Solving Dynamic Economic Models By Decomposition

Jeffrey Sun

July 5, 2024

Click here for current version.

Abstract

I describe a specific way in which many dynamic economic models can be decomposed into a "within-period model" and a "between-period model." I then describe a sense in which these two parts can be solved separately. That is, a solver for the overall model can be constructed by combining a "within-period model solver" and a "between-period model solver." This document does not attempt to introduce new solution algorithms *per se*. Indeed, it tries to nest many existing algorithms. This document is an exercise in splitting the complex "model solver" into smaller, more manageable parts—to facilitate model solving, transparency, code reuse, standardization, and the application to economic models of tools from computer science and machine learning.

1 What is this document?

This whole document is an attempt to formalize a certain numerical solution strategy for dynamic heterogeneous-agent models which I have found to be quite robust and useful. It involves, roughly:

- Splitting the model into a "within-period model" and a "between-period model," in a way which I will make precise.
- Implementing the within-period model and the between-period model.
- Selecting or implementing an algorithm to "solve" the model.

In addition to being a formal definition of this "process" or "strategy" or "class of model solvers," I am endeavoring to make this definition *constructive*, in the sense that you should be able to actually numerically solve your desired model by following the construction process. I am also building up a library of components which you can hopefully use to build up this solver quickly and robustly.

In the next few sections, I will describe through examples each of the key elements of a "solved model":

1. The Within-Period Model
2. The Between-Period Model
3. The Overall Objective Function
4. The Within-Period Model Solver
5. The Between-Period Model Solver

(I also discuss the notion of the "True Model", which the computable model is approximating, in Section 7.)

The reason why I break the model solver up in this way is because each component can be solved *separately*. That is, you can use this strategy to break the problem up into simple, understandable pieces, implement each piece (or take it off the shelf), then combine them to get a full model solver.

1.1 Terminology

Throughout, I use "solution" to refer to the specific numerical solution to a model, and "solver" to refer to the algorithm which finds that solution.

1.2 Caveat

Some ideas expressed here are novel and unusual. Some are old and some are basically trivial. Unfortunately, I don't always know which is which. I will continue to update this document to add appropriate attributions and as it becomes clearer to me how the ideas here fit in to the broader literature.

2 The Within-Period Model

A "within-period model" is a sort of one-period model, out of which complex dynamic models can be built. I will describe how in later sections.

For now, consider a one-period model in which some distribution of agents enter the model at the beginning of the period, then receive payoffs at the end of the period. Formally, let X^{start} denote the space of idiosyncratic states at the beginning of the period and X^{end} denote the space of idiosyncratic states at the end of the period. For simplicity, suppose that X^{start} is discrete. Then let

$$\Lambda^{\text{start}} : X^{\text{start}} \longrightarrow \mathbb{R}$$

denote the start-of-period population distribution, and

$$V^{\text{end}} : X^{\text{end}} \longrightarrow \mathbb{R}$$

denote the end-of-period payoffs. Finally, let $S^{\text{start}} \in \mathbb{R}^{|S|}$ be a vector of exogenous aggregate state variables. Then a "within-period model solver" is a function, for now called IPP for "intra-period problem,"

$$\text{IPP} : \left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}\right) \mapsto \left(V^{\text{start}}, \Lambda^{\text{end}}, S^{\text{end}}\right).$$

This IPP function might also be called an *operator*. It is a function which takes functions as inputs and gives functions as outputs. It is the operator taking the end-of-period value function and start-of-period state to the start-of-period value function and the end-of-period state.

This type of "within-period model" will form the computational core of a dynamic model. To explain the strategy, please assume for now that such an IPP function exists. Constructing the IPP function is what I refer to as "solving the within-period model," and I address it in Section 5.

3 Between-Period Model

The "within-period model" described in Section 2 is not terribly useful yet. What does it tell us about equilibrium? Dynamics? Calibration? I show that all of these problems can be formulated as the problem of solving a "between-period model" takes takes a "within-period model" as an input.

The first issue is that sometimes we don't want to take V^{end}, Λ^{start}, and S^{start} for granted, but instead solve for them somehow. The key to all of this is that many complex models can be thought of as simply representing different notions of "solving for" V^{end}, Λ^{start}, and S^{start}. That is, as the solution to a system of equations,

$$H\left(\left\{V_m^{\text{end}}\right\}_m, \left\{\Lambda_m^{\text{start}}\right\}_m, \left\{S_m^{\text{start}}\right\}_m, \text{IPP}\right) = 0,$$

where $\{\Lambda_m^{\text{start}}\}_m$ is a vector of different starting populations Λ_m^{start}, and similarly for $\left\{V_m^{\text{end}}\right\}_m$ and $\{S_m^{\text{start}}\}_m$. The following examples can hopefully illustrate this idea.

