Jordan's Canonical Form Just Doesn't Compute

Diagonalizing matrices can be dangerous to your digits

Matrices and differential equations are the key mathematical tools in MATLAB and SIMULINK. The Jordan Canonical Form is the key relationship between matrices and differential equations. So, why doesn't MATLAB use the JCF in any of its computations? In fact, until the Symbolic Math Toolbox came along, we didn't even have a function to compute the JCF.

The difficulty with the Jordan Canonical Form is that it is wildly sensitive to perturbation. Any kind of error—uncertainty in experimental data, arithmetic roundoff error, linearization of nonlinear functions—completely changes the JCF and, more importantly, the transformations which generate it.

Let's start with a system of n linear, constant coefficient ordinary differential equations.

$$x' = A\,x$$

Here $x(t)$ is a vector-valued function of t, the prime denotes differentiation with respect to t, and A is a given n-by-n matrix which is independent of t. If you've never heard of—or want to forget—the Jordan Canonical Form, then you have to hope there is a matrix V which diagonalizes A, that is $V^{-1}AV$ is a diagonal matrix Λ. Then the change of variables,

$$x = Vy$$

reduces the n-by-n problem to n instances of the 1-by-1 case,

$$y' = \Lambda y$$

The elements of the diagonal matrix Λ are the eigenvalues of A and the columns of V are the eigenvectors of A. The k-th component of the solution $y(t)$ is the exponential function of t determined by the eigenvalue λ_k and the initial value $y_k(0)$

$$y_k(t) = y_k(0)\ exp(\lambda_k t)$$

The components of the solution $x(t)$ of the original problem are then linear combinations of the exponential functions determined by the eigenvalues and eigenvectors and the initial values.

But this fantasy about diagonalizing matrices leaves out some very important differential equations, like the single, second order

$$u'' = 0$$

The solution, of course, is a straight line—a linear polynomial in t. The solution does *not* involve any exponentials. This equation can be converted to a 2-by-2, first-order system by taking x to be the vector with components

$$x_1 = u$$
$$x_2 = u'$$

Then the differential equation is

$$x_1' = x_2$$
$$x_2' = 0$$

This can be written in matrix form

$$x' = Jx$$

where J is the 2-by-2 matrix generated by the MATLAB statement

```
J = [ 0  1;  0  0 ]
```

This matrix *cannot* be diagonalized; no change of variable which preserves the differential equation will produce a zero in the upper right hand corner. That had better be the case, because the differential equation has polynomial, not exponential, solutions. The matrix J is the simplest example of a nondiagonal Jordan Canonical Form. The eigenvalues—in this case, a pair of zeros—are on the diagonal, but that crucial 1 off the diagonal couples the two differential equations.

A different change of variable leads to a matrix whose structure is not quite so obvious. Let

$$x_1 = u + u'$$
$$x_2 = u - u'$$

Then the equation becomes

$$x' = A\,x$$

where A is generated by the MATLAB statement

```
A = [ 1/2  -1/2;  1/2  -1/2 ]
```

This new change of variables really hasn't altered things very much. The Jordan Canonical Form of A is the matrix J from our first system. The differential equation still has solutions which are quadratic polynomials in t; it can't be transformed into a pair of decoupled equations with exponential solutions.

But now, let's perturb the matrix A very slightly to

```
A = [ 0.5 -0.49999999; 0.49999999 -0.5 ]

A =
   0.50000000000000     0.49999999000000

   0.49999999000000    -0.50000000000000
```

The eigenvalues are changed from a pair of zeros to

```
λ = eig(A)
λ =
1.0e-004 *

   1.0000
  -1.0000
```

A change of 10^{-8} in the matrix elements changes the eigenvalues by 10^{-4}. Now this perturbed matrix has distinct eigenvalues, and can be diagonalized. The eigenvector matrix is

```
[V,λ] = eig(A); V
V =
   0.70717748832907    0.70703606697295
   0.70703606697295    0.70717748832907
```

