SIAM REV.
Vol. 40, No. 4, pp. 947–956, December 1998

013

JORDAN NORMAL FORM VIA ELEMENTARY TRANSFORMATIONS*

A. BUJOSA†, R. CRIADO‡, AND C. VEGA†

Abstract. This paper presents a method based on elementary transformations which may be applied to a matrix $\mathbf{A}$, whose characteristic polynomial has been decomposed into linear factors, in order to obtain a nonsingular matrix $\mathbf{P}$ such that $\mathbf{P^{-1}AP}$ is in Jordan normal form. This method can be used in the classroom, among other problems, to directly solve a linear ODE with constant coefficients. We also present a symbolic Maple program implementing the method.

Key words. Jordan normal form, elementary transformations

AMS subject classification. 15A21

PII. S0036144597329346

1. Introduction. In the applications of the Jordan matrix to engineering, it is important to determine the Jordan form of a given matrix $\mathbf{A}$, i.e., to determine a matrix $\mathbf{P}$ such that $\mathbf{P^{-1}AP}$ is in the Jordan normal form. Nevertheless, in mathematics books where the Jordan matrix is used, the manner of building the matrix $\mathbf{P}$ is not well described (see [1], [8], or [9]).

This fact is not due to scientific or conceptual difficulties but is due to didactic difficulties (the complete exposition and its justification is too long). So, if the characteristic polynomial has multiple roots, apparently it is preferable to use the triangular form of the given matrix (see [2]). Another possible reason is that if the roots are close, the computer can not distinguish if they are equal or not.

The objective of this article is to give a symbolic algorithm such that from a matrix $\mathbf{A}$ and its eigenvalues, we build a matrix $\left(\frac{\mathbf{J}}{\mathbf{P}}\right)$ where $\mathbf{J}$ is the Jordan matrix of $\mathbf{A}$ and $\mathbf{P}$ is the matrix of the change of basis. Furthermore, this algorithm presents the following advantages:

1. It requires substantially fewer operations than other methods.
2. Its justification, which requires little mathematical framework, constitutes a constructive proof of the existence of the Jordan normal form.

Also, we give a Maple program that implements this algorithm. It is important to recall that although the ideas involved in this algorithm are conceptually clear, there does not exist a similar method presented in text books.

Also, it is important to stress that this method is only useful in terms of symbolic computing. For a comprehensive discussion of problems related to the numerical computation of the Jordan normal form (clustering of eigenvalues and extraction of Jordan structure using the staircase algorithm) see [4], [6], [7], [5], and the standard text book in matrix computations [3].

2. The method. In this method the Jordan normal matrix is built through the successive application of elementary operations. To assure that such elementary

*Received by the editors February 17, 1997; accepted for publication (in revised form) October 9, 1997.
http://www.siam.org/journals/sirev/40-4/32934.html
†Departamento de Matemáticas, ETSI Telecomunicación, Ciudad Universitaria, 28040 Madrid, Spain (abujosa@mat.upm.es, cvega@mat.upm.es).
‡Departamento de Informática y Matemática, Universitaria Rey Juan Carlos ESCET, Móstoles, Spain (r.criado@escet.urjc.es).

operations preserve, similarity is necessary to compensate each elementary row operation with its corresponding inverse elementary column operation. We show this correspondence below.

$\mathbf{A \cdot P}$	$\mathbf{P^{-1} \cdot A}$
$col_i \leftarrow col_i + \alpha col_j$	$row_j \leftarrow row_j - \alpha row_i$
$col_i \leftarrow \alpha col_i$	$row_i \leftarrow \alpha^{-1} row_i$
$col_i \leftrightarrow col_j$	$row_i \leftrightarrow row_j$

.

