AN ALGORITHM FOR THE NUMERICAL SOLUTION OF MULTIPORT BOUNDARY VALUE PROBLEMS ARISING FROM SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS IN WHICH JUMP DISCONTINUITIES ARE PERMITTED AND FOR WHICH BOTH THE DYNAMICS AND BOUNDARY CONDITIONS MAY BE NONLINEAR. NUMERICAL RESULTS ARE GIVEN FOR SEVERAL EXAMPLES AND THE ALGORITHM IS ALSO APPLIED TO A NOISY DYNAMICAL SYSTEM IN WHICH THE STATES ARE ESTIMATED BY USING A VARIATIONAL TECHNIQUE.

1. INTRODUCTION

Multipoint boundary value problems (MPBVP's) for ordinary differential equations have been studied as early as 1897 by Niccolletti [1], and later by Wilder, [2], Polya [3], Bocher [4], and Whyburn [5], and recently by many authors (see the bibliographies in Bellman and Kalaba [6], Brown [7], and Agarwal [8]. In MPBVP's considered here the dynamics of a system are described by a first order differential equation of dimension n

\[ \dot{x} = f(x, t), \quad a < t < b, \]  

and the boundary conditions are of the form

\[ g[x(t_1), x(t_2), \ldots, x(t_m)] = 0, \]  

where \( a = t_1 < t_2 < \ldots < t_m = b. \)

Boundary conditions of the type

\[ x(t_i) = y_i, \quad i = 1, \ldots, m, \]  

correspond to the measurement or observation of a system at m different times so the description (1.1)-(1.2) includes problems that arise naturally in many different fields. Boundary value problems for partial differential equations can often be reduced to this description also [9]. While initial value and two point problems are included, we are concerned mainly with problems for which \( m > 2. \)

Discontinuities in the solutions at interior boundary points occur in many applications [9-14], and several theoretical results concerning existence and uniqueness of solutions have been given, especially for linear and separated boundary conditions corresponding to (1.2), [7, 15]. For continuous problems some numerical methods, including quasilinearization [6], invariant imbedding [16], finite differences [17], collocation [18], and the method of adjoints [8], have been proposed and recently an initial value adjusting method, extending quasilinearization techniques and multiple shooting techniques for TPBVP's has also been proposed [19]. For this latter method the functions for the dynamics and boundary conditions, while being required to be sufficiently smooth, may both be nonlinear.

In this paper we further extend this method by giving an algorithm to solve nonlinear MPBVP's in which discontinuities are permitted. The boundary conditions for such problems can be described by equations of the form

\[ h[x(t_{1}^{+}), x(t_{2}^{+}), x(t_{2}^{+}), \ldots, x(t_{m-1}^{+}), x(t_{m}^{+})] = 0 \]  

(1.4)

where \( x(t_{i}^{+}) \) means the usual right or left limit at \( t = t_i. \)

The algorithm proposed for such problems is an iterative method that adjusts initial values at each boundary point except the last, so is a kind of parallel shooting method.

It is possible to extend the method given in [19], so that only initial values at one point (usually \( t = a \)) are adjusted, along the lines proposed here, but such an algorithm will contain a large number of transitional matrices, so will have numerical drawbacks. In addition, the present method can be used over longer
intervals and for stiff problems by introducing more boundary points with corresponding continuity conditions. The algorithm also contains features similar to those in [19] and [20] which overcome some difficulties of quasilinearization, namely, the equations are dealt with directly, the partial derivatives (Jacobians) need not be calculated analytically, only initial approximations at boundary points, rather than initial functions, need be prescribed, and storage requirements from previous iterations are greatly reduced. In part II [21] a quadratic convergence proof is given for the method.

In the next section the notation and many of the definitions are given as well as the details for an extended quasilinearization algorithm. The main method and its computational algorithm, together with the theoretical relationships between the two algorithms are given in section 3, and several numerical examples are presented in the following section. The first few examples represent a variety of conditions and analytical solutions are given for comparative purposes. In the last example the states of a dynamical system with given noisy measurements are estimated, using a least-squares estimator. Such problems occur in optimal control and many areas of engineering [22-24], and we use a simple example considered in [22], where invariant imbedding was used, to illustrate the method. The Euler-Lagrange conditions applied to the problem give rise to a nonlinear system with boundary conditions containing discontinuities and the algorithm proposed here is applied to obtain numerical results. The application of the algorithm to compartment models from pharmacokinetics in which discontinuities arise from bolus inputs will appear later.

