Proceedings of IDETC/CIE 2005:
ASME 2005 International Design Engineering Technical Conferences
& Computers and Information in Engineering Conference
September 24-28, Long Beach, CA

DETC 2005-84038

DISCRETE FOURIER SERIES APPROXIMATION TO PERIODIC SOLUTIONS OF AUTONOMOUS DELAY DIFFERENTIAL EQUATIONS

David E. Gilsinn
Mathematical and Computational Sciences Division
National Institute of Standards and Technology
Gaithersburg, MD 20899-8910

ABSTRACT

This paper describes the algorithmic details involved in developing high-order Fourier series representations for periodic solutions to autonomous delay differential equations. Although the final approximate Fourier coefficients are computed by way of a nonlinear minimization algorithm, the steps to set up the objective function are shown to involve a sequence of matrix-vector operations. By proper coordination, these operations can be made very efficient so that high-order approximations can be obtained easily. An example of the calculations is shown for a Van der Pol equation with unit delay

NOMENCLATURE

a	=	vector of unknown Fourier coefficients
b	=	vector of residual Fourier coefficients
f	=	differentiable function
h	=	discrete delay
r	=	residual bound
t	=	scaled time vector
x	=	state variable
x_m	=	approximate state variable
A	=	coefficient matrix for perturbed state variable
B	=	coefficient matrix for perturbed delay state variable
R	=	residual state
U	=	monodromy operator
Z	=	fundamental solution of variational equation
γ_p	=	norm of p-th derivative of f
σ_p	=	bound function for approximate Fourier coefficients
ω	=	unknown frequency

INTRODUCTION

Numerically approximating a solution to a delay differential equation has been studied by many authors. See, for example, Engelborghs et al. [1], Paul [2], Shampine and Thompson [3], Willé, D. R. and Baker, C. T. H. [4]. However, some problems require knowledge of a representation of the periodic structure of a high-order approximate solution to a differential equation. The need for this form of representation has arisen in cases where several authors have been able to show that under certain conditions there exist exact periodic solutions for differential equations and functional differential equations in a computable neighborhood of the approximate solution, e. g. a Galerkin approximation. For some of these results see Cesari [5], Stokes [6,7], and Urabe [8]. An application of one of the results due to Stokes [7] has been reported by Gilsinn [9] for delay differential equations.

To the author's knowledge there have been few reports on procedures to develop representations of solutions to delay differential equations. MacDonald [10] has applied harmonic balance to special cases. Poincaré-Lindstedt methods have been used by Casal and Freedman [11] and Morris [12]. However, the methods described in these papers do not lend themselves easily to developing high-order approximations.

In this paper we extend to a class of autonomous delay differential equations a method used by Urabe and Reiter [13] to construct high-order trigonometric approximations to periodic solutions of nonautonomous ordinary differential equations. In particular, we wish to show that by a careful organization of operations we can structure the computation in an efficient matrix-vector form. Current compilers, e. g. FORTRAN 90/95, and processing systems, such as MATLAB,

provide high performance computing capability that allows vectorized matrix-vector operations. The approach to approximations presented here allows for potentially very high-order approximations.

The paper is divided as follows. We first describe the class of problems we consider and develop the form of the determining equations. Then we develop the vectorization steps for evaluating the coefficients for the approximate representation. Next, we estimate the approximation error through the residual. We then consider one approach to estimating the stability of the approximate solution. Finally, we examine a number of cases of approximate solutions to a Van der Pol equation with delay.

CONSTRUCTING THE DETERMINING EQUATIONS

We will consider the class of autonomous delay differential equations of the form

$$\ddot{x} + x = X(x(t-h), \dot{x}(t-h)) \quad (1)$$

where $x, X \in R,\ X(0,0) = 0.$ We assume that X is sufficiently differentiable. It is known that solutions exist and are unique if continuous, initial-condition functions are specified on the delay interval $[-h, 0]$, see Hale [14]. In order to simplify the notation, we normalize the delay to unity. This can be done by substituting th for *t*. Furthermore, since the period is unknown in Eq. (1) we can introduce a normalized period of 2π by replacing *t* by t/ω, where ω is an unknown frequency. Then we can put (1) in the form

$$\omega^2 \ddot{x} + x = X(x(t-\omega), \dot{x}(t-\omega)). \quad (2)$$

Thus, in the process of developing an approximate periodic solution we also approximate the frequency. For illustration, below we use the Van der Pol equation with

$$X(x(t-\omega), \dot{x}(t-\omega)) = \omega\lambda(1 - x(t-\omega)^2)\dot{x}(t-\omega)$$

In order to determine the frequency we need to constrain one of the coefficients of our representation. Since we will be using finite trigonometric polynomials we will set the most significant sine term coefficient to zero. We then write the trigonometric polynomial as

$$x_m(t) = a_2 \cos t + \sum_{n=2}^{m} \left[a_{2n} \cos nt + a_{2n-1} \sin nt \right] \quad (3)$$

where we look for a periodic solution around the origin. We will set $a_1 = \omega$ for the unknown frequency. The first and second derivatives of the trigonometric polynomial become

$$\dot{x}_m(t) = -a_2 \sin t + \sum_{n=2}^{m} \left[n a_{2n-1} \cos nt - n a_{2n} \sin nt \right] \quad (4)$$

$$\ddot{x}_m(t) = -a_2 \cos t + \sum_{n=2}^{m} \left[-n^2 a_{2n} \cos nt - n^2 a_{2n-1} \sin nt \right]$$

