Differential Algebraic Formulation of Normal Form Theory

Martin Berz

Department of Physics and Astronomy and National Superconducting Cyclotron Laboratory, Michigan State University, East Lansing, Mi 48824

Abstract.
A differential algebraic (DA) formulation of a normal form theory for repetitive systems is presented. Contrary to previous approaches, no Lie algebraic tools are used. The resulting algorithm is very transparent and not restricted to the treatment of symplectic systems.

In the case of symplectic systems, the normal form algorithm provides a nonlinear coordinate transformation in which the motion is confined to circles. The transformation exists if the tunes are not on a resonance; in this case, it can be used to compute tune shifts in a similar way as in the Lie algebraic picture.

In the case of nonsymplectic systems, the motions in the new coordinates are growing or shrinking exponential spirals. In the case all spirals are shrinking, which occurs in electron rings, all amplitude dependent tune shifts vanish and in a formal sense tune resonances do not occur.

The algorithm has been implemented in the code COSY INFINITY. For symplectic systems, which can also be studied with the DA-Lie algorithm also implemented in COSY INFINITY, identical results are obtained at a reduced computational expense.

1. Introduction

The famous Courant Snyder theory [1] completely describes the repetitive behaviour of linear symplectic systems. It provides a unique criterion for stability of the system, it provides an invariant of the system, and allow the calculation of important quantities like the tune.

In the nonlinear case, the situation becomes substantially more involved. The question of stability is very difficult to answer, invariants usually do not exists, and the tune depends on the amplitude of the particle under consideration. Normal form theory [2, 3, 4, 5] comes closest to a nonlinear extension of the Courant Snyder theory in that it answers the questions of the amplitude dependence of the tune. It also produces a
set of pseudo-invariants which in special cases are real invariants, and at least allows the
description of the motion in coordinates which are more suitable than the original ones.

Normal form ideas were introduced to the field by Dragt and Finn [2] in the Lie
algebraic framework [6, 7]. While the original paper [2] contains all of the core ideas,
some simplifications were necessary [8] before a first implementation for realistic systems
was obtained by Neri and Dragt [4]. Difficulties inherent in the Lie algebraic formulation
limited the efforts to relatively low orders, and only the combined DA - Lie approach
[5] circumvented this problem, resulting in the first arbitrary order algorithm, and also
allowed the use of system parameters.

In the following sections we present an arbitrary order normal form algorithm
that does not require any Lie algebraic methods, and in particular does not require
ongoing changes between the factored Lie operator representation and the DA Taylor
series representation. The resulting algorithm is more direct than the hybrid algorithm,
and it allows the treatment of non-symplectic systems.

In the next section we summarize some tools discussed elsewhere in detail. Section
3 shows how to perform a nonlinear change of variable to a rotationally invariant
map. Sections 4 and 5 discuss the results of the transformation for symplectic and
non-symplectic systems. A summary and an appendix follow.

2. The Linear Transformation of the Map

The goal of the normal form algorithm is to provide a nonlinear change of variables such
that the map in the new variables has a significantly simpler structure than before. So
we assume we are given the transfer map of a particle optical system

\[ \mathcal{Z}_F = \mathcal{M}(\mathcal{Z}_i, \delta) \]  
(1)

where \( \mathcal{Z} \) are the 2v phase space coordinates and \( \delta \) are system parameters. While it is in
general impossible to obtain the exact map \( \mathcal{M} \), the DA methods [9, 10, 11, 12] allow us
to compute the partial derivatives \( [\mathcal{M}]_n \) of the map to any order \( n \). This can be done in
a particularly elegant way using the code COSY INFINITY [13, 14, 15], but also with
other DA based codes.