2. QUASILINEARIZATION

We suppose that the dynamics of the problem under consideration are governed by the first order system of differential equations written as:

\[ x = f(x, t), \quad a < t < b, \]  

(2.1)

where \( t \) is the independent variable and \( x \) is an n-dimensional vector. We also assume that the boundary conditions for the problem are prescribed at several points, and it is convenient to use two vector equations to describe them:

\[ g(x(t_1), \ldots, x(t_{m_1})) = 0, \quad a = t_1 < \ldots < t_{m_1} = b, \]  

(2.2a)

\[ v(x(t_1), \ldots, x(t_{m_1}), x(r_1), x_{\downarrow_1}, \ldots, x(r_{m_2})), \]  

\[ x(r_{m_2}) = 0, \quad t_1 < r_1 < \ldots < r_{m_2} < t_{m_1}, \quad r_i \neq t_j, \]  

(2.2b),

\[ m_1 > 2, \quad m_2 > 0. \]  

Equation (2.2a) deals exclusively with boundary points \( t = t_j \), where \( x \) is continuous, and (2.2b) permits \( x \) or some component to be discontinuous at \( t = r_j \), with corresponding right and left limits denoted \( x(r_j^+) \) and \( x(r_j^-) \). The vector \( g \) will be required to have the same number of components as \( f \); \( v \) must have \( n(m_1 + m_2 - 2) \) or less components so that the problem will not be over-determined. Continuity conditions of the type

\[ x_k(t_j^-) - x_k(t_j^+) = 0 \]  

(2.3)

are allowed in (2.2b) and in fact the number of such conditions that will make \( v \) exactly \( n(m_1 + m_2 - 2) \) dimensional are added to (2.2b) so that the problem (2.1), (2.2) is not under-determined.

Simple jump discontinuities for \( x \) would be included in (2.2b) in the form (2.3) with a constant added to one side of the equation. We assume that the function \( f \) is twice continuously differentiable in \( x \) and continuous in \( t \) on subintervals between boundary points, and that the functions \( g \) and \( v \) are twice continuously differentiable in their arguments. By a solution of (2.1), (2.2), we mean a piecewise \( C^1 \) vector function \( x \) satisfying (2.2) everywhere and satisfying (2.1) on subintervals between the boundary points.

In order to give an iterative method to solve (2.1), (2.2), we first order and re-label the boundary points as

\[ t_1 = r_1 < r_2 < \ldots < t_{m_1} + m_2 = t_{m_1}, \]  

and consider \( r \) as the new independent variable. At the \( k \)-th step of the procedure, initial approximations \( kx(r_i) \) are given at \( r = r_i \), \( i = 1, \ldots, m_1 + m_2 - 1 \) and the resulting initial value problems

\[ \dot{x} = f(x, r), \quad t_1 < r < t_{i+1}, \]  

(2.4a)

\[ x(t_i) = kx(r_i^-), \]  

(2.4b)

are solved, with solutions denoted \( kx^{(i)}(r) \), \( i = 1, \ldots, m_1 + m_2 - 1 \). The function \( kx^{(i)}(r) \), given by \( kx^{(i)}(r) \) on \( [r_i, r_{i+1}] \) is then substituted into the boundary conditions to obtain

\[ k = g[kx(r_1^+), \ldots, kx(r_{m_1}^-), kx(r_{m_1}^+ + m_2 - 1)], \]  

(2.5a)

and

\[ k = v[kx(r_1^-), kx(r_2^+), kx(r_2^-), \ldots, kx(r_{m_1}^- + m_2 - 1)]. \]  

(2.5b)

If \( k_x \) and \( k_v \) are sufficiently small, then \( kx(r) \) is the required approximate solution to (2.1), (2.2), otherwise the initial approximations are adjusted, the index \( k \) is replaced by \( k + 1 \) and the procedure is repeated. The algorithms proposed to adjust the initial approximations are given in (2.10) and (3.6) below. The first one

Journal of Computational and Applied Mathematics, volume 6, no 2, 1980. 134
(2.10) corresponds to a quasilinearization technique and the second one (3.6) is the extended initial value adjusting method with discontinuities, that is used for the computing.

