APPLICATION OF DYNAMIC PROGRAMMING TO THE AIRPLANE MINIMUM TIME-TO-CLimb PROBLEM

T. F. Cartaino
S. E. Dreyfus

RM-1710

22 March 1956

This research is sponsored by the United States Air Force under contract No. AF 49(638)-700 monitored by the Directorate of Development Planning, Deputy Chief of Staff, Research and Technology, Hq USAF.

This is a working paper. It may be expanded, modified, or withdrawn at any time. The views, conclusions, and recommendations expressed herein do not necessarily reflect the official views or policies of the United States Air Force.
SUMMARY

The dynamic-programming technique developed by R. Bellman is applied to the solution of the airplane minimum time-to-climb problem. A brief introduction to dynamic programming is presented, followed by an exposition of the climb problem and the solution of a typical example case. It is concluded that dynamic programming offers a method of solution for the climb problem which is fast, is readily adaptable to routine engineering calculation, and allows the inclusion of the effects of variations in airplane weight and drag along the flight path.
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUMMARY</td>
<td>11</td>
</tr>
<tr>
<td>LIST OF SYMBOLS</td>
<td>iv</td>
</tr>
<tr>
<td>I. INTRODUCTION TO DYNAMIC PROGRAMMING</td>
<td>1</td>
</tr>
<tr>
<td>II. APPLICATION TO CLIMB PROBLEM</td>
<td>4</td>
</tr>
<tr>
<td>III. EXAMPLE PROBLEM</td>
<td>7</td>
</tr>
<tr>
<td>IV. CONCLUSIONS</td>
<td>10</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>17</td>
</tr>
</tbody>
</table>
LIST OF SYMBOLS

\( a, b \)  
matrix designation

\( C_{D_0} \)  
zero-lift drag coefficient

\( D \)  
airplane drag, lb

\( g \)  
acceleration due to gravity, ft/sec^2

\( H \)  
altitude, ft

\( I, J \)  
matrix elements

\( K \)  
drag-due-to-lift factor

\( M \)  
flight Mach number

\( m \)  
number of columns in matrix

\( N \)  
maneuver load factor

\( n \)  
number of rows in matrix

\( q \)  
dynamic pressure, lb/ft^2

\( S \)  
reference wing area, ft^2

\( T \)  
available thrust, lb

\( t \)  
time, sec

\( V \)  
airplane velocity, ft/sec

\( V_S \)  
speed of sound, ft/sec

\( W \)  
airplane gross weight, lb

\( X \)  
horizontal distance, n mi

\( \delta \)  
ambient pressure ratio

\( \phi \)  
climb angle, deg
I. INTRODUCTION TO DYNAMIC PROGRAMMING

Dynamic programming, as developed by R. Bellman at The RAND Corporation, is a method for solving problems arising from multi-stage decision processes. In particular, problems in the calculus of variations can be considered to arise from multi-stage decision processes of a continuous type. In this method neither constraints nor discontinuities in the expressions involved present any special difficulties. The mathematical theory of dynamic programming has been treated in Refs. 1 and 2 and will not be discussed here. To explain the method we resort to a simple example. Consider the matrix (a):

\[
\begin{array}{cccc}
8 & 3 & 4 & 4 \\
9 & 4 & 7 & 3 \\
1 & 6 & 5 & 2 \\
2 & 5 & 1 & 6 \\
\end{array}
\]

Suppose we wish to travel from position \( a_{41} \) to position \( a_{44} \) so that the sum of the integers encountered along the way is a minimum. Let us restrict ourselves to moving one integer at a time, either one row up or one column to the right. The obvious technique for finding the minimum-sum or optimal path is to start at \( a_{41} \) and try all possible paths through the matrix to \( a_{44} \), finding the solution by inspection. This method of solution, the brute-force or enumerative approach, is usually prohibitive timewise, since the number of separate calculations involved is

\[
\frac{(n + m - 2) \cdot (n + m - 2)!}{(n - 1)! \cdot (m - 1)!} \quad (1)
\]

where \( n \) is the number of rows and \( m \) the number of columns. A 10 x 10 matrix \((n = m = 10)\) requires \(875,160\) separate calculations to try all paths according to our rule for possible moves.
In the theory of dynamic programming it has been shown that by starting
at the final position $a_{14}$ and moving through the matrix backwards, using the
rules given below, we in effect test every possible path without actually per-
forming each numerical operation. The number of calculations now required is

$$2(n-1)(m-1) + (n + m - 2)$$

(2)

which for a $10 \times 10$ matrix becomes 180 separate calculations. The advantage
of the dynamic programming technique is that it reduces the number of calcula-
tions to be performed to a small fraction of those required by the brute-force
method.

