Computation of the simplest normal forms with perturbation parameters based on Lie transform and rescaling

P. Yu

Department of Applied Mathematics, University of Western Ontario, London, Ont., Canada N6A 5B7

Received 2 April 2001; received in revised form 6 April 2001

Abstract

Normal form theory is one of the most powerful tools for the study of nonlinear differential equations, in particular, for stability and bifurcation analysis. Recently, many researchers have paid attention to further reduction of conventional normal forms (CNF) to so-called the simplest normal form (SNF). However, the computation of normal forms has been restricted to systems which do not contain perturbation parameters (unfolding). The computation of the SNF is more involved than that of CNFs, and the computation of the SNF with unfolding is even more complicated than the SNF without unfolding. Although some author mentioned further reduction of the SNF, no results have been reported on the exact computation of the SNF of systems with perturbation parameters. This paper presents an efficient method for computing the SNF of differential equations with perturbation parameters. Unlike CNF theory which uses an independent nonlinear transformation at each order, this approach uses a consistent nonlinear transformation through all order computations. The particular advantage of the method is able to provide an efficient recursive formula which can be used to obtain the \( n \)th-order equations containing the \( n \)th-order terms only. This greatly saves computational time and computer memory. The recursive formulations have been implemented on computer systems using Maple. As an illustrative example, the SNF for single zero singularity is considered using the new approach. © 2002 Elsevier Science B.V. All rights reserved.

MSC: 34C20; 58F36; 34A34

Keywords: Differential equation; The simplest normal form (SNF); Nonlinear transformation; Lie transform; Rescaling; Unfolding; Computer algebra

1. Introduction

Normal form theory for differential equations can be traced back to the original work of one hundred years ago, and most credit should be given to Poincaré [11]. The theory plays an important role in the study of differential equations related to complex behavior patterns such as...
bifurcation and instability [4,8,9]. The basic idea of normal form theory is employing successive, near-identity nonlinear transformations to obtain a simple form. The simple form is qualitatively equivalent to the original system in the vicinity of a fixed point, and thus greatly simplify the dynamical analysis. However, it has been found that conventional normal forms (CNF) are not the simplest form which can be obtained, and may be further simplified using a similar near-identity nonlinear transformation (e.g., see [2,3,5,6,10,12,14,15,17]). Roughly speaking, CNF theory uses the $k$th-order nonlinear transformation to possibly remove the $k$th-order nonlinear terms of the system, while in the computation of the simplest normal form (SNF) the terms in the $k$th-order nonlinear transformation are not only used to simplify the $k$th-order terms of the system, but also used to eliminate higher order nonlinear terms. Since the computation for the SNF is much more complicated than that of CNFs, computer algebra systems such as Maple, Mathematics, Reduce, etc. have been used (e.g., see [1,7,16–19]). Recently, researchers have paid particular attention to the development of efficient computation methodology for computing the SNF [19].

The computation of normal forms has been restricted to systems which do not contain perturbation parameters (unfolding). However, in general a physical system or an engineering problem always involves some system parameters, usually called perturbation parameter or unfolding. In practice, finding such a normal form is more important and applicable. There are two ways for finding such a normal form. One way is to extend the dimension of a system by including the dimension of the parameter and then apply normal form theory to the extended system. The other way employs normal form theory directly to the original system. The former may be convenient for proving theorems while the later is more suitable for the computation of normal forms, which is particularly useful when calculating an explicit normal form for a given system. However, in most cases of computing such CNFs with unfolding, people are usually interested in the normal form only. Thus one may first ignore the perturbation parameter and compute the normal form for the corresponding “reduced” system (by setting the parameters zero), and then add unfolding to the resulting normal form. In other words, the normal form of the original system with parameters is equal to the normal form of the “reduced” system plus the unfolding. This way it greatly reduces the computation effort, with the cost that it does not provide the nonlinear transformation between the original system and the normal form. This “simplified” approach is based on the fact that the normal form terms (besides the unfolding) for the original system (with perturbation parameters) are exactly same as that of the “reduced” system, implying that all higher order nonlinear terms consisting of the parameters can be eliminated by nonlinear transformations.

The computation of the SNF is more involved than that of CNFs, and the computation of the SNF with unfolding is even more complicated than the SNF without unfolding. Although some authors mentioned further reduction of the SNF, no results have been reported on the exact computation of the SNF of systems with perturbation parameters. One might suggest that we may follow the “simplified” way used for computing CNFs. That is, we first find the SNF for the “reduced” system via a near-identity nonlinear transformation, and then add an unfolding to the SNF. However, it can be shown that this “simplified” way is no longer applicable for computing the SNF of systems with perturbation parameters. In other words, in general it is not possible to use only near-identity transformations to remove all higher order terms which involve the perturbation parameters. In this paper, we propose, in addition to the near-identity transformation, to incorporate the rescaling on time to form a systematic procedure. The particular ad-
vantage of the method is to provide an efficient recursive formula which can be used to obtain the \( n \)th-order equations containing the \( n \)th-order terms only. This greatly saves computational time and computer memory. The recursive formulation can be easily implemented using a computer algebra system such as Maple. Moreover, unlike CNF theory which uses an independent nonlinear transformation at each order, this approach uses a consistent nonlinear transformation through all order computations. This provides a one step transformation between the original system and the final SNF, without the need of combining the multiple step nonlinear transformations at the end of computation.

