Economic Methodology and Computability: Implications for The Evaluation of Econometric Forecasts

Scott Moss and Bruce Edmonds[1]

1 Introduction

The purpose of this paper is to bring standard propositions of economics together with computability theory in order to assess the use of econometric forecasting in policy analysis. Our criterion for the applicability of econometric forecasting will be in terms of the relative accuracies of alternative forecasts as a result of improved model specification, estimation and/or data collection and processing techniques. We will justify this criterion by demonstrating that standard economic methodology going back to Friedman (1953) implies that elementary welfare considerations impose a duty on forecasters to identify the conditions in which their forecasts are the best available.

This approach gives our arguments particular importance when forecasts are used for policy analysis by rational decision-makers and their advisors. It also removes from us any requirement to specify correctly what is in the heads of econometricians. Nonetheless, we note that econometricians often write and speak as if they view econometric forecasting as a progressive science in which the reasons for specific forecasting failures are recognized, understood and corrected. Consequently, revised models will be more accurate in the sense of making correct forecasts should the circumstances of the last forecasting failure recur. The arguments of this paper apply equally to the possibility that econometric forecasting can be a progressive science in this sense.

We formulate the question of whether it is possible for forecasters to execute their duty to choose the best available model or for econometric forecasting to be a progressive science so that it can be answered by an appeal to computability theory. We find that in some conditions individual forecasting procedures can be compared for systematic differences in accuracy. However, there are no general procedures which can be used to make *a priori* comparisons of forecasting model goodness.

---

[1]This paper follows on from previous work by Moss, Michael Artis and Paul Ormerod funded by the ESRC under contract R000232821. The authors wish to express their gratitude to Michael Artis of the University of Manchester and European University Institute for extended discussions about previous draughts of this paper. He does not bear any responsibility and does not necessarily agree with our arguments.

2 The Methodological Issues[2]

Our concern with forecasting in particular together with the importance of forecasting in the formulation of economic and business policies leads us naturally to associate the use of forecasting models and methods with policy analysis. For the present, we simply assume that econometric models should be assessed according to the accuracy of their forecasts. We also recognize that many econometric modellers would not accept that forecasting accuracy is an appropriate test of their models. For this reason, we return to the issue in section 4.

In line with the foregoing remarks, we will take it for granted that a forecasting model should be used for policy prescription only when that application maximizes the policy analyst's subjective expectation of policy benefit. In effect, a standard cost-benefit approach is applied to methodological issues.

Define a policy as a set of individual acts *P*. We suppose the set of acts to be supported by a forecasting model whenever all of the conditions in which that model is the best available model are satisfied. This will mean that the model is specified in some appropriate way, that the estimation techniques yield the best possible estimates of the “true” parameters of the model, that the observed data either conform to the definitions of the variables of the model or the model has been specified to take account of the differences between the true and the observed data.

Suppose that there is some set of conditions in which the model specification and estimation, the data and the computational procedures are all appropriate to the determination of some optimal set of policy decisions. That is, there is some forecasting model which, when applied to the current data set implies some policy.

A particular policy is implied by a forecasting model whenever that model is the best available and, at least, is no worse than any previous or alternative current forecasting model. Let us suppose that there is some set of conditions *C* which gives us confidence that the conditions of application of a forecasting model are satisfied (and, therefore, we have confidence in the policies supported by that model). Let there be *n* such conditions. Each condition is tagged with the variable $C_i \in \{true, false\}$. Thus, *C* implies that $C_i = true\ (i=1...n)$.

---

[2]The arguments of this section closely follows the argument in Moss (1993).

Let B be the image of the mapping $[P|C \to \Re]$, the value of the benefits expected from the set of policy actions P.

The "observation tag" for the ith condition is $\phi_i \in \{true, false\}$ which takes the value *true* if it is intended to observe the ith condition and *false* otherwise. The intention of the forecaster to observe conditions of application is captured by the set $\Phi = \{\phi_i | (i = 1 \ldots n)\} \cap true$. In addition, we denote by $C(\Phi)$ the cost of observing all of the conditions $\phi_i \in \Phi$.

To complete our notation we require some means of representing degrees of prior belief in the satisfaction of the conditions of application which it is intended to observe. The standard representation is in terms of subjective probabilities. For this reason, we adopt the mapping $\Psi(\Phi) \to [0,1]$ which we interpret as the subjective probability that all conditions of application which it is intended to observe will be satisfied.