This eigenvector matrix V defines a change of variables that transforms the system into a decoupled system whose solutions involve *exp(t/10000)* and *exp(-t/10000)*. But we know the solution should be very close to a linear function of *t*. It is probably a lousy idea to try to represent the solution in terms of these two exponentials. This is reflected in the fact that *V* is badly conditioned.

```
cond(V) = 1.0000e+004
```

This matrix *A* is an example of why we can't use the Jordan Canonical Form for practical computation. Technically, *A* has a full set of linearly independent eigenvectors and a diagonal JCF. But any attempt to use this fact will be plagued by the condition number of *V* and a resulting loss of accuracy. On the other hand, if we try to say that *A* does not have a full set of eigenvectors and use a nondiagonal JCF, we are also making possibly significant errors.

Another example can be generated starting with MATLAB's test matrix,

```
A = gallery(5)

A =

      -9      11     -21      63    -252
      70     -69     141    -421    1684
    -575     575   -1149    3451  -13801
    3891   -3891    7782  -23345   93365
    1024   -1024    2048   -6144   24572
```

This matrix was constructed so that it's characteristic polynomial is

$$\lambda^5 = 0$$

All five eigenvalues are zero. The Jordan Canonical Form can be obtained from the Symbolic Math Toolbox which, in this case, does exact integer calculations with *no* roundoff error.

```
J = jordan(A)
J =
[0, 1, 0, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 1, 0]
[0, 0, 0, 0, 1]
[0, 0, 0, 0, 0]
```

With a well-conditioned change of variables, the equation

$$x' = Ax$$

essentially becomes

$$u'''' = 0$$

All solutions are cubic polynomials in *t*. So far, so good. But, now change the second diagonal element of *A* from *-69* to

```
A(2,2) = -70

A =

      -9      11     -21      63    -252
      70     -70     141    -421    1684
    -575     575   -1149    3451  -13801
    3891   -3891    7782  -23345   93365
    1024   -1024    2048   -6144   24572
```

I want to regard that as a significant change; I am not willing to approximate this new matrix by the unaltered `gallery(5)`. The characteristic polynomial is now

```
pretty(factor(charpoly(A,'l')))
               4       3        2
(λ - 1) (λ  + 2 λ  - 67 λ  + 234 λ - 168)
```

One of the eigenvalues is *1*. The other four are the distinct, irrational numbers which are the roots of an irreducible quartic. The Jordan Canonical Form is diagonal. We could ask the Symbolic Math Toolbox to compute it, but the result is the kind of intimidating expression I discussed in the last newsletter. Instead we can simply ask for

```
[V,λ] = eig(A); λ = diag(λ)

λ =

 -10.5726
   3.7953 + 1.3331i
   3.7953 - 1.3331i
   1.0000
   0.9820
```

The form of the solution to the differential equation has changed dramatically, from a cubic polynomial in *t*, to a function involving terms like *exp(-10.5726 t)* and *exp(3.7953 t) sin(1.3331 t)*. Moreover, the change of variables required to obtain this representation is badly conditioned.

```
cond(V) = 2.3890e+004
```

The fundamental difficulty is with matrices, like this last *A*, which are close to, but not exactly equal to, matrices with nondiagonal JCFs. If we try to diagonalize such matrices, then errors are magnified by the condition number of the eigenvector matrix, which can be arbitrarily large. On the other hand, to use the "nearby" nondiagonal JCF may also represent an unacceptably large error. We're damned if we do and damned if we don't.

The numerically reliable approach to all this is to avoid the Jordan Canonical Form altogether. The portions of MATLAB and SIMULINK that deal with matrix differential equations use other solution methods, including the Schur Form (which is triangular, instead of diagonal) and the matrix exponential (which is *not* computed from eigenvalues and eigenvectors). Maybe we'll have a chance to discuss such things in a future newsletter.

I can't resist finishing up with a personal story. Prof. G. W. "Pete" Stewart of the University of Maryland introduces speakers at conferences by composing original limericks for them. Several years ago, he honored me with:

Said Professor Cleve Moler one day,
I'd like to get e *to the* A.
  It's as easy as pie,
  To get e *to the* y,
But I can't get e *to the* J.

Thanks, Pete.