In the next section, the efficient computation method is presented and the general explicit recursive formula is derived. Section 3 applies the new approach to derive the SNF for single zero singularity, and conclusions are drawn in Section 4.

2. Computation of the SNF using Lie transform

Consider the general nonlinear differential equation, described by

\[
\frac{dx}{dt} = f(x, \mu), \quad x \in \mathbb{R}^n, \quad \mu \in \mathbb{R}^m,
\]

where \( x \) and \( \mu \) are the \( n \)-dimensional state variable and \( m \)-dimensional parameter variable, respectively. It is assumed that \( x = 0 \) is an equilibrium of the system for any values of \( \mu \), i.e., \( f(0, \mu) = 0 \). Further, we assume that the nonlinear function \( f(x, \mu) \) is analytic with respect to \( x \) and \( \mu \), and thus we may expand Eq. (1) as

\[
\frac{dx}{dt} = Lx + f_2(x, \mu) + f_3(x, \mu) + \cdots + f_k(x, \mu) + \cdots,
\]

where \( Lx \triangleq v_1(x) \) represents the linear part and \( L \) is the Jacobian matrix, \( D_x f \), evaluated on the equilibrium \( x = 0 \) at the critical point \( \mu = 0 \). It is assumed that all eigenvalues of \( L \) have zero real parts and, without loss of generality, is given in Jordan canonical form. (Usually \( J \) is used to indicate Jacobian matrix. Here we use \( L \) in order to be consistent with the Lie bracket notation.) \( f_k(x, \mu) \) denotes the \( k \)th-degree vector homogeneous polynomials of \( x \) and \( \mu \).

To show the basic idea of normal form theory, we first discuss the case when system (1) does not involve the perturbation parameter, \( \mu \), as normal forms are formulated in most cases. In such a case \( f_k(x, \mu) \) is reduced to \( f_k(x) \). The basic idea of normal form theory is to find a near-identity nonlinear transformation

\[
x = y + h(y) \triangleq y + h_2(y) + h_3(y) + \cdots + h_k(y) + \cdots
\]

such that the resulting system

\[
\frac{dy}{dt} = Ly + g(y) \triangleq Ly + g_2(y) + g_3(y) + \cdots + g_k(y) + \cdots
\]
becomes as simple as possible. Here \( h_k(y) \) and \( g_k(y) \) denote the \( k \)th-degree vector homogeneous polynomials of \( y \).

According to Takens’ normal form theory [13], we may first define an operator as follows:

\[
L_k : \mathcal{H}_k \mapsto \mathcal{H}_k,
\]

\[
U_k \in \mathcal{H}_k \mapsto L_k(U_k) = [U_k, v_1] \in \mathcal{H}_k,
\]

where \( \mathcal{H}_k \) denotes a linear vector space consisting of the \( k \)th-degree homogeneous vector polynomials. The operator \([U_k, v_1]\) is called Lie bracket, defined by

\[
[U_k, v_1] = Dv_1 \cdot U_k - DU_k \cdot v_1.
\]

Next, define the space \( \mathcal{R}_k \) as the range of \( L_k \), and the complementary space of \( \mathcal{R}_k \) as \( \mathcal{K}_k = \text{Ker}(L_k) \).

Thus,

\[
\mathcal{H}_k = \mathcal{R}_k \oplus \mathcal{K}_k,
\]

and we can then choose bases for \( \mathcal{R}_k \) and \( \mathcal{K}_k \). Consequently, a vector homogeneous polynomial \( f_k(x) \in \mathcal{H}_k \) can be split into two parts: one of them can be spanned by the basis of \( \mathcal{R}_k \) and the other by that of \( \mathcal{K}_k \). Normal form theory shows that the part belonging to \( \mathcal{R}_k \) can be eliminated while the part belonging to \( \mathcal{K}_k \) must be retained in the normal form.

It is easy to apply normal form theory to find the “form” of the CNF given by Eq. (4). In fact, the coefficients of the nonlinear transformation \( h_k(y) \) being determined correspond to the terms belonging to space \( \mathcal{R}_k \). The “form” of the normal form \( g_k(y) \) depends upon the basis of the complementary space \( \mathcal{K}_k \), which is induced by the linear vector \( v_1 \). We may apply the matrix method [9] to find the basis for space \( \mathcal{R}_k \) and then determine the basis of the complementary space \( \mathcal{K}_k \).

Since the main attention of this paper is focused on finding further reduction of CNFs and computing the explicit expressions of the SNF and the nonlinear transformation, we must find the “form” of \( g_k(y) \). Similar to finding CNFs, the SNFs have been obtained using near-identity nonlinear transformations. It should be mentioned that some author has also discussed the use of “rescaling” to obtain a further reduction (e.g., see [1,7]). However, no results have been reported on the study of the SNF of system (1). In this paper, we present a method to explicitly compute the SNF of system (1). The key idea is still same as that of CNFs: finding appropriate nonlinear transformations so that the resulting normal form is the simplest. The simplest here means that the terms retained in the SNF is the minimum up to any order.