The residual of Eq. (2) is

$$R(t,a) = a_1^2 \ddot{x}_m(t) + x_m(t) - X(x_m(t-a_1), \dot{x}_m(t-a_1)) \quad (5)$$

where $a = (a_1, a_2, \cdots, a_{2m})^T$ Expanding Eq. (5) in discrete Fourier series gives

$$R(t,a) = \frac{1}{\pi} \sum_{n=1}^{m} [\sin nt \int_0^{2\pi} R(s,a) \sin nsds + \cos nt \int_0^{2\pi} R(s,a) \cos nsds] \quad (6)$$

If we equate Eq. (6) to zero, the determining equations can be written as

$$R_{2n}(a) = \frac{1}{\pi} \int_0^{2\pi} R(s,a) \cos nsds = 0$$

$$R_{2n-1}(a) = \frac{1}{\pi} \int_0^{2\pi} R(s,a) \sin nsds = 0 \quad (7)$$

for $n = 1, 2, \cdots, m$.

If this set of 2m determining equations in 2m unknowns has a solution $\bar{a} = (\bar{a}_1, \bar{a}_2, \cdots, \bar{a}_{2m})^T$, the trigonometric polynomial

$$\bar{x}_m(t) = \bar{a}_2 \cos t + \sum_{n=2}^{m} \left[\bar{a}_{2n} \cos nt + \bar{a}_{2n-1} \sin nt \right] \quad (8)$$

will be taken as an approximate periodic solution of Eq. (2) of period 2π with approximate frequency $\omega = \bar{a}_1$.

The key to vectorizing an algorithm to solve the determining equations Eq. (7) rests on a vectorizable representation of the integrals in Eq. (7). For this we use a result proved by Urabe and Reiter [13].

THEOREM: Let $f(t)$ be a p-times (p ≥ 1) continuously differentiable periodic function with period 2π and let its Fourier series be

$$f(t) = a_1 + \sum_{n=1}^{\infty} \left[a_{2n} \cos nt + a_{2n+1} \sin nt \right].$$

Set

$$\gamma_p = \left[\frac{1}{2\pi} \int_0^{2\pi} \left| f^{(p)}(t) \right|^2 dt \right]^{\frac{1}{2}}.$$

Then, for any positive integer N,

$$\left| a_1 - \frac{1}{2N} \sum_{i=1}^{2N} f(t_i) \right| \leq \gamma_p \sigma_p (N-1)$$

$$\left| a_{2n} - \frac{1}{2N} \sum_{i=1}^{2N} f(t_i) \cos nt_i \right| \leq 2\gamma_p \sigma_p (N-1)$$

$$\left| a_{2n+1} - \frac{1}{2N} \sum_{i=1}^{2N} f(t_i) \sin nt_i \right| \leq 2\gamma_p \sigma_p (N-1)$$

for $n = 1, 2, \cdots, N-1,$

$$t_i = \frac{2i-1}{2N}\pi,$$

for $i = 1, 2, \cdots, 2N$, and

$$\sigma_p(N-1) < \sqrt{\frac{2}{2p-1}}(N-1)^{-p+\frac{1}{2}}.$$

This theorem implies that for a sufficiently fine mesh specified by $\{t_i : i = 1, \cdots, 2N\}$ in $[0, 2\pi]$ the determining equations can be written as

$$R_{2n}(a) = \frac{1}{N}\sum_{i=1}^{2N} R(t_i, a)\cos nt_i = 0$$

$$R_{2n-1}(a) = \frac{1}{N}\sum_{i=1}^{2N} R(t_i, a)\sin nt_i = 0 \qquad (9)$$

$$\text{for } n = 1, 2, \cdots, m$$

for

$$t_i = \frac{2i-1}{2N}\pi,$$

for $i = 1, 2, \cdots, 2N$, and $N \geq m+1.$ For a more complete discussion of the interpolative ability of finite Fourier series see Dahlquist and Björck [15], Hamming [16], or Stoer and Bulirsch [17].

Equation (9) can then be used to find an approximation for the vector a by a minimization process, such as

$$\min_a \sum_{k=2}^{2m} \left| R_k(a) \right|^2 \qquad (10)$$

There are existing high-quality software packages that are optimized to perform such minimization. Our object in this paper is to indicate how the determining equations Eq. (9) can be efficiently vectorized. Since the functions Eq. (9) are evaluated for each iteration of a it is important that their evaluation be as efficient as possible.

VECTORIZATION ISSUES

The significance of the formulation of the determining equations in Eq. (9) is that the various operations can be written in matrix-vector form. In this section we assume that the optimization package used requires a user-supplied subroutine function that would take the vector of parameters a and produce the vector

$$R(a) = \left(R_2(a), \cdots, R_{2m}(a) \right)^T \qquad (11)$$

Such packages exist so this is not a constraining assumption.

Our objective in this section is to produce the basic vectorization steps needed to create Eq. (11), given an a . To start, we suppose a user has specified the number *m* of harmonics desired in Eq. (3), the trigonometric polynomial for $x_m(t)$. We also assume a mesh index *N* is specified with $N \geq m+1$. In describing the steps we will use the notation of MATLAB as well as in Golub and VanLoan [18].

