APPENDIX E. DESCRIPTION OF MODEL MECHANICS

This appendix explores in greater detail some of the modeling choices and mechanics described in Chapter 6. Specifically, it provides further documentation on the array structure employed, and the decision rules guiding contest-related promotion flows.

Arrays

In theory, the entire simulation could record YOS movements from one stock to the next using a single array. In practice, however, this was not feasible. Creating a unique array with 37 elements (as many as the maximum years of service before retirement) proved impossible: ITHINK models featuring large arrays and correspondingly large transition matrices are extremely computing-intensive and require disproportionate amounts of storage space.\(^{162}\)

To address this problem, smaller arrays were created—in most cases one per grade. These encompass the entire YOS range that could conceivably be found in that grade. The array size for a given grade depends on the size of arrays from more junior grades that are linked to it, as well as the maximum number of years a soldier could stay in that grade. For instance, assume individuals from the VSP1 grade (which only requires one year in grade before promotion) being promoted as VPS2s can have between 7 and 11 years of service. Assume further that the VSP2 grade takes 5 years to complete. As shown in Figure 6.5, this implies that the first element of the VSP2 grade array needs be YOS8 (no one could have 7 or less YOS), and that the last element needs to be YOS16 (no one could stay in the grade beyond 16 YOS).

\(^{162}\) In fact, the ITHINK platform proved to be inefficient at handling arrays. The software program does not allow for easy construction of transition matrices (each transition probability has to be specified using a series of equations), and large matrices and array sizes create giant program files and turn model operation into an extremely slow process.
In general, the range for grade \( i \) can be summarized in equation E.1:

\[
\text{ArrayRange}_i = [YOS_y, \ldots, YOS_o + \text{MaxYears}_i]
\]  

(E.1)

where \( YOS_y \) is represents the years of service of the youngest possible entrant, \( YOS_o \) represents the YOS of the oldest possible entrant, and \( \text{MaxYears}_i \) specifies the maximum number of years that can be spent in year \( i \).\(^{163}\)

The specific arrays used and their respective names are summarized in Table E.1 below. This table shows that there are three exceptions to the one-array-per-grade rule: (1) the VFB category, which only requires array use for those who moved beyond the first 3- or 5-year term; (2) the S1 and S2 grades, for which a special array (SYOSAlt) was created to keep track of the YOS progression of unusually young cohorts during the first years of the simulation; and (3) the M3 grade, where three arrays were created in order to limit a ballooning file size.

\(^{163}\) Promotions through competition entail an extra year spent in the grade one occupied when he/she competed. This is factored into the calculation of the number of YIG stocks for each grade, and therefore in the sizes of arrays. For each non-terminal grade (e.g., VSP2s and VSP3s) that in theory could produce a promotion contest winner, an extra YIG stock was created.
Table E.1 Arrays used in the model and their respective sizes

<table>
<thead>
<tr>
<th>Array Name</th>
<th>Range of array elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFBYOS</td>
<td>YOS4 to YOS10</td>
</tr>
<tr>
<td>VSP1YOS</td>
<td>YOS5 to YOS12</td>
</tr>
<tr>
<td>VSP2YOS</td>
<td>YOS6 to YOS18</td>
</tr>
<tr>
<td>VSP3YOS</td>
<td>YOS11 to YOS23</td>
</tr>
<tr>
<td>VSP4YOS</td>
<td>YOS16 to YOS37</td>
</tr>
<tr>
<td>SYOS</td>
<td>YOS8 to YOS37</td>
</tr>
<tr>
<td>SYOSAlt</td>
<td>YOS4 to YOS10</td>
</tr>
<tr>
<td>M0&amp;1&amp;2YOS</td>
<td>YOS1 to YOS22</td>
</tr>
<tr>
<td>M3YOS</td>
<td>YOS12 to YOS31</td>
</tr>
<tr>
<td>M3YOS2</td>
<td>YOS20 to YOS37</td>
</tr>
<tr>
<td>M3YOS3</td>
<td>YOS31 to YOS37</td>
</tr>
<tr>
<td>M4YOS</td>
<td>YOS14 to YOS37</td>
</tr>
</tbody>
</table>

Multiple flows

The basic challenge for this promotion mechanism is to devise a procedure that allocates promotion flows from multiple stocks into another stock, under the constraint that

\[ \sum_{\text{inflows}_{\text{stock D}}} = X \]  \hspace{1cm} (E.2)

where \( X \) is a quota specified by the Army.

Solving this with the use of logical statements (IF...THEN...ELSE) can be overwhelming, since in some situations the number of YIG stocks involved in the multiple flows are very large.\(^{164}\) A more "emergent" and simple solution involves calculating inflows from A, B, and C as fractions of a time period (delta time, or dt), updating the size of stock D each dt.\(^{165}\) In essence, each YIG stock contributes to D a fraction of X at each time period. At the moment when D=X, all inflows stop for that time period. Stocks continue sending their allotted quota at each dt until they reach zero quantity; once they do, the remaining stocks still able to contribute increase their relative share of inflows until X, or the maximum quantity lower than X, is sent. By default, the

---

\(^{164}\) This is especially true in the terminal grades, where one could conceivably spend twenty years of his/her career. The IF...THEN...ELSE combinations that would have to be considered in these cases are extremely large and therefore unworkable.

\(^{165}\) These stock names refer to the stylized case first illustrated in section 6.2.
model makes 128 calculations per time period. Such a small delta time implies that each stock sends only a small fraction of what D requires at each calculation; this in turn ensures that flows to D do not under- and/or overshoot X.\textsuperscript{166}

In order to make this approach function properly, ITHINK needs to "know" how many calculations are left in a particular time period. To achieve this, a variable (DT.Counter) was created which sums the number of elapsed dts in a period. For instance, if there are 4 calculations per time period, then 1 dt equals 0.25 (1 divided by 4). In this case DT.Counter cycles through the following four values: 0.25, 0.5, 0.75, 0. Based on DT.Counter, the following decision rules regulating flows to D apply for material in stock A:

- **Attrition (priority 1):** If DT.Counter=dt, then Attrition\*(A/dt), else 0

This implies that all attrition-related outflows take place only at the first calculation (dt), and that the inflow happens instantaneously. The instantaneous flow is ensured by the fact that attrition exits are divided by dt.\textsuperscript{167}

- **Flows to D (priority 2):** If DT.Counter=1-dt then 0, else (Quota-D)/(1-DT.Counter)

The expression above instructs the model to incrementally send individuals to D until the next to last dt (0.75 if dt=1/4)--the last flow actually will be counted by the program in last dt (0) of the time period. The promotion flow is divided by "1-DT.Counter" as a way to weigh the outflows by the remaining dts (the fewer the remaining dts, the larger the outflow).

- **Flows to B (priority 3):** If DT.Counter= 0 then A/dt else 0

\textsuperscript{166} This approach assumes units are divisible. Given the scale of the flows and stock sizes, however, fractional promotions do not sensibly affect overall model output for each time period.

\textsuperscript{167} In each dt, the outflow would equal the expected attrition quantity--say L--divided by the number of calculations. It follows that to get an instantaneous flow of L the flow amount for that calculation needs to be L/dt, since (L/dt)/dt = L.
In essence, whatever is left in A after the exits due to attrition and promotion into D are accounted for is taken and moved to B at the end of the period. The material flows from A in the last dt, and appears in B's inventory at the beginning of the next period.