The fundamental difference between the CNF and the SNF is explained as follows: finding the coefficients of the nonlinear transformation and normal form requires for solving a set of linear algebraic equations at each order. Since in general the number of the coefficients is larger than that of the algebraic equations, some coefficients of the nonlinear transformation cannot be determined. In CNF theory, the undetermined coefficients are set zero and therefore the nonlinear transformation is simplified. However, in order to further simplify the normal form, one should not set the undetermined coefficients zero but let them carry over to higher order equations and hope that they can be used to simplify higher order normal form terms. This is the key idea of
the SNF computation. It has been shown that computing the SNF of a “reduced” system (without perturbation parameters) is much complicated than that for CNFs. Therefore, it is expected that computing the SNF of system (1) is even more involved than finding the SNF of the “reduced” system. Without a computer algebra system, it is impossible to compute the SNF. Even with the aid of symbolic computation, one may not be able to go too far if the computational method is not efficient.

To start with, we extend the near-identity transformation (3) to include the parameters, given by

\[ x = y + h(y, \mu) \triangleq y + h_2(y, \mu) + h_3(y, \mu) + \cdots + h_k(y, \mu) + \cdots \]  

(8)

and then add the rescaling on time:

\[ t = (T_0 + T(y, \mu)) \tau \triangleq (T_0 + T_1(y, \mu) + T_2(y, \mu) + \cdots + T_k(y, \mu) + \cdots) \tau. \]  

(9)

Further we need to determine the “form” of the normal form of system (1). In generic case, we may use the basis of \( \mathcal{X}_k \) (see Eq. (7)) to construct \( g_k(y) \), which is assumed to be the same as that for the CNF of system (1), plus the unfolding given in the general form, \( L_1(\mu)y \), so that Eq. (4) becomes

\[ \frac{dy}{d\tau} = (L + L_1(\mu))y + g_2(y) + g_3(y) + \cdots + g_k(y) + \cdots, \]  

(10)

where \( L_1(\mu) \) is an \( n \times n \) matrix linear function of \( \mu \), to be determined in the process of computation, representing the unfolding of the system.

Now differentiating Eq. (8) with respect to \( t \) and then applying Eqs. (2), and (8)–(10) yields a set of algebraic equations at each order for solving the coefficients of the SNF and the nonlinear transformation. A further reduction from a CNF to the SNF is to find appropriate \( h_k(y, \mu) \)'s such that some coefficients of \( g_k(y) \)'s can be eliminated.

When one applies normal form theory (e.g., Takens normal form theory) to a system, one can easily find the “form” of the normal form (i.e., the basis of the complementary space \( \mathcal{X}_k \)), but not the explicit expressions. However, in practical applications, the solutions for the normal form and the nonlinear transformation are both important and need to be found explicitly. To do this, one may assume a general form of the nonlinear transformation and substitute it back to the original differential equation, with the aid of normal form theory, to obtain the \( k \)th-order equations by balancing the coefficients of the homogeneous polynomial terms. These algebraic equations are then used to determine the coefficients of the normal form and the nonlinear transformation. Thus, the key step in the computation of the \( k \)th-order normal form is to find the \( k \)th-order algebraic equations, which takes the most of the computation time and computer memory. The solution procedures given in most of normal form computation methods contain all lower order and many higher order terms in the \( k \)th-order equations, which extremely increases the memory requirement and the computation time. Therefore, from the computation point of view, the crucial step is first to derive the \( k \)th-order algebraic equations which only contain the \( k \)th-order nonlinear terms.

The following theorem summarizes the results for the new recursive and computationally efficient approach, which can be used to compute the \( k \)th-order normal form and the associated nonlinear transformation.
Theorem 1. The recursive formula for computing the coefficients of the simplest normal form and the nonlinear transformation is given by

\[ g_k = f_k + [h_k, v_1] + \sum_{i=2}^{k-1} \{[h_i, f_{k-i+1}] + Dh_i(f_{k-i+1} - g_{k-i+1})\} \]

\[ + \sum_{m=0}^{k-2} T_m \sum_{i=2}^{k-m} \frac{1}{i!} \sum_{j=i}^{k-m-i} \sum_{l_1+l_2+\cdots+l_i=k-m-(j-i)} T_l (f_{k-i} + Lh_{k-i}) \]

for \( k = 2, 3, \ldots \).

Here, \( f_j(y, \mu), g_j(y), \) and \( h_j(y, \mu) \) are all \( j \)th-degree vector homogeneous polynomials in their arguments, and \( T_j(y, \mu) \) is a scalar function of \( y \) and \( \mu \). The variables \( y \) and \( \mu \) have been dropped for simplicity. The notation \( D^j f_j h_{j_1} h_{j_2} \ldots h_{j_l} \) denotes the \( l \)th-order terms of the Taylor expansion of \( f_j(y + h(y), \mu) \) about \( y \). More precisely,

\[ D^j f_j(y + h, \mu) = D(D(\cdots D((Df_j h_{i_1} h_{i_2} \cdots h_{i_{j-1}}) h_{i_j}) \cdots h_{i_{j-1}}) h_{i_j}), \]

where each differential operator \( D \) affects only function \( f_j \), not \( h_{i_m} \) (i.e., \( h_{i_m} \) is treated as a constant vector in the process of the differentiation), and thus \( i \leq j \). Note that at each level of the differentiation, the \( D \) operator is actually a Frechét derivative to yield a matrix, which is multiplied with a vector to generate another vector, and then to another level of Frechét derivative, and so on.

