Equation-free analysis of agent-based models: Equation-free methods and implementation

Spencer A. Thomas, David J.B. Lloyd, and Anne C. Skeldon

Department of Mathematics, Evolution and Resilience of Industrial Ecosystems (ERIE), University of Surrey, Guildford, GU2 7XH, UK

May 29, 2015

1 Numerical continuation

In a numerical continuation approach, the focus is on fixed points (or periodic cycles). Rather than time evolving $x_{n+1} = f(x_n, \lambda)$ to find fixed points, solutions for $\tilde{x}$ such that $f(\tilde{x}, \lambda) - \tilde{x} = 0$ are sought. Having found a solution for one particular set of parameter values, the solution is 'followed' by stepping in λ and resolving $f(\tilde{x}, \lambda) - \tilde{x} = 0$. The prediction method enables continuation along solution branches (Fig. 1). This is faster than finding fixed points using simulations and has the advantage that both stable and unstable solutions can be found. These techniques are well-established in their value in understanding deterministic systems [3]. The also enable bifurcation and stability analysis of the system uncovering its behaviour under parameter variation. Although well studied for deterministic (no noise) systems, the application of this to stochastic (noisy) systems is extremely limited. Moreover this method requires the system to have an analytical form for the macroscopic state, i.e. an equation describing the systems behaviour.

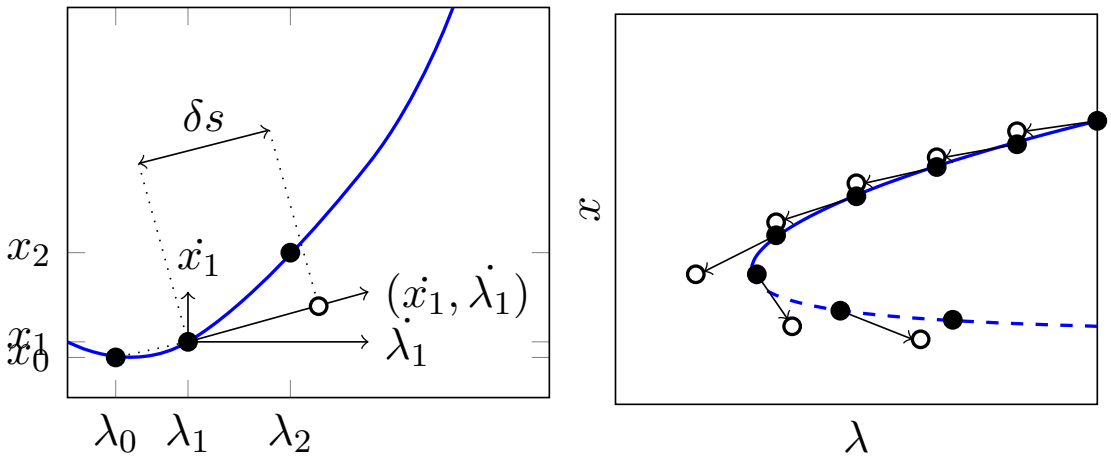

Figure 1: *Pseudo-arclength predictor-corrector method (left) enabling continuation of both stable (solid) and unstable (dashed) solution branches (right). Based on known solutions (black) predictions can be made (white) for solutions under parameter perturbation.*

2 Equation-free Methods

Equation-free methods [6, 7] enable numerical continuation of a system by replacing the macroscopic dynamical equation, $x_{n+1} = f(x_n, \lambda)$, with an ensemble of simulations at the microscopic level. Using an appropriate microscopic state, initialised from the macroscopic state (lifting), the

macroscopic behaviour of the model can be estimated (restricting) from an ensemble of microscopic simulations (N realisations) over a small time window τ (time-horizon). N independent microscopic states (x) are initialised from the macroscopic state (X) at time $t = 0$. After a simulating each microscopic state (i.e. time evolution) from $t = 0$ to $t = \tau$, the $X(t = \tau)$ state is estimated from the distribution of microscopic final state, i.e. $g(x)$ at $x(t = \tau)$. That is, for a stochastic system, after the simulation (time evolution) the ensemble of renationalisations will be in different states, yielding a distribution, $g(x)$. We can estimate the macroscopic state, $X(t = \tau)$, from this distribution by taking some properties such as mean, variance, etc. An overview of this generic process is given in Fig. 2. This enables the macroscopic analysis of a system, i.e. using numerical continuation, without the need for an explicit form. That is it does not require an equation of the system, such as a partial differential equation, to describe its behaviour, thus the term equation-free. Additionally this provides a platform for continuation of stochastic systems.

