1 Notation

{sec:notati

1.1 Periods, Stages, Steps

The problem so far assumes that the agent has only one decision problem to solve in any period. But it is increasingly common to model agents who have multiple choice stages per period; a problem might have, say, a consumption decision (call it the c stage), a labor supply stage (call it ℓ) and a choice of what proportion ς of their assets to invest in a risky asset (the portfolio-choice stage).

The modeler might well want to explore whether the order in which the stages are solved makes any difference, either to the substantive results or to aspects of the computational solution like speed and accuracy.

If, as in section **??**, we hard-wire into the solution code for each stage an assumption that its successor stage will be something in particular (say, the consumption stage assumes that the portfolio choice is next), then if we want to change the order of the stages (say, labor supply after consumption, followed by portfolio choice), we will need to re-hard-wire each of the stages to know particular things about its new successor (for example, the specifics of the distribution of the rate of return on the risky asset must be known by whatever stage precedes the portfolio choice stage).

But one of the cardinal insights of Bellman's (1957, "Dynamic Programming") original work is that *everything that matters* for the solution to the current problem is encoded in a 'continuation-value function.' Using Bellman's insight, we describe here a framework for isolating the stage problems within a period from each other, and the period from its successors in any future period; the advantage of this is that the isolated stage and period problems will then be 'modular': We can solve them in any order *without changing any code* (only transitions need to be rewired). After considering the stage-order $[\ell, \mathrm{c}, \varsigma]$, the modeler can costlessly reorder the stages to consider, say, the order $[\ell, \varsigma, \mathrm{c}]$.[1]

1.2 Steps

{subsec:ste

The key is to distinguish, within each stage's Bellman problem, three steps:

1. **Arrival**: Incoming state variables (e.g., k) are known, but any shocks associated with the period have not been realized and decision(s) have not yet been made
2. **Decision**: The agent solves the decision problem for the period
3. **Continuation**: After all decisions have been made, their consequences are measured by evaluation of the continuing-value function at the values of the 'outgoing' state variables (sometimes called 'post-state' variables).

Notice that this specification is silent about when the stochastic shocks are realized; this may occur either before or after the decision stage. In the consumption problem we

[1]As long as the beginning-of-stage and end-of-stage value functions for the stages all depend on the same state variables; see the discussion in section **??**.

are studying, the natural choice is to assume that the shocks have been realized before the decision is made so that the consumer knows what their income has been for the period. In the portfolio problem we will examine below, the portfolio share decision must be made before the stochastic returns are realized.

When we want to refer to a specific step in the stage we will do so by using an indicator which identifies that step. Here we use the consumption stage problem described above to exemplify the usage:

Step	Indicator	State	Usage	Explanation
Arrival	$\leftharpoonup$	k	$\mathrm{v}_{\leftharpoonup}(k)$	value at entry to stage (before shocks)
Decision(s)	(blank)	m	$\mathrm{v}(m)$	value of stage-decision (after shocks)
Continuation	$\rightharpoondown$	a	$\mathrm{v}_{\rightharpoondown}(a)$	value at exit (after decision)

Notice that the value functions at different steps of the stage have distinct state variables. Only k is known at the beginning of the stage, and other variables take on their values with equations like $b = k\mathcal{R}$ and $m = b + \theta$. We will refer to such within-the-stage creation of variables as 'evolutions.' So, the consumption stage problem has two evolutions: from k to m and from m to a.

1.3 Transitions

{subsec:tra

In the backward-induction world of Bellman solutions, to solve the problem of a particular period we must start with an end-of-period (continuation) value function, which we designate by explicitly including the period indicator in the subscript (the := symbol denotes that the object on the right hand side is assigned to the object on the left hand side; the left object 'gets' the right object):needs discussion: It's made at the time of execution of Matt's link structure; but is it a pointer, a deepcopy, an algorithm, or what?

$$\mathrm{v}_{t\rightharpoondown}(a) := \beta \mathrm{v}_{\leftharpoonup(t+1)}(\overbrace{a}^{=k}), \tag{1}$$

{eq:trns-si

and we are not done solving the problem of period t until we have constructed a beginning-of-period value function $\mathrm{v}_{\leftharpoonup t}(k)$.

Similarly, in order to solve the problem of any stage, we must endow it with an end-of-stage continuation-value function. For the last stage in a period, the end-of-stage function is taken to be end-of-period value function; in our case where there is only one stage, this can be written cleanly as:

$$\mathrm{v}_{\rightharpoondown}(a) := \mathrm{v}_{t\rightharpoondown}(a). \tag{2}$$

{eq:last-st

pseudocode?

1.4 The Decision Problem in the New Notation

{subsec:dec

From ‘inside’ the decision stage, the Decision problem can now be written much more cleanly than in equation (**??**):

$$\mathrm{v}(m) = \max_{c}\ \mathrm{u}(c) + \mathrm{v}_{\neg}(\overbrace{m - \mathrm{c}}^{=a}) \tag{3}$$

{eq:vMidStg