Proof. First differentiating Eq. (8) results in

\[ \frac{dx}{dt} = \frac{dy}{dt} + Dh(y, \mu) \frac{dy}{dt} + (I + Dh(y, \mu)) \frac{dy}{d\tau} \frac{d\tau}{dt}, \]

then substituting Eqs. (2), (9) and (10) into Eq. (13) yields

\[ (T_0 + T(y, \mu))(Lx + f_2(x, \mu) + f_3(x, \mu) + \cdots + f_k(x, \mu) + \cdots) \]

\[ = (I + Dh(y, \mu))(Ly + L_1(\mu)y + g_2(y) + g_3(y) + \cdots + g_k(y) + \cdots). \]

Note that the \( T_0 \) can be used for normalizing the leading nonzero nonlinear coefficient of the SNF. Since this normalization may change stability analysis if time is reversed, we prefer to leave the leading nonzero coefficient unchanged and thus set \( T_0 = 1 \).

Next substituting Eq. (8) into Eq. (14) and re-arranging gives

\[ L_1(\mu)y + g_2(y) + g_3(y) + \cdots + g_k(y) + \cdots \]

\[ = Lh(y, \mu) - Dh(y, \mu)Ly + T(y, \mu)Ly - Dh(y, \mu)L_1(\mu)y \]

\[ - Dh(y, \mu)(g_2(y) + g_3(y) + \cdots + g_k(y) + \cdots) \]
\[ + f_2(y + h(y, \mu)) + f_3(y + h(y, \mu)) + \cdots + f_k(y + h(y, \mu)) + \cdots \]
\[ + T(y, \mu)(L h(y, \mu) + f_2(y + h(y, \mu)) + \cdots + f_k(y + h(y, \mu)) + \cdots). \tag{15} \]

Then we use Taylor expansions of \( f \) near \( h = 0 \) to rewrite Eq. (15) as
\[
L_1(\mu)y + g_2(y) + g_3(y) + \cdots + g_k(y) + \cdots = (1 + T(y, \mu)) \sum_{i=2}^{\infty} f_i(y, \mu) - Dh(y, \mu)L_1(\mu)y + T(y, \mu)(Ly + Lh(y, \mu))
\]
\[ + Lh(y, \mu) - Dh(y, \mu)Ly - Dh(y, \mu) \sum_{i=2}^{\infty} g_i(y) + (1 + T(y, \mu)) \sum_{i=2}^{\infty} Df_i(y, \mu)h(y, \mu) \]
\[ + (1 + T(y, \mu)) \left\{ \frac{1}{2!} \sum_{i=2}^{\infty} D^2 f_i(y, \mu)h^2(y, \mu) + \frac{1}{3!} \sum_{i=3}^{\infty} D^3 f_i(y, \mu)h^3(y, \mu) \right. \]
\[ + \cdots + \frac{1}{k!} \sum_{i=k}^{\infty} D^k f_i(y, \mu)h^k(y, \mu) + \cdots \right\}. \tag{16} \]

Further, applying Eqs. (8) and (9) yields
\[
L_1(\mu)y + \sum_{i=2}^{\infty} g_i(y) = \left\{ 1 + \sum_{i=2}^{\infty} T_i(y, \mu) \right\} \sum_{i=2}^{\infty} f_i(y, \mu) - \left\{ \sum_{i=2}^{\infty} Dh_i(y, \mu) \right\} L_1(\mu)y
\]
\[ + \left\{ \sum_{i=2}^{\infty} T_i(y, \mu) \right\} \left\{ v_1(y) + L \sum_{i=2}^{\infty} h_i(y, \mu) \right\}
\]
\[ + \sum_{i=2}^{\infty} [h_i(y, \mu), v_1(y)] + \sum_{i=2}^{\infty} \sum_{j=2}^{\infty} [h_j(y, \mu), f_i(y, \mu)]
\]
\[ + \sum_{j=2}^{\infty} Dh_j(y, \mu) \sum_{i=2}^{\infty} \left\{ f_i(y, \mu) - g_i(y) \right\}
\]
\[ + \sum_{i=2}^{\infty} T_i(y, \mu) \left\{ \sum_{i=2}^{\infty} Df_i(y, \mu) \right\} \sum_{i=2}^{\infty} h_i(y, \mu)
\]
\[ + \left\{ 1 + \sum_{i=1}^{\infty} T_i(y, \mu) \right\} \sum_{i=2}^{\infty} \frac{1}{i!} \sum_{j=i}^{\infty} D^j f_j(y, \mu) \left\{ \sum_{i=2}^{\infty} h_i(y, \mu) \right\}^i, \tag{17} \]

where \( v_1(y) = Ly \), representing the linear part of the system, and the Lie operator with respect to \( y \) has been used. Note in Eq. (17) that the expansion of \( f \) and \( h \) do not have the purely parameter
terms since \( f(0, \mu) = 0 \) for any values of \( \mu \). Finally, comparing the same order terms in Eq. (17) results in

\[
L_2 + g_2 = f_2 + [h_2, v_1] + T_1 v_1,
\]

\[
g_3 = f_3 + [h_3, v_1] + [h_2, \dot{f}_2] + Dh_2(f_2 - g_2) + T_1(f_2 + Lh_2) + T_2 v_1 - Dh_2 L_2,
\]

\[
g_4 = f_4 + [h_4, v_1] + [h_3, \dot{f}_2] + [h_2, \dot{f}_2] + Dh_2(f_3 - g_3) + Dh_3(f_2 - g_2) + \frac{1}{2} D^2f_2 h_2^2
\]

\[+ T_1(f_3 + Lh_3 + Df_2 h_2) + T_2(f_2 + Lh_2) + T_3 v_1 - Dh_3 L_2,
\]

\[
\vdots
\]

(18)

where \( L_2 \triangleq L_2(y, \mu) = L_1(\mu)y \), and the variables in \( g_i, f_i, h_i, v_i \) and \( T_i \) have been dropped for simplicity. For a general \( k \), one can obtain the recursive formula \( g_k \) given in Eq. (11), and thus the proof is completed.