Furthermore, elementary column operations allow us to build a matrix of change of basis, as follows:

$$\left(\frac{\mathbf{A}}{\mathbf{I}}\right) \rightarrow \boxed{\text{Applying the same elementary column operations to both matrices.}} \rightarrow \left(\frac{\mathbf{A \cdot P}}{\mathbf{P}}\right) \rightarrow \boxed{\text{Applying the elementary row operations only to the top matrix.}}$$

$$\rightarrow \left(\frac{\mathbf{P^{-1} \cdot A \cdot P}}{\mathbf{P}}\right).$$

The algorithm has the following operational plan:

$$\left(\begin{array}{c|c} \mathbf{J_{n-1}} & \\ \hline 0 & \mathbf{R_{n-1}} \end{array}\right) \xrightarrow{\text{step n}} \left(\begin{array}{c|c} \mathbf{J_n} & \mathbf{R_n} \\ \hline 0 & \end{array}\right).$$

In step n a matrix similar to $\mathbf{A}$ is generated in which the first n columns contain a Jordan matrix. Now, we show how to accomplish this step. Suppose that initially the algorithm began with a matrix $\mathbf{A} \in Mat_m(\mathcal{C})$ whose characteristic polynomial is $(X - \lambda_1)^{\mu(\lambda_1)} \cdot \ldots \cdot (X - \lambda_r)^{\mu(\lambda_r)}$ and that the first $n - 1$ steps have generated the following matrix similar to $\mathbf{A}$:

$$\mathbf{A_n} = \left(\begin{array}{c|c|c} \mathbf{J}_{\lambda_1} & & \\ \hline & \boxed{\mathbf{J}_{\lambda_2}} & \\ \hline & & \mathbf{T} \\ \hline & \boxed{\mathbf{J}_{\lambda}} & \mathbf{M} \\ \hline & & \mathbf{B} \end{array}\right),$$

where $\mathbf{J}_{\lambda_i}$ denotes a Jordan matrix associated with the eigenvalue λ_i. Also, suppose $\lambda_1, \lambda_2, \ldots, \lambda$ are distinct complex numbers and that the order of $\mathbf{J}_\lambda$ is k where $k < \mu(\lambda)$. Now, we subtract $\lambda\mathbf{I}$ from the matrix $\mathbf{A_n}$ and get the following:

$$\mathbf{A_n} - \lambda\mathbf{I} = \left(\begin{array}{c|c|c} \mathbf{J}_{\lambda_1 - \lambda} & & \\ \hline & \boxed{\mathbf{J}_{\lambda_2 - \lambda}} & \\ \hline & & \mathbf{T} \\ \hline & \boxed{\mathbf{J_0}} & \mathbf{M} \\ \hline & & \mathbf{B} - \lambda\mathbf{I} \end{array}\right).$$

Each step is formed by six stages.

2.1. Stage 1: To make null the first column of the matrix $\mathbf{B} - \lambda\mathbf{I}$. Since λ is a root of the characteristic polynomial of the matrix $\mathbf{B}$, the matrix $\mathbf{B} - \lambda\mathbf{I}$ is singular. Therefore, it is possible to make null the first column of the matrix $\mathbf{B} - \lambda\mathbf{I}$ accomplishing elementary operations of $m - n + 1$ last columns, i.e., through elementary operations of type $col_i \leftarrow col_i + \alpha col_j$, $col_i \leftarrow \alpha col_i$, and $col_i \leftrightarrow col_j$, where $\boxed{n \leq i, j \leq m}$. Moreover, as the corresponding inverse operations are of type $row_j \leftarrow row_j - \alpha row_i$, $row_i \leftarrow \alpha^{-1} row_i$, and $row_i \leftrightarrow row_j$, where $\boxed{n \leq i, j \leq m}$, we will have built a matrix similar to $\mathbf{A_n} - \lambda\mathbf{I}$ with the following form:

$$\mathbf{Q}^{-1}(\mathbf{A_n} - \lambda\mathbf{I})\mathbf{Q} = \begin{pmatrix} \mathbf{J}_{\lambda_1-\lambda} & & & \mathbf{T}' \\ & \boxed{\mathbf{J}_{\lambda_2-\lambda}} & & \\ & & \ddots & \\ & & \boxed{\mathbf{J_0}} & \mathbf{M}' \\ & & & \mathbf{B}' \end{pmatrix}$$

$$\text{where } \mathbf{B}' = \begin{pmatrix} 0 & b_{1,2} & \cdots & b_{1,s} \\ \vdots & \vdots & & \vdots \\ 0 & b_{s,2} & \cdots & b_{s,s} \end{pmatrix}$$