3.1 Example: Stationary Steady State

The notion of a stationary steady state can be constructed as,

$$\begin{aligned} V^{\text{end}} &= \beta V^{\text{start}} \\ \Lambda^{\text{start}} &= \Lambda^{\text{end}} \\ S^{\text{start}} &= S^{\text{end}} \\ \left(V^{\text{start}}, \Lambda^{\text{end}}, S^{\text{end}}\right) &= \text{IPP}\left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}\right) \end{aligned}$$

Formally, you can construct H as some vector-valued function

$$\begin{aligned} &H\left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}\right) = \left(V^{\text{start}} - V^{\text{end}}, \Lambda^{\text{start}} - \Lambda^{\text{end}}, S^{\text{start}} - S^{\text{end}}\right) \\ &\text{where } \left(V^{\text{start}}, \Lambda^{\text{end}}, S^{\text{end}}\right) = \text{IPP}\left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}\right). \end{aligned}$$

But this is sort of an arbitrary encoding, and in general cannot always be computed so simply, so for the remainider of this section I stick with just describing the system of equations. In Section 4, I describe more generally how to "evaluate" a candidate numerical solution to the between-period-problem.

3.2 Example: Two-Period Model

A two-period model, taking Λ_1^{start}, S_1^{start}, and V_2^{end} as given, can be constructed as,

$$\begin{aligned}\left(V_t^{\text{start}}, \Lambda_t^{\text{end}}, S_t^{\text{end}}\right) &= \text{IPP}\left(V_t^{\text{end}}, \Lambda_t^{\text{start}}, S_t^{\text{start}}\right) \ \forall t \in \{1,2\} \\ V_1^{\text{end}} &= \beta V_2^{\text{start}} \\ \Lambda_2^{\text{start}} &= \Lambda_1^{\text{end}} \\ S_2^{\text{start}} &= S_1^{\text{end}}.\end{aligned}$$

Note that, if a function IPP^{back} exists, as in Section 2 but not depending on S^{start}, then a the two-period model can be solved in a loop-free fashion according to the algorithm:

Algorithm 1 Standard Two-Period Model Solver

1. Take Λ_1^{start}, S_1^{start}, and V_2^{end} as given
2. $V_2^{\text{start}} \longleftarrow \text{IPP}^{\text{back}}\left(V_2^{\text{end}}\right)$
3. $V_1^{\text{end}} \longleftarrow V_2^{\text{start}}$
4. $V_2^{\text{start}} \longleftarrow \text{IPP}^{\text{back}}\left(V_2^{\text{end}}\right)$
5. $\Lambda_1^{\text{end}} = \text{IPP}\left(V_1^{\text{end}}, \Lambda_1^{\text{start}}, S_1^{\text{start}}\right)_2$
6. $S_1^{\text{end}} = \text{IPP}\left(V_1^{\text{end}}, \Lambda_1^{\text{start}}, S_1^{\text{start}}\right)_3$
7. $\Lambda_2^{\text{end}} = \text{IPP}\left(V_2^{\text{end}}, \Lambda_2^{\text{start}}, S_2^{\text{start}}\right)_2$
8. $S_2^{\text{end}} = \text{IPP}\left(V_2^{\text{end}}, \Lambda_2^{\text{start}}, S_2^{\text{start}}\right)_3$

However, it is important to note that this is not the *only* way in which you might solve a two-period model as described above. In fact, one of the main things I'm trying to do is to *separate* the system of equations defining the model and the particular solution algorithm or "solver." The construction of the solver will therefore be introduced separately in Section 6.