By hypothesis, if all of the conditions of application of the theory are true, then the acts in P will imply some expected benefit, $\mathrm{E}(B|C)$. Otherwise some different benefit, $\mathrm{E}(B|\neg C)$, will result. Thus, the prior expected benefit of P when the set of conditions to be observed is empty is

(1) $$\mathrm{E}(B|\Phi = \varnothing) = \mathrm{E}(B|C) \cdot \mathrm{E}(C) + \mathrm{E}(B|\neg C) \cdot (1 - \mathrm{E}(\neg C))$$

More generally, the expected benefit given any arbitrary set of conditions to be observed will be

(2) $$\begin{aligned} \mathrm{E}(B|\Phi) = \Psi(\Phi) \cdot \{ & \mathrm{E}(C|\Psi(\Phi)) \cdot \mathrm{E}(B|C) \\ & + [1 - \mathrm{E}(C|\Psi(\Phi))] \cdot \mathrm{E}(B|\neg C) - \mathrm{c}(\Phi) \} \\ & - [(1 - \Psi(\Phi)) \cdot \mathrm{c}(\Phi)] \end{aligned}$$

where $\mathrm{c}(\Phi)$ is the cost of observing the conditions of application in Φ. Expanding and simplifying equation (2), we get

(3) $$\begin{aligned} \mathrm{E}(B|\Phi) = {} & \mathrm{E}(C|\Psi(\Phi)) \cdot \Psi(\Phi) \cdot \mathrm{E}(B|C) \\ & + [1 - \mathrm{E}(C|\Psi(\Phi))] \cdot \mathrm{E}(B|\neg C) \\ & - \mathrm{c}(\Phi) \end{aligned}$$

Since $\mathrm{E}(C|\Psi(\Phi)) \cdot \Psi(\Phi) = \mathrm{E}(\Psi(\Phi)|C) \cdot \mathrm{E}(C)$ and, from the definition of C, $\mathrm{E}(\Psi(\Phi)|C) = 1$, equation (3) can be written

(4) $$\begin{aligned} \mathrm{E}(B|\Phi) = {} & \mathrm{E}(B|C) \cdot \mathrm{E}(C) + (1 - \mathrm{E}(C)) \cdot \mathrm{E}(B|\neg C) \\ & - (1 - \Psi(\Phi)) \cdot \mathrm{E}(B|\neg C) - \mathrm{c}(\Phi) \end{aligned}$$

Substituting into equation (4) from equation (1), we have

(5) $\mathrm{E}(B|\Phi) = \mathrm{E}(B|\Phi = 0) - (1 - \Psi(\Phi)) \cdot \mathrm{E}(B|\neg C) + \mathrm{c}(\Phi)$

For the regime in which no conditions of application are observed to entail rationality, it must be the case that the negative term on the right side of equation (5) is non-negative. That is, to adopt a policy based on some forecasting model without a prior assessment of the extent to which the conditions of application of that model are satisfied is efficient if and only if

(6) $(1 - \Psi(\Phi)) \cdot \mathrm{E}(B|\neg C) + \mathrm{c}(\Phi) > 0$

for every possible set Φ - i.e. for every possible combination of conditions of application of the model.

For example, presuming that there is some cost to observing conditions of application, equation (6) will always be satisfied if $(1\text{-}\Psi(\Phi))=0$. This would be the case if the policy analyst were convinced that all of the conditions of application of the forecasting model were always satisfied. As a result, any subset of those conditions will also always be satisfied.

Another forecaster might believe that the policies implied by the model and forecast yield substantial and positive benefits even when the conditions of application are violated. Formally, $\mathrm{E}(B|\neg C)$ is so high that allowing for the probability of $\neg C$, the benefit when conditions of application are known not to be fulfilled swamp the cost of observation. Presumably, some theory supports that belief.