and $s = m + 1 - n$.

```
stage_1 := proc(S,B)
  local S0,dim_S,i,pivot,col,coef,inverse;

  S0 := S;
  dim_S := linalg[coldim](S0);
  inverse :=array(identity,1 .. linalg[rowdim](S0),1 .. linalg[rowdim](S0));

  for col from B to dim_S do
    for pivot from B to dim_S while S0[pivot,col] = 0 do   od;
    if pivot <= dim_S then
      for i from col+1 to dim_S do
           if S0[pivot,i] = 0 then next fi;
           coef := -S0[pivot,i]/S0[pivot,col];
           S0 := linalg[addcol](S0,col,i,coef);
           inverse := linalg[addrow](inverse,i,col,-coef)
      od
    else
      S0 := linalg[swapcol](S0,B,col);
      inverse := linalg[swaprow](inverse,B,col);
      S0 := linalg[multiply](inverse,S0);
      RETURN(evalm(S0))
    fi
  od;
  RETURN('error')
  end
```

2.2. Stage 2: To make null the first column of the matrix $\mathbf{T}'$. Now, as $\lambda_1 \neq \lambda, \lambda_2 \neq \lambda, \ldots$ it is possible to make null the coefficients of the first column of $\mathbf{T}'$ through elementary transformations of type $col_n \leftarrow col_n + \alpha col_i$, where $\boxed{1 \leq i < n - k}$ (recall that k is the order of $\mathbf{J_0}$). Furthermore, as the inverse operations have the form $row_i \leftarrow row_i - \alpha row_n$, where $\boxed{1 \leq i < n - k}$, we have built a matrix similar to $\mathbf{A_n} - \lambda\mathbf{I}$ with the following form:

$$\mathbf{Q'}^{-1}(\mathbf{A_n} - \lambda\mathbf{I})\mathbf{Q'} = \begin{pmatrix} \mathbf{J}_{\lambda_1-\lambda} & & & & \\ & \mathbf{J}_{\lambda_2-\lambda} & & & \mathbf{T''} \\ & & \ddots & & \\ & & & \mathbf{J_0} & \mathbf{M'} \\ & & & & \mathbf{B'} \end{pmatrix}$$

$$\text{where } \mathbf{T''} = \begin{pmatrix} 0 & t_{1,2} & \cdots & t_{1,s} \\ \vdots & \vdots & & \vdots \\ 0 & t_{r,2} & \cdots & t_{r,s} \end{pmatrix}$$

and $r = n - k$.

```
stage_2 := proc(S,T,J0)

      local S0,i,coef;
      S0 := S;
      for i from J0-1 by -1 to 1 do
          if S0[i,T] = 0 then next fi;
          coef := -S0[i,T]/S0[i,i];
           S0 := linalg[addcol](S0,i,T,coef);
           S0 := linalg[addrow](S0,T,i,-coef)
      od;
      RETURN(evalm(S0))
      end
```

Now we consider the following submatrix formed by $\mathbf{J_0}$ and the first column of $\mathbf{M'}$ and the row immediately above:

$$row_r \rightarrow \begin{pmatrix} \begin{matrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ \mathbf{eJ_1} & & & 0 \end{matrix} & & & \begin{matrix} m_{r+1} \\ \vdots \\ m_{r+p} \\ \mu_1 \end{matrix} \\ & \ddots & & \vdots \\ & & \begin{matrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ \mathbf{eJ_h} & & & 0 \end{matrix} & \begin{matrix} m_{r+p'} \\ \vdots \\ m_{r+p''} \\ \mu_h \end{matrix} \\ row_n \rightarrow \begin{matrix} 0 & 0 \end{matrix} & \cdots & \begin{matrix} 0 & 0 \end{matrix} & \begin{matrix} 0 \end{matrix} \end{pmatrix},$$

where $\mathbf{eJ_1}, \ldots, \mathbf{eJ_h}$ are elementary Jordan matrices.