The normal form algorithm consists of a sequence of coordinate transformations \( \mathcal{A} \)
of the map:

\[ \mathcal{A} \circ \mathcal{M} \circ \mathcal{A}^{-1} \]  
(2)

The first such coordinate transformation is the move to the parameter dependent
fixed point \( \mathcal{Z}_F \) which satisfies

\[ \mathcal{Z}_F = \mathcal{M}(\mathcal{Z}_F, \delta) \]  
(3)

This transformation can be performed to arbitrary order using DA methods. For
details we refer to [16]. After the fixed point transformation, the map is origin preserving;
this means that for any \( \delta \), we have

\[ \mathcal{M}(\mathcal{Z}, \delta) = \mathcal{Z} \]  
(4)
As explained in [16], we note that the fixed point transformation is possible if and only if 1 is not an eigenvalue of the linear map.

In the next step we perform a linear coordinate transformation that diagonalizes the linear part of the map. For this process, we have to assume that there are \(2n\) pairwise distinct eigenvalues. This, together with the fact that no eigenvalue should be unity and that their product is positive are the only requirements we have to demand for the map; under normal conditions, accelerators are always designed such that these conditions are met.

As shown in detail in [16], it is possible to perform a diagonalization such that the linear map assumes the following form:

\[
\begin{pmatrix}
  r_1 e^{+i\mu_1} & & 0 \\
  & \ddots & \\
  0 & & r_v e^{+i\mu_v} \\
  0 & & & r_v e^{-i\mu_v}
\end{pmatrix}
\]

Here the tunes \(\mu_j\) are either purely real or purely imaginary. For stable systems, none of the \(r_j e^{\pm i\mu_j}\) must exceed unity in modulus.

For symplectic systems, the determinant is unity, which entails that the product of the \(r_j\) must be unity. This implies that for symplectic systems, for any \(r_j < 1\) there is another with \(r_j > 1\). Thus stable symplectic systems have \(r_j = 1\) for all \(j\), because otherwise there would be one \(j\) for which \(r_j\) exceeds unity, and thus at least one of \(r_j e^{\pm i\mu_j}\) would have modulus larger than unity. This would also happen if a \(\mu_j\) were imaginary. So all \(\mu_j\) are real, and they are even nonzero because we demanded distinct eigenvalues.

To the eigenvector pair \(s_j^\pm\) belonging to the eigenvalue \(r_j e^{\pm i\mu_j}\), we associate another pair \(t_j^\pm\) of variables as follows:

\[
\begin{align*}
t_j^+ &= (s_j^+ + s_j^-)/2 \\
t_j^- &= (s_j^+ - s_j^-)/2i.
\end{align*}
\]

In case of complex \(s_j^\pm\), which corresponds to the stable case, the \(t_j^\pm\) are just the real and imaginary parts and thus are real. In the unstable case, \(t_j^+\) is real and \(t_j^-\) is imaginary. Obviously the \(s_j^\pm\) can be expressed in terms of the \(t_j^\pm\) as

\[
\begin{align*}
s_j^+ &= t_j^+ + i t_j^- \\
s_j^- &= t_j^+ - i t_j^-
\end{align*}
\]

In the rest of the paper, it is advantageous to perform the manipulations in the \(s_j^\pm\), while the results are most easily interpreted in the \(t_j^\pm\).
3. The DA Normal Form Algorithm

In this section we will show a map in the $s^\pm_j$ can be subjected to nonlinear coordinate transformations that considerably simplify the nonlinear terms. The advertised transformation to the new coordinates is carried out in an iterative manner. The first step consists of the fixed point transformation and the linear diagonalization. All further steps are purely nonlinear and do not affect the linear part anymore. The $m$th step transforms only the $m$th order of the map and leaves the lower orders unaffected.

We begin the $m$th step by splitting the momentary map $M$ into its linear and nonlinear parts $R$ and $S_m$, i.e. $M = R + S_m$. The linear part $R$ has the form of Eq. 5. Then we perform a transformation using a map that to $m$th order has the form

$$A_m = \mathcal{E} + T_m$$

(8)

where $T_m$ vanishes to order $m - 1$. Because the linear part of $A_m$ is the unity map, $A_m$ is invertible. Moreover, inspection of the algorithm to invert transfer maps reveals that up to order $m$, we have

$$A_m^{-1} = \mathcal{E} - T_m$$

(9)

Of course, the full inversion of $A_m$ contains higher order terms, which will turn out to be one of the reasons why iteration is needed. It is also worth noting that in principle the higher order parts of $T_m$ can be chosen freely. It seems to be particularly useful to choose these terms in such a way that they represent the flow of a dynamical system by interpreting $T$ as the first term in the Lie derivative series [9]. This has the advantages that the computation of the inverse is trivial and that the transformation map comes out to be symplectic if the original map is.