The more general possibility is that, even if conditions of application might be violated and, if they are, negative benefits might result from the implied policies, the cost of observing the conditions of application could in principle be so great that they exceed the expected opportunity costs associated with the inapplicability of the forecast. This possibility seems reasonable when observation requires detailed and expensive investigations which themselves yield no collateral benefits. But is it never sufficiently cheap and error never sufficiently costly and is the world never sufficiently risky as to make it worthwhile *a priori* to investigate the validity of anyconditions of application of any forecasting model? If it is possible that equation (6) will not be satisfied in non-trivial cases, then the-conditions of application of forecasting models become a serious and important issue..[3]

---

[3]Compare this position with that in Friedman's (1953) classic essay on methodology where conditions of application are never the descriptive accuracy of the theory though, implicitly, different models would apply to different problems. The particular model to be used for a particular problem "will doubtless be recognized before the event." (p. 36)

3 Arbitrary Models and Computability Theory

The argument of the previous section suggests that elementary welfare considerations require forecasters to define and assess the conditions of alternative forecasting models before choosing any of them for policy prescription. In this section we show that such comparisons are not in general computable *a priori*. Obviously, we can make *a postiori* comparisons of finite forecasts and reject models that fail the test. This is not sufficient either to justify the use of particular forecasting models in policy analysis or to sustain the general proposition that econometric forecasting is or can be a progressive science. We can either try to identify the conditions in which particular models yield relatively accurate forecasts or we can adopt an approach to policy formation which does not rely on the accuracy of any particular forecasting model or set of models.

We begin with some definitions.

Computable model: a model for which all of its variable values can be computed given unlimited computational resources.

Switch: an endogenous means of selecting variable values or equations which is not itself an equation. Rules, dummy variables and programming code are examples of switches.

Standard model: a computable, linear econometric model over discrete time with switches and at least one time lag.

Correct model: a standard model that generates empirically correct values of observable variables to within some predetermined accuracy. Since any finite sequence of numbers can be generated by some algorithmic process, there must always be at least one correct model.

The basic result of this section is the proof that there is no general algorithmic means of knowing if an arbitrary *standard* trial model will ultimately converge to a *correct* model

We first show that the class of economic models is identical to that of the well known class of computable functions. As a corollary of this we deduce that it is undecidable whether such standard models will actually converge. This has further consequences for the possibility that sequences of such models will converge on the correct model.

There are many ways to characterize the class of processes that can be mechanically computed. The first such characterisation was by Turing (1936). This was followed by a host of other such formalisations (*e.g.* Gödel-Kleene (1936), Church (1941), Post (1943), Markov(1951)). All of these turned out to be equivalent. Since then many processes have turned out to be equivalent, some of them quite surprising like solving diophantine equations and tiling the plane. This has lead to the Church's famous thesis that *all* mechanical processes are thus equivalent. It should not, therefore, surprise us that the class of econometric forecasts should also be equivalent.[4]

The field of Computability (also know as Recursive Function) Theory has now become a well-established field of mathematics. Some of its most important results show that many important questions *about* computable functions are themselves not computable. In particular, there is no computable means of settling the question of whether a computable process will ever halt and come to an answer.

While there has been substantial and important work on the implications of computability for economic games[5] and choice functions[6], we are not aware of any applications to econometric forecasting.

We will use here a formalisation of computable functions called an Unlimited Register Machine (URM) formulated by Sheperdson and Sturgis (1963). This is a much easier formalism to deal with than Turing's original machine. Like other such formalisations it is functionally equivalent to that of Turing's.

A *URM* is a computer with an unlimited number of memory locations, called "registers", available to it. Each of these registers can hold a natural number (0, 1, 2,...). Call these registers $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$,....

Each URM also has a program consisting of a sequence of four kinds of instruction:

Z(***n***)	- Make register number ***n*** zero.
S(***n***)	- Increase register number ***n*** by one.
M(***n***,***m***)	- Copy the contents of register number ***n*** to register number ***m***, erasing its previous contents.
J(***n***,***m***,***q***)	- If the contents of register number ***n*** are the same as register number ***m***, then jump to instruction number ***q***.

The URM executes the program starting at instruction **1** and progressing to the next unless it meets an instruction of type J(***n***,***m***,***q***) where the condition holds, in which case it continues execution at step ***q***.

The starting state of the registers represents the input to this machine. The program terminates when ***q*** becomes zero. In this case the output is the end state of the registers.

---

[4] A good but mathematical introductory text is Cutland (1980) where all of the standard computability results on which we rely are found. An introduction to applications of computability theory to economics is Anderlini (1992).