2.3. Stage 3: To make null the coefficients m_{r+i}. To annihilate the coefficient m_{r+i} it is enough to apply the following elementary operations: $col_n \leftarrow col_n - m_{r+i} \cdot col_{r+i+1}$ and $row_{r+i+1} \leftarrow row_{r+i+1} + m_{r+i} \cdot row_n$.

```
stage_3 := proc(S,M,J0)

      local S0,i,coef;
      S0 := S;
      for i from J0 to M-2 do
          if S0[i,i+1] = 0 or S0[i,M] = 0 then next fi;
          coef := -S0[i,M];
          S0 := linalg[addcol](S0,i+1,M,coef);
          S0 := linalg[addrow](S0,M,i+1,-coef)
      od;
      RETURN(evalm(S0))
      end
```

2.4. Stage 4: To make units the nonnull coefficients μ_i. Suppose that $\mu_i \neq 0$ and that its associated elementary Jordan block starts at the p row and ends at the q row. Then to make the coefficient μ_i equal to one it is enough to accomplish the following succession of elementary operations: $row_q \leftarrow \mu_i^{-1} \cdot row_q$, $row_{q-1} \leftarrow \mu_i^{-1} \cdot row_{q-1}$, ..., $row_p \leftarrow \mu_i^{-1} \cdot row_p$, and then $col_q \leftarrow \mu_i \cdot col_q$, $col_{q-1} \leftarrow \mu_i \cdot col_{q-1}$, ..., $col_p \leftarrow \mu_i \cdot col_p$.

```
stage_4 := proc(S,M,J0)

      local S0,i,coef;
      S0 := S;
      for i from M-1 by -1 to J0 do
           if S0[i,i+1] = 0 or i = M-1 then coef := S0[i,M] fi;
           if coef = 0 then next fi;
           S0 := linalg[mulrow](S0,i,1/coef);
           S0 := linalg[mulcol](S0,i,coef)
      od;
      RETURN(evalm(S0))
      end
```

2.5. Stage 5: To get at most a single unit. We suppose that $\mu_i = \mu_j = 1$ and that their associated elementary Jordan blocks start, respectively, at p_i and p_j rows, and end, respectively, at q_i and q_j rows. We can suppose without loss of generality that $q_i - p_i \leq q_j - p_j$. Then to annul μ_i it is enough to accomplish the following succession of elementary operations: $row_{q_i} \leftarrow row_{q_i} - row_{q_j}$, $row_{q_i-1} \leftarrow row_{q_j-1} - row_{q_j-1}$, ..., $row_{p_i} \leftarrow row_{p_i} - row_{q_j-(q_i-p_i)}$, and then $col_{q_j} \leftarrow col_{q_j} + col_{q_i}$, $col_{q_j-1} \leftarrow col_{q_j-1} + col_{q_i-1}$, ..., $col_{q_j-(q_i-p_i)} \leftarrow col_{q_j-(q_i-p_i)} + col_{p_i}$.

```
stage_5 := proc(S,M,J0)

    local S0,i,k,box,dim_box,coef;
    S0 := S;
    dim_box := 0;
    for i from M-1 by -1 to J0 do
         if S0[i,i+1] = 0 or i = M-1 then
              if S0[i,M] = 1 then k := i else k := J0 fi
         fi;
         if dim_box < k-i+1 then box := k; dim_box := k-i+1 fi
    od;
    if dim_box = 0 then RETURN(evalm(S0)) fi;
    for i from M-1 by -1 to J0 do
         if S0[i,i+1] = 0 or i = M-1 then
              if not i = box and S0[i,M] = 1 then coef := -1; k := i else coef := 0 fi
         fi;
         if coef <> 0 then
              S0 := linalg[addrow](S0,box-k+i,i,coef);
              S0 := linalg[addcol](S0,i,box-k+i,-coef)
         fi
    od;
    RETURN(evalm(S0))
    end
```

2.6. Stage 6: To organize the enlarged Jordan matrix $\mathbf{J_0}$. It is enough to exchange rows such that the Jordan block whose $\mu_i = 1$ (if it occurs) remains at the bottom, and then to do the same column exchange.

```
stage_6 := proc(S,M,J0)

    local S0,i,k,box;
    S0 := S;
    box := M;
    for i from M-1 by -1 to J0 do  if S0[i,M] = 1 then box := i fi od;
    if M-1 <= box then RETURN(evalm(S0)) fi;
    for i from M by -1 to box+2 do
         S0 := linalg[swapcol](S0,i,i-1); S0 := linalg[swaprow](S0,i,i-1)
```

```
    od;
    RETURN(evalm(S0))
    end
```