In order to derive the algorithm, we first define

\[ k_x(\tau_i, \tau_{i+1}) \] to be the \( n \times n \) transitional matrix satisfying

\[ \Psi(\tau_i, \tau_{i+1}) = f_x(k_x, \tau) \Psi(\tau_i, \tau_{i+1}), \quad \tau_i < \tau < \tau_{i+1}, \]

\[ \Psi(\tau_i, \tau_{i+1}) = I_n, \quad i = 1, \ldots, m_1 + m_2 - 1, \quad (2.6a) \]

where \( f_x \) denotes the Jacobian of \( f \) and \( I_n \) is the \( n \times n \) identity matrix. Then for \( i = 2, \ldots, m_1 + m_2 \) we have

\[ k_x^{i+1}(\tau_i) = k_x(\tau_i) + k_x(\tau_i, \tau_{i+1})[k_x^{i+1}(\tau_{i+1}) - k_x(\tau_{i+1})], \quad \tau_{i-1} < \tau < \tau_i. \quad (2.7) \]

Using this value when \( i = m_1 + m_2 \) in \( k_x^{i+1} \) and expanding in a Taylor's series about \( [k_x(\tau_{m_1} + m_2)] \) we obtain from the boundary condition \( k_x^{i+1} = 0 \) a linearized boundary condition

\[ \frac{\partial k_x}{\partial x}(\tau_i) + \cdots + \frac{\partial k_x}{\partial x}(\tau_{m_1} + m_2 - 1) = 0. \quad (2.8) \]

Similarly, for \( i = 2, \ldots, m_1 + m_2 \), the values from (2.7) are used in \( k_x^{i+1} \) and the Taylor's series expansion about \( [k_x(\tau_{m_1} + m_2)] \) yields the linearized boundary condition

\[ \frac{\partial k_x}{\partial x}(\tau_i) + \cdots + \frac{\partial k_x}{\partial x}(\tau_{m_1} + m_2 - 1) = 0. \quad (2.9) \]

From (2.8) and (2.9) we have

\[ k_x(k_x^{i+1} - k_x) = -\beta, \quad (2.10) \]

where

\[ k_x = [k_x(\tau_{m_1}^{1}), \ldots, k_x(\tau_{m_1 + m_2 - 1})]^T, \quad (2.11) \]

\[ \beta = [k_g, k_v]^T, \quad (2.12) \]

and the adjusting matrix \( k_S \) contains matrices \( S_{ij} \), of the form

\[ S_{ij} = \frac{\partial k_g}{\partial x}(\tau_{i+1}) + \frac{\partial k_v}{\partial x}(\tau_{i+1}^j), \quad j = 1, \ldots, m_1 + m_2 - 2, \quad (2.13a) \]

\[ S_{1,1} + \cdots + S_{1,j} + \cdots + S_{1,m_1 + m_2 - 1} = 0. \quad (2.13b) \]

Thus new initial conditions are determined from (2.10) if \( k_S \) is nonsingular, and this algorithm corresponds to the quasilinearization technique.

### 3. INITIAL VALUE ADJUSTING METHOD

In order to make the above method more practical numerically, we now approximate the partial derivatives and transitional matrices in \( k_S \) by introducing perturbed initial value problems at each boundary point and in each direction.

Again, consider the \( k \)-th step with initial approximation \( k_x(\tau_{i+1}) \) at the \( i \)-th boundary point. Let \( k_j(i)(\tau_i) \) denote the solution to the problem

\[ x = f(x, \tau), \quad \tau_i < \tau < \tau_{i+1}, \quad (3.1a) \]

\[ \frac{\partial k_j(i)(\tau_i)}{\partial x}(\tau_i) = e_j, \quad (3.1b) \]

where \( e_j \) is the unit vector in the \( j \)-th direction and \( e \) is a small positive number called the perturbation parameter. Define an \( n \times n \) dimensional matrix

\[ k_j(\tau, \tau_{i+1}; e) \] having \( j \)-th column, \( k_j(\tau, \tau_{i+1}; e) \), at \( \tau = \tau_{i+1}^j \) \((i = 1, \ldots, m_1 + m_2 - 1)\) given by

\[ k_j(\tau_{i+1}^j, \tau_{i+1}^j; e) = -\frac{1}{e} k_j(i)(\tau_{i+1}^j - k_x(i)(\tau_{i+1}^j)), \quad (3.2) \]

where \( k_x(i)(\tau_i) \) solves the original initial value problems.
For the boundary conditions corresponding to (2.2a) we now define \( n \) dimensional vectors for \( j = 1, \ldots, n \) as follows:

\[
k^j \equiv \{k_x(\tau^+_j), \ldots, k_x(\tau^-_1), \ldots, \}
\]

for \( i = 1, \ldots, m_1 + m_2 - 2 \), and

\[
k^j(m_1 + m_2 - 1) = \{k_x(\tau^+_1), \ldots, k_x(\tau^-_{m_1 + m_2 - 2}), \}
\]