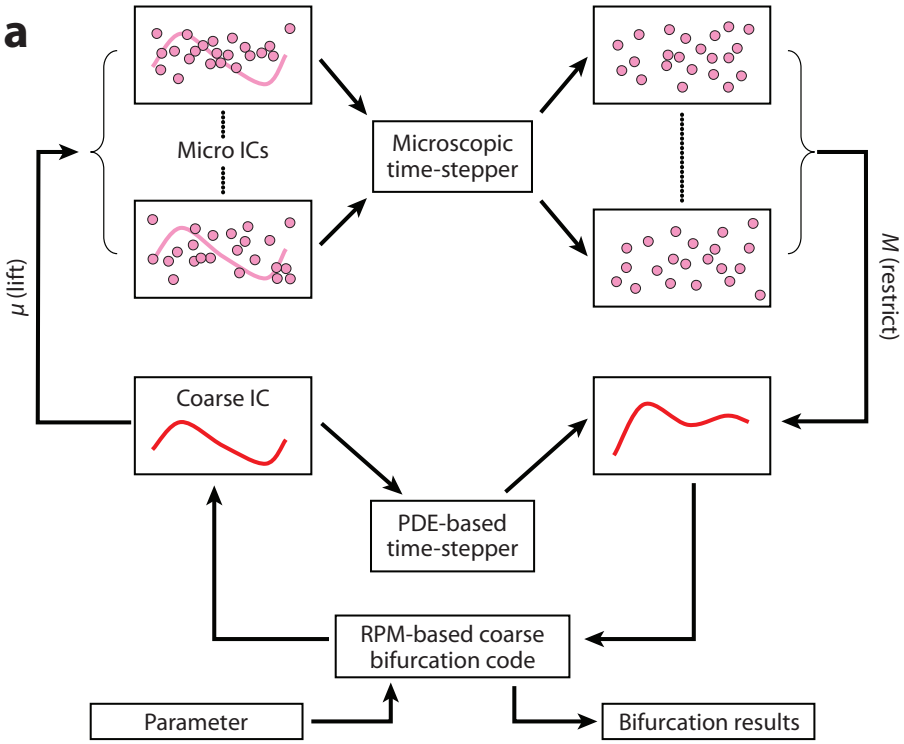

Figure 2: *Equation-free method for macroscopic analysis figure from [8]. IC-initial conditions, RPM-recursive predictor method and PDE-partial differential equation.*

3 Implementation for agent-based models

In this instance we are interested in the analysis of agent-based models using the equation-free method described in Section 2. Specifically, we use this process to perform numerical continuation of agent-based models as described in Section1, which may be stochastic and rarely have an analytical form. We require our implemented algorithm to be generic, i.e. use-able for any agent-based model, and robust, i.e. is not sensitive to noise, parameters or the models themselves.

Here we focus on NetLogo based agent-based models as it is commonly used in the research community, is compatible with all major operating system, and has the ability to interface with other languages enabling the construction of an equation-free 'wrapper' to the models. In our code the Lifting step corresponds to the `setup` procedure, the ensemble of realisations correspond to

N separate simulations run independently, the simulation (or time evolution) is the `go` procedure. After the simulation of the realisations we take the mean of $g(x)$ as an estimator of $X(t = \tau)$. Our implementation of the equation-free process for agent-based models is illustrated in Fig. 3.