Finally we add $\lambda\mathbf{I}$ to the final matrix to obtain the matrix $\mathbf{A_{n+1}}$. Therefore, the main program which obtains the Jordan matrix of a given matrix $\mathbf{A}$ is the following:

```
Jordan := proc(A,roots)

    local root,step,J0,dim_A,A0,A1;
    dim_A := linalg[coldim](A);
    A0 := linalg[stack](A,array(identity,1 .. dim_A,1 .. dim_A));

    step := 1;
    for root in roots do
        J0 := step;
        A0 := evalm(A0-root*array(identity,1 .. 2*dim_A,1 .. dim_A));
        A1 := stage_1(A0,step);
        while A1 <> 'error' do
            A0 := stage_2(A1,step,J0);
            A0 := stage_3(A0,step,J0);
            A0 := stage_4(A0,step,J0);
            A0 := stage_5(A0,step,J0);
            A0 := stage_6(A0,step,J0);
            step := step+1;

            A1 := stage_1(A0,step)
         od;
         A0 := evalm(A0+root*array(identity,1 .. 2*dim_A,1 .. dim_A))
     od;
     RETURN(evalm(A0))
     end
```

3. Example. Consider the matrix

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & -2 & 0 & 1 \\ -1 & 1 & 3 & -4 & 0 & 2 \\ -1 & 0 & 4 & -5 & 0 & 3 \\ -1 & 0 & 3 & -5 & 0 & 4 \\ -1 & 0 & 3 & -7 & 1 & 5 \\ -1 & 0 & 3 & -7 & -1 & 7 \end{pmatrix}$$

whose characteristic polynomial is $(X-1)^4(X-2)^2$. Then

$$\begin{array}{c} \mathbf{A}-\mathbf{I} \\ \left(\begin{array}{cccccc} -1 & 1 & 1 & -2 & 0 & 1 \\ -1 & 0 & 3 & -4 & 0 & 2 \\ -1 & 0 & 3 & -5 & 0 & 3 \\ -1 & 0 & 3 & -6 & 0 & 4 \\ -1 & 0 & 3 & -7 & 0 & 5 \\ -1 & 0 & 3 & -7 & -1 & 6 \\ \hline 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{array}\right) \end{array} \quad \begin{array}{l} \longrightarrow \\ \text{Step 1 : stage 1} \end{array}$$

$$\mathbf{P^{-1}(A-I)P}$$

$$\left(\begin{array}{cccccc}
0 & 0 & -1 & -1 & 0 & 1 \\
0 & 0 & 0 & 2 & 4 & -6 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & -2 & 0 & 2 \\
0 & 0 & -1 & -3 & 0 & 3 \\
0 & 0 & -1 & -3 & -1 & 4 \\
\hline
3 & 0 & 1 & -4 & 0 & 2 \\
2 & 1 & 0 & -3 & -2 & 4 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)
\xrightarrow[\text{Step 2 : stage 1}]{}$$

$$\mathbf{P^{-1}(A-I)P}$$

$$\left(\begin{array}{cccccc}
0 & 0 & -1 & 1 & -2 & 0 \\
 & 0 & -1 & -1 & 1 & 0 \\
 & 0 & 0 & 2 & -2 & 1 \\
 & 0 & 0 & 1 & 1 & 0 \\
 & 0 & 0 & 0 & 1 & 0 \\
 & 0 & 0 & 2 & 0 & 0 \\
\hline
3 & -2 & 1 & -6 & 12 & 0 \\
2 & -1 & 0 & -3 & 4 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & -2 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0
\end{array}\right)
\xrightarrow[\text{Step 3 : stage 4}]{}$$

$$\mathbf{P^{-1}(A-I)P}$$

$$\left(\begin{array}{cccccc}
0 & 0 & 1 & -1 & 2 & 0 \\
 & 0 & 1 & 1 & -1 & 0 \\
 & & 0 & 2 & -2 & 1 \\
 & & 0 & 1 & 1 & 0 \\
 & & 0 & 0 & 1 & 0 \\
 & & 0 & 2 & 0 & 0 \\
\hline
-3 & 2 & 1 & -6 & 12 & 0 \\
-2 & 1 & 0 & -3 & 4 & 1 \\
-1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & -2 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 \\
0 & -1 & 0 & 0 & 0 & 0
\end{array}\right)
\xrightarrow[\text{Step 3 : stage 5}]{}$$

