Enhanced IVT Calculation Algorithms

Landon Sego, Brett Amidan, Tom Ferryman
Pacific Northwest National Laboratory

February 1, 2008

To improve the processing capability of the System Level Morning Report (SLMR), we revised the algorithms and the programming used to calculate the Intermediate Value Tables (IVTs). The revised code is contained and documented in the R library, *calcIVT* . Through a series of extensive QA checks, we demonstrated that the new algorithms run about 4.5 times faster and produce results that are equivalent to the original algorithms that were developed previously. A summary of the speed improvements, general improvements, and algorithmic changes for the continuous and discrete IVT matrices are summarized below.

Summary of speed improvements

Time (in seconds) to process 19 flights:

Type	Original	Improved	Times faster (Original / Improved)
Discrete alone	73.4	35	2
Continuous alone	393	67	5.8
Continuous and Discrete	421	94	4.5

These tests were run a Dell Precision Workstation with 3.2 GHz CPU and 3 GB RAM.

General improvements to IVT processing

1. Both the improved continuous and discrete IVT codes handle the data objects more efficiently than before.
2. Where it makes sense to do so, the R library calls compiled C code which greatly improves processing speed.
3. The improved discrete IVT code will now recognize if a transition occurs at a phase boundary. Previously, these transitions were not counted.
4. The new discrete IVT code will now recognize transitions that are separated by missing values: e.g. 0, 0, 0, 0, NA, 1, 1, 1, 1.
5. The new discrete IVT code allows the state labels to differ for every parameter. Previously, the state labels were assumed to be the same for all variables that had the same measurement frequency (Hz).
6. The improved continuous IVT code now correctly fits the quadratic moving window to parameters that are measured more than once a second. For these high-frequency parameters, windows of size ± 5 seconds are used, and all the data within that window are used to fit the quadratic regression model. For example, for an 8Hz parameter, the ± 5 data window actually consists of 81 data points.

The table below summarizes how variables measured with different frequencies are handled by the continuous IVT code:

Hz	Skip (seconds)	Skip (data points)	Bandwidth (seconds)	Bandwidth (data points)	Right side of the sequence of the linear predictor
8	1	8	5	40	0,0.125,...,5
4	1	4	5	20	0,0.25,...,5
1	1	1	5	5	0,1,..,5
0.5	2	1	10	5	0,2,...,10
0.25	4	1	20	5	0,4,...,20
skip.actual (in seconds) = max(1, 1/Hz)			1 second is the nominal skip		
bandwidth.actual (in seconds) = max(5, 5/Hz)			5 seconds is the nominal bandwidth		

Algorithm for the continuous flight parameters

The core of the continuous IVT calculation is accomplished using compiled C code. The C code fits the quadratic model to a moving window across each of the continuous flight parameter vectors. To simplify the solution of the least square estimators (and thus avoid complicated expressions and/or matrix inversion), the linear and quadratic predictors are centered so that they are both orthogonal to the intercept, and if possible, orthogonal to one another. The usual quadratic regression model can be written as:

$$y_i = a + bx_i + cx_i^2 + e_i \tag{1.1}$$

We can create an equivalent model by centering the linear and quadratic predictors as follows:

$$y_i = a' + b'x_i^* + c'x_i^{**} + e_i' \tag{1.2}$$

where

$$x_i^* = x_i - \frac{1}{n}\sum_{i=1}^{n} x_i \ , \quad x_i^{**} = \left(x_i^*\right)^2 - \frac{1}{n}\sum_{i=1}^{n}\left(x_i^*\right)^2 \ , \tag{1.3}$$

and n is the number of data points in the window. Note that $\sum_{i=1}^{n} x_i^* = 0$ and $\sum_{i=1}^{n} x_i^{**} = 0$, which implies that x_i^* and x_i^{**} are orthogonal to the intercept. Furthermore, if the original x_i are evenly spaced, then $\sum_{i=1}^{n} x_i^* x_i^{**} = 0$. (However, in the event there are missing data points in the window, the original x_i are not evenly spaced). Using (1.3) and some algebra, we can rewrite (1.2) as:

$$y_i = \left(a' - b'\overline{x} + c'\left(\overline{x}^2 - \overline{\overline{x}}\right)\right) + \left(b' - 2c'\overline{x}\right)x_i + c'x_i^2 + e_i' \tag{1.4}$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad \bar{\bar{x}} = \frac{1}{n}\sum_{i=1}^{n}\left(x_i^*\right)^2 \tag{1.5}$$

Using (1.4), we can equate the original parameters in (1.1) to those in (1.2):

$$\begin{aligned} a &= a' - b'\bar{x} + c'\left(\bar{x}^2 - \bar{\bar{x}}\right) \\ b &= b' - 2c'\bar{x} \\ c &= c' \\ e_i &= e_i' \ , \quad \forall i \end{aligned} \tag{1.6}$$

