Remark on Algorithm 982: Explicit solutions of triangular systems of first-order linear initial-value ordinary differential equations with constant coefficients

W. VAN SNYDER

Algorithm 982: Explicit solutions of triangular systems of first-order linear initial-value ordinary differential equations with constant coefficients provides an explicit solution for an homogeneous system, and a brief description of how to compute a solution for the inhomogeneous case. The method described is not directly useful if the coefficient matrix is singular. This remark explains more completely how to compute the solution for the inhomogeneous case, and for the singular coefficient matrix case.

CCS Concepts: •**Mathematics of computing →Solvers; Ordinary differential equations;**

General Terms: Mathematics of computing, Mathematical software, Solvers, Mathematical analysis, Differential equations, Ordinary differential equations

Additional Key Words and Phrases: ordinary differential equations, initial value problem, first order, constant coefficient, triangular system, explicit solution

ACM Reference format:
W. Van Snyder. 2020. Remark on Algorithm 982: Explicit solutions of triangular systems of first-order linear initial-value ordinary differential equations with constant coefficients. *ACM Trans. Math. Softw.* 0, 0, Article 0 (2020), 4 pages.
DOI: 0000001.0000001

The formal solution for the homogeneous system

$$\frac{\mathrm{d}\mathbf{N}(t)}{\mathrm{d}t} = \mathbf{A}\,\mathbf{N}(t) \tag{1}$$

in which $\mathbf{A}$ is a constant triangular matrix, is

$$\mathbf{N}(t) = \exp(\mathbf{A}\,t)\,\mathbf{N}(0)\,. \tag{2}$$

Because this requires to compute the exponential of the matrix $\mathbf{A}\,t$, Algorithm 982 [1] provides an explicit solution

$$N_i(t) = N_i(0)\,e^{a_{ii}t} + \sum_{j=1}^{i-1} z_{ij}\left(N_j(t) - N_j(0)\,e^{a_{ii}t}\right) \tag{3}$$

where the constant coefficients z_{ij} are calculated using the recurrence

Author's e-mail address: van.snyder@sbcglobal.net.

DOI: 0000001.0000001

$$z_{ij} = 0 \text{ if } i \le j, \text{ and}$$
$$z_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-j-1} a_{i-k,j}\, z_{i,i-k}}{a_{jj} - a_{ii}} \text{ if } i > j\,. \tag{4}$$

This solution requires to compute the exponential of only the diagonal elements of $\mathbf{A}t$. This is, of course, equivalent to computing

$$\mathbf{N}(t) = \exp(\mathbf{A}\,t)\,\mathbf{N}(0) = \mathbf{M}\,\exp(\mathbf{\Lambda}\,t)\,\mathbf{M}^{-1}\,\mathbf{N}(0)\,, \tag{5}$$

where $\mathbf{M}$ is the modal matrix of $\mathbf{A}$, and $\mathbf{\Lambda}$ is the diagonal matrix consisting of the eigenvalues of $\mathbf{A}$, which are the diagonal elements of $\mathbf{A}$. Computing $\exp(\mathbf{\Lambda}\,t)$ requires to compute only the exponential of the diagonal elements of $\mathbf{A}t$. But this method does not require to compute and invert the modal matrix. It is well known, and obvious from Equation (4), that this method does not work if $\mathbf{A}$ is defective.

The solution given in [1] for an inhogeneous equation

$$\frac{\mathrm{d}\mathbf{N}(t)}{\mathrm{d}t} = \mathbf{A}(\mathbf{N}(t) + \mathbf{B}) \tag{6}$$

with $\mathbf{B}$ constant is

$$\mathbf{N}(t) = \exp(\mathbf{A}t)\,\hat{\mathbf{N}}(0) - \mathbf{B}\,, \tag{7}$$

where $\hat{\mathbf{N}}(0) = \mathbf{N}(0) + \mathbf{B}$. For a system of the form

$$\frac{\mathrm{d}\mathbf{N}(t)}{\mathrm{d}t} = \mathbf{A}\mathbf{N}(t) + \mathbf{C} \tag{8}$$

the general solution if $\mathbf{A}$ is nonsingular is

$$\mathbf{N}(t) = \exp(\mathbf{A}t)\,(\mathbf{N}(0) + \mathbf{A}^{-1}\mathbf{C}) - \mathbf{A}^{-1}\mathbf{C} \tag{9}$$

where $\mathbf{B}$ in Equations (6) and (7) has been replaced by $\mathbf{A}^{-1}\mathbf{C}$. But this is not useful if $\mathbf{A}$ is singular. Singular systems arise when modeling nuclear decay if nonradioactive daughter isotopes are included.

Assume that if $\mathbf{A}$ is singular, it has been permuted to a form that has a leading nonsingular triangle $\mathbf{A}_m$ with m rows and columns, i.e., $a_{ii} \neq 0$ for $i \le m$, and $a_{ii} = 0$ for $i > m$. The solutions for the first m equations $\mathbf{N}_m$ are given by the first m rows of Equation (7), *viz.*

$$\mathbf{N}_m(t) = \exp(\mathbf{A}_m t)\,\hat{\mathbf{N}}_m(0) - \mathbf{B}_m\,. \tag{10}$$

where $\mathbf{B}_m = \mathbf{A}_m^{-1}\mathbf{C}_m$.