It has been observed from Eqs. (11) and (18) that

(i) If system (1) does not have the parameter \( \mu \), then \( T = 0, L_1 = 0, f(y, \mu) = f(y), h(y, \mu) = h(y) \) and thus Eq. (17) is reduced to

\[
\sum_{i=2}^{\infty} g_i(y) = \sum_{i=2}^{\infty} f_i(y) + \sum_{i=2}^{\infty} [h_i(y), v_1(y)] + \sum_{i=2}^{\infty} \sum_{j=2}^{\infty} [h_j(y), f_i(y)]
\]

\[+ \sum_{j=2}^{\infty} Dh_j(y) \sum_{i=2}^{\infty} \{f_i(y) - g_i(y)\} + \sum_{i=2}^{\infty} \frac{1}{i!} \sum_{j=i}^{\infty} D^j f_j(y) \{h_2(y) + h_3(y) + \cdots\}]

(19)

which has been obtained in [19] for the “reduced” system. This indicates that computing the SNF with unfolding needs much more computation effort than that for computing the SNF without unfolding.

(ii) The only operation involved in the formula is the Frechet derivative, involved in \( Dh_i, D^j f_j \) and the Lie bracket \([\cdot, \cdot, \cdot]\). This operation can be easily implemented using a computer algebra system.

(iii) The \( k \)-th order equation contains all the \( k \)-th order and only the \( k \)-th order terms. The equation is given in a recursive form.

(iv) The \( k \)-th order equation depends upon the known vector homogeneous polynomials \( v_1, f_2, f_3, \ldots, f_{k-1} \), and on the results for \( h_2, h_3, \ldots, h_{k-1}, T_1, T_2, \ldots, T_{k-1}, L_1 \), as well as \( g_2, g_3, \ldots, g_{k-1} \) obtained from the lower order equations.

(v) The equation involves coefficients of the nonlinear transformation \( h \), rescaling \( T \) and the coefficients of the \( k \)-th order normal form \( g_k \). If the \( j \)-th order \( (j < k) \) coefficients of \( h_j \) and \( T_j \) are completely determined from the \( j \)-th order equation, then the \( k \)-th order equation only involves the unknown coefficients of \( h_k, T_k \) and \( g_k \), which yields a CNF.

(vi) If the \( k \)-th order equation contains lower order coefficients of \( h_j, T_j \), \( (j < k) \) which have not been determined in the lower order \( (< k) \) equations, they may be used to eliminate more coefficients of \( g_k \), and thus the CNF can be further simplified.
(vii) In most of the approaches for computing the SNFs without unfolding (e.g., see [1,5–7,10,15–18]), the nonlinear vector field $f$ is assumed to be in a CNF for the purpose of simplifying symbolic computations. Since for such an approach, the $k$th-order equations usually include all the lower order terms as well as many higher order terms, it is extremely time consuming for symbolic computation and takes too much computer memory. For the approach proposed in this paper, the $k$th-order equation exactly only contains the $k$th-order terms, which greatly saves computer memory and computational time. This is particularly useful for computing the SNF with unfolding. Therefore, for our approach, the vector field $f(x, \mu)$ can be assumed as a general analytic function, not necessary a CNF form.

Now we can use Eq. (18) to explain the idea of SNF. Consider the first Equation of (18) for $L_2$ and $g_2$, we can split the right-hand side into two parts: one contains $y$ only while the other involves both $y$ and $\mu$. The part containing only $y$ can determine $g_2$. That is, the part from $f_2$ that cannot be eliminated by $h_2$ and $T_1$ is the solution for $g_2$. Similarly, the other part containing both $y$ and $\mu$ can be used to find the unfolding $L_1(\mu)$. However, it can be seen that some coefficients of $h_2$ and $T_1$ are not used at this order. Setting these “unnecessary” coefficients zero results in the next equation of (18) for $g_3$ in the same situation: it only requires to use $h_3$ and $T_2$ to remove terms from $f_3$ as much as possible, since other terms $h_2$, $g_2$, $L_2$ and $T_1$ have been solved from the second order equations. This procedure can be continued to any higher order equations. This is exactly CNF theory. However, when we solve for $g_2$ and $L_2$ let the “unnecessary” coefficients of $h_2$ and $T_1$ be carried over to next step equation, then it is clear to see from the second equation of (18) that four terms may contain these “unnecessary” coefficients. These “unnecessary” coefficients can be used to possibly further eliminate a portion or the whole part of $f_3$ which cannot be eliminated by the CNF approach. If these “unnecessary” coefficients are not used at this step, they can be carried over further to higher order equations and may be used to remove some higher order normal form terms.