There are several quantities in Eq. (9) that need only be computed once. These can be done in the main calling program and be declared as global quantities or placed in a module in FORTRAN 90/95 and accessed by a USE statement in the subroutine that computes the vector $R(a)$.

The first vector that should be produced in the main program is the mesh vector *t*. This can be done easily with the three statements

```
N2 = 2*N;
i = (1:N2)';
t = (1/N2)*(2*i-1)*pi;
```

The second statement produces a column vector from 1 to N2. The third produces a column vector from t_1 to t_{2N} . Once the mesh vector is generated, four matrices can be formed.

The first matrix that can be formed in the main program is associated with the determining equations Eq. (9). These equations can be written in matrix-vector form as

```
R = CS'*r                    (12)
```

where the prime designates transpose and

$$r = (r_1, r_2, \cdots, r_{2N})^T$$

$$r_i = R(t_i, \alpha) = a_1^2 \ddot{x}_m(t_i) + x_m(t_i) - X\left(x_m(t_i - a_1), \dot{x}_m(t_i - a_1)\right)$$

$$\text{for } i = 1, \cdots, 2N$$

$$CS = \begin{bmatrix} \sin t_1 & \sin t_2 & \cdots & \sin t_{2N} \\ \cos t_1 & \cos t_2 & \cdots & \cos t_{2N} \\ \sin 2t_1 & \sin 2t_2 & \cdots & \sin 2t_{2N} \\ \cos 2t_1 & \cos 2t_2 & \cdots & \cos 2t_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \sin mt_1 & \sin mt_2 & \cdots & \sin mt_{2N} \\ \cos mt_1 & \cos mt_2 & \cdots & \cos mt_{2N} \end{bmatrix} \qquad (13)$$

The vector *r* is calculated in the function subroutine for $R(a)$ and we will consider this vector below. The matrix *CS* can be computed once in the main program and passed to the subroutine as a global array. The portion of the script given below depends on the fact that the intrinsic functions for sine and cosine can be applied to an array. This feature is available in MATLAB and FORTRAN 90/95.

```
m2 = 2*m;
m2m1 = m2-1;
j = (1:m)';
tm = t*j';
cm = cos(tm);
sm = sin(tm);
CS(1:N2,1:2:m2m1) = sm;
CS(1:N2,2:2:m2N2) = cm;
```

The second line applies *t* to the transpose of *j* to form the $m \times N2$ array

$$\begin{bmatrix} t_1 & \cdots & t_{2N} \\ 2t_1 & \cdots & 2t_{2N} \\ \vdots & \vdots & \vdots \\ mt_1 & \cdots & mt_{2N} \end{bmatrix}$$

The next matrix of interest is associated with $x_m(t)$ in Eq. (3). We can write Eq. (3) over the mesh *t* as

$$xm = M0 * a \qquad (14)$$

where

$$xm = \left(x_m(t_1), \cdots, x_m(t_{2N})\right)^T$$

$$M0 = \begin{bmatrix} 0 & \cos t_1 & \sin 2t_1 & \cos 2t_1 & \cdots & \sin mt_1 & \cos mt_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cos t_{2N} & \sin 2t_{2N} & \cos 2t_{2N} & \cdots & \sin mt_{2N} & \cos mt_{2N} \end{bmatrix} \qquad (15)$$

We can compute *M0* as follows

M0(1:N2,1) = 0;
M0(1:N2,3:2:m2m1) = sm(1:N2,2:m);
M0(1:N2,2:2:m2) = cm.

This same matrix can be used to compute $\ddot{x}_m(t_i),\ i = 1, \cdots, N2.$ We do this by defining a global work column array in the main program of length *m*2 and setting it to $W = (0, -a_2, -2^2 a_3, -2^2 a_4, \cdots, -m^2 a_{2m-1}, -m^2 a_{2m})^T$.

We will discuss an efficient procedure for creating this vector below. The computation of the second derivative is them simply

$$xmdd = M0 * W \qquad (16)$$

Next, $x_m(t - a_1)$ can be computed using the array CS in Eq. (13). We can write

$$xm\_delay = CS * b \qquad (17)$$

where

$$xm\_delay = \left(x_m(t_1 - a_1), \cdots, x_m(t_{2N} - a_1)\right)^T$$

$$b = \begin{pmatrix} a_2 \sin a_1 \\ a_2 \cos a_1 \\ a_3 \cos 2a_1 + a_4 \sin 2a_1 \\ -a_3 \sin 2a_1 + a_4 \cos 2a_1 \\ \vdots \\ a_{2m-1} \cos ma_1 + a_{2m} \sin ma_1 \\ -a_{2m-1} \sin a_1 + a_{2m} \cos ma_1 \end{pmatrix} \qquad (18)$$

We will discuss the efficient computation of *b* below. Forming $\dot{x}_m(t_i - a_1)$ can be done in a similar manner. In this case

$$xmd\_delay = CS * c \qquad (20)$$

where

$$xmd\_delay = (\dot{x}_m(t_1 - a_1), \dot{x}_m(t_2 - a_1), \cdots, \dot{x}_m(t_{2N} - a_1))^T$$

$$c = \begin{bmatrix} -a_2 \cos a_1 \\ a_2 \sin a_1 \\ 2a_3 \sin 2a_1 - 2a_4 \cos 2a_1 \\ 2a_3 \cos a_1 + 2a_4 \sin a_1 \\ \vdots \\ ma_{2m-1} \sin ma_1 - ma_{2m} \cos ma_1 \\ ma_{2m-1} \cos ma_1 + ma_{2m} \sin ma_1 \end{bmatrix} \qquad (21)$$

For each call to the function that produces $R(a)$, each of the vectors *W*, *b*, and *c* must be formed. The forming of each of the vectors depends on the operations with which each is composed. We will consider the efficient computation of each of the vectors in turn.