[5] e.g. Prasad, (1991).

[6] Binmore (1987); Rustem & Velupillai, (1990).

An Alternative Version of the URM (AURM)

We will be using an altered version of this standard type that is equivalent to the standard version in that any program for the standard URM can be simulated on the altered version and vice-versa.

This version is identical to the standard URM, except that it has only two types of instruction available:

S(***n***)	- Increase register number ***n*** by one.
DJZ(***n***,***q***)	- Decrease the contents of register number ***n*** if it is greater than zero. If the result is zero continue execution of the program from step ***q***.

Assumptions

Assume that there is a correct but unknown model of some economic process. We will consider the case where we are trying to construct trial models that will converge to the correct model after an initial "settling-down" period. We will restrict ourselves to *standard* models of the above type.

It should be noted that most questions about processes with known bounds upon their computation time *are* computable. Trivially you can run the process and see. Similarly any finite sequence of numbers can be generated by some algorithmic process by copying the output from an internal table as required. Thus there must always be at least one correct model for such finite sequences. This does not mean that this model is known (or even *knowable*) by us. For the sake of the arguments below we are assuming that there is a correct model which is *universally* valid[7].

Lemma 1: AURM machines can simulate any URM Machine

For any URM there is a AURM machine that is equivalent to it, such that it always gives the same output for every input and it only terminates if the URM terminates (see Appendix A for a detailed proof).

Lemma 2: URM machines can simulate any AURM Machine

Likewise for any AURM there is a URM machine that is similarly equivalent to it, such that it always gives the same output for every input and it only terminates if the URM terminates (see Appendix A for a detailed proof).

[7] Note that if you have a series of models and known criteria for swapping between them (e.g. at certain times) then this can be combined to form a single universal model.

Lemma 3: AURM machines can be simulated by a "Standard" *Model*

We now show that an arbitrary AURM machine can be simulated by an linear switched model on a number with one time lag.

Proof:

Let the maximum register referred to by the AURM be **max**.

Consider a AURM machine with registers: $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3,\ldots,\mathbf{r}_{max}$ with initial values: $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3,\ldots, \mathbf{a}_{max}$. and a program with instructions: $\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3,\ldots,\mathbf{i}_{max}$.

Each register $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3,\ldots,\mathbf{r}_{max}$ will be simulated by the variables: $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3,\ldots, \mathbf{X}_{max}$.

The starting value of the model ($\mathbf{X}_1(0), \mathbf{X}_2(0),\ldots \mathbf{X}_{max}(0)$) will be the input values of the AURM registers $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3,\ldots,\mathbf{r}_{max}$ before the program starts. **Pc** will start at **1** as does **Rt.**

Let the model have three sections (as above).

Let the first and third sections be initially empty and the second section as follows:

Inc := 0
If Pc(T-1)>0 Then Inc := 1
Pc(T) := Pc(T-1) + Inc
Rt := Rt + Inc

For each instruction in the program add equations to the model as follows (considering the $\mathbf{j}^{th}$ instruction):

If the $\mathbf{j}^{th}$ instruction is a S($\boldsymbol{n}$) instruction add a block of equations of the form:

Ps := 0
If Pc(T) = j Then Ps = 1
$X_n(T) := X_n(T-1) + Ps$

to the first section of the model.

If the $\mathbf{j}^{th}$ instruction is a DJZ(**n**,**q**) instruction add a block of equations of the form:

Ps := 0
If Pc = j And $X_n(T-1)>0$ Then Ps = 1
$X_n(T) := X_n(T-1) - Ps$

to the first section and a switch of the form:

If Pc(T)=j and $X_n(T)=0$ Then Pc :=q

to the third section.

Now we have a model which simulates the AURM. Each time step simulates one instruction of the AURM being executed. The variable **Pc** keeps track of which instruction is to be executed next.

The model will converge to settled values for all the variables (including **Pc** and **Rt)** if **Pc** is set to zero (i.e. the AURM terminates).

This model simulates the arbitrary AURM (see Appendix A for detailed proof).

Lemma 4: An AURM can simulate any standard *model*

As we have shown above that the AURM is equivalent to the URM as a definition of computability, it can compute any *computable* function. We thus appeal to the Church-Turing Thesis to show this.

Main Theorem: The class of functions that are computable by a standard model are identical to those computable by a Turing Machine.