3. The SNF for the single zero singularity

In this section, we shall apply the results and formulas obtained in the previous section as well as the Maple program developed on the basis of recursive formula (11) to compute the SNF of single zero singularity. We use this simple example to demonstrate the solution procedure in finding the explicit SNF of a general system. For the single zero singularity, the linear part, $Lx$, becomes zero, and we may put the general expanded system in a slightly different form for convenience:

$$\frac{dx}{dt} = \sum_{i=1}^{\infty} a_{1i} \mu^i x + \sum_{i=0}^{\infty} a_{2i} \mu^i x^2 + \sum_{i=0}^{\infty} a_{3i} \mu^i x^3 + \cdots .$$

Similarly, the near-identity nonlinear transformation and the time scaling are, respectively, given by

$$x = y + \sum_{i=1}^{\infty} h_{1i} \mu^i y + \sum_{i=0}^{\infty} h_{2i} \mu^i y^2 + \sum_{i=0}^{\infty} h_{3i} \mu^i y^3 + \cdots .$$
and
\[ T_0 + T(y, \mu) = 1 + \sum_{i=1}^{\infty} T_{0i} \mu^i + \sum_{i=0}^{\infty} T_{1i} \mu^i y + \sum_{i=0}^{\infty} T_{3i} \mu^i y^2 + \cdots \] (22)
with \( t = (1 + T(y, \mu)) \tau \). In order to have a comparison between system (20) and its “reduced” system, given by (obtained by setting \( \mu = 0 \) in Eq. (20)),
\[ \frac{dx}{dt} = a_{20} x^2 + a_{30} x^3 + a_{40} x^4 + \cdots \] (23)
we list the SNF of the “reduced” system below, which were obtained only using a near-identity transformation [16]:

(i) If \( a_{20} \neq 0 \), then the SNF is
\[ \frac{dy}{dt} = a_{20} y^2 + a_{30} y^3. \] (24)

(ii) If \( a_{20} = \cdots = a_{k-10} = 0 \) but \( a_{k0} \neq 0 \) \((k \geq 3)\), then the SNF is
\[ \frac{dy}{dt} = a_{k0} y^k + b_{(2k-1)0} y^{2k-1} \] \((k \geq 3)\), (25)
where the coefficient \( b_{(2k-1)0} \) is given explicitly in terms of \( a_{j0}'s \).

This shows that the SNF of the “reduced” system (23) contains only two terms up to any order.

To obtain the unfolding, we assume \( a_{11} \neq 0 \). Other cases can be similarly discussed. Since Eq. (20) describes a codimension one system, we expect that the final SNF should have only one linear term for the unfolding, and all higher order terms in Eq. (20) are eliminated except for, at most, the terms with the coefficients \( a_{20} \) and \( a_{30} \). In other words, the SNF of equation (20) is expected to have the form:
\[ \frac{dy}{d\tau} = L_1 \mu y + g_k y^k + g_{2k-1} y^{2k-1} \] \((k \geq 2)\). (26)

We start from the case \( a_{20} \neq 0 \), and then discuss the general case.

Generic case: Suppose \( a_{20} \neq 0 \), and in addition \( a_{11} \neq 0 \). This is a generic case. Applying the first equation of (18) gives
\[ L_1 \mu y + g_2 y^2 = a_{11} \mu y + a_{20} y^2 \] \( (27) \)
which in turn results in
\[ L_1 = a_{11} \quad \text{and} \quad g_2 = a_{20}, \] (28)
as expected, i.e., no second order terms can be removed. Similarly, from the second equation of (18) one can find the following equations:
\[ a_{30} + T_{10} a_{20} = g_3, \]
\[ a_{20} a_{21} - a_{30} a_{11} + a_{20}^2 (a_{11} + T_{01}) = 0, \]
\[ a_{12} + a_{11} T_{01} = 0. \] (29)
First it is observed from Eq. (29) that if we only use the near-identity transformation (21), then only the coefficient $h_{11}$ can be used in Eq. (29), and thus $a_{30}$ must be retained in the normal form as expected. However, an additional term $a_{12}$ must be retained too in the normal form. More terms (in addition to $a_{11}, a_{20}$ and $a_{30}$) will be found in higher order normal forms. This can also happen in other singularities. This shows that one cannot apply the “simplified” way to find SNF of a system with perturbation parameters. In other words, unlike CNFs, the SNF of system (20) is not equal to the SNF of the “reduced” system (23), given by Eq. (24) or (25), plus an unfolding (which is $a_{11}\mu$ for this case).

Secondly, note from Eq. (29) that with the aid of rescaling, we can remove $g_3$ which appears in the SNF of the “reduced” system. There are three coefficients $T_{10}, T_{01}$ and $h_{11}$ for three equations in (29), and thus the three equations can be solved when $g_3 = 0$. It is further seen from Eq. (29) that $T_{10}$ must be chosen for the first equation, indicating that the rescaling must include the state variable in order to obtain the SNF of the system. $h_{11}$ and $T_{01}$ are used for the second and the third equations of (29), respectively. Before continuing to the next order equation, we summarize the above results as the following theorem.

**Theorem 2.** The general SNF of system (20) for the single zero singularity cannot be obtained using only a near-identity transformation. When the rescaling on time is applied, the SNF of the system can be even further simplified. However, the rescaling must contain the state variable.