To study the effect of the transformation, we now infer up to order $m$:

$$A \circ M \circ A^{-1} = \mathcal{E} + T_m \circ (R + S_m) \circ (E - T_m)$$

(10)

For the first step, we have used $S_m \circ (E - T_m) = S_m$ which holds because $S_m$ is nonlinear and $T_m$ is of order $m$. In the second step we used $T_m \circ (R + S_m - R \circ T_m) = T_m \circ R$ which holds because $T_m$ is of exact order $m$ and everything in the second term is nonlinear except $R$.

A closer inspection of the last line reveals that $S_m$ can be simplified by choosing the commutator $C_m = \{T_m, R\} = (T_m \circ R - R \circ T_m)$ appropriately. Indeed, if the range of $C_m$ is the full space, then $S_m$ can be removed entirely. However, as we shall see, most of the time this is not the case.

Let $(T_{mj}^\pm | k^+_1, k^-_1, \ldots, k^+_n, k^-_n)$ be the Taylor expansion coefficient of $T_{mj}$ with respect to $(s^+_1)^{k^+_1}(s^-_1)^{k^-_1} \cdots (s^+_n)^{k^+_n}(s^-_n)^{k^-_n}$ in the $j$-th component pair of $T_m$. So $T_{mj}^\pm$ is written as

$$\sum (T_{mj}^\pm | k^+_1, k^-_1, \ldots, k^+_n, k^-_n) \cdot (s^+_1)^{k^+_1}(s^-_1)^{k^-_1} \cdots (s^+_n)^{k^+_n}(s^-_n)^{k^-_n}$$

(11)
Similarly we identify the coefficients of $C$ by $(C_j^\pm|k_1^+, k_1^-, ..., k_n^+, k_n^-)$. Because $R$ is diagonal, it is easily possible to express the coefficients of $C$ in terms of the ones of $T$. One obtains

$$
(C_j^\pm|k_1^+, k_1^-, ..., k_n^+, k_n^-) = \left(\prod_{i=1}^{n} r_i^{(k_i^+ + k_i^-)} \cdot e^{i\mu_i(k_i^+ - k_i^-) - r_j \cdot e^{i\mu_j}}\right) \cdot (T_j^\pm|k_1^+, k_1^-, ..., k_n^+, k_n^-)
$$

$$
= C_j^\pm(k_1^+, k_1^-) \cdot (T_j^\pm|k_1^+, k_1^-, ..., k_n^+, k_n^-) \quad (12)
$$

Now it is apparent that a term in $S_j^\pm$ can be removed if and only if the factor $C(k_1^+, k_1^-)$ is nonzero; if it is nonzero, then the required term in $T_j^\pm$ is just the negative of the respective term in $S_j^\pm$ divided by $C(k_1^+, k_1^-)$.

So the outcome of the whole normal form transformation depends upon the conditions under which the term $C(k_1^+, k_1^-)$ vanishes. This is obviously the case if and only if the moduli and the arguments of $r_j \cdot e^{i\mu_j}$ and $\left(\prod_{i=1}^{n} r_i^{(k_i^+ + k_i^-)} \cdot e^{i\mu_i(k_i^+ - k_i^-)}\right)$ are identical.

In the next sections we will discuss the conditions of this for various special cases and draw conclusions.