These vectors are then used to define \( n \times n \) dimensional matrices \( kQ(i)(e) \) having \( j \)-th columns given by

\[
k^jQ(i)(e) = \frac{1}{e} \left\{ k^j(i)(e) - k^j \right\},
\]

where again \( k^j(i)(\tau) \) solves (3.1). Analogous to (2.10), we now let

\[
kS(e) = kS.
\]

\[
\text{Proof} \quad \text{Relationships (i) and (ii) are proved similarly to those for the continuity case [20], and (iv) follows immediately from (ii) and (iii). In order to prove (iii), we fix i, k, and j \((1 < i < m_1 + m_2 - 1, k > 0, 1 < j < n)\).}
\]

By expanding in a Taylor's series about \( k^j(i)(\tau) \), we have

\[
k^j(i)(\tau) = k^j(i)(\tau^-_j) + \frac{\partial k^j}{\partial x(\tau^-_1)} \left( k^j(i)(\tau^-_1) - k^j(i)(\tau^-_1) \right) + \text{higher order terms}
\]

Therefore

\[
\lim_{e \to 0} \frac{1}{e} \{v[k^j(i)(\tau^-_1), \ldots, k^j(i)(\tau^-_1)] - k^j(i)(\tau^-_1)\}
\]

and

\[
\lim_{e \to 0} kS(e) = kS.
\]

Let the matrices \( k\Psi(\tau, \tau^+), k\Psi(\tau^-_1, \tau^-_1; e) \), \( kQ(i)(e), kV(i)(e), kS \) and \( kS(e) \)

\((i = 1, \ldots, m_1 + m_2 - 1; k = 0, 1, \ldots)\) be defined by

\[
(2.6), (3.2), (3.4), (3.5), (2.13) \text{ and (3.7) respectively. Then}
\]

(i) \( \lim_{e \to 0} k\Psi(\tau^-_i + 1, \tau^-_i; e) = k\Psi(\tau^-_i + 1, \tau^-_i), \)

(ii) \( \lim_{e \to 0} kQ(i)(e) = \frac{\partial k^j}{\partial x(\tau^-_1)} \) for \( i = 1, 2, \ldots, m_1 + m_2 - 2, \)

\[
\frac{\partial k^j}{\partial x(\tau^-_1 + 1)} + \frac{\partial k^j}{\partial x(\tau^-_1 + 1)} k\Psi(\tau^-_1, \tau^-_1),
\]

(iii) \( \lim_{e \to 0} kV(i)(e) = \frac{\partial k^j}{\partial x(\tau^-_1)} - \frac{\partial k^j}{\partial x(\tau^-_1 + 1)} k\Psi(\tau^-_1, \tau^-_1); \)

and

(iv) \( \lim_{e \to 0} kS(e) = kS. \)
i.e. the j-th columns of (3.10) are true, and thus (3.10) is valid.

If the adjusting matrix, $kS$, from the quasilinearization technique has an inverse, then it can be shown that $kS(e)$ will also have an inverse for sufficiently small $e$. As seen above, the elements of $kS(e)$ can be obtained from (3.4) and (3.5), so that partial derivatives need not be calculated analytically for the algorithm (3.6), and only initial values from a previous iteration need be stored. Quasilinearization techniques and the initial value adjusting method possess a quadratic convergence property (cf. [6], [25]) and details concerning such a property for this method will appear later. The examples provided in the next section also indicate this property. The computational algorithm is summarized as follows.

1. Set $k = 0$, prescribe the perturbation parameter $e$, the convergence criterion $\delta$, and the initial approximations $kx(i)_{j} (i = 1, ..., m_1 + m_2 - 1)$;

2. for each $i (1 \leq i \leq m_1 + m_2 - 1)$, compute the initial value problems (2.4) by a Runge-Kutta-Gill algorithm and store $kx(i)_{j} (r_{i+1})$;

3. calculate $k_g$ and $k_v$ from (2.5) and set $k(x) = (k_g, k_v)^T$ and obtain $k_G = \left( \frac{1}{n(m_1 + m_2 - 1)} \right)^{1/2}$;

4. if $k_G < \delta$, then $kx$ is the required approximate solution, otherwise proceed;

5. for each $i (1 \leq i \leq m_1 + m_2 - 1)$ and each $j (1 \leq j \leq n)$ compute the perturbed initial value problems (3.1) and store $kx(i)_{j} (r_{i+1})$;

6. calculate $k_q(i)_{j}$ and $k_y(i)_{j}$ from (3.4) and (3.5) and use them to form $kS(e)$ from (3.7);

7. solve the algebraic system (3.6) by Gaussian elimination for the new initial approximations $k + 1 x(r_{i+1}) (1 \leq i \leq m_1 + m_2 - 1)$, set $k = k + 1$ and return to step 1.