To illustrate the method of moving through the matrix backwards, we return
to matrix (a), and construct a new matrix (b) by the rule

$$b_{ij} = a_{ij} + \min(b_{i,j+1}, b_{i-1,j}),$$

(3)

where the symbol $\min(x,y)$ means the smaller of the two quantities. At the upper
row

$$b_{ij} = a_{ij} + b_{i,j+1},$$

(4)

and at the right-hand column

$$b_{ij} = a_{ij} + b_{i-1,j}.$$

(5)

Using rules (3)-(5) we obtain matrix (b):

\[
\begin{array}{cccc}
19 & 11 & 8 & 4 \\
24 & 15 & 14 & 7 \\
21 & 20 & 14 & 9 \\
22 & 20 & 15 & 15 \\
\end{array}
\]
Matrix (b) has the property that an element \( b_{ij} \) is the sum of elements of \( a_{ij} \) encountered along the optimal path from \( a_{ij} \) to the final position \( a_{14} \). Consequently, when we start at position \( b_{41} \) in matrix (b) and move successively according to our restrictions to the smaller of the two possible integers, we are following the path of minimum-sum in matrix (a). By inspection, the path we should follow through matrix (b) is:

\[
\begin{array}{cccc}
19 & 11 & 8 & 4 \\
24 & 15 & 14 & 7 \\
21 & 20 & 14 & 9 \\
22 & 20 & 15 & 15 \\
\end{array}
\]

This path through matrix (b) corresponds to the path through matrix (a), shown below, which is the desired solution to our example problem:

\[
\begin{array}{cccc}
8 & 3 & 4 & 4 \\
9 & 4 & 7 & 3 \\
1 & 6 & 5 & 2 \\
2 & 5 & 1 & 6 \\
\end{array}
\]

It will be noted that the sum of integers along the optimal path from \( a_{41} \) to \( a_{14} \) is 22, which corresponds to the element \( b_{41} \) in matrix (b). Consequently, matrix (b) gives us not only the indication of the optimal path through matrix (a) but also the sum of the integers along this path.
II. APPLICATION TO CLIMB PROBLEM

Turning to consideration of airplane performance, time-to-climb is given by the expression

\[ t = \int_{H_1}^{H_2} \frac{1}{(dH/dt)} \ dH , \]  

(6)

where

\[ \frac{dH}{dt} = \frac{V}{W} (T - D) \]

\[ = \frac{V}{1 + \frac{V}{g} \frac{dV}{dH}} \]

\[ = V \sin \theta . \]  

(7)

Time-to-climb can therefore be expressed functionally as

\[ t = f \left( H, V, \text{throttle setting}, \frac{dH}{dt}, \frac{dV}{dH} \right) . \]  

(8)

Since climb is usually at a fixed throttle setting, especially for the minimum-time case, we can write

\[ t = f \left( H, V, \frac{dH}{dt}, \frac{dV}{dH} \right) . \]  

(9)

The problem of finding the minimum-time path from one point to another in the altitude-velocity plane is to minimize Eq. (9). Problems of this type are subject to solution by the calculus of variations, and this approach has been tried by Kirkwood (Ref. 3), Behrbom (Ref. 4), and others. Solution by the calculus of variations, however, presents a variety of difficulties, among which are complexities of formulation and, in many cases, indeterminate solutions. For engineering use this approach is intractable. Another approach, the energy method, is presented by Rutowski in Ref. 5. This approach, while better suited to engineering use than the calculus of variations is objectionable because only the initial and final altitudes, and not the associated velocities, can be specified. Further, the unrealistic assumption is made
that dives and zooms are accomplished in zero time. By formulation and solution of a typical example case it will be shown how the application of dynamic programming eliminates the objectionable features of both the calculus of variations and energy approaches, and introduces several advantages as well.