We assume the vector $a = \left(a_1, a_2, \cdots, a_{2m}\right)^T$ is available to the function $R(a)$. We initialize the work array W and two more column work arrays W1 and W2 of length 2*m* to zero in the main program and pass them as global arrays. We can initialize the vectors to zero in the main program by

W = zeros(m2,1);
W1 = zeros(m2p1,1);
W2 = zeros(m2,1);

Instead of defining extra arrays *b* and *c* we will reuse *W*. We can also initialize some other arrays and variables in the main program as

mv = (1:m)';
mvsqr = mv.*mv;
m2 = 2*m;
m2m1 = m2 – 1;

and pass them as global values also.

In the function for $R(a)$ we can immediately compute *xm* in Eq. (14) as

xm = M0*a;

To compute *xmdd* in Eq. (16) we compute the array *W* as

W(1:2:m2m1,1) = -mvsq.*a(1:2:m2m1,1);
W(2:2:m2,1) = -mvsq.*a(2:2:m2,1);

The dot-asterisk operation takes two vectors of the same length and creates a new vector of the same length with each element of the new vector equal to the product of the same element entries of the two vectors in the product. Thus if w = u.*v then w(i) = u(i)*v(i). This operation is defined in MATLAB. Although it is not defined as an intrinsic operation of FORTRAN 90/95, there is a facility to create user defined operations by using an INTERFACE OPERATOR block. We

should note that the first element in *W* when multiplied by *M*0 will produce a zero. It is more efficient to leave it in W than to expend extra operations to create a zero there.

We can generate *b* in Eq. (17) using the work arrays *W*, *W*1 and *W*2 as follows. We first break *b* in Eq. (18) into the sum of two vectors

$$\begin{bmatrix} 0 \\ 0 \\ a_3 \cos 2a_1 \\ -a_3 \sin 2a_1 \\ \vdots \\ a_{2m-1} \cos ma_1 \\ a_{2m-1} \sin ma_1 \end{bmatrix}$$

and

$$\begin{bmatrix} a_2 \sin a_1 \\ a_2 \cos a_1 \\ a_4 \sin 2a_1 \\ a_4 \cos 2a_1 \\ \vdots \\ a_{2m} \sin ma_1 \\ a_{2m} \cos ma_1 \end{bmatrix}$$

To form *xm_delay* in Eq. (17) we compute as follows. Note the reuse of the work arrays.

```
smva = sin(mv*a(1,1));
cmva = cos(mv*a(1,1));
W(1,1) =0;
W(2,1) = 0;
W1(4:2:m2,1) = -a(3:2:m2m1,1);
W1(3:2:m2m1,1) = a(3:2:m2m1,1);
W2(1:2:m2m1,1) = cmva;
W2(2:2:m2,1) = smva;
W = W1.*W2;
W1(1:2:m2m1,1) = a(2:2:m2,1);
W1(2:2:m2,1) = a(2:2:m2,1);
W2(1:2:m2m1,1) = smva;
W2(2:2:m2,1) = cmva;
W = W + W1.*W2;
Xm_delay = CS*W;
```

We can form *xmd_delay* in Eq. (21) in a similar manner. We break *c* into the sum of two vectors

$$\begin{bmatrix} 0 \\ 0 \\ 2a_3 \sin 2a_1 \\ 2a_3 \cos 2a_1 \\ \vdots \\ ma_{2m-1} \sin ma_1 \\ ma_{2m-1} \cos ma_1 \end{bmatrix}$$

and

$$\begin{bmatrix} -a_2 \cos a_1 \\ a_2 \sin a_1 \\ -2a_4 \cos 2a_1 \\ 2a_4 \sin 2a_1 \\ \vdots \\ -ma_{2m} \cos ma_1 \\ ma_{2m} \sin ma_1 \end{bmatrix}$$

Then, to form *xmd_delay* in Eq. (21) we can write using W2 from the previous calculation

```
W1(1,1) = 0;
W1(3:2:m2m1,1) = mv(2:m).*a(3:2:m2m1,1);
W1(2:2:m2,1) = W1(1;2:m2m1,1);
W = W1.*W2;
W1(1:2:m2m1,1) = -mv.*a(2:2:m2,1);
W1(2:2:m2,1) = -W1(1:2:m2m1,1);
W2(1:2:m2m1,1) = cmva;
W2(2:2:m2,1) = smva;
W = W + W1.*W2;
Xmd_delay = CS*W;
```

Finally, we can form the residual vector $R(a)$ by setting

```
r = (a(1,1)*a(1,1)*xmdd+xm+a(1,1)*lambda…
      (xm_delay.*xm_delay – 1).*xmd_delay;
R = CS'*r;
```

Where the three dots at the end of the line represent a a line continuation. The objective function for (10) can be formed by

```
y = sum(R.*R);
```

where y is the scalar returned by the function $R(a)$ to the minimization routine.