3.3 Example: Perfect-Foresight Transition

The notion of a perfect-foresight transition, taking the initial state Λ_1^{start} and S_1^{start} as given, with exponential discounting at rate β, can be constructed as,

$$\begin{aligned} V_t^{\text{end}} &= \beta V_{t+1}^{\text{start}} \ \forall t \in \mathbb{N} \\ \Lambda_{t+1}^{\text{start}} &= \Lambda_t^{\text{end}} \ \forall t \in \mathbb{N} \\ S_{t+1}^{\text{start}} &= S_t^{\text{end}} \ \forall t \in \mathbb{N} \\ \left(V_t^{\text{start}}, \Lambda_t^{\text{end}}, S_t^{\text{end}}\right) &= \text{IPP}\left(V_t^{\text{end}}, \Lambda_t^{\text{start}}, S_t^{\text{start}}\right) \ \forall t \in \mathbb{N}. \end{aligned}$$

Note that this defines an infinite number of equations, and thus is not computable in practice. Thus, when we go to solve this system of equations, we will have to construct an objective function that does not line up exactly with the "true model." In particular, the usual trick is to simulate forward T periods. Formally, to define a sequence of models, $\{M_T \mid T \in N\}$, with M_T being defined by

$$\begin{aligned} V_t^{\text{end}} &= \beta V_{t+1}^{\text{start}} \ \forall t \in \{1, \ldots, T-1\} \\ \Lambda_{t+1}^{\text{start}} &= \Lambda_t^{\text{end}} \ \forall t \in \{1, \ldots, T-1\} \\ S_{t+1}^{\text{start}} &= S_t^{\text{end}} \ \forall t \in \{1, \ldots, T-1\} \\ V_T^{\text{end}} &= \beta V_T^{\text{start}} \\ \Lambda_T^{\text{start}} &= \Lambda_T^{\text{end}} \\ S_T^{\text{start}} &= S_T^{\text{end}} \\ \left(V_t^{\text{start}}, \Lambda_t^{\text{end}}, S_t^{\text{end}}\right) &= \text{IPP}\left(V_t^{\text{end}}, \Lambda_t^{\text{start}}, S_t^{\text{start}}\right) \ \forall t \in \{1, \ldots, T\}, \end{aligned}$$

together with an argument that the sequence of models M_T converges, in some sense, to the "true model."

However, just as I am keen to distinguish between the system of equations and its solution algorithm, I am keen to distinguish between the "true model" and "approximate model." This will be discussed more fully in Section 7.

In particular, the restriction to a small subset of the full set of equations in the true model, as here, is in some sense a *sampling procedure*. These will become crucial for solving large state space models, such as those with heterogeneous agents and aggregate shocks. This is the reason why I take pains to distinguish between the "true" set of equations defining the between-period model and the numerical "objective function" used to evaluate a candidate numerical solution in Section 4.

3.4 Example: Market Clearing

To construct market-clearing conditions, suppose that period-t prices, p, are contained within S^{start} and period-t excess demand quantities, q, are contained

within S^{end}. Then, market clearing conditions can be constructed as,

$$\begin{aligned} q\left(S^{\text{end}}\right) &= 0 \\ \left(V^{\text{start}}, \Lambda^{\text{end}}, S^{\text{end}}\right) &= \text{IPP}\left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}\right) \end{aligned}$$

3.5 Example: Overlapping Generations

A stationary steady state with two overlapping generations can be constructed in a couple ways. One is to just treat it as a two-period model, but that is a sort of useful lie—there is just a single period, and setting it up in this way messes up the strategies presented here for cleanly introducing new features such as market clearing and dynamics. Instead, the more compatible formulation would look involve introducing age-rollover operators Age_V and Age_Λ which apply the "change of age" to V and Λ. That is, if the idiosyncratic state of each household i consists of an age a_i and other state variables x_i, such that for any V and Λ

$$\begin{aligned} \left(\text{Age}_V V\right)(x, 1) &= V(x, 2) \\ \left(\text{Age}_V V\right)(x, 2) &= V^{\text{bequest}}(x) \\ \left(\text{Age}_\Lambda \Lambda\right)(x, 2) &= \Lambda(x, 1) \\ \left(\text{Age}_\Lambda \Lambda\right)(x, 1) &= \Lambda^{\text{birth}}(x) \end{aligned}$$

where V^{bequest} and Λ^{birth} are some exogenous terminal utility and entry population distribution, respectively. Then the model can be written,

$$\begin{aligned} V^{\text{end}} &= \beta \text{Age}_V V^{\text{start}} \\ \Lambda^{\text{start}} &= \text{Age}_\Lambda \Lambda^{\text{end}} \\ S^{\text{start}} &= S^{\text{end}} \\ \left(V^{\text{start}}, \Lambda^{\text{end}}, S^{\text{end}}\right) &= \text{IPP}\left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}\right) \end{aligned}$$

3.6 Example: Search and Matching

A model with job search and matching can be constructed similarly to the model with market clearing conditions described in Section 3.4, except that:

- In addition to prices (wages), the market tightness parameters are also included in the state S^{start}.
- In addition to excess demand, the job finding rates are also included in the state S^{end}.