The advantage of the parameterization given in (1.2) is that it simplifies the computation of the least squares estimates of the model parameters. To begin with, suppose the x_i are evenly spaced so that x_i^* and x_i^{**} are orthogonal. It can be readily shown that the least squares estimates are then given by:

$$\begin{aligned} \hat{a}' &= \frac{1}{n}\sum_{i=1}^{n} y_i \\ \hat{b}' &= \sum_{i=1}^{n} x_i^* y_i \Big/ \sum_{i=1}^{n}\left(x_i^*\right)^2 \\ \hat{c}' &= \sum_{i=1}^{n} x_i^{**} y_i \Big/ \sum_{i=1}^{n}\left(x_i^{**}\right)^2 \end{aligned} \tag{1.7}$$

And the residual is calculated in the usual fashion:

$$e_i' = y_i - \hat{a}' - \hat{b}'x_i^* - \hat{c}'x_i^{**} \tag{1.8}$$

If x_i are not evenly spaced so that x_i^* and x_i^{**} are not orthogonal (which would occur if the window contained missing data), it can be shown that the least squares estimates are given by:

$$\begin{aligned} \hat{a}' &= \frac{1}{n}\sum_{i=1}^{n} y_i \\ \hat{b}' &= \frac{1}{D}\left[\left(\sum_{i=1}^{n}\left(x_i^{**}\right)^2\right)\left(\sum_{i=1}^{n} x_i^* y_i\right) - \left(\sum_{i=1}^{n} x_i^* x_i^{**}\right)\left(\sum_{i=1}^{n} x_i^{**} y_i\right)\right] \\ \hat{c}' &= \frac{1}{D}\left[\left(\sum_{i=1}^{n}\left(x_i^{*}\right)^2\right)\left(\sum_{i=1}^{n} x_i^{**} y_i\right) - \left(\sum_{i=1}^{n} x_i^* x_i^{**}\right)\left(\sum_{i=1}^{n} x_i^{*} y_i\right)\right] \end{aligned} \tag{1.9}$$

where

$$D = \left[\sum_{i=1}^{n}\left(x_i^*\right)^2\right]\left[\sum_{i=1}^{n}\left(x_i^{**}\right)^2\right] - \left[\sum_{i=1}^{n} x_i^* x_i^{**}\right]^2 \tag{1.10}$$

Note that (1.9) reduces to (1.7) when $\sum_{i=1}^{n} x_i^* x_i^{**} = 0$, i.e., when x_i^* and x_i^{**} are orthogonal. The enhanced IVT code fits the quadratic model shown in (1.1) by calculating the parameter estimates for each window using (1.7) or (1.9) and then "backtransforming" to the original parameterization using (1.6).

Algorithm for the discrete flight parameters

The discrete IVT contains a summary of the percentage of time that each discrete flight parameter spent in each state for each phase. It also contains the number of transitions that occurred within each phase between the various discrete states of each parameter. Calculating the percentage of time that is spent within each state is a relatively simple matter and will not be discussed here. The principal changes we made to the existing algorithm involve the methodology used to count the number transitions between the states.

The basic strategy was to create a mapping of the values of the discrete parameters whose first difference would have values that uniquely identify the various transitions. The approach is easiest to explain using a simple example:

Suppose we have a discrete vector that has four states labeled 0, 1, 2, and 3. We found that the mapping $m(x) = 2^x$ will produce unique differences among each of the states. That is, $m(x) - m(y)$ will be unique for every combination of $x \neq y$, where *x* and *y* correspond to state labels 0, 1, 2, …. The approach works for 2 to 55 unique states. Beyond 55, some combinations of $m(x) - m(y)$ begin to repeat due to what appears to be machine error. The transition matrix below shows the values of $m(x) - m(y)$ for each of the 16 possible ways to move from state *x* (on the rows) to state *y* (on the columns).

	0	**1**	**2**	**3**
0	0	-1	-3	-7
1	1	0	-2	-6
2	3	2	0	-4
3	7	6	4	0

The diagonal elements correspond to the situation where the discrete parameter value remains in the same state, which is why their matrix elements are zero. For purposes of this discussion, these diagonal elements are not of concern to us since they are summarized by the percentage of time that the discrete parameter remains in that particular state. Note that the off-diagonal elements of the matrix are unique. For example, if the discrete parameter transitions from state 3 to state 1, then that transition is uniquely identified in the matrix by $m(3) - m(1) = 6$. And the transition from state 1 to state 2 is represented by $m(1) - m(2) = -2$.

To count the number transitions from one state to another for a discrete parameter vector *Y*, the first (or successive) differences of *m*(*Y*) are tabulated by phase. The number of "6's" that occur in a given phase will indicate the number of times the parameter transitions from state 3 to state 1. Likewise, the number of "-2's" that occur in a given phase will indicate the number of times the parameter transitions from state 1 to state 2, etc.

This approach helps to minimize the number of passes that need to be made through the discrete parameter vector in order to count all the possible transitions.