ESTIMATING THE RESIDUAL

We wish to estimate a value for *r* such that

$$r \geq \left| \bar{a}_1^2 \ddot{\bar{x}}_m(t) + \bar{x}_m(t) - X\left(\bar{x}_m(t-\bar{a}_1), \dot{\bar{x}}_m(t-\bar{a}_1)\right) \right| \qquad (22)$$

for $t \in [0, 2\pi]$. The overbar is used to represent the estimated solution and parameters.

In this section and the next, we will present the general methods and not detail the possible vectorization steps that

could be used. In general, they would be similar to those in the previous section.

Let

$$R(t) = \bar{a}_1^2 \ddot{\bar{x}}_m(t) + \bar{x}_m(t) - X\left(\bar{x}_m(t-\bar{a}_1), \dot{\bar{x}}_m(t-\bar{a}_1)\right). \quad (23)$$

To estimate the value of r we can consider the approximate trigonometric polynomial

$$R(t) \approx \sum_{n=1}^{m_0} \left(b_{2n} \cos nt + b_{2n+1} \sin nt\right) \quad (24)$$

for some large positive integer m_0. As in the second section the coefficients are approximated by

$$b_{2n} = \frac{1}{N} \sum_{i=1}^{2N} R(t_i) \cos nt_i \quad (25)$$

$$b_{2n+1} = \frac{1}{N} \sum_{i=1}^{2N} R(t_i) \sin nt_i$$

for $n = 1, 2, \cdots, m_0,\ N \geq m_0 + 1,$

$$t_i = \frac{2i-1}{2N}\pi$$

for $i = 1, 2, \cdots, 2N.$ Then we estimate the *r* in Eq. (22) by

$$\max_{0 \leq t \leq 2\pi} \left| \sum_{n=1}^{m_0} \left(b_{2n} \cos nt + b_{2n+1} \sin nt\right) \right| \quad (26)$$

An obvious upper bound for Eq. (26) is given by

$$\sum_{n=1}^{m_0} \left(\left|b_{2n}\right| + \left|b_{2n+1}\right|\right) \quad (27)$$

but refined meshes for $t \in [0, 2\pi]$ could lead to a better estimate in Eq. (26).

STABILITY OF THE APPROXIMATE SOLUTION

The stability of the approximate periodic solution is determined by the characteristic multipliers of the variational equation with respect to the approximate solution. Characteristic multipliers for delay differential equations have been computed by several authors. See Butcher et al. [19] and Luzyanina and Engelborghs [20], for example. For this paper we will use a method proposed by Gilsinn [9] based upon representing the monodromy operator in terms of a variation-of- constants formula developed by Halanay [21].

For the class of equations considered here the variational equation about the approximate solution is given by

$$\dot{z}(t) = A(t)z(t) + B(t)z(t-\omega) \quad (28)$$

where

$$A(t) = \begin{bmatrix} 0 & 1 \\ -1/\omega^2 & 0 \end{bmatrix}$$

$$B(t) = \begin{bmatrix} 0 & 0 \\ \left(1/\omega^2\right) X_1(x(t-\omega), \dot{x}(t-\omega)) & \left(1/\omega^2\right) X_2(x(t-\omega), \dot{x}(t-\omega)) \end{bmatrix}$$

The subscripts to *X* represent the partial derivatives with respect to the first and second variables, respectively. Clearly $A(t) = A(t+2\pi),\ B(t) = B(t+2\pi).$

Let $Z(t,s)$ be the fundamental solution of Eq. (28) such that $Z(s,s) = I,\ Z(t,s) = 0$ for $t < s.$ Define the operator

$$(U\phi)(s) = z(s+2\pi) \quad (29)$$

where ϕ is the initial continuous function on $[-\omega, 0]$. The eigenvalues of U are the characteristic multipliers (Halanay [21]). Furthermore Halanay [21] has shown that the operator U can be written as

$$(U\phi)(s) = Z(s+2\pi, 0)\phi(0) + \int_{-\omega}^{0} Z(s+2\pi, \alpha+\omega)B(\alpha+\omega)\phi(\alpha)d\alpha \quad (30)$$

We can approximate the operator equation by discretizing $[-\omega, 0]$ into *k* equal intervals by $-\omega = s_1 < \cdots < s_{k+1} = 0$ and setting the interval as $\Delta = \omega / k.$ *U* is then represented by a matrix $\left[U_{ij}\right]$ where for $i = 1, \cdots, k+1,\ j = 1, \cdots, k,$

$$U_{ij} = Z(s_i + 2\pi, s_j + \omega)B(s_j + \omega)\Delta.$$

For $i = 1, \cdots, k+1,\ j = k+1,$

$$U_{i,k+1} = Z(s_i + 2\pi, s_{k+1}) + Z(s_i + 2\pi, s_{k+1} + \omega)B(s_{k+1} + \omega)\Delta.$$

The fundamental solution can be computed numerically using, for example, the program of Shampine and Thompson [3], or by other means as discussed earlier.