4. Stable Symplectic Maps

As discussed above, in the stable symplectic case all the $r_j$ are equal to one, and the $\mu_j$ are purely real. So the moduli of the first and second terms in $C_j^\pm(k_1^+, k_1^-)$ are equal if and only if their phases agree modulo $2\pi$. This is obviously the case if

$$
\bar{\mu} \cdot (k_i^+ - k_i^-) = \pm \mu_j \quad (\text{mod} \ 2\pi)
$$

where the different signs apply for $C_j^+(k_1^+, k_1^-)$ and $C_j^-(k_1^+, k_1^-)$, respectively. This can occur in two possible ways:

1. $k_i^+ = k_i^-$ \ \forall \ i \neq j$, and $k_j^+ = k_j^- \pm 1$

2. $\bar{\mu} \cdot n = 0 \ (\text{mod} \ 2\pi)$ has nontrivial solutions.

The first case is of mathematical nature and lies at the heart of the normal form algorithm. It yields terms that are responsible for amplitude dependent tune shifts. We will discuss its consequences below. The second case is equivalent to the system lying on a higher order resonance and is of more physical nature. In case the second condition is satisfied, there will be resonance driven terms that cannot be removed and that prevent a direct computation of amplitude tune shifts.

Before proceeding in the discussion, we note that the second condition entails complications even if it is almost, but not exactly, satisfied. In this case, the removal of the respective term produces a small denominator that generates terms that become larger and larger, depending on the proximity to the resonance. In the removal process, this resonance proximity factor is multiplied by the respective expansion coefficient, and so this product obviously is an excellent characteristic of the significance of the resonance.
With higher and higher orders, i.e. larger $k^+$ and $k^-$, the number of relevant
resonances increases. Since the resonances lie dense in tune space, eventually the growth
of terms is almost inevitable and hence produces a map that is much more nonlinear
than the underlying one. As we shall see in the next section, this problem is alleviated
by damping.

We now discuss the form of the map if no resonances occur. In this case, the
transformed map will have the form

$$
\mathcal{M}^+_j = s^+_j \cdot f_j(s^+_1 s^-_1, \ldots, s^+_v s^-_v) \\
\mathcal{M}^-_j = s^-_j \cdot \tilde{f}_j(s^+_1 s^-_1, \ldots, s^+_v s^-_v)
$$

(14)

The variables $s^+_j$ are not particularly well suited for the discussion of the result,
and we express the map in terms of the adjoined variables $t^+_j$ introduced in 6. Simple
arithmetic shows that

$$
s^+_j \cdot s^-_j = (t^+_j)^2 + (t^-_j)^2
$$

(15)

It is now advantageous to write $f_j$ in terms of amplitude and phase as $f_j = a_j \cdot e^{i \phi_j}$. Performing the transformation to the coordinates $t^+_j$, we thus obtain

$$
\mathcal{M}^+_j = \begin{pmatrix} 1/2 & 1/2 \\ 1/2i & -1/2i \end{pmatrix} \cdot \begin{pmatrix} (t^+_j + it^-_j) \cdot f_j[(t^+_1)^2 + (t^-_1)^2, \ldots, (t^+_v)^2 + (t^-_v)^2] \\ (t^+_j - it^-_j) \cdot \tilde{f}_j[(t^+_1)^2 + (t^-_1)^2, \ldots, (t^+_v)^2 + (t^-_v)^2] \end{pmatrix} \\
= a_j \cdot \begin{pmatrix} \cos(\phi_j) & -\sin(\phi_j) \\ \sin(\phi_j) & \cos(\phi_j) \end{pmatrix} \cdot \begin{pmatrix} t^+ \\ t^- \end{pmatrix}
$$

(16)

Here $\phi_j = \phi_j[(t^+_1)^2 + (t^-_1)^2, \ldots, (t^+_v)^2 + (t^-_v)^2]$ depends on a rotationally invariant
quantity.

So in these coordinates, the motion is now given by a rotation, the frequency of
which depends only on the amplitudes $(t^+_j)^2 + (t^-_j)^2$ and some system parameters and
thus does not vary from turn to turn. As we will show now, these frequencies are precisely
the tunes of the nonlinear motion.

For any repetitive system, the tune of one particle is the total polar angle advance
divided by the number of turns in the limit of turn number going to infinity, if this limit
exists. If we now express the motion in the new coordinates, we pick up an initial polar
angle for the transformation to the new coordinates; then, every turn produces an equal
polar angle $\phi_j$ which depends on the amplitude and parameters of the particle; at the
end, we produce a final polar angle for the transformation back to the old coordinates.