Proof

As the class of functions computable by a standard model and those computable by an AURM is identical and that class is identical to those computable by an URM, that class must be the commonly recognised class of *computable* functions.This URM is a well-known equivalent of a Turing Machine[8].

Corollary 1: There Is No Algorithmic Means Of Knowing If An Arbitrary Standard Model Converges

If there were an algorithm to predict whether all linear models with switches and at least one time lag would converge to fixed values then that algorithm would predict whether a model of the above type would converge. We would thus have an algorithm to decide if the AURM it simulated terminated; this would give us a general algorithm for determining whenever the corresponding URM halted - this is impossible.

Hence there is no such algorithm.

Corollary 2: Convergence by an arbitrary such linear switched model to a computable "correct" model is undecidable.

The question of whether an arbitrary linear switched model converges to the "correct" model is equivalent to the question of whether the difference between the two models on any important variable is sufficiently small after a certain time.

The difference of such models is just another linear switched model with at least one time lag as the two sets of equations and switches could be collected together (renaming any variables which occur in both) and adding an extra series of equations calculating the differences as new variables. Thus the problem of convergence of the trial model to the correct one is equivalent to the convergence of another standard model to the zero function (we will call this function ψ).

---

8 See Cutland (1980)

We will now use Rice's Theorem. This states that if ϕ_1, ϕ_2, ϕ_3,... is any (effective) enumeration of computable functions and $\Im$ is any non-trivial class of computable functions (i.e. $\Im \neq \emptyset$ and $\exists \phi \notin \Im$) then the problem '$\phi_x \in \Im$' is undecidable.

If we call the correct model ψ and the trial model ϕ_e then the problem can be expressed as $\exists T \forall t>T(|\phi_e(t)-\psi(t)|<\varepsilon)$, i.e. is there a time after which the trial model and the correct model differ by less than ε. Let $\Re=\{\phi \mid \exists T \forall t>T(|\phi(t)-\psi(t)|<\varepsilon)\}$. Then the problem can be expressed as '$\phi_x \in \Re$'. Now, by definition $X \in \Re$ and $(\psi+x) \notin \Re$, thus the problem '$\phi_x \in \Re$' is undecidable by Rice's Theorem.

Thus you can not, in general tell whether such an arbitrary standard model converges to a similar *correct* model.

Corollary 3: There is no general algorithm for determining the existence of a method for improving arbitrary models so that it will converge to the "correct" one.

Call the arbitrary trial model ϕ_e and the correct model ψ. Let $I=\{f \mid \exists N \forall n>N \exists T \forall t>T(|\phi_{f^n(e)}(t)-\psi(t)|<\varepsilon)\}$, i.e. this is the set of computable functions such that after a certain number of them to this arbitrary trial model, it is suitably close to the correct model (in the same sense as above). Now, since $\psi \in \mathbf{C}$ then there is a natural number k such that $\psi=\phi_k$ in an enumeration of $\mathbf{C}$, then the constant function $\mathbf{k}(x)=k$ is in I (so $I \neq \emptyset$) and if t is the index of the function $(\psi+x)$ (i.e. $\phi_t=(\psi+x)$) then the constant function $\mathbf{t}(x)=t$ is not in I (so $\mathbf{t} \notin \mathbf{I}$). Thus by Rice's Theorem the problem '$\phi_x \in I$' is undecidable. In other words there is no effective means of deciding *a priori* whether an algorithmic process will improve a model so that it converges to a correct model.

This does not stop any improvements of a trial model as, at least, some trivial methods of improvement are possible: e.g. taking one point at a time in the series and fixing them to the desired values. It does show that you can not assume that there is a method for uniformly improving forecasts.

Corollary 4: There is no general algorithm for determining which of several alternative standard models will converge most closely to a "correct" model.

If there were a general algorithm for determining which of several model converges most closely to a "correct" model then this algorithm would also determine whether a given algorithm improved a trial model as much as the perfect constant function (**k** in the above). Thus we would have a positive procedure for deciding whether an algorithm improved the trial model perfectly. This we showed was impossible in the last corollary.

Thus we have shown that there is no general algorithmic means for determining whether a standard model converges to a solution or which of the solutions of several models converges most closely to the solution of the correct model.