EXAMPLE

For an example of the approximation process described in the previous sections we consider the Van der Pol equation with unit delay

$$\ddot{x}(t) + x(t) = \lambda\left(1 - x(t-1)^2\right)\dot{x}(t-1). \quad (31)$$

As before, we introduce an unknown frequency and write Eq. (31) in the form

$$\omega^2 \ddot{x}(t) + x(t) = \omega\lambda\left(1 - x(t-\omega)^2\right)\dot{x}(t-\omega) \quad (32)$$

for $t \in [0, 2\pi]$.

We will study the effects of various values of λ and various orders of harmonic approximation. In all of these cases, we will estimate the residual error and compute the characteristic multipliers relative to the approximate solution. If the approximate solution were in fact a true periodic solution, then one of the characteristic multipliers would be identically one. Since we are dealing with variations about an approximate

solution, however, we can expect that there may not be any characteristic multipliers equal to one. We can still expect that if all of the characteristic multipliers lie within the unit circle in the complex plane, then the approximate solution is stable and if any characteristic multiplier lies outside of the unit circle the approximate solution is unstable (Halanay [21]).

The results reported here were performed on a system with a 2.4 GHz Intel Pentium 4 processor, 2 GB RAM, under a Windows 2000, Service Pack 4, operating system. The algorithms were coded in MATLAB 6.5.

The first case we consider is for λ=0.01. The first subcase is for a Galerkin expansion of 5 harmonics. Figure 1 shows the error distribution for the residual in Eq. (26).

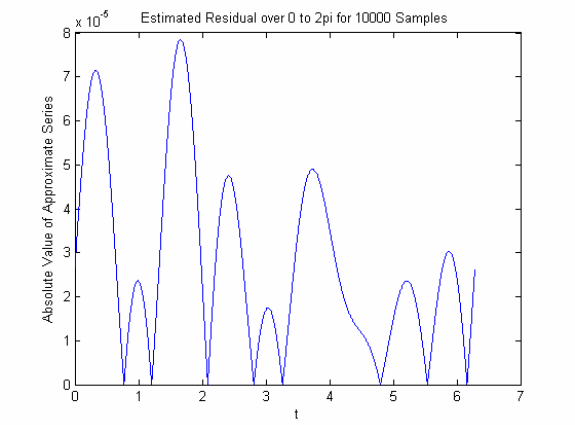

Figure 1: Error distribution for 10000 samples from 0 to 2π for the Galerkin approximation residual for λ = 0.01 and 5 harmonics.

Figure 2 shows an overlay phase plot of the Galerkin approximate solution and the numerical solution of the Van der Pol Eq. (32) where the frequency was set to the estimated frequency. The numerical integration was done over a range of 8π. Although it is hard to distinguish it the Galerkin phase plot is done in dashed lines.

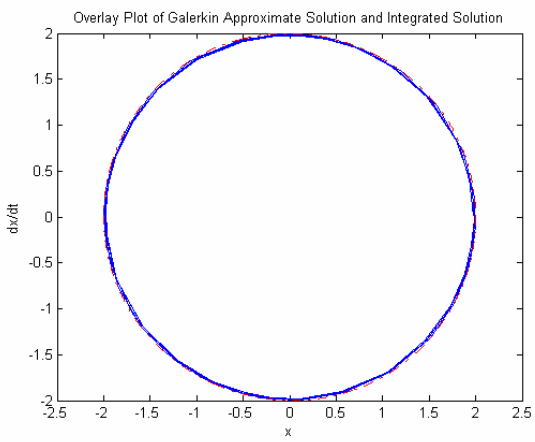

Figure 2: Overlay of Galerkin approximation and numerically integrated Van der Pol for λ=0.01 with 5 harmonics

For each case, for λ three harmonics cases will be examined. The results will be summarized in a table. Table 1 represents the summary results for the λ=0.01 case. The six rows of the table are from top to bottom, the λ case, the harmonics in the Galerkin approximation. T1 in milliseconds represents the sum of the milliseconds consumed in setting up the CS and M0 matrices in the main program plus the time consumed in one evaluation of the function called by the optimization subroutine. T2 in milliseconds is the sum of T1 and the time it takes to run the optimization subroutine (fminuc in MATLAB). The fifth row is the maximum absolute value of the residual error computed in Eq. (26). The final row is the maximum of the absolute values of the characteristic multipliers.

λ=0.01			
Harmonics	5	10	25
T1 (milliseconds)	0	10	0
T2 (milliseconds	190	1091	3044
Max. Abs. Resid.	7.8481e-5	4.4003e-5	1.1390e-4
Max. Abs. Char. Mult.	0.9962	0.9962	0.9961

Table 1: Summary table for λ=0.01 case showing timings, absolute residual error and maximum absolute characteristic multiplier.

The fact that T1 shows zero milliseconds indicates the vectorization is efficient enough to evaluate all of the matrices and vectors in less than a millisecond. That T1 is 10 in the 10 harmonic case is possibly due to a system interrupt during the evaluations, since the other cases were zero. T2 shows the effect of the optimization call. Note that the absolute residual error increases at 25 harmonics along with a change in the maximum absolute characteristic multiplier. This indicates that there is an optimum setting for the harmonics that is less than 25. Since the maximum absolute value of the characteristic multipliers is less than one, this indicates the approximate solution is stable. This is clearly indicated by Figure 2 which shows the attraction of the numerically integrated Van der Pol equation.