$$\mathbf{P^{-1}(A-I)P}$$

$$\left(\begin{array}{cccccc} 0 & 0 & 0 & -2 & 3 & 0 \\ & 0 & 1 & 1 & -1 & 0 \\ & & 0 & 2 & -2 & 1 \\ & & & 1 & 1 & 0 \\ & & & 0 & 1 & 0 \\ & & & 2 & 0 & 0 \\ \hline -3 & -1 & 1 & -6 & 12 & 0 \\ -2 & -1 & 0 & -3 & 4 & 1 \\ -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & -2 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \end{array}\right) \xrightarrow[\text{Steps } 4,5,6+2\mathbf{I}]{} \left(\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ & 1 & 1 & 0 & 0 & 0 \\ & & 1 & 1 & 0 & 0 \\ & & & 1 & 0 & 0 \\ & & & & 2 & 1 \\ & & & & 0 & 2 \\ \hline -3 & -1 & 1 & 0 & -1 & 3 \\ -2 & -1 & 0 & 1 & -2 & 6 \\ -1 & -1 & 0 & 0 & -3 & 9 \\ 0 & -1 & 0 & 0 & -4 & 12 \\ 0 & -1 & 0 & 0 & -5 & 15 \\ 0 & -1 & 0 & 0 & -5 & 14 \end{array}\right).$$

$-\mathbf{I}$		
Step1	**Step2**	**Step3**
$C_3 \leftarrow C_3 + 3C_1$	$C_4 \leftarrow C_4 - 2C_3$	$C_2 \leftarrow -1C_2$
$R_1 \leftarrow R_1 - 3R_3$	$R_3 \leftarrow R_3 + 2R_4$	$R_2 \leftarrow -1R_2$
$C_4 \leftarrow C_4 - 4C_1$	$C_6 \leftarrow C_6 + 2C_3$	$C_1 \leftarrow -1C_1$
$R_1 \leftarrow R_1 + 4R_4$	$R_3 \leftarrow R_3 - 2R_6$	$R_1 \leftarrow -1R_1$
$C_6 \leftarrow C_6 + 2C_1$	$C_5 \leftarrow C_5 - 2C_4$	$C_2 \leftarrow C_2 + 1C_1$
$R_1 \leftarrow R_1 - 2R_6$	$R_4 \leftarrow R_4 + 2R_5$	$R_1 \leftarrow R_1 - 1R_2$
$C_3 \leftarrow C_3 + 2C_2$	$C_6 \leftarrow C_6 + 3C_4$	
$R_2 \leftarrow R_2 - 2R_3$	$R_4 \leftarrow R_4 - 3R_6$	
$C_4 \leftarrow C_4 - 3C_2$	$C_6 \leftarrow C_6 + 1C_5$	
$R_2 \leftarrow R_2 + 3R_4$	$R_5 \leftarrow R_5 - 1R_6$	
$C_5 \leftarrow C_5 - 2C_2$	$C_2 \leftrightarrow C_6$	
$R_2 \leftarrow R_2 - 2R_5$	$R_2 \leftrightarrow R_6$	
$C_6 \leftarrow C_6 + 4C_2$		
$R_2 \leftarrow R_2 - 4R_6$		
$C_1 \leftrightarrow C_3$		
$R_1 \leftrightarrow R_3$		

	$+\mathbf{I}, -\mathbf{2I}$	
Step4	**Step5**	**Step6**
$C_4 \leftrightarrow C_6$	$C_5 \leftrightarrow C_6$	$C_6 \leftarrow C_6 - 2C_4$
$R_4 \leftrightarrow R_6$	$R_5 \leftrightarrow R_6$	$R_4 \leftarrow R_4 + 2R_6$
	$C_5 \leftarrow C_5 + 2C_4$	$C_6 \leftarrow C_6 - 8C_3$
	$R_4 \leftarrow R_4 - 2R_5$	$R_3 \leftarrow R_3 + 8R_6$
	$C_5 \leftarrow C_5 + 4C_3$	$C_6 \leftarrow C_6 - 14C_2$
	$R_3 \leftarrow R_3 - 4R_5$	$R_2 \leftarrow R_2 + 14R_6$
	$C_5 \leftarrow C_5 + 5C_2$	$C_6 \leftarrow C_6 + 5C_1$
	$R_2 \leftarrow R_2 - 5R_5$	$R_1 \leftarrow R_1 - 5R_6$
	$C_5 \leftarrow C_5 - 2C_1$	$+\mathbf{2I}$
	$R_1 \leftarrow R_1 + 2R_5$	

