ation **CK** ≤ **D**, where **C** is a matrix of constants, **K** is the fit coefficient vector, and **D** is a vector of limits. The matrix has dimensions N by M where N is the number of constraint expressions and M is the number of fit coefficients. In most cases, almost all the elements of the matrix are zeroes. In the previous example, the **C** matrix and **D** vector are:

C matrix					**D vector**
0	1	0	1	0	5
0	-1	0	0	0	-3.3
0	0	0	-1	0	-2.2

This is the form used internally by the curve-fitting code. If you wish, you can build a matrix wave and one-dimensional wave containing the *C* and *D* constants. Instead of using /C=*textwave* to request a fit with constraints, use `/C={matrix, 1Dwave}`.

In general, it is confusing and error-prone to create the right waves for a given set of constraints. However, it is not allowed to use the textwave method for curve fitting from a threadsafe function (see **Constraints and ThreadSafe Functions** on page III-250). Fortunately, Igor can create the necessary waves when it parses the expressions in a text wave.

You can create waves containing the **C** matrix and **D** vector using the /C flag (note that this flag goes right after the CurveFit command name, not at the end). If you have executed the examples above, you can now execute the following commands to build the waves and display them in a table:

```
CurveFit/C dblExp expData /D/R/C=CTextWave
Edit M_FitConstraint, W_FitConstraint
```

The **C** matrix is named M_FitConstraint and the **D** vector is named W_FitConstraint (by convention, matrix names start with "M_", and 1D wave names start with "W_").

In addition to their usefulness for specifying constraints in a threadsafe function, you can use these waves later to check the constraints. The following commands multiply the generated fit coefficient wave (W_coefs) by the constraint matrix and appends the result to the table made previously:

```
MatrixMultiply M_FitConstraint, W_coef
AppendToTable M_Product
```

The result of the MatrixMultiply operation is the matrix wave M_Product. Note that the values of M_Product[1] and M_Product[2] are larger than the corresponding values in W_FitConstraint because the constraints were infeasible and resulted in constraint violations.

M_F	M_F	M_F	M_F	M_F	W_FitConstraint	M_product[][0]
0	1	2	3	4		0
0	1	0	1	0	5	4.99998
0	-1	0	0	0	-3.3	-3.05782
0	0	0	-1	0	-2.2	-1.94217

Constrained Curve Fitting Pitfalls

Bad Values on the Constraint Boundary

The most likely problem with constrained curve fitting is a value on a constraint boundary that produces a singular matrix error during the curve fit. For instance, it would be reasonable in many applications to require a pre-exponential multiplier to be positive, in this case K1>0:

$$y = K_0 + K_1 e^{K_2 x}$$

It would be natural to write a constraint expression "$K_1 > 0$", but this can lead to problems. If some iteration tries a negative value of K_1, the constraints will set K_1 to be exactly zero. But when K1 is zero, the function has no dependence on K_2 and a singular matrix error results. The solution is to use a constraint that requires K_1 to be greater than some small positive number: "$K_1 > 0.1$", for instance. The value of the limit must be tailored to your application. If you expect the constraint to be inactive when the solution is found, and the fit reports that the constraint was active, you can decrease the limit and do the fit again.