4. NUMERICAL EXAMPLES

Several examples are presented to show the effectiveness and scope of the above method. The first three deal with the same system having linear dynamics and various types of boundary conditions, and for which actual solutions are known; the second deals with a system having nonlinear dynamics, and corresponds to a well-known problem in science. The tables contain a few of the calculated values, initial approximations, $0 x(t)$, the number of grid points, $p$, the number of iterations, $k$, and the error, $G$. A few of the exact values are also included for the first three examples. All calculations were done in double precision on an IBM 370/158.

**Example 1**

We consider the second order equation $\ddot{x} + x = 0$ on $[0, \pi]$ with corresponding linear system

$$\begin{align*}
\ddot{x}_1 &= x_2, \\
\ddot{x}_2 &= -x_1
\end{align*}$$

and simple boundary conditions corresponding to (2.2a) given by

$$
\begin{align*}
x_1(0) &= 0, & x_2(0) &= 1.
\end{align*}
$$

For discontinuities we assume that $x_1$ has a jump decrease of size 1 and $x_2$ is continuous at $t = \pi/6$, $x_2$ has a jump increase of size 1 and $x_1$ is continuous at $t = \pi/3$, $x_1$ and $x_2$ have jump decreases of size 1 and $3^{1/2} - 1$ respectively at $t = \pi/2$ and both are continuous at $t = 2 \pi/3$. The unique solution to this problem is

$$
x = x_1 = (\sin t) \chi[0, \pi/6) + 1/2 (\sin t - 3^{1/2}\cos t) \chi[\pi/6, \pi/3),
$$

and

$$
x = x_1 = (\sin t) \chi[\pi/3, \pi/2) - (\cos t) \chi[\pi/2, \pi),
$$

where $\chi[a, b)$ is the usual characteristic function that has value 1 for $x \in [a, b)$ and is 0 otherwise. The results are given in table 1. The convergence criterion was arbitrarily set at $5 \times 10^{-8}$ for this example.

**Example 2**

We again consider the system (4.1) but with nonlinear boundary conditions given by

$$
\begin{align*}
x_1(0) + x_2(\pi/4) + x_1(3\pi/4) &= 0, \\
(4.3a)
\end{align*}
$$

and jump decreases of size 1 in both components at $t = \pi/2$. The two solutions for this problem are

$$
x = x_1 = \begin{cases}
(2^{-1/2} - 1) \sin t + (2^{3/2} - 1) \cos t/(1 - 2^{1/2}) \chi[0, \pi/2) \\
(2^{-1/2} - 2) \sin t + 2^{1/2} \cos t/(1 - 2^{1/2}) \chi[\pi/2, \pi),
\end{cases}
$$

and

$$
x = x_1 = (\sin t) \chi[0, \pi/2) + (\cos t) \chi[\pi/2, \pi).
$$

Table 2 contains results for both solutions. For the second one the convergence criterion was decreased to $5 \times 10^{-14}$, but the number of grid points was maintained at 100.

**Example 3**

For the last example of this type, we again use system (4.1) and boundary conditions (4.2) but with highly nonlinear boundary conditions corresponding to (2.2b) using all boundary points, given by
\[ x_1 (\pi/4^+)^2 x_2 (\pi/4^+)^3 \]
\[ - \exp \left[ x_2 (\pi/2^-) \right] x_2 (\pi/4^-)^2 x_3 (3\pi/4^+)^2 \]
\[ + x_2 (\pi)^2 x_1 (3\pi/4^-)^2 = .1859767072 \]
\[ x_1 (\pi/2^-)^2 x_1 (\pi/4^-)^3 x_1 (3\pi/4^+)^2 \]
\[ + x_2 (\pi/2^+)^2 \exp \left[ x_2 (3\pi/4^-) \right] + x_1 (\pi/2^+)^2 x_1 (\pi)^2 \]
\[ = .1261677772 \]

and with both components continuous at \( t = \pi/4, 3\pi/4 \). The values of one of the actual solutions
\[ x = x_1 = (\sin t) x [0, \pi/2) + (.269767584177 \cos t - .257683622961 \sin t) x [\pi/2, \pi) \]
and its approximations are listed in table 3. Again the convergence criterion was set at \( 5 \times 10^{-14} \), but the initial approximations were very inaccurate for this example.

The number of iterations \( k \), and corresponding error criteria \( G_i \), are listed for the above three examples in table 4. The convergence rates appear to be quadratic when the initial approximations are sufficiently close to the exact values.