Analogous to our original example, we now consider the \( a_{ij} \) matrix to be an \( a_{HV} \) matrix, where each row represents an altitude and each column a velocity. Let the top row be the desired final altitude, the bottom row the starting altitude, the left-hand column the initial velocity, and the right-hand column the final or limiting velocity. What we require is a path of minimum time from \( a_{H_1, V_1} \) to \( a_{H_f, V_f} \), where \( H_1 \) is the initial altitude, \( V_1 \) the initial velocity, and \( H_f \) and \( V_f \) the final altitude and velocity, respectively. The elements \( a_{HV} \) are now taken to be the time required to climb from one row to the next at constant velocity or to accelerate from one column to the next at constant altitude. When \( \Delta H \), the altitude change between adjacent rows, and \( \Delta V \), the velocity increment between adjacent columns, are sufficiently small, any path can be arbitrarily closely approximated by combinations of incremental climbs and incremental accelerations. We have chosen to limit the airplane to these two moves, although this is not a necessary restriction upon the method.

We now define the \( (b_i) \) matrix as follows:

\[
    b_{HV} = \min \left[ \begin{array}{c}
        a_{H+\Delta H, V} * b_{H+\Delta H, V} \\
        a_{H, V+\Delta V} * b_{H, V+\Delta V}
    \end{array} \right] \quad (10)
\]

This matrix is formed, as in the example, by starting in the upper right-hand position, the end point, and moving backwards through the matrix, so that \( b_{H+\Delta H, V} \) and \( b_{H, V+\Delta V} \) are known before \( b_{HV} \) is calculated. In other words, the minimum time to climb from any point \((H, V)\) to the end point is the minimum of
two possibilities: the time to climb $\Delta H$ plus the time to fly from $(H + \Delta H, V)$ to the end point, or the time to accelerate $\Delta V$ plus the time to fly from $(V + \Delta V, H)$ to the end point. Once matrix $(b)$ is completed, it is necessary only to proceed through it, starting at the initial point, and to select the smaller of the two possible adjacent elements to establish the optimal path. Again, as in the example, each element $b_{HV}$ represents the minimum time to fly from that element to the end point.

As a consequence of the formulation of matrix $(b)$ (a process requiring only the number of separate calculations indicated by Eq. (2)), the optimal path from any point $(H, V)$ to the end point and the time required along this path have been established. The treatment of the effects of airplane weight-change along the path and the effects of the climb path on airplane drag is presented in the next section.
III. EXAMPLE PROBLEM

To show numerical results of the application of dynamic programming to
the minimum time-to-climb problem, a hypothetical interceptor airplane was
selected capable of flight at a speed equivalent to 2.0 Mach number and an
altitude of 60,000 ft. Drag for this airplane can be represented by

\[ D = C_{D_0} q S + \frac{K N^2 w^2}{q S} , \]  \hspace{1cm} (11)

where

\[ q = 1481.5 \text{ ft}^2 , \]  \hspace{1cm} (12)

and

\[ M = \cos \theta + \frac{V^2}{g} \frac{d\theta}{dh} \sin \theta . \]  \hspace{1cm} (13)

\[ C_{D_0} \] and \[ K, \] the zero-lift drag coefficient and the drag-due-to-lift factor re-
spectively, are functions of Mach number, as shown in Figs. 1 and 2. The avail-
able thrust is a function of altitude and speed if throttle setting is fixed
and is shown in Fig. 3. The equivalence between Mach number and velocity is
given by

\[ M = \frac{V}{V_S} , \]  \hspace{1cm} (14)

where \[ V_S \] is a function of altitude only.