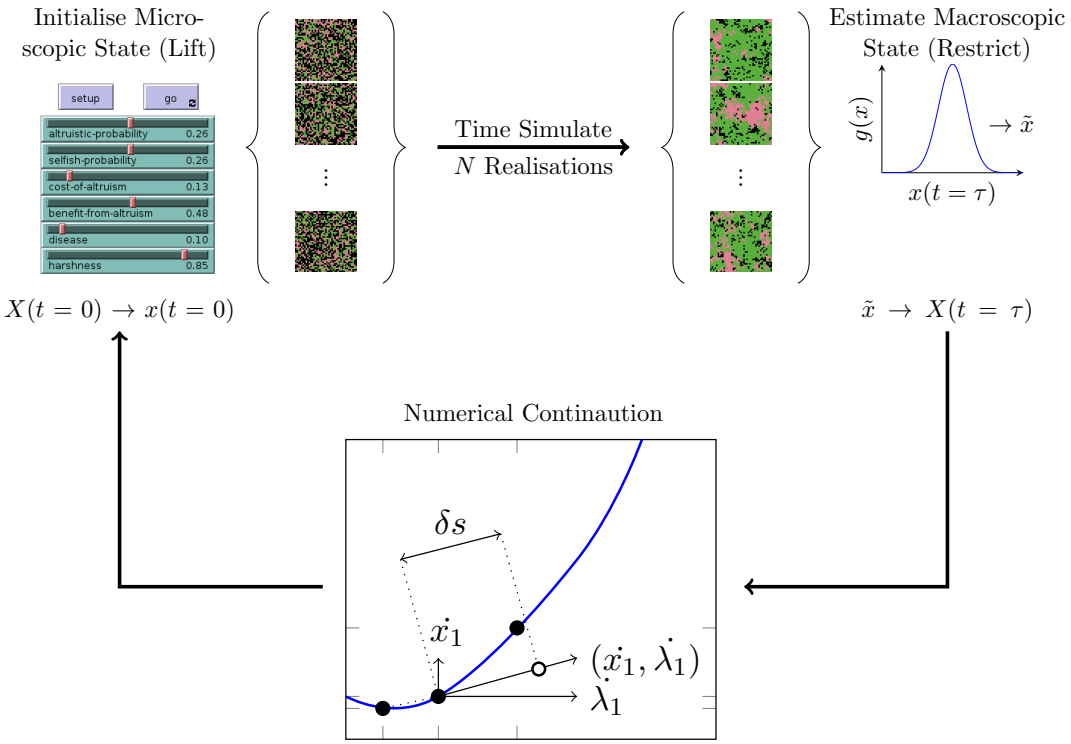

Figure 3: *Equation-free method for numerical continuation of agent-based models.*

4 Structure of the algorithm

Our implementation is in Java due to its compatibility with all major operating systems and ability to interface with NetLogo directly. As NetLogo is also universal across the major operating systems (Mac, Windows and Linux) our implementation can be used by the widest possible audience. Moreover, Java also enables the design of a user interface that enables user to run our algorithm without any knowledge of Java or programming (outside of NetLogo), or how to execute programs. Details of the user interface are given in Section 6, users comfortable with using terminals can see Section 8 to run the algorithm. The structure if the algorithm is represented in Fig. 4.

4.1 Overall design

We employ and object-orientated design providing abstraction to our algorithm. That is, our implementation is highly modular, thus individual elements can be edited or substituted without changing the rest of the code. This enables simple changes to be made, such as the predictor method, convergence method, lift and restrict operators and system exploration phase. With this design advanced users can substitute or edit any of these elements as desired without effecting the overall program.

4.2 System exploration stage

To perform equation-free analysis of any system one must tune some problem specific variables, these are; the number of realisations (N), the simulation time (τ) and the step size in the parameter space (δs). As no such process exists in the literature, we additionally developed a systematic method for determining these values essential for equation-free analysis. This process allows users

to configure our equation-free algorithm to their model without requiring any knowledge of the underlying mathematics. This stage should be run priori to the equation-free analysis to provide the parameters required, then the algorithm can perform the analysis using these values. Details of this procedure are beyond the scope of a user manual, though are detailed in our paper **cite paper** if readers are interested. Note that this procedure requires the user to define model parameters where are some variable dynamics in the system. If a parameter set is given where there is no change in the state of the model this procedure will fail. If the model is deterministic (has no noise) then this process will also fail. In the instance of a deterministic model $N = 1$, δs can be selected based on the magnitude of the parameter under investigation and τ can be easily obtained through a single simulation of the agent-based model in NetLogo (i.e. how many ticks). Our algorithm works for deterministic models, though currently our systematic parameter determination is based on the stochasticity of the model and will not work for systems with no noise.

For first use of the system exploration algorithm the `testing` variable should be set to true and the `tauSpreadThreshold` set to 10,000. The program is then run and will give values for τ and γ, after some time the curve of γ as a function of τ will reach a maximum value, or possibly increase for ever. This gives a bounds for `tauSpreadThreshold` as $<$ the maximum γ, or in the second case one can select a `tauSpreadThreshold` based on feasible run times. Once `tauSpreadThreshold` is obtained the test phase is repeated and will provide values for τ, N and δs. These can subsequently be used to perform the equation-free analysis by setting `testing` variable to false.