Apologies that this example is a little light on details. I will work on expanding it.

3.7 Example: Epstein-Zinn Preferences

A stationary steady state with Epstein-Zinn preferences can be defined by allowing consumption c to persist as a state variable in Λ. Then the line in Section 3.1,

$$V^{\text{end}} = \beta V^{\text{start}}$$

is replaced with,

$$V^{\text{end}} = \left[(1-\beta)\, c \left(V^{\text{start}}\right)^{\rho} + \beta\mu \left(V^{\text{start}}\right)^{\rho}\right]^{1/\rho}.$$

3.8 Example: Aggregate Uncertainty

3.9 Example: Calibration by Indirect Inference

Another intriguing example is that calibration by indirect inference can be computed quite similarly to prices. In the case of a one-period model, suppose that the parameters to be calibrated, θ, are included in S^{start}, and the simulated moments, Y, are included in S^{end}. Let the data moments be Y_0. Then the problem becomes,

$$\begin{aligned} &\min_{\theta} c\left(Y\left(S^{\text{end}}(\theta)\right), Y_0\right) \\ &\text{where } \left(V^{\text{start}}, \Lambda^{\text{end}}, S^{\text{end}}(\theta)\right) = \text{IPP}\left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}(\theta)\right). \end{aligned}$$

4 Overall Objective Function

In some cases, the between-period model might be characterized by a very large or infinite number of equations. In this case, it is necessary to develop an approximate objective function with which to evaluate the model.

4.1 Full Set of Defining Equations

In cases when the between-period model is characterized by a feasibly-small number of equations, the objective function can simply be constructed as that full set of equations, as in the H function in Section 3.1.

4.2 Finite-Horizon Approximations to Infinite-Horizon Transition Paths

4.3 Sampling Procedure as Objective Function: Monte Carlo

5 Within-Period Model Solver

For the sake of many of the between-period model solvers described in Section 6, it is often necessary to implement not only IPP, but also a backwards-iteration

function IPP^{back}, such that,

$$\begin{gathered}\text{For all } V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}, \\ \text{If } \left(V^{\text{start}}, \Lambda^{\text{end}}, S^{\text{end}}\right) = \text{IPP}\left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}\right) \\ \text{Then } \left(V^{\text{start}}, S^{\text{start}}\right) = \text{IPP}^{\text{back}}\left(V^{\text{end}}, S^{\text{start}}\right).\end{gathered}$$

Different authors have independently hit upon the idea of constructing such functions through further decomposing the IPP functions into multiple sub-period "stages" or "phases." For example, mechanisms for constructing IPP functions in this way appear in the ECON-ARK and SSJ toolkits. I describe my particular formulation of this decomposition here, and am working on provided some of the necessary code on my website.

It can also often be useful to differentiate *through* the IPP function. In particular, this allows us to use gradient-based solvers. Automatic differentiation can be a very powerful tool for differentiating through IPP. In the case where the IPP function is implicitly defined, and internally implemented using iterative methods, automatic differentiation can still be used with a little extra setup, which I discuss here.

6 Between-Period Model Solver

6.1 Example: Two Period Model

I describe in Section 3.2 a simple algorithm for solving a two-period model.

6.2 Example: Brute Force Solver

In some sense, the "simplest" idea would be to "stack" all of the equations into a single objective function, then just use a standalone nonlinear equation solver. That is, assuming we haveimplemented the H function as in 3.1,

Algorithm 2 Brute Force Solver

1. Take the H and IPP functions as given
2. Guess $\left\{V_m^{\text{end}}\right\}_m, \{\Lambda_m^{\text{start}}\}_m$, and $\{S_m^{\text{start}}\}_m$.
3. $err \longleftarrow H\left(\{\Lambda_m^{\text{start}}\}_m, \left\{V_m^{\text{end}}\right\}_m, \{S_m^{\text{start}}\}_m, \text{IPP}\right)$
4. Use the nonlinear equation solver to obtain a new guess for $\{\Lambda_m^{\text{start}}\}_m, \left\{V_m^{\text{end}}\right\}_m$, and $\{S_m\}_m$, given err.
5. Return to Step 3.