By repeatedly applying the recursive formula (11), one can continue the above procedure to find the algebraic equations for the third, fourth, etc. order equations and determine the nonlinear coefficients. The recursive algorithm has been coded using Maple and executed on a PC. The results are summarized in Table 1, where NT stands for nonlinear transformation. The table shows the coefficients which have been computed. The coefficients in the first row, $g_2 = a_{20}$ and $L_1 = a_{11}$, are actually the two coefficients of the SNF.

It is observed from Table 1 that except for the two coefficients $g_2$ and $L_1$, all other coefficients are lined diagonally in the ascending order, according to one of the subscripts of $h_{ij}$ or $T_{ij}$. The Maple program has been executed up to 10th order. But the general rule can be easily proved by the method of mathematical induction. Note that the coefficients $h_{2j}$ are not presented in Table 1, instead the coefficients $T_{1j}$ are used. In fact, the coefficients $h_{2j}$ do not appear in the algebraic equations and that’s why the coefficients $T_{1j}$ have to be introduced, which causes the state variable involved in the time rescaling. Further it is seen that the coefficients $h_{0j}$ follows $L_1$, the coefficients $h_{1j}$ follows $g_2$, and the coefficients $T_{1j}$ are below $g_2$. After this, $h_{30}, h_{40},$ etc. are followed. This rule will be seen again in other nongeneric cases discussed later. Each row of the table corresponds to a certain order algebraic equations obtained from the recursive equation (11). For example, the two coefficients in the top row are used for solving the second algebraic equations, corresponding to the coefficients of $y^2$ and $y\mu$, and thus the SNF of system (20) for $a_{11} \neq 0$ and $a_{20} \neq 0$ is given by

$$\frac{dy}{d\tau} = a_{11}\mu y + a_{20}y^2, \quad \text{(30)}$$

up to any order. The three coefficients, $T_{10}, h_{11}$ and $T_{01}$ in the second row are used for solving the third order algebraic equations (see Eq. (11)) corresponding to the coefficients $y^3$, $y^2\mu$ and $y\mu^2$. In other words, all the three third order nonlinear terms in system (20) can be removed by the three
Table 1
NT coefficients for $a_{20} \neq 0$

<table>
<thead>
<tr>
<th>$g_2$</th>
<th>$L_1$</th>
<th>$h_{11}$</th>
<th>$T_{01}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{10}$</td>
<td>$h_{12}$</td>
<td>$T_{02}$</td>
<td></td>
</tr>
<tr>
<td>$h_{30}$</td>
<td>$T_{12}$</td>
<td>$h_{13}$</td>
<td>$T_{03}$</td>
</tr>
<tr>
<td>$h_{40}$</td>
<td>$T_{13}$</td>
<td>$h_{14}$</td>
<td>$T_{04}$</td>
</tr>
<tr>
<td>$h_{50}$</td>
<td>$T_{14}$</td>
<td>$h_{15}$</td>
<td>$T_{05}$</td>
</tr>
<tr>
<td>$h_{60}$</td>
<td>$T_{15}$</td>
<td>$h_{16}$</td>
<td>$T_{06}$</td>
</tr>
<tr>
<td>$h_{70}$</td>
<td>$T_{16}$</td>
<td>$h_{17}$</td>
<td>$T_{07}$</td>
</tr>
</tbody>
</table>

Table 2
NT coefficients for $a_{30} \neq 0$

<table>
<thead>
<tr>
<th>$g_3$</th>
<th>$L_1$</th>
<th>$h_{20}$</th>
<th>$T_{01}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{10}$</td>
<td>$h_{21}$</td>
<td>$T_{02}$</td>
<td></td>
</tr>
<tr>
<td>$T_{20}$</td>
<td>$h_{22}$</td>
<td>$T_{03}$</td>
<td></td>
</tr>
<tr>
<td>$h_{40}$</td>
<td>$T_{13}$</td>
<td>$h_{23}$</td>
<td>$T_{04}$</td>
</tr>
<tr>
<td>$h_{50}$</td>
<td>$T_{14}$</td>
<td>$h_{24}$</td>
<td>$T_{05}$</td>
</tr>
<tr>
<td>$h_{60}$</td>
<td>$T_{15}$</td>
<td>$h_{25}$</td>
<td>$T_{06}$</td>
</tr>
<tr>
<td>$h_{70}$</td>
<td>$T_{16}$</td>
<td>$h_{26}$</td>
<td>$T_{07}$</td>
</tr>
</tbody>
</table>

coefficients, and so on. All the coefficients listed in Table 1 are explicitly expressed in terms of the original system coefficients $a_{ij}$'s. Therefore, the two nonlinear transformations given by Eqs. (21) and (22) are now explicitly obtained.

**Simple nongeneral case:** Now suppose $a_{20} = 0$, but $a_{30} \neq 0$, and again we assume $a_{11} \neq 0$. This is a nongeneric case. Then the coefficient of the term $y^2$ in the second algebraic equation is identically equal to zero due to $a_{20} = 0$. Since the solution procedure is similar to Case 1, we omit the detailed discussion and list the results in Table 2, where $L_1 = a_{11}$ and $g_3 = a_{30}$.