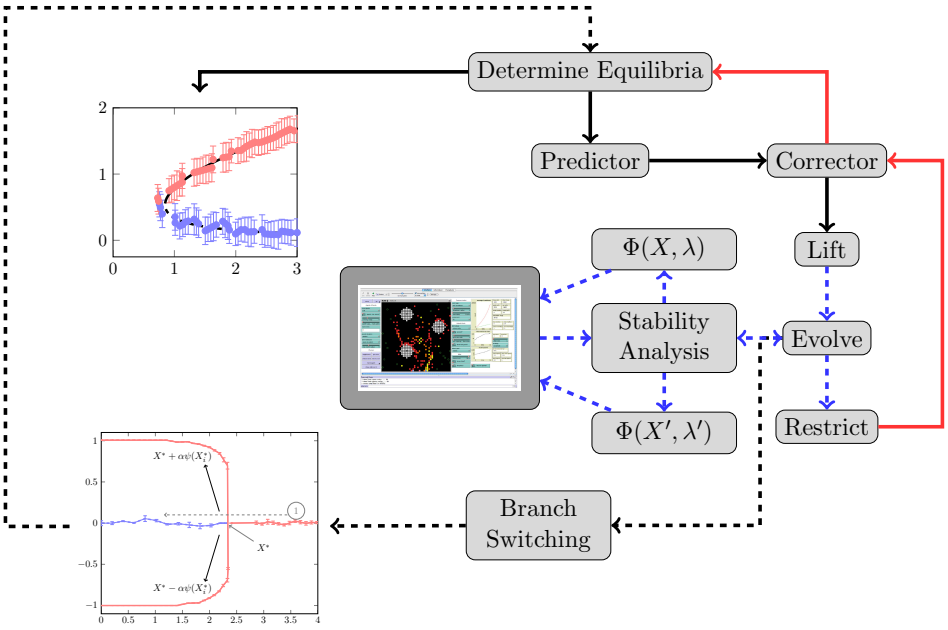

Figure 4: *Structure of our equation-free wrapper algorithm. Grey boxes indicate users supplied information (model and parameter settings) and blue boxes are Java classes that do not need to be accessed by general users. Read lines indicate iterative processes and dashed lines correspond to the system exploration stage only (see text). The wrapper produces the parameter dependency curves with bifurcation and stability analysis (red-stable, blue-unstable) of the model.*

5 Application to your model

Included in this repository is a number of examples of application to specific agent-based models. This is updated when a new model is analysed and is a good resource for understanding what is required for the application of our tool to your system.

The basic requirements are that the user provide a working NetLogo model that they wish to analyse, and the parameter names and values required to run the model. The user must also ensure

that the `setup` procedure is sufficient to initialise the model at an appropriate state. This is model specific and can not be generalised. The 'default' `setup` procedure may not have to be edited or an additional one written in order to perform the equation-free analysis. A simple example of this is given for the Ising model in the repository. Essentially the `setup` procedure must be able to initialise the agent-based model to any of its possible states. That is, if I run a simulation for some period of time, can I recreate this state of the model with some parameter values or settings? This is required for the numerical continuation path following. The user is also required to define a useful measure of the system, this is also model specific and can be a variable(s) or a combination of them. This must also correspond in with the Lift operation in order to evaluate fixed points in the system. Again it is worth looking at the repository for examples for this.

Beyond the model parameters there are some variables that are required for the equation-free process. These can be determined through the system exploration stage in Section 4.2. Some other variables can be tuned, though this is only recommended for advanced users, and are set to default values for general use.

6 User interface

The user interface can be opened by double clicking on the `EFNC_GUI.jar` file, or right clicking and selecting `open with` then `Java Runtime`. Alternatively this can be run from a terminal by navigating to the equation-free folder and typing `java EFNC_GUI`. The interface layout is illustrated in Fig. 5.

Pressing the *Model Parameters* button will open up a window asking how many parameters are in the model. This includes all parameters that have variable inputs, i.e. slides, etc. Next a window will open with two columns of boxes and rows equal to the number of parameters you just entered. On the left hand side type the name of the variable, e.g. `size-of-population`, then in the right hand side type the value associated with this parameter and can be integers, floats, boolean values or strings. Note if they are strings then the value will need to be put into quotation marks, i.e. parameter name = `name-of-agent`, parameter value = "`Alice`". Once complete another message will follow asking if the system is initialised. This is in reference to the `setup` procedure and the fixed point calculation in the continuation. In most cases the response will be no but this is included for generality.

The *Continuation Parameter button*

The *NetLogo Path* button opens a window asking you to input the location (or path) to your NetLogo code. This can be absolute path, or relative to the *EFNC* parent directory. There is a folder within the *EFNC* directory called *netlogo/* where some of the repository codes are. You can add additional NetLogo models to this folder.

The *Define Measurements* button opens a window asking the user to input the number of measurements of the system. Note as stated in Section 5, this is problem specific and should also link to the lift operation in some way. Once a number has been defined a new window opens with this many boxes where the measurements should be entered. See the repository for examples of measurements. Some brief examples are, for the *Forest Fire* model the measure is "(burned - trees / initial - trees) * 100.0 " , and for the *Altruism* model the measure is "count patches with [pcolor = pink] / count patches" and " count patches with [pcolor = green] / count patches ".