As the number of turns increases, the contribution of the initial and final polar
angles due to the transformation becomes more and more insignificant, and in the limit
the tune comes out to nothing but $\phi_j$. So altogether, we showed that the limit exists and
that it can be computed analytically as a by product of the normal form transformation.
5. Stable Non-Symplectic Maps

In the case of stable, non-symplectic maps, all \( r_j \) must satisfy \( r_j \leq 1 \), because otherwise at least one of the \( r_j e^{\pm i \mu_j} \) is larger than unity in modulus. Since in the normal form transformation, terms can be removed if and only if the phases or amplitudes for the two contributions in \( C(k^+, k^-) \) are different and the amplitudes contribute, more terms can be removed.

Of particular practical interest is the totally damped case in which \( r_j < 1 \) for all \( j \) and all \( \mu_j \) are real, which describes damped electron rings. In this case an inspection of equation (12) reveals that now every nonlinear term can be removed. Then a similar argument as in the previous section shows that now the motion assumes the form

\[
M_j = r_j \cdot \begin{pmatrix} \cos(\phi_j) & -\sin(\phi_j) \\ \sin(\phi_j) & \cos(\phi_j) \end{pmatrix} \cdot \begin{pmatrix} t_j^+ \\ t_j^- \end{pmatrix}
\]

where now the angle \( \phi_j \) does not depend on the phase space variables anymore but only on the parameters. This means that the normal form transformation of a totally damped system leads to exponential spirals with constant frequency \( \phi_j \). In particular this entails that totally damped systems do not have any amplitude dependent tune shifts, and that they eventually collapse into the origin. Since in practice the damping is of course usually very small, these effects are usually covered by the short term sensitivity to resonances.

It is quite illuminating to consider the small denominator problem in the case of totally damped systems. Clearly the denominator can never fall below \( 1 - \max(r_j) \) in magnitude. This puts a limit on the influence of any low order resonance on the dynamics; in fact, even sitting exactly on a low order resonance does not have any serious consequences if the damping is strong enough. In general, the influence of a resonance now depends on two quantities: the distance in tune space and the contraction strength \( r_j \). High order resonances are suppressed particularly strongly because of the contribution of additional powers of \( r_j \).

Because all systems exhibit a residual amount of damping, the arguments here are generally relevant. It is especially noteworthy that residual damping suppresses high order resonances by the above mechanism even for proton machines, which entails that from a theoretical view, ultimately high order resonances become insignificant.

6. Unstable Maps

Clearly the normal form algorithm also works for unstable maps. The number of terms that can be removed will be at least the same as in the symplectic case, and sometimes it is possible to remove all terms. Among the many possible combinations of \( r_j \) and \( \mu_j \), the most common case in which the \( \mu_j \) are real is worth studying in more detail. In this case, all terms can be removed unless the logarithms of the \( r_j \) and the tunes satisfy the same resonance condition, i.e.

\[
\mathbf{n} \cdot (\log(r_1), ..., \log(r_v)) = 0 \\
\mathbf{n} \cdot \mathbf{\mu} = 0 \pmod{2\pi}
\]

(18)
have simultaneous nontrivial solutions. This situation characterizes a new type of resonance, the coupled phase-amplitude resonance.

Phase-amplitude resonances can never occur if all $r_j$ are greater than unity in magnitude. This case corresponds to a totally unbound motion, and the motion in normal form coordinates moves along growing exponential spirals.

Symplectic systems, on the other hand, satisfy $\prod_{l=1}^n r_l = 1$. So if there are $r_j$ with both signs of the logarithm, and thus the possibility for amplitude resonances exists. In fact, any symplectic system lies on the fundamental amplitude resonance characterized by $\vec{n} = (1, 1, ..., 1)$. In this light, the stable symplectic case is a degeneracy in which all logarithms vanish and so the system lies on every amplitude resonances. Thus it is susceptible to any phase resonance, and it suffices to study just these.