We now consider the λ=0.1 case. The only figures we will show for the rest of the cases are the overlay figures comparable to Fig. 2. Figure 3 shows some deforming of the limit cycle but again the Galerkin approximate solution overlays the numerically integrated Van der Pol equation, although as Table 2 shows the maximum absolute residuals in this case are larger than in the λ=0.01 case. Again, the characteristic multipliers lie within the unit circle indicating stability. This is indicated by the maximum absolute value of the characteristic multipliers being less than one. Table 2 shows that there is no distinct advantage in adding harmonics beyond five.

Figure 4 shows further deforming of the phase plot for the λ=0.5 case. Table 3 shows the maximum absolute residual increasing. The maximum absolute characteristic multiplier still remains less than one showing that stability still holds.

Figure 5 and Table 4 indicate that stability still holds at λ=1.0 but Figure 6 and Table 5 show that instability begins to occur at λ=1.14 although, even with numerical integration over 64π, the numerical solution still remains close to the approximate solution.

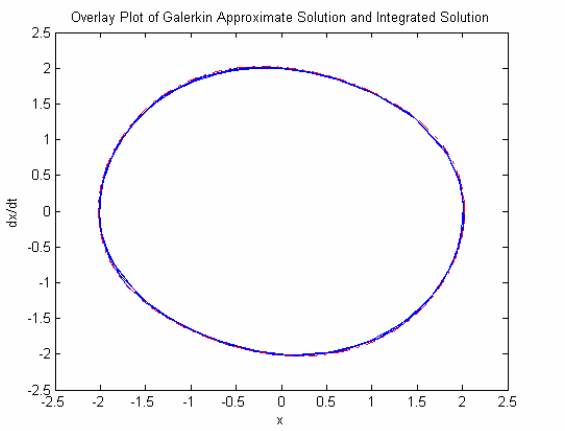

Figure 3: Overlay of Galerkin approximation and numerically integrated Van der Pol for λ=0.10 with 5 harmonics

λ=0.10			
Harmonics	5	10	25
T1 (milliseconds)	0	0	10
T2 (milliseconds	110	641	2724
Max. Abs. Resid.	5e-3	5e-3	5e-3
Max. Abs. Char. Mult	0.9979	0.9979	0.9979

Table 2: Summary table for λ=0.10 case showing timings, absolute residual error and maximum absolute characteristic multiplier.

Instability at λ=1.14 for the Van der Pol equation with delay in the nonlinear terms is in marked contrast with the Van der Pol equation with no delay. In that case the Van der Pol equation is stable for large values of λ, far exceeding one.

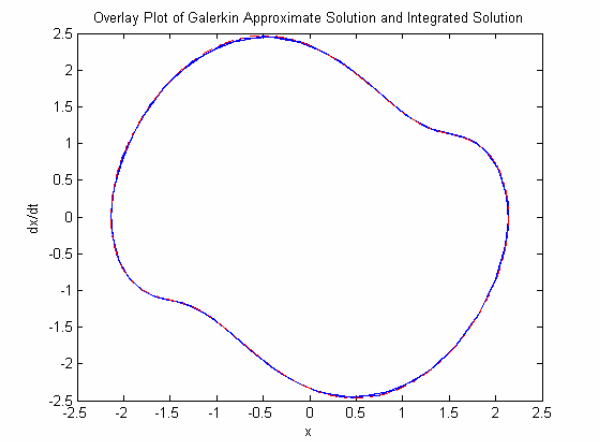

Figure 4: Overlay of Galerkin approximation and numerically integrated Van der Pol for λ =0.50 with 5 harmonics

λ =0.50			
Harmonics	5	10	25
T1 (milliseconds)	0	0	0
T2 (milliseconds	121	381	2994
Max. Abs. Resid.	0.2389	0.2459	0.2454
Max. Abs. Char. Mult	0.9963	0.9962	0.9962

Table 3: Summary table for λ =0.50 case showing timings, absolute residual error and maximum absolute characteristic multiplier.

SUMMARY

This paper has shown that approximate periodic solution for autonomous differential equations with constant delay can be efficiently computed by using discrete Fourier series. The efficiency arises from the ability to write the various algorithm operations in vector-matrix form. With existing scientific computing languages and processors these operations can be vectorized adding to the efficiency and speed-up of the algorithm.

DISCLAIMER

Certain trade names and company products are mentioned in the text or identified in an illustration in order to adequately specify the experimental procedure and equipment used. In no case does such an identification imply recommendation or endorsement by the National Institute of standards and Technology, nor does it imply that the products are necessarily the best available for the purpose.

λ =1.00			
Harmonics	5	10	25
T1 (milliseconds)	0	0	0
T2 (milliseconds	200	561	2994
Max. Abs. Resid.	1.4980	2.2655	2.2766
Max. Abs. Char. Mult	0.9474	0.9930	0.9930

Table 4: Summary table for λ =1.0 case showing timings, absolute residual error and maximum absolute characteristic multiplier.

ACKNOWLEDGMENTS

The author wishes to acknowledge the assistance of Christopher Copeland of Vanderbilt University for his assistance in testing earlier versions of the algorithms described in this paper.