The *Testing* button configures the code to run in the system exploration stage and opens a message saying *if gamma unknown run with a value of 10,000*, followed by a box to input a value of gamma. This button does not run the code but does configure it to the test mode. To return the code to the analysis stage press the *Testing off* button.

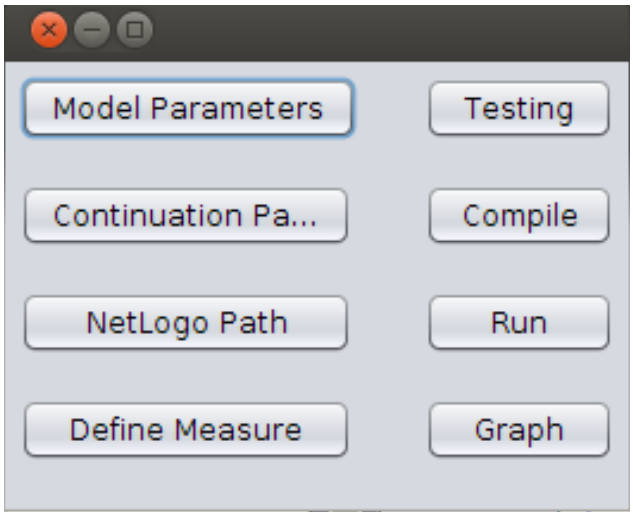

Figure 5: *Graphic user interface for our equation-free analysis tool.*

7 Running without the interface

Open a terminal and navigate to the EFNC directory. Once here place your NetLogo code in the `netlogo/` directory. Open the file `codes/ContinuationParameters.java`, essentially this is a text file of parameter values. This file is separated into 1) continuation parameters, 2) model parameters, and 3) system exploration parameters each with a comment describing with they are. Firstly in the model parameters section you should include the parameter names and values and model location as required. Run the model in the system exploration stage as in Section 4.2 and input these values into the file. To compile the code for both the system exploration and analysis stage type in to the terminal `./compile` then when a message appears saying compilation complete you can run the program (again in both stages) by typing `./run`. That's it!

8 Output files and graphs

The program will then produce all the output files into the folder `outputfiles/` including the parameter dependencies (with variance, errors, etc) and distribution of realisations at each point. Additionally it will produce a *GNUplot* script to plot this data. If you have *GNUplot* installed then simply type `gnuplot` or `./gnuplot` followed by `load "continuation.plot"` then `exit` to exit *gnuplot*. If you are using the the user interface simply press the `Graph` button to generate the plot files. The output files are all text files so can also be imported in to Excel or other graphing packages.

9 Further Reading

The following table contains a reference list for further reading on the topic contained in this method and example.

Topic	Reference
Introduction to bifurcation analysis	[1]
Introduction to continuation	[2–5]
Introduction to equation-free methods	[6–8]

Acknowledgments

The support of the UK Engineering and Physical Sciences Research Council for programme grant EP/H021779/1 (Evolution and Resilience of Industrial Ecosystems (ERIE)) is gratefully acknowl-

edged.

References

[1] C. Meunier and A. D. Verga, *Noise and Bifurcations* J. Stat. Phys., 1988, 50(1-2), pp 345-375

[2] E. Doedel, H. B. Keller and J. P. Kernevez, *Numerical Analysis And Control of Bifurcation Problems (I) Bifurcation in Finite Dimensions* Int. J. Bifurcation Chaos, 1991, 493(3), pp 493-520

[3] E. L. Allgower and K. Georg, *Numerical Continuation Methods, An Introduction* Springer-Verlag Berlin Heidelberg 1990

[4] W. C. Rheinboldt, *Numerical continuation methods: a perspective* Journal of Computational and Applied Mathematics, 200, 124, pp 229-244

[5] B. Krauskopf, H. M. Osinga and J. Galàn-Vioque (Eds.), *Numerical Continuation Methods for Dynamical Systems* Springer 2007

[6] C. Theodoropoulos, Y. H. Qian and I. G. Kevrekidis IG *Coarse stability and bifurcation analysis using time-steppers: a reaction-diffusion example* Proc. Natl. Acad. Sci. 2000, 97, pp 9840-9845

[7] I. G. Kevrekidis et al. *Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level tasks* Comm. Math. Sci. 2003, 1, pp 715-762

[8] I. G. Kevrekidis and G. Samaey, *Equation-Free Multiscale Computation: Algorithms and Applications*, Annual Review of Physical Chemistry, 2009, 60(1), pp 321-344