4 Conditions of Application of Forecasting Models

That there are some conditions in which the applicability of different forecasting models can be compared is without doubt. We have proved in section 3 and Appendix A that, in general, there is no systematic means of knowing whether one forecasting model is more appropriate than another. There must be a separate proof of the preferability of one over another in each instance.

There is, however, substantial evidence that the accuracy of forecasting models can be improved systematically by intervention based on the judgement of the model operators. In these cases, forecasters set residual values of individual equations in order to reflect their judgements about the values which the LHS variables should take. Moreover, Moss, Artis and Ormerod (1994) have exhibited a forecasting model of the UK economy in which some aspects of judgement-based interventions were implemented by expert-system-type rules. The rules described the actual intervention behaviour of the operator of the London Business School Quarterly Model of the United Kingdom Economy.

Rules, we have seen, are switches as defined in section 3. The evidence on the effects of judgement together with the work of Moss, *et. al.* imply that some switches do systematically improve forecasts. In other words, forecasting models which include a particular type of switch are known systematically to generate more accurate forecasts than models which do not include those switches. It therefore seems likely and is certainly possible that there are some (possibly weak) conditions in which a particular kind of switch is associated with closer convergence to the output of a correct model than is achieved without such switches.

We have also seen that, provided we are considering discrete-time forecasts over finite forecasting periods, there are always standard models which will generate correct forecasts. Consequently, the analysis of section 3 must always apply to finite forecasting periods. It is therefore certainly correct to suggest, as does the ESRC, that “there is no single ‘correct’ model.”[9] There are in principle many correct standard models which could cor-

[9] *cf.* the ESRC’s specification of its *Round 4 Macroeconomic Modelling Program* (ESRC, 1994).

rectly forecast the variables of concern in the large macroeconometric forecasting models as well as the variables of concern in smaller more specialized models. The problem identified here is the importance on welfare grounds of knowing the conditions in which any model's forecasts converge more closely to the forecasts of a correct model than do the forecasts of any other model combined with the impossibility of any general means of determining what are those conditions.

The alternative to forecasting is scenario modelling. Certainly the consortium managing the ESRC's Macroeconomic Modelling Programme now emphasize their support for "increasing the use of the models in carrying out policy simulations and in the analysis of policy recommendation." In any event, using macroeconometric forecasting models for simulating policy has a long and honourable history. Nonetheless, simulations are not less subject to the strictures we have identified than are forecasts. The argument in section 2, for example, was developed first in relation to the general use of models to support policy recommendations. The certain existence of standard models which would generate correct simulations of the effects of policy measures implies that our computability theoretic arguments apply to policy simulation without change. Finally, the applicability of our arguments apply equally whether we consider a temporal sequence of models or a cross-section of alternative models. A "horses-for-courses" approach to modelling does not escape the need to identify the conditions of application of the individual models when they are to be used to inform policy.

What might be useful is to conduct a wide range of policy simulations under a variety of simulated conditions in order to gain some feel for the conditions in which one policy or another will be successful. The purpose of such an extended simulation exercise is to identify conditions in which particular classes of forecasting model are more accurate than the alternatives or particular classes of policy strategy yield greater benefits. Perhaps, as with judgement-based interventions, we will find a set of modelling procedures which yields unambiguously more accurate results than the alternatives.

5 Conclusion

The purpose of this paper has been to argue (i) that economic forecasters should in general identify conditions in which one forecasting model or another is applicable to a given policy analysis and (ii) that there are no general algorithmic means of determining which of several models best satisfies its conditions of application. The first point follows from conventional economic welfare theory and the second from computability theory.

We do not see this result as a dilemma. We do infer from it that the use of a single forecaseting model to generate policy prescriptions has no scientific basis in either economics or logic. It may also have substantial costs as, for example, when UK interest-rate policy was predicated in the early 1990s on Treasure Model forecasts of an early upturn in macroeconomic activity which was not, in the event, realized until much later.

We propose more flexible use of models to generate a variety of scenarios which will help policy analysts formulate an appreciation of policy opportunities and pitfalls. Perhaps this approach will lead in time to the identificaion of conditions in which the relative accuracies of well specified classes of forecasting models can be compared *a priori*.

6 References

Binmore, K. (1987), 'Modeling Rational Players, Part I', *Economics and Philosophy* **3**. pp. 179-214.