The first term in Equation (10) is of the same form as the formal solution given by Equation (2), and is therefore given by the first m solutions in Equation (3), with $\mathbf{N}_m(t)$ replaced by $\hat{\mathbf{N}}_m(t)$. Equation (3) becomes

$$N_i(t) = \hat{N}_i(0)\,e^{a_{ii}t} + \sum_{j=1}^{i-1} z_{ij}\left(\hat{N}_j(t) - \hat{N}_j(0)\,e^{a_{ii}t}\right) - B_i\,,\ i \le m\,. \tag{11}$$

If $\mathbf{A}$ is singular, remaining equations are of the form

$$\frac{\mathrm{d}N_k(t)}{\mathrm{d}t} = \sum_{l=1}^{k-1} a_{kl}\, N_l(t) + C_k\,,\ k > m\,, \tag{12}$$

for which the solutions are simply

$$N_k(t) = \sum_{l=1}^{k-1} a_{kl} \int_0^t N_l(t)\,\mathrm{d}t + C_k\, t + N_k(0)\,,\ k > m\,, \tag{13}$$

because the right-hand sides do not depend upon $N_k(t)$. The $C_k\, t$ term in Equation (13) does not appear in Equation (11). Therefore, it is clear that Equation (11) does not produce a correct solution for $k > m$.

From Equation (11),

$$\int_0^t N_l(t)\,\mathrm{d}t = \hat{N}_l(0)\,\frac{e^{a_{ll}t} - 1}{a_{ll}} + \sum_{j=1}^{l-1} z_{lj} \left(\int_0^t \hat{N}_j(t)\,\mathrm{d}t - \hat{N}_j(0)\,\frac{e^{a_{ll}t} - 1}{a_{ll}} \right) - B_l\, t\,,\ l \le m\,. \tag{14}$$

Algorithm 982 was motivated by study of nuclear decay. If isotope j does not decay, it cannot contribute to accumulation of isotope i. Therefore, if $a_{jj} = 0$, then $a_{ij} = 0$ for $i > j$, it is not necessary to compute $\int_0^t N_l(t)\,\mathrm{d}t$ for $l > m$, and the upper bound for the summations in Equations (12) and (13) is m instead of $k - 1$.

In equations where $a_{jj} = 0$ and $a_{ij} \neq 0$ for $i > j$, integrals of Equation (13) are required in the right-hand side of Equation (13). This results in multi-dimensional quadratures, which can be reduced to one dimension using integration by parts. The provided software does not compute solutions for this case.

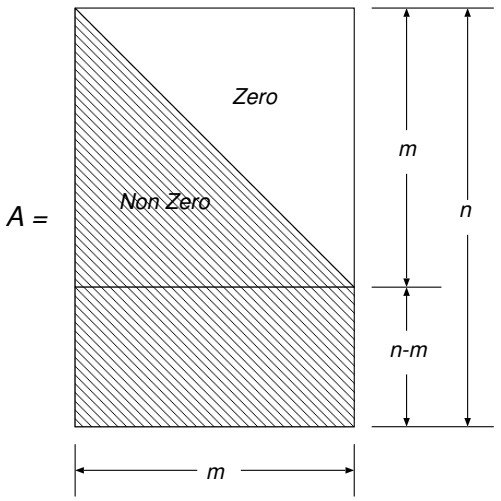

Fig. 1. Shape of nondefective matrix for which this method works

The `Compute_Solution` subroutine provided with Algorithm 982 has been revised to allow to provide **B**:

```
subroutine Compute_Solution ( T, A, Z, N0, N_T, B )
  real(rk), intent(in) :: T         ! Independent variable
  real(rk), intent(in) :: A(:,:)    ! Coefficient matrix for ODE
  real(rk), intent(in) :: Z(:,:)    ! Coefficient matrix solutions
  real(rk), intent(in) :: N0(:)     ! Initial condition
  real(rk), intent(out) :: N_T(:)   ! Solution N(T)
  real(rk), intent(in), optional :: B(:) ! For inhomogeneous equations.
    ! Let M be the index in A of the last nonzero diagonal element.
    ! Then B(1:m) is B in d N(t)/d t = A ( N(t) + B )
    ! and B(m+1:) is C in d N(t)/d t = A N(t) + C. Remember that here,
    ! columns of A after M are zero, so this is just a quadrature.
```

```
    ! Backsolve_A_C_B can compute B(1:m) if all you have is C.
  end subroutine Compute_Solution
```

Because the new argument B is optional, the revised procedure can be used in software that solves the homogeneous case by simply omitting the B argument, and used in existing software that used the original version, without adding the B argument.

As a convenience, a generic subroutine is included to solve $A_m B_m = C_m$:

```
  subroutine Solve_A_C_B ( A, C, B, Status )
    real(rk), intent(in) :: A(:,:) ! Lower triangular coefficient matrix
    real(rk), intent(in) :: C(:)   ! Right hand side of A B = C
    real(rk), intent(out) :: B(:)  ! Solution of A_m B_m = C_m, where M is the
                                   ! index of the last nonzero diagonal in A
    integer, intent(out) :: Status ! M, the index of the last nonzero diagonal
                                   ! in A, if there are no nonzeroes after M,
                                   ! else the negative of the index of the
                                   ! first column after M containing any
                                   ! nonzero element.
  end subroutine Solve_A_C_B
```

The named constant RK has the value KIND(0.0d0) in the double precision specific subroutines, and KIND(0.0e0) in the single precision specific subroutines.

REFERENCES

[1] Snyder, W. V. 2017. Algorithm 982: Explicit solutions of triangular systems of first-order linear initial-value ordinary differential equations with constant coefficients. *ACM Transactions on Mathematical Software 44,* 2, Article No,: 19.

Received December 2020