**Example 4**

For the last example we apply the algorithm in order to estimate the states of a dynamical system when noisy measurements are given. For this problem a standard variational technique is used to optimize a least squares criterion. When many measurements are used, this technique produces a multipoint boundary value problem with discontinuities at the points where measurements are taken, as discussed theoretically in [26]. As an illustration, we consider the problem of estimating the three angular velocities about the principal body axes of a rotating body, given noisy measurements on only one angular velocity [22]. Suppose the equations of motion are given as

\[ \dot{x}_1 = -2x_2x_3 + u_1 \]  
(4.4a)

\[ \dot{x}_2 = 1.5x_1x_3 + u_2 \]  
(4.4b)

\[ \dot{x}_3 = -0.25x_1x_2 + u_3 \]  
(4.4c)

for \( 0 < t < 20 \) (seconds), where \( x_i \) represents the angular velocity (radians/second) about the \( i \)th principal axis, \( u_i \) is an unknown disturbance torque for that axis, and suppose further that measurements are made only on one angular velocity \( x_1 \). We write the measurements as

\[ y_i = x_1 (t_i) + v_i, \quad 1 < i < m, \]  
(4.5)

where \( x_1 (t_i) \) represents the actual value at \( t = t_i \), \( v_i \) represents noise and \( m \) is the number of measurements.

If \( 0x = (0x_1, 0x_2, 0x_3) \) is an initial estimate for the unknown initial condition, \( x_0 \), for (4.4) and (4.5), then we wish to minimize the criterion

\[ J(x_0, x, u) = \frac{1}{2} \sum (x_0_i - 0x_i)^2 + \frac{1}{2} \sum v_i^2 \]

\[ + \frac{1}{2} \int_0^2 \|u(t)\|^2 dt. \]  
(4.6)

If we introduce Lagrange multipliers, \( \lambda(t) = (\lambda_1, \lambda_2, \lambda_3) \), and if \( 0x \) is sufficiently accurate, then a local solution to (4.4)-(4.6) exists and is given by a solution to the system [26]

\[ \dot{x} = f(x, t) - \lambda(t) \]  
(4.7a)

\[ \dot{\lambda} = -f_x (x, t)' \lambda(t), \]  
(4.7b)

with boundary conditions

\[ \lambda(0) = 0x - x(0) \]  
(4.8a)

\[ \lambda(t_{i-1}^*) - \lambda(t_i^*) = y_i - x_1 (t_i), \quad i = 1, \ldots, m-1 \]  
(4.8b)

\[ \lambda(t_m^*) = x_1 (t_m) - y_m, \]  
(4.8c)

where \( f(x, t) \) represents the given dynamics and \( f_x (x, t)' \) is the transpose of the usual Jacobian. In this case,

\[ u(t) = -\lambda(t) \]  
(4.7)

\[ \lambda(t_{i+1}^*) = \lambda(t_i^*) \]  
(4.8)

\[ \lambda(t_m^*) = x_1 (t_m) - y_m, \]  
(4.8)

The measurements used for the simulation were obtained as follows. First, equations (4.4) with \( u_1 = u_2 = u_3 = 0 \), and with initial values arbitrarily taken as \( x_1(0) = .9, x_2(0) = .6 \) and \( x_3(0) = .15 \), were integrated for \( 0 < t < 20 \), using steps of size 0.04. The resulting values, which were considered to be the actual solutions, are listed at a few points in table 5. The step size and number of steps given in the table were used throughout.

Ten values for the first angular velocity, \( x_1(t) \), at times \( t = 1, \ldots, 10 \), were used as measurements in (4.8) for the next stage. The exact initial values above were also used for \( 0x \) in (4.8). This stage of the experiment
was simply done to check that the method would produce the actual values if the actual measurements were used. For the last stage, the measurements were corrupted with noise by using the model

\[ y_i = x(t_i) + 0.1 r_1(t_i) x(t_i) + 0.1 r_2(t_i) \]

