that problems encountered in solving these dynamics would likely coincide with some of the problems that the solution algorithms would have in solving non-linear models. Focussing on the case when $\eta = 100$, we showed that the choice of ODE solver has a significant impact on the likely success of the shooting algorithms.

In Herbert and Stemp (2008), we considered both the linear model and the non-linear model. By attempting to solve these models using forward-shooting and reverse-shooting algorithms, we showed that the forward-shooting algorithm is likely to choose a "wrong" solution (defined as a solution that converges to a steady-state equilibrium that is not economically meaningful), but the "right" solution for the linear model. The reverse-shooting algorithm chooses the "right" solution in both cases. Since for this paper we intend to compare computational solutions of both the linear and non-linear models, we will employ the reverse-shooting algorithm in this paper as our preferred computational algorithm for solving both models.

In the case of the standard two-dimensional model, reverse-shooting involves just one search in time starting from the neighbourhood of the steady-state, with the model dynamics throwing the dynamic solution onto the stable arm of the saddle-path. Reverse-shooting for higher dimensional models with more than two stable eigenvalues requires search over a grid (the stable manifold) with the dimension of the grid equal to the number of stable eigenvalues.

Our approach is to compare the "goodness of fit" of reverse-shooting solutions for both the linear and non-linear model, by comparing the computational solutions with the benchmark solution, given by the closed-form solution of the linear model. To make all three solutions comparable, each solution is expressed in levels, with the closed-form and computational solutions of the linear model being converted to levels by taking exp of endogenous variables as in equations (4a-4c).

The reverse-shooting algorithm has two essential components: a differential equation solver to solve for each candidate path and a search routine that chooses among possible candidate paths and determines when an acceptable candidate path has been found. Our choice of routines is based, as well as on computational efficiency, on our aim that the chosen software can solve a wide variety of models with limited knowledge of the model. For the search method we use a Nelder-Mead direct simplex search (Lagarias et al., 1998). We implement this search by the MATLAB function *fminsearch*. There is a range of differential equation solvers available in MATLAB (Shampine and Reichelt, 1997). We consider three candidate solvers: The Runge-Kutta solver, the Adams-Bashforth-Moulton predictor-corrector algorithm, and a Stiff solver. Each of these candidate solvers is described below in turn.

3.2 Runge-Kutta (RK) solver

We use the variable step-size Runge-Kutta algorithm as implemented as a solver by the MATLAB function *ode45* (see Mathworks, 2009). The algorithm uses a 5th-order Runge-Kutta formula to estimate the local truncation error for the 4th order Runge-Kutta formula. We use the step-size that gives an absolute tolerance in the local truncation error of at least 10^{-6} and a default relative tolerance of the truncation error relative to the size of the solution of at least 10^{-3}. The Runge Kutta algorithm is a single-step solver so that only the solution at the previous time-step is used in determining the current solution.

3.3 Adams-Bashforth-Moulton (ABM) solver

This is a multi-step, variable step-size, algorithm, which usually needs several preceding time-point solutions to compute the current time point solution. The ABM solver is regarded as better than the Runge-Kutta solver when the ODE function is expensive to calculate and has been implemented using the MATLAB function *ode113*. We first implement the ABM solver using the default maximum step-size parameter. Again we use the default absolute tolerance in the local truncation error of at least 10^{-6} and the default relative tolerance of at least 10^{-3}.

3.4 Stiff solver

With a model such as the one in this paper where there are multiple differential equations the issue of stiffness can arise. If stiffness is an issue then the usual solution methods often become unstable. Stiffness is not precisely defined but "... typically occurs in a problem where there are two or more very different time scales of the independent variable on which the dependent variables are changing" (Press et al, 2007, p. 931). This could be indicated by a large relative variation in the eigenvalues. The Stiff solver has been implemented using the MATLAB function *ode115s*.

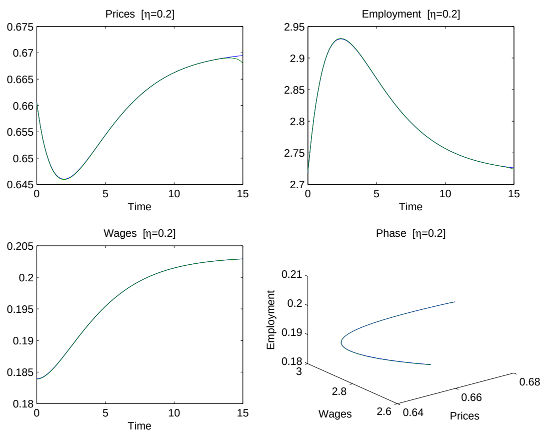

Figure 4. Solution to linear model using ABM solver with maximum step-size 0.01: $\eta = 0.2$

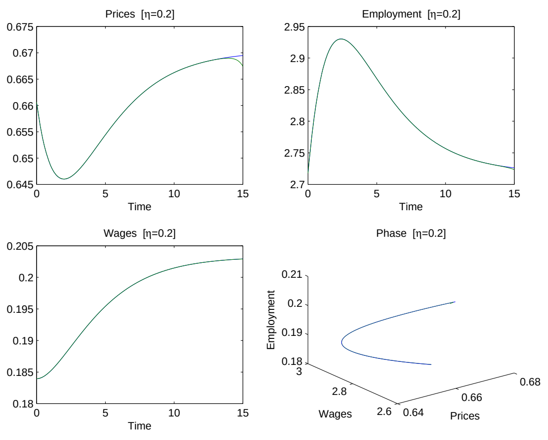

Figure 5. Solution to non-linear model using ABM solver with maximum step-size 0.01: $\eta = 0.2$

4. RESULTS

4.1 Current approach

In Stemp and Herbert (2006) we focused on the computational solution of the linear model in the most oscillatory case, which is the outcome for wages when $\eta = 100$. Using the RK solver, the reverse-shooting algorithm had difficulty reproducing the correct solution. The ABM solver was able to do much better. It turns out that the considerable improvement in "goodness of fit" was driven by the smaller step-size rather than the choice of solver. In this paper we have a truly non-linear version of the model, rather than just a linear version that we use to simulate some of the likely properties of a non-linear model through oscialltory behaviour. Accordingly, here we focus on the case when eigenvalues are real-valued. In particular, we consider the case when $\eta = 0.2$.