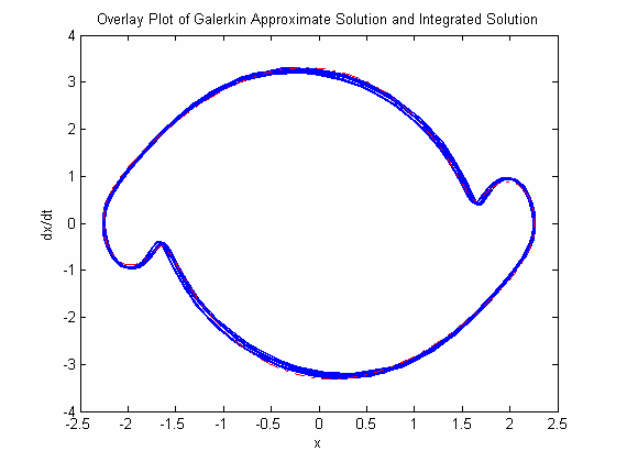

Figure 6: Overlay of Galerkin approximation and numerically integrated Van der Pol for λ =1.14 with 5 harmonics

λ =1.14			
Harmonics	5	10	25
T1 (milliseconds)	0	0	0
T2 (milliseconds	271	631	3065
Max. Abs. Resid.	3.1132	4.2665	4.2745
Max. Abs. Char. Mult	1.0121	1.0026	1.0025

Table 5: Summary table for λ =1.14 case showing timings, absolute residual error and maximum absolute characteristic multiplier.

REFERENCES

[1] Engelborghs, K., Luzyanina, T., IN 'T Hout, K. J., and Roose, D., 2000, "Collocation Methods for the Computation of Periodic Solutions of Delay Differential Equation," SIAM J. Sci. Comput., **22**, (5), pp. 1593-1609.

[2] Paul, C. A. H., 1992, "Developing a Delay Differential Equation Solver," Applied Numerical Mathematics, **9**, pp. 403-414.

[3] Shampine, L. F., and Thompson, S., 2001, "Solving DDE's in MATLAB," Applied Numerical Mathematics, **37**, pp. 441-458.

[4] Willé, D. R., and Baker, C. T. H., 1992, "DELSOL – a Numerical Code for the Solution of Systems of Delay-Differential Equations,' **9**, pp.223-234.

[5] Cesari, L., 1962, "Functional Analysis and Periodic Solutions of Nonlinear Differential Equations," **1**, (2), pp. 149-187.

[6] Stokes, A., 1972, "On the Approximation of Nonlinear Oscillations," Journal of Differential Equations, **12**, pp. 535-558.

[7] Stokes, A. P., 1976, "On the Existence of Periodic Solutions of Functional Differential Equations," Journal of Mathematical Analysis and Applications, **54**, pp. 634-652.

[8] Urabe, M., 1965, "Galerkin's Procedures for Nonlinear Periodic Systems," Arch. Rational Mech., **20**, pp. 120-152.

[9] Gilsinn, D. E., 2004, "Approximating Limit Cycles of a Van der Pol Equation with Delay," Proc. Of Dynamic Systems and Applications, **4**, To appear.

[10] MacDonald, N., 1995, "Harmonic Balance in Delay-Differential Equations," Journal of Sound and Vibration, **186**, (4), pp. 649-656.

[11] Casal, A., and Freedman, M., 1980, "A Poincaré-Lindstedt Approach to Bifurcation Problems for Differential-Delay Equations," IEEE Transactions on Automatic Control, **AC-25**, (5), pp. 967-973.

[12] Morris, H. C., 1976, "A Perturbative Approach to Periodic Solutions of Delay-Differential Equations," J. Inst. Maths Applics, **18**, pp. 15-24.

[13] Urabe, M., and Reiter, A., 1966, "Numerical Computation of Nonlinear Forced Oscillations by Galerkin's Procedure," Journal of Mathematical Analysis and Applications, **14**, pp. 107-140.

[14] Hale, J., 1971, *Functional Differential Equations*, Springer-Verlag, New York, pp. 13-23.

[15] Dahlquist, G., and Björck, Å, 1974, *Numerical Methods*, Prentice-Hall, Inc., Englewood Cliffs, pp. 410-411.

[16] Hamming, R. W., 1973, *Numerical Methods for Scientists and Engineers*, McGraw-Hill Book Company, New York, pp. 510-515.

[17] Stoer, J. and Bulirsch, R., 1993, *Introduction to Numerical Analysis*, Springer-Verlag, New York, pp. 72-77.

[18] Golub, G. H., and Van Loan, C. F., 1989, *Matrix Computations*, The Johns Hopkins University Press, Baltimore.

[19] Butcher, E. A., Ma, H., Bueler, E., Averina, V., and Szabo, Z., 2004, "Stability of Linear Time-Periodic Delay-Differential Equations via Chebyshev Polynomials," International Journal for Numerical Methods in Engineering, **59**, pp. 895-922.

[20] Luzyanina, T., and Engelborghs, K., 2002, "Computing Floquet Multipliers for Functional Differential Equations," International Journal of Bifurcation and Chaos, **12**, (12), pp. 2977-2989.

[21] Halanay, A., 1966, *Differential Equations: Stability, Oscillations, Time Lags*, Academic Press, New York, Chap. 4.