7. Conclusion

In this paper we have presented a DA normal form algorithm for complex periodic systems. It is applicable as long as the linear transfer map has no multiple eigenvalues, all eigenvalues differ from 1, and their product is positive. All these conditions are basic requirements for linear stability and are usually satisfied by circular accelerators.

The algorithm is very transparent and computationally efficient and does not require Lie algebraic tools. It works to arbitrary order and allows the treatment of system parameters. In the case of symplectic systems, identical results as with the hybrid DA-Lie algorithm discussed in [5] are obtained at a reduced effort. In particular, the algorithms allows the computation of amplitude and parameter tune shifts if the linear tunes are not in resonance. For systems near a resonance, the characteristic small denominator problem occurs.

The algorithm also applies to damped systems. In this case, it can be used to show that formally there are no amplitude dependent tune shifts. In addition, the transformation denominators now also contain a damping dependent term which prevents them from shrinking beyond a certain size, corresponding to the favorable long term behaviour of damped systems. Since any machine has nonzero residual damping, this also explains an old paradox of accelerator physics: in the strict sense, every resonance has to be avoided, but on the other hand, the resonances lie dense in tune space. Under the presence of ever so slight residual damping, resonances of high enough order turn out to be also mathematically irrelevant.

8. Appendix: The DA Normal Form Algorithm in COSY INFINITY Language

Besides the transparency of the normal form algorithm in the DA picture, its strength lies in the possibility to implement it completely and in full detail. As with most DA operations, the resulting programs are not only powerful but also compact and easy to understand. This is particularly true for programs written in the COSY INFINITY language [15, 13, 14].

To stress this point, we present here the COSY INFINITY source of the above normal form algorithm which amounts to about five dozen lines of code. This excludes
the routines for the eigenvalue solver, the closed orbit transformations, and the routines to compute Twiss parameters as well as low level DA routines. For the sake of comparison we mention that the hybrid DA - Lie program, which is written in precompiled FORTRAN [17] excluding the same routines is about 20 times longer.