\(i = 1, \ldots, 10\) where \(r_1\) and \(r_2\) are statistically independent random variables, uniformly distributed between -1 and +1. With these ten exact or noisy measurements and initial guesses, \(0x\), the boundary conditions (4.8) are then written in the following way for the algorithm. The components for \(g\) (points of continuity) are

\[
\begin{align*}
\lambda_1(0) &= 0, x_1(0) - x_1(0) \\
\lambda_2(0) &= 0, x_2(0) - x_2(0) \\
\lambda_3(0) &= 0, x_3(0) - x_3(0) \\
\lambda_1(10) &= x_1(10) - y(10) \\
\lambda_2(10) &= 0 \\
\lambda_3(10) &= 0
\end{align*}
\]

and the components for \(v\) (points of discontinuity) are

\[
\begin{align*}
\lambda_1(1^+) - \lambda_1(1^-) &= y(1) - x_1(1) \\
\vdots & \quad \vdots \\
\lambda_1(9^+) - \lambda_1(9^-) &= y(9) - x_1(9)
\end{align*}
\]

with the remaining \(v\)-components corresponding to missing continuity conditions for \(x_i(t), i = 1, 2, 3, \) and \(\lambda_2(t)\) and \(\lambda_3(t)\) at \(t = 1, 2, \ldots, 9\) being provided automatically by the algorithm. When the exact measurements were used, the algorithm applied to (4.9) and (4.11) produced the actual solutions to an accuracy of \(1 \times 10^{-8}\) or better. The measurement noise, which should turn out to be zero in this case, was \(1 \times 10^{-10}\) or better. In this, and the other runs as well, the convergence criterion was set at \(1 \times 10^{-8}\).

When noisy measurements (4.10) were used, several values for \(0x\) were tested, and the results for two of these are listed in table 4. As expected, the results are more accurate when \(0x\) is closer to the actual initial values. The results here agree with those in [22], where invariant imbedding was used with the same estimator. Some remarks indicating feasibility of the estimator are given there. Finally, in table 7 error criteria is given for the iterates for all three trials reported. The initial approximations for \(x_i\) and \(\lambda_i\) used in all three cases were

\[
x_1(0) = 1, x_2(0) = x_3(0) = 0.2, \text{ and } x_i(t_j) = \lambda_i(t_j) = 1 \times 10^{-4} \text{ for all other points,}
\]

\(t_j = 1, \ldots, 10\). As indicated in this table, the rate of convergence appears to be quadratic for the last few iterates in each case.

5. CONCLUSION

A numerical algorithm has been given for the solution of very general multipoint problems involving ordinary differential equations, and for which jump discontinuities are allowed. In order to use the code that was developed for the algorithm, only the differential equations (1.1), the boundary conditions (1.2), and initial approximations at boundary points need be prescribed, in addition to the usual computing parameters. Thus it can be applied to a wide variety of problems with very little preliminary work. It appears to be relatively insensitive to the accuracy of the initial guesses, and quadratic convergence is proved in part II [21].

6. ACKNOWLEDGEMENT

It is a pleasure to acknowledge many interesting discussions on multipoint problems with Dr. Richard Bellman. The authors are also grateful for the generous use of computing facilities at the University of Southern California.

REFERENCES


Table 1. Results for example 1, p = 120, k = 1, G = 7.1 × 10^{-10}

<table>
<thead>
<tr>
<th>t</th>
<th>0x1(t)</th>
<th>0x2(t)</th>
<th>x1(t)</th>
<th>kx1(t)</th>
<th>x2(t)</th>
<th>kx2(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.1</td>
<td>0.9</td>
<td>0.0</td>
<td>.5263554869D -10(a)</td>
<td>1.0</td>
<td>.1000000000D 01</td>
</tr>
<tr>
<td>π/6⁻</td>
<td>0.5(b)</td>
<td>.4999999981D 00</td>
<td>.8660254038D 00</td>
<td>.8660254048D 00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>π/6⁺</td>
<td>-0.6</td>
<td>0.8</td>
<td>-0.5</td>
<td>-.5000000037D 00</td>
<td>.8660254038D 00</td>
<td>.8660254048D 00</td>
</tr>
<tr>
<td>π/3⁻</td>
<td>0.0</td>
<td>-.4791049739D -08</td>
<td>1.0</td>
<td>.1000000002D 01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>π/3⁺</td>
<td>0.1</td>
<td>2.1</td>
<td>0.0</td>
<td>-.4847221360D -08</td>
<td>2.0</td>
<td>.2000000002D 01</td>
</tr>
<tr>
<td>π/2⁻</td>
<td>1.0</td>
<td>.9999999936D 00</td>
<td>1.7</td>
<td>.1732050814D 01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>π/2⁺</td>
<td>0.1</td>
<td>0.9</td>
<td>0.0</td>
<td>-.6073722467D -08</td>
<td>1.0</td>
<td>.10000000007D 01</td>
</tr>
<tr>
<td>2π/3⁻</td>
<td>0.5</td>
<td>.4999999968D 00</td>
<td>.8660254038D 00</td>
<td>.8660254146D 00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2π/3⁺</td>
<td>0.4</td>
<td>0.8</td>
<td>0.5</td>
<td>.4999999970D 00</td>
<td>.8660254038D 00</td>
<td>.8660254144D 00</td>
</tr>
<tr>
<td>π</td>
<td>1.0</td>
<td>.1000000000D 01</td>
<td>1.0</td>
<td>.1200569022D -07</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) -.526D -10 = −0.526 × 10^{-10}
(b) x1(π/6⁻) = lim _{t→π/6⁻} x1(t) = .5