4. Remark on the solution of a linear ODE with constant coefficients. Now, if A is the matrix of the preceding paragraph, we can use the obtained matrix P to obtain the solutions of the linear system of ODEs as follows:

$$\frac{d\mathbf{X}}{dt} = \mathbf{AX}.$$

So, if $\mathbf{H_j}$ is the jth column of the matrix $\mathbf{P}$, we have

$(\mathbf{A}-\mathbf{I})\mathbf{H_1}=\mathbf{0}$		
$(\mathbf{A}-\mathbf{I})\mathbf{H_2}=\mathbf{0}$	$(\mathbf{A}-\mathbf{I})\mathbf{H_3}=\mathbf{H_2}$	$(\mathbf{A}-\mathbf{I})\mathbf{H_4}=\mathbf{H_3}$
$(\mathbf{A}-2\mathbf{I})\mathbf{H_5}=\mathbf{0}$	$(\mathbf{A}-2\mathbf{I})\mathbf{H_6}=\mathbf{H_5}$	

and we can write the solutions of $\frac{d\mathbf{X}}{dt} = \mathbf{AX}$ as follows:

$$\mathbf{X} = e^t\left(c_1\mathbf{H_1} + c_2\mathbf{H_2} + c_3\left[\mathbf{H_3} + \frac{t}{1!}\mathbf{H_2}\right] + c_4\left[\mathbf{H_4} + \frac{t}{1!}\mathbf{H_3} + \frac{t^2}{2!}\mathbf{H_2}\right]\right)$$

$$+e^{2t}\left(c_5\mathbf{H_5} + c_6\left[\mathbf{H_6} + \frac{t}{1!}\mathbf{H_5}\right]\right),$$

where

$$\mathbf{H_1} = \begin{pmatrix} -3 \\ -2 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \mathbf{H_2} = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \mathbf{H_3} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{H_4} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \mathbf{H_5} = \begin{pmatrix} -1 \\ -2 \\ -3 \\ -4 \\ -5 \\ -5 \end{pmatrix} \mathbf{H_6} = \begin{pmatrix} 3 \\ 6 \\ 9 \\ 12 \\ 15 \\ 14 \end{pmatrix}.$$

It is curious to note that we can check the correctness of our calculus by replacing this solution in the equation $\frac{d\mathbf{X}}{dt} = \mathbf{AX}$ and verifying that this equality holds.

5. Conclusion. The above method has been used succesfully in the classroom to demonstrate the solution of a linear ODE with constant coefficients. Further study for students could include the use of a computer with a system for symbolic or mathematical computation (as Maple or Mathematica) in order to create a program which executes the above method. Such a program could be performed by students working in groups of two with each computer, since this gives students some experience working on a project such as they might do in industry.

REFERENCES

[1] M. Brown, *Differential Equations and their Applications*, Springer-Verlag, New York, 1983.
[2] M.V. Fedoruk, *Ordinary Differential Equations*, Nauka, Moscow, 1980.
[3] G. Golub and C. Van Loan, *Matrix Computations*, 2nd ed., The John Hopkins Press, Baltimore, MD, 1989.
[4] G. H. Golub and J. H. Wilkinson, *Ill-conditioned eigensystems and the computation of the Jordan canonical form*, SIAM Rev., 18 (1976), pp. 578–619.
[5] B. Kagström, *How to Compute the Jordan Normal Form—The Choice Between Similarity Transformations and Methods Using the Chain Relations*, Report UMINF-91.81, Department of Computing Science, Umea University, S-901 87 Umea, Sweden, 1981.
[6] B. Kagström and A. Ruhe, *An algorithm for the numerical computation of the Jordan normal form of a complex matrix*, ACM Trans. Math. Software, 6 (1980), pp. 389–419.
[7] B. Kagström and A. Ruhe, *ALGORITHM* 560: *An algorithm for the numerical computation of the Jordan normal form of a complex matrix*, [F2], ACM Trans. Math. Software, 6 (1980), pp. 437–443.
[8] S. Lang, *Linear Algebra*, 2nd ed., Addison–Wesley, Reading, MA, 1973.
[9] G. Shilov, *Linear Algebra*, Dover, New York, 1977.