PROCEDURE DANF M MN MA IMA EPS ; {Computes Normal Form MN of a map M.}
MA is the transformation map, which is only computed if IMA#0.
EPS is the tolerance below which resonance denominators are not removed
VARIABLE J 1 ; VARIABLE K 1 ; VARIABLE L 2 ; VARIABLE NCM 1 ;
VARIABLE IER 1 ; VARIABLE XF 100 NV ; VARIABLE T 2=NM1 ;
VARIABLE F 100 TWOND ; VARIABLE MUU 100 ND ; VARIABLE AA 100 ND ;
VARIABLE BB 100 ND ; VARIABLE GG 100 ND ; VARIABLE RR 100 ND ;
VARIABLE MU 2 ; VARIABLE A 2 ; VARIABLE B 2 ; VARIABLE D 2 ;
VARIABLE PHI 1 TWOND ; VARIABLE R 1 TWOND ;
VARIABLE M1 2=NM1 NV ; VARIABLE M2 2=NM1 NV ; VARIABLE M3 2=NM1 NV ;
VARIABLE I 2 ; IMUNIT I ; NOM := NOC ; DSET 1E-14 ;
FM M XF MN IER ; BM MN MN M1 IER ;
IF IMA#0 ; LOOP J 1 TWOND ; MA(J) := -XF(J) + DD(J) ; ENDDO;
POLVAL 1 M1 TWOND MA TWOND MA TWOND ; ENDDIF ;
GT MN F MU AA BB GG RR ;
LOOP J 1 ND ; K:= 2*J-1 ; L := K + 1 ; MU := CONS(MUU(J))*2*PI ;
A := CONS(AA(J)) ; B := CONS(BB(J)) ; D := CONS(RR(J)) ;
IF TYPE(MU)=TYPE(1) ; PHI(K) := MU ; PHI(L) := -MU ;
R(K) := D ; R(L) := D ;
M2(K) := ( I*B *DD(K) + I*B *DD(L) )/SQRT(2*I*B) ;
M2(L) := ( (-1-I*A) *DD(K) + (1-I*A) *DD(L) )/SQRT(2*I*B) ;
M1(K) := ( ( 1-I*A)/2/I*B*DD(K) - 1/2*DD(L) )*SQRT(2*I*B) ;
M1(L) := ( ( 1+I*A)/2/I*B*DD(K) + 1/2*DD(L) )*SQRT(2*I*B) ;
ELSEIF 1=1 ; PHI(K) := 0 ; PHI(L) := 0 ; MU := IMAG(MU) ;
IF IMAG(D)#0 ; WRITE 6 '$$$ ERROR DANF ' ; ENDDIF ; D := REAL(D) ;
R(K) := D*EXP(-MU) ; R(L) := D*EXP(MU) ;
M2(K) := DD(K) ; M2(L) := DD(L) ;
M1(K) := DD(K) ; M1(L) := DD(L) ; ENDDIF ;
ENDDO;
ANN MN M2 MN ; CPOLVAL 1 M1 TWOND MN TWOND MN TWOND ;
IF IMA#0 ; CPOLVAL 0 M1 TWOND MA TWOND MA TWOND ; ENDDIF ;
NOM := NOC ; RS := 0 ; LOOP J 2 NOM ; LOOP K 1 TWOND ; CO J ;
CDNFDA MN(K) R PHI K TWOND EPS T ; M3(K) := -T ; CO NOM ;
IF (K/2)=INT(K/2) ; RS := RS + T*DD(K-1) ; ENDDIF ; ENDDIFF ;
LOOP K 1 TWOND ; CDFLO M3 DD(K)+O*I T TWOND ; M1(K) := T ; ENDDIF ;
LOOP K 1 TWOND ; M2(K) := -M3(K) ; ENDDIF ;
LOOP K 1 TWOND ; CDFLO M3 DD(K)+O*I T TWOND ; M2(K) := T ; ENDDIF ;
LOOP K TWOND+1 NV ; M1(K) := DD(K) ; M2(K) := DD(K) ; ENDDIF ;
CPOLVAL 1 MN TWOND M1 NV M3 TWOND ;
LOOP K TWOND+1 NV ; M3(K) := M1(K) ; ENDDIF ;
CPOLVAL 1 M2 TWOND M3 NV MN TWOND ;
IF IMA#0 ; LOOP K 1 TWOND ; M3(K) := MA(K) ; ENDDIF ;
LOOP K TWOND+1 NV ; M3(K) := DD(K) ; ENDDIF ;
CPOLVAL 1 M2 TWOND M3 NV MA TWOND ; ENDDIF ;
ENDDIFF ; LOOP J 1 ND ; K:= 2*J-1 ; L := K + 1 ; IF PHI(K)#0 ;
M1(K) := ( DD(K) - I*DD(L) )/SQRT(2*I) ;
M1(L) := ( DD(K) + I*DD(L) )/SQRT(2*I) ;
M2(K) := ( DD(K) + DD(L) )*SQRT(I/2) ;
M2(L) := ( I*DD(K) - I*DD(L) )*SQRT(I/2) ;
ELSEIF 1=1 ;
M1(K) := DD(K) ; M1(L) := DD(L) ; M2(K) := DD(K) ; M2(L) := DD(L) ;
ENDDIF ; ENDDIFF ; LOOP K TWOND+1 NV ; M1(K) := DD(K) ; ENDDIFF ;
CPOLVAL 1 MN TWOND M1 NV M3 TWOND ;
LOOP K TWOND+1 NV ; M3(K) := M1(K) ; ENDDO
CPOLVAL 1 M2 TWOND M3 NV MW TWOND ;
IF IMA0 ; CPOLVAL 1 M2 TWOND MA TWOND MA TWOND ; LOOP K 1 TWOND ;
   IF ABS(IMAG(MA(K)))<1E-6 ; MA(K) := REAL(MA(K)) ; ENDF ; ENDDO
ENDIF ; LOOP K 1 TWOND ; IF ABS(IMAG(MW(K)))<1E-6 ;
   MW(K) := REAL(MW(K)) ; ENDF ; ENDDO ; DSET 1E-16 ; ENDPREPROC

References