4.2 The case when $\eta = 0.2$

Figure 2 shows the dynamic comparison of the solution to the linear model for $\eta = 0.2$, using the most basic solver: the RK solver with default settings. Using the "eyeball" metric, there is virtually no difference between the computational solution and the benchmark closed-form solution. This can be contrasted with Figure 3 which shows a similar solution for the non-linear model. There, the "eyeball" metric shows significant difference between the computational solution for the non-linear model and the benchmark closed-form solution.

In Stemp and Herbert (2006), we focused on the case when $\eta = 100$. Our preferred solver in this case was an ABM solver with small step-size. Figures 4 and 5 show dynamic comparisons for the case when $\eta = 0.2$ with the same solver, an ABM solver with maximum step-size of 0.01; for these figures, our "eyeball" metric shows that the linear model remains a good fit using this solver and the non-linear model is now also a good fit using the ABM solver with small step-size.

Since major improvement for the different solvers has come about for the non-linear model, we will discuss our approach to improving the solution to this model in greater detail.

4.3 The non-linear model

Table 2 documents the different approaches considered in determining the best method for solving the non-linear model. In all cases we report the RMSE (root mean square error) norm which is an L_2 norm adjusted for the number of sample points. The norm is reported for the aggregate model as well as for the three endogenous variables separately. It can be observed that the outcome under the preferred solver, ABM solver with maximum step size of 0.01, is by far the best outcome.

Table 2. Norm results for all methods: non-linear model when $\eta = 0.2$

Method	RMSE All	RMSE P	RMSE N	RMSE W
RK Solver with defaults	0.007744	0.001996	0.021559	0.000506
RK Solver with tolerances half the defaults	0.000478	0.000936	0.001303	8.60E-06
ABM Solver with defaults	0.000267	0.000521	0.000743	5.58E-06
ABM Solver with half tolerances	0.000826	0.001622	0.002254	1.45E-05
ABM Solver with half tolerances on ode and refine	0.000811	0.001591	0.002212	1.42E-05
ABM Solver with max step-size 0.01	0.000133	0.000248	0.000364	5.03E-06
Stiff Solver with defaults	0.125903	0.034000	0.305671	0.007573
Stiff Solver with half tolerances and refine	0.000762	0.001470	0.002083	1.61E-05

As can be seen from the table, different methods have been used to augment basic defaults. The computational effort for the solver is directly related to the step size and so the solvers used here choose a step size based on the error tolerances. When more accuracy is needed reducing the step size (through specifying a smaller maximum step size) is usually considered the approach of last resort. The recommended approach is to reduce the error tolerances. In the approach we used here we have halved the default error tolerances (to 10^{-6}) and the solvers still fail to produce accurate solutions. Another approach is to produce more output points by using equations that are part of the solver to generate solutions between the steps. These require low computational cost. The Matlab ode solvers have an additional 'refine' factor to obtain more solutions between the step points. (Mathworks, 2009). We have used these approaches but the solvers still fail. The Stiff solver has also not been especially successful for this problem.

5. CONCLUSION

In this paper, we have considered linear and non-linear versions of a dynamic macroeconomic model. We have shown that standard default settings in proprietary software give unsatisfactory results for solving the dynamic path of the non-linear model. Our results demonstrate how the solution of the model can be substantially improved by changes to parameter settings.

REFERENCES

Blanchard, O. J., and C. M. Kahn (1980), “The solution of linear difference models under rational expectations,” *Econometrica*, 48, pp. 1305-1311.

Cagan, P. (1956), “The monetary dynamics of hyperinflation,” in M Friedman (editor), *Studies in the Quantity Theory of Money*, University of Chicago Press, Chicago.

Herbert, R. D., and P. J. Stemp (2008), “Solving a non-linear model: The importance of model specification for deriving a suitable solution,” *Mathematics and Computers in Simulation*, doi:10.1016/j.matcom.2008.04.010

MathWorks (2009), website: www.mathworks.com,

Lagarias, J. C., J. A. Reeds, M. H. Wright and P. E. Wright (1998), Convergence properties of the Nelder-Mead simplex method in low dimensions, *SIAM Journal of Optimization*, 9(1), p.112-147.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (2007), *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, 3rd Edition.

Sargent, T. J., and N. Wallace (1973), “The stability of models of money and growth with perfect foresight,” *Econometrica*, 41, pp. 1043-1048.

Shampine, L. F., and M. W. Reichelt (1997), "The MATLAB ODE Suite," *SIAM Journal on Scientific Computing*, 18(1).

Stemp, P. J., and R. D. Herbert (2006), “Solving non-linear models with saddle-path instabilities,” *Computational Economics*, 28, pp. 211-231.

Turnovsky, S. J. (2000), *Methods of Macroeconomic Dynamics*, Second Edition, MIT Press, Cambridge Massachusetts and London England..