Thus the SNF of system (20) for this case is given by

$$\frac{dy}{d\tau} = a_{11}y + a_{30}y^3.$$  \hspace{1cm} (31)

Note from Table 2 that the top left entry is empty due to $a_{22} = 0$, while $g_3$ moves downwards by one row. The coefficients $h_{1j}$ still follow $g$ coefficient, and the $T_{ij}$ coefficients are still below the $g$ coefficients. $T_{0j}$ coefficients do not change. However, comparing Table 2 with Table 1 shows that Table 2 has one more line of $T_{2j}$ coefficients, in addition to $T_{1j}$. Moreover, there is a new line given by the coefficients $h_{2j}$ following the empty box where $g_2 = 0$.

One may continue to apply the above procedure and execute the Maple program to compute the SNF of the single zero singularity for the case $a_{20} = a_{30} = 0$, $a_{40} \neq 0$, etc. Tables similar to Tables 1 and 2 can be found. In general, we may consider the following nongeneric case.

**General nongeneric case:** $a_{20} = a_{30} = \cdots = a_{k-1,0} = 0$, $a_{k0} \neq 0$. As usual, we assume $a_{11} \neq 0$. The results are listed in Table 3, where $L_1 = a_{11}$ and $g_k = a_{k0}$. This indicates that the SNF of system
executing the Maple program. For example, suppose possible unfolding may not be so simple as this case. However, they can be easily obtained by 

\[
x = h_{2(k-3)}T_{0(k-1)}
\]

\[
g_{k} = h_{(k-1)0}h_{(k-2)1}...h_{2(k-3)}T_{0(k-1)}
\]

\[
T_{10} = h_{(k-1)1}h_{(k-2)2}...h_{2(k-2)}T_{0k}
\]

\[
T_{20} = h_{(k-1)2}h_{(k-2)3}...h_{2(k-3)}T_{0(k+1)}
\]

... 

\[
T_{k(2k-2)} = h_{2(k-2)1}h_{2(k-1)2}...h_{k(k+1)}T_{0(k-1)}
\]

\[
h_{k+2} = h_{(k+1)1}h_{(k+2)2}...h_{k+1}T_{0(k+2)}
\]

(20) for the general nongeneric case is

\[
\frac{dy}{d\tau} = a_{11}\mu y + a_{k0}y^k.
\]

The general table has the similar rule as that of Tables 1 and 2: \(T_{0j}\)’s follow \(L_1\), \(h_{ij}\)’s follow the empty box where \(g_i = 0\). \(h_{ij}\)’s follow the nonzero \(g_k\), and then \(k - 1\) lines of coefficients \(T_{ij}\)’s, \(i = 1, 2, \ldots, k - 1\).

Summarizing the above results yields the following theorem.

**Theorem 3.** For the system

\[
\frac{dx}{dt} = \sum_{i=1}^{\infty} a_{11}\mu^i x + \sum_{i=0}^{\infty} a_{21}\mu^i x^2 + \sum_{i=0}^{\infty} a_{31}\mu^i x^3 + \cdots
\]

which has a zero singularity at the equilibrium \(x = 0\). Suppose \(a_{11} \neq 0\) and the first nonzero coefficients of \(a_{jo}\)’s is \(a_{i0}\), then the SNF of the system is given by

\[
\frac{dy}{d\tau} = a_{11}\mu y + a_{k0}y^k \quad (k \geq 2),
\]

up to any order.

In the above we only discussed the case \(a_{11} \neq 0\) which results in the unfolding \(a_{11}\mu y\). Other possible unfolding may not be so simple as this case. However, they can be easily obtained by executing the Maple program. For example, suppose \(a_{11} = a_{12} = 0\), but \(a_{13} \neq 0\) and \(a_{21} \neq 0\), then the SNF is found to be

\[
\frac{dy}{d\tau} = a_{13}\mu^3 y + a_{21}\mu^2 y^2 + a_{k0}y^k \quad (k \geq 2),
\]

up to any order.

Table 3

<table>
<thead>
<tr>
<th>NT coefficients for (a_{i0} \neq 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(L_1)</td>
</tr>
<tr>
<td>(h_{20} = T_{01})</td>
</tr>
<tr>
<td>(h_{30} = h_{21} = T_{02})</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>(g_{k} = h_{(k-1)0}h_{(k-2)1}...h_{2(k-3)}T_{0(k-1)})</td>
</tr>
<tr>
<td>(T_{10} = h_{(k-1)1}h_{(k-2)2}...h_{2(k-2)}T_{0k})</td>
</tr>
<tr>
<td>(T_{20} = h_{(k-1)2}h_{(k-2)3}...h_{2(k-3)}T_{0(k+1)})</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>(T_{k(2k-2)} = h_{2(k-2)1}h_{2(k-1)2}...h_{k(k+1)}T_{0(k-1)})</td>
</tr>
<tr>
<td>(h_{k+2} = h_{(k+1)1}h_{(k+2)2}...h_{k+1}T_{0(k+2)})</td>
</tr>
</tbody>
</table>

...
4. Conclusions

An efficient method is presented for computing the simplest normal forms of differential equations involving perturbation parameters. The main advantages of this approach are: (i) it provides an algorithm to compute the $k$th-order algebraic equations which only contain the $k$th-order terms. This greatly saves computational time and computer memory. (ii) The nonlinear transformation is given in a consistent form for the whole procedure. The zero singularity is particularly considered using the new approach. It is shown that the SNF for the single zero singularity with unfolding has a generic form which contains only two terms up to any order.

Acknowledgements

This work was supported by the Natural Science and Engineering Research Council of Canada (NSERC).

References