We proceed by forming matrix \[ \Delta H, \] for which we have chosen \[ \Delta H = 1000 \text{ ft} \]
and \[ \Delta M = .02. \] A starting point of \((H = 0 \text{ ft}, M = .8)\) and a final point of
\((H = 60,000 \text{ ft}, M = 2.0)\) result in a 61 x 61 element matrix. The time to
climb \(\Delta H\) at constant Mach number is given by

\[ \Delta t = \frac{\Delta H \left(1 + \frac{V}{w} \frac{\Delta V}{\Delta H}\right)}{\frac{V}{w} (T - D)} , \]  \hspace{1cm} (15)
and the time to accelerate $\Delta M$ at constant altitude is given by

$$
\Delta t = \frac{W \Delta V}{F (T - D)}
$$

(16)

For small $\Delta t$ the horizontal distance covered in both climbs and accelerations is computed as the product of $\Delta t$ and the average horizontal velocity component.

Using expressions (11)-(16), matrix $b_{HM}$ was developed by a high-speed digital computer and minimum-time paths were determined for several cases. The results are shown in Figs. 4-6. In Case A, the gross weight of the airplane $W$ was held constant at 40,000 lb and the load factor $N$ was assumed equal to 1.0. The time-to-climb along the optimal path for Case A is 277 sec. Figure 4 shows the altitude - Mach number trace of this path, while Fig. 5 shows the altitude - horizontal distance profile.

In Case B, the effect of changing gross weight along the flight path was examined by introducing the specific fuel consumption as a function of altitude and speed. Since we must move backwards through the matrix $b_{HM}$, the technique employed is an iterative process, starting with several assumed values of final gross weight and calculating the increase in gross weight as we move backward through the matrix. The interesting by-product of this technique is that we have available time-to-climb data for a variety of starting gross weights, as shown in Fig. 6. For a starting gross weight of 40,000 lb and a load factor of 1.0, time along the optimal path is 252 sec and the fuel used is 3452 lb. The altitude - Mach number trace and flight profile for Case B are shown in Figs. 4 and 5 for comparison with Case A.

For Case C, the effect of including the climb angle in the airplane drag ($N = \cos \Theta$) was investigated. The effect on drag of normal accelerations, the right-hand term in Eq. (13), was ignored. The technique for including $\cos \Theta$
in the drag is an iterative process similar to that used for a gross-weight change. As in Case B, a weight change was considered for Case C. The results are shown in Figs. 4 and 5. The time-to-climb for Case C for a starting gross weight of 40,000 lb and \( N = \cos \theta \) is 251 sec. Further refinement to include the effect of normal accelerations on drag could be introduced but is not felt to be justified in view of the small change in time between Cases B and C.

The results show that, for the example problem, the altitude-velocity trace and the altitude-distance profile of the optimal path are fairly insensitive to gross-weight changes and drag corrections, and that the minimum time is insensitive to drag corrections due to flight-path inclination. The time-to-climb is, however, seen to be quite sensitive to gross-weight changes.

Turning now to practical application of these results, it is clear that no pilot would be expected to follow complicated altitude-velocity climb schedules such as those in Fig. 4. Fortunately, it can be shown quite readily that any of the paths in Fig. 4, for example, can be closely approximated timewise by a path which employs two constant-Mach-number climbs and a constant-altitude acceleration only. Thus, the most useful purpose of the theoretical solution may well be to establish practical minimum time-to-climb paths.
IV. CONCLUSIONS

The following conclusions can be drawn from this study of the application of dynamic programming to the airplane minimum time-to-climb problem:

1. Dynamic programming offers a method of solution to the climb problem which is fast, is readily adaptable to routine engineering calculation, and allows the inclusion of the effects of weight changes and drag corrections along the flight path.

2. Practical climb schedules for minimum time from any starting point in the flight spectrum can be established by a single application of the dynamic-programming method.

3. Dynamic programming can also be employed to determine paths of least fuel consumption, and may be applicable to the general missile-guidance problem.
Fig. 2 — Drag due to lift factor
Fig. 3—Thrust available

Sea level
20,000 ft
40,000
60,000

0.5
1.0
1.5
2.0

Thrust available
Fig. 4—Altitude—Mach number trace of minimum-time paths
Fig. 5—Altitude–horizontal distance profile of minimum-time paths
Fig. 6—Effect of initial gross weight on minimum time-to-climb
REFERENCES