Cutland, N. J. (1980), *Computability*, Cambridge University Press, Cambridge.

Friedman, Milton (1953), 'Essay on the Methodology of Positive Economics' in *Essays on Positive Economics*, (Chicago: University of Chicago Press), pp. .

Moss, S. (1993), 'The Economics of Positive Methodology' in R Blackwell, J. Chatta and E. Nell (eds.), *Economics as Worldly Philosophy*, (Basingstoke: Macmillan).

Moss, S., M. Artis and P. Ormerod (1994), 'A Smart Automated Forecasting System', *Journal of Forecasting*, **13**, pp. 299-312.

Prasad, K. (1991), 'Computability and Randomness of Nash equilibria in infinite games', *Journal of Mathematical Economics,* pp. 429-442.

Rustem, B., Velupillai, K. (1990), 'Rationality, Computability and Complexity', *Journal of Economic Dynamics and Control,* **14**, pp. 419-432.

Sheperdson,J. C., Sturgis, H. E. (1963): 'Computability of recursive functions', *J. Ass. Comp. Mach.,* **10**, pp. 217-255.

Thomas, J. (1993): 'Non-Computable rational expectations equilibria', *Math. Soc. Sci.,* **25**, pp. 133-142.

Turing, A. M. (1936): 'On Computable Numbers, with an application to the Entscheidungsproblem'. *Proc.London Math.Soc.*, **42**, pp. 230-265.

Appendix A

For any URM there is a AURM machine that is equivalent to it, such that it always gives *the same output for every input and it only terminates if the URM terminates.*

We construct an AURM program which simulates an arbitrary URM program. We need to also refer to two registers not used by the URM program, we call these d, e and f., f always has the value 0 in it. This is possible as the URM will use only a limited number of registers.

We show that each instruction in the URM program there is an equivalent set of instructions in an AURM that has the same effect. Of course, some renumbering of the jump addresses is needed to accommodate the substituted block of AURM instructions.

Consider an arbitrary instruction in this program at position ***s***. This instruction can one of the forms S(n), Z(n), M(n,m) or J(n,m,q).

Case S(n)

Trivial, this is simulated by an identical instruction in the AURM program. No renumbering is necessary.

Case Z(n)

Replaced by the code

```
s      DJZ(n,s+2); decrement n and goto end if reached zero
       DJZ(f,s)     ; goto s
s+2
```

All jump addresses subsequent to the Z(n) instruction now need incrementing by 2.

Case M(n,m)

```
s      DJZ(e,s+2)   ;
       DJZ(f,s)     ; make e zero
s+2    DJZ(m,s+4)   ;
       DJZ(f,s)     ; make m zero
s+4    S(n)         ;
       DJZ(n,s+9)   ; if n is zero goto s+9
       S(m)         ; increment m
       S(e)         ; increment e
       DJZ(f,s+4)   ; goto s+4
s+9    S(e)         ;
       DJZ(e,s+14)  ; if e is zero goto s+14
       DJZ(e,s+12)  ; decrement e
s+12   S(n)         ;
       DJZ(f,s+9)   ; goto s+9
s+14
```

All jump addresses subsequent to the M(n,m) instruction now need incrementing by 14.

Case J(n,m,q)

```
s       M(n,d)          ; copy n to d
s+14    M(m,e)          ; copy m to e
s+28    S(d)            ;
        DJZ(d,s+35)     ; if d is zero goto s+35
        S(e)            ;
        DJZ(e,s+37)     ; if e is zero goto s+37
        DJZ(d,s+33)     ; decrement d
s+33    DJZ(e,s+34)     ; decrement e
s+34    DJZ(f,s+28)     ; goto s+28
s+35    S(e)            ;
        DJZ(e,q')       ; if e is zero goto q
s+37
```

All jump addresses subsequent to the J(n,m,q) instruction now need incrementing by 37. Here where it says ***M(n,d)*** or ***M(m,e)*** then all of the instructions listed under the previous case should be inserted with the appropriate numbers substituted. q' is equal to the q if q is before s and is equal to q+37 if after it.

For any AURM there is a URM machine that is similarly equivalent to it, such that it always gives the same output for every input and it only terminates if the URM terminates.

Similar to the above proof but easier. For each instruction at position q there are two cases.