6.3 Example: Value Function and State Iteration

In the case of a stationary steady state model as in Section 3.1,

$$\begin{aligned} V^{\text{end}} &= \beta V^{\text{start}} \\ \Lambda^{\text{start}} &= \Lambda^{\text{end}} \\ S^{\text{start}} &= S^{\text{end}} \\ \left(V^{\text{start}}, \Lambda^{\text{end}}, S^{\text{end}}\right) &= \text{IPP}\left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}\right), \end{aligned}$$

value function iteration and state iteration is often effective.

Algorithm 3 Value Function and State Iteration

1. Take V^{end}, Λ^{start}, and S^{start} as given
2. $\left(V^{\text{start}}, \Lambda^{\text{end}}, S^{\text{end}}\right) \longleftarrow \text{IPP}\left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}\right)$
3. $V^{\text{end}} \longleftarrow \beta V^{\text{start}}$
4. $\Lambda^{\text{start}} \longleftarrow \Lambda^{\text{end}}$
5. $S^{\text{start}} \longleftarrow S^{\text{end}}$

6.4 Example: Tatonnement

In the case of a market clearing model as in Section 3.4,

$$\begin{aligned} q\left(S^{\text{end}}\right) &= 0 \\ \left(V^{\text{start}}, \Lambda^{\text{end}}, S^{\text{end}}\right) &= \text{IPP}\left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}\right), \end{aligned}$$

Tattonnement is often effective.

Algorithm 4 Tatonnement

1. Take V^{end} and Λ^{start} as given. Select some update rate α.
2. Guess prices p
3. Construct the state: $S^{\text{start}} \longleftarrow S^{\text{start}}(p)$
4. $\left(V^{\text{start}}, \Lambda^{\text{end}}, S^{\text{end}}\right) \longleftarrow \text{IPP}\left(V^{\text{end}}, \Lambda^{\text{start}}, S^{\text{start}}\right)$,
5. Extract excess demand: $q \longleftarrow q\left(S^{\text{end}}\right)$
6. Update prices: $p \longleftarrow p + \alpha q$
7. Return to Step 3.

6.5 Example: Krusell-Smith

6.6 Example: Deep Approximate Dynamic Programming

I describe this in more depth in my slides herehere.

6.7 Example: Perturbation Around Perfect-Foresight Transition

7 True Model vs. Approximate Model

What does it mean for one model to approximate another? This can be formalized by thinking about a sequence of approximator models that converge to a "true model." That is, if M_∞ is a model, and $\{M_k \mid k \in \mathbb{N}\}$ is a sequence of models, then the models M_k are said to "approximate" M_∞ if $M_k \to_{k\to\infty} M_\infty$. Note that this can only formally be understood by keeping track of the whole sequence, for instance by parameterizing it.

7.1 Example: Idiosyncratic State Discretization

7.2 Example: Aggregate Shock Discretization

7.3 Example: Finite-Horizon Approximations to Infinite-Horizon Transition Paths

7.4 Example: Time-Discretized Continuous-Time Models

It is sometimes remarked that when we transform a discrete time model to continuous and then solve it on a computer, we are making the model discrete-time again in order to solve it. This is an interesting insight and suggests to me that some of the benefits of continuous-time methods can be realized without touching the daunting machinery of SDEs.

I begin with the following observation. When we numerically solve a continuous-time model, we are in effect:

1. Starting with a convergent sequence of less-tractable discrete-time models, $\{M_k \mid k \in \mathbb{N}\}$. For instance, with time step $\Delta t = k^{-1}$.
2. Finding a different sequence of more-tractable discrete-time models with the same limit, $\left\{\widetilde{M}_k \mid k \in \mathbb{N}\right\}$.
3. Solving one of the models $\widetilde{M}_k$ in the second sequence.

What do I mean that the models $\widetilde{M}_k$ are "more tractable" than the models M_k? Some examples are the assumptions that, in $\widetilde{M}_k$,

1. Shock realizations are mutually exclusive within time steps. For instance, the probability that the Calvo fairy visits twice within the same time step ends up being dominated by the probability that it visits once.

2. Demand and supply in each time step depend on prices in the previous period. (Possibly subject to some sort of renormalization.) That is, as the time step gets small, the change in prices between time steps gets small so that, in the limit, prices in each time step are well-approximated by prices in the previous time step.

This gives me the interesting idea of avoiding some of the formalism of continuous-time models by instead working directly with the sequences of M_k and $\widetilde{M}_k$, and to solve $\widetilde{M}_k$, applying transparently discrete-time techniques.