Case S(n)

Is trivial.

Case DJZ(n,q)

is replaced by the URM code:

```
s       J(n,f,s+9)      ; if n is already zero goto s+9
        Z(d)            ; set d as 0
        Z(e)            ;
        S(e)            ; set e as 1
s+4     J(n,e,s+8)      ; if n = e goto s+8
        S(d)            ; d := d + 1
        S(e)            ; e := e + 1
        J(n,n,s+4)      ; goto s+4
s+8     M(d,n)          ; copy d to n
s+9     J(n,f,q)        ; if n is zero goto q
s+10
```

All jump addresses subsequent to the DJZ(n,q) instruction now need incrementing by 10.

There is a standard model that simulates an arbitrary AURM.

We will show that there is a model of the form shown above, such that for each AURM instruction and each possible state of the registers in the AURM that one time iteration of the model has the identical effect as that AURM instruction.

Consider an arbitrary AURM instruction at position **q** in the program. The variable **Pc** keeps track of the number of the instruction being executed. Thus **Pc(T)** will have value **q** when instruction q is to be simulated. There are two cases:

Case S(n)

This instruction adds one to the register $\mathbf{r}_n$, and then execution passes on to the next instruction - number **q+1**.

In this case a block of the form:

Ps := 0
If Pc(T) = q Then Ps = 1
$\mathbf{X_n(T) := X_n(T\text{-}1) + Ps}$

exists in the model.

As **Pc(T)** has value **q** then **Ps** will be set to 1 and this will have the effect of incrementing variable $\mathbf{X}_n$.

The block of form:

Inc := 0
If Pc(T)>0 Then Inc := 1
Pc(T+1) := Pc(T) + Inc

will cause the value of **Pc** to be incremented by one as q>0, as the AURM program has not terminated (otherwise instruction S(n) would not be being executed).

None of the other blocks in the first section of form:

Ps := 0
If Pc = <No> And $\mathbf{X_n(T\text{-}1)>0}$ **Then Ps = 1**
$\mathbf{X_n(T) := X_n(T\text{-}1) - Ps}$

will have effect as **<No>** will not be **q** in these cases. Similarly for the switches in the third section of form:

If Pc(T)=<No> Then Pc:=<No>

Thus the sole effect of the model in this iteration is to increment the $\mathbf{X}_n$ and **Pc** variables.

Case DJZ(n,p)

This instruction subtracts one from the register $\mathbf{r}_n$ if this is non-zero, then if the value of rn is now zero execution continues at instruction number **p,** otherwise execution passes on to the next instruction - number **q+1**.

In this case a block of the form:

Ps := 0
If Pc(T) = q Then Ps = 1
$\mathbf{X_n(T) := X_n(T\text{-}1) - Ps}$

exists in the model as well as a switch in the third section of the form:

If Pc(T)=0 and X_n(T)=0 Then Pc:=p

As **Pc(T)** has value **q** then **Ps** will be set to 1 and this will have the effect of decrementing the variable $\mathbf{X_n}$.

The block of form:

Inc := 0
If Pc(T)>0 Then Inc := 1
Pc(T+1) := Pc(T) + Inc

will cause the value of **Pc** to be incremented by one as q>0, as the AURM program has not terminated (otherwise the instruction would not be being executed) but now the value of **Pc** will be reset to **p** if, in addition $\mathbf{X}_n$ is now 0.

None of the other blocks in the first section of form:

Ps := 0
If Pc = <No> And X_n(T-1)>0 Then Ps = 1
X_n(T) := X_n(T-1) - Ps

will have effect as **<No>** will not be **q** in these cases. Similarly for the other switches in the third section of form:

If Pc(T)=<No> Then Pc:=<No>.

Thus the sole effect of the model in this iteration is to decrement the $\mathbf{X}_n$ if it is greater than zero and set the **Pc** variable to **p** if the result is zero and **Q+1** otherwise.

Halting

The AURM halts only if **q** is ever set to zero. The model only ever settles down into a steady state if **Pc** is set to zero, for in this case none of the first or third sections will have any effect and the second section is of form:

Inc := 0
If Pc(T)>0 Then Inc := 1
Pc(T+1) := Pc(T) + Inc
Rt := Rt + Inc

Which will only have effect if **Pc**>0.