### Table 2. Results for example 2

First solution: \( p = 100, k = 2, G = 4.3 \times 10^{-8} \)
Second solution: \( p = 100, k = 3, G = 1.4 \times 10^{-16} \)

<table>
<thead>
<tr>
<th>( t )</th>
<th>( 0x_1(t) )</th>
<th>( 0x_2(t) )</th>
<th>( x_1(t) )</th>
<th>( k_1(t) )</th>
<th>( x_2(t) )</th>
<th>( k_2(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-4.5</td>
<td>0.3</td>
<td>-.44142138D01</td>
<td>-.29289322D01</td>
<td>-.28289323D00</td>
<td></td>
</tr>
<tr>
<td>( \pi/4^- )</td>
<td>-.33284271D01</td>
<td>-.33284273D01</td>
<td>.29142136D01</td>
<td>.29142137D01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \pi/4^+ )</td>
<td>-3.4</td>
<td>3.0</td>
<td>-.33284271D01</td>
<td>-.33284273D01</td>
<td>.29142136D01</td>
<td>.29142137D01</td>
</tr>
<tr>
<td>( \pi/2^- )</td>
<td>-.29289322D00</td>
<td>-.29289329D00</td>
<td>.44142136D01</td>
<td>.44142138D01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \pi/2^+ )</td>
<td>-1.3</td>
<td>3.5</td>
<td>-.12928932D01</td>
<td>-.12928933D01</td>
<td>.34142136D01</td>
<td>.34142138D01</td>
</tr>
<tr>
<td>( 3\pi/4^- )</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 3\pi/4^+ )</td>
<td>1.6</td>
<td>3.4</td>
<td>1.5</td>
<td>1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \pi )</td>
<td>.34142136D01</td>
<td>.34142138D01</td>
<td>.12928932D01</td>
<td>.12928933D01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 3. Results for example 3, \( p = 100, k = 7, G = 2.8 \times 10^{-17} \)

<table>
<thead>
<tr>
<th>( t )</th>
<th>( 0x_1(t) )</th>
<th>( 0x_2(t) )</th>
<th>( x_1(t) )</th>
<th>( k_1(t) )</th>
<th>( x_2(t) )</th>
<th>( k_2(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0.1</td>
<td>1.1</td>
<td>0.0</td>
<td>.198466068759D-07</td>
<td>1.0</td>
<td>.999999993233D00</td>
</tr>
<tr>
<td>( \pi/4^- )</td>
<td>.707106781187D00</td>
<td>.707106785811D00</td>
<td>.707106781187D00</td>
<td>.707106766675D00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \pi/4^+ )</td>
<td>0.7</td>
<td>0.7</td>
<td>.707106781187D00</td>
<td>.707106785811D00</td>
<td>.707106781187D00</td>
<td>.707106766675D00</td>
</tr>
<tr>
<td>( \pi/2^- )</td>
<td>1.0</td>
<td>0</td>
<td>.999999992900D00</td>
<td>0.0</td>
<td>-.710027897284D-08</td>
<td>-.710027897284D-08</td>
</tr>
<tr>
<td>( \pi/2^+ )</td>
<td>0.1</td>
<td>-1.1</td>
<td>0.0</td>
<td>-.710027857516D-08</td>
<td>-1.0</td>
<td>-.100000000710D01</td>
</tr>
<tr>
<td>( 3\pi/4^- )</td>
<td>-0.7</td>
<td>3.7</td>
<td>.707106781187D00</td>
<td>.707106786603D00</td>
<td>-.707106781187D00</td>
<td>-.707106785575D00</td>
</tr>
<tr>
<td>( 3\pi/4^+ )</td>
<td>-0.7</td>
<td>-0.7</td>
<td>.707106781187D00</td>
<td>.707106786603D00</td>
<td>-.707106781187D00</td>
<td>-.707106785575D00</td>
</tr>
<tr>
<td>( \pi )</td>
<td>-1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>.100000000000D01</td>
<td>0.0</td>
<td>-.564604950394D-08</td>
</tr>
</tbody>
</table>

Journal of Computational and Applied Mathematics, volume 6, no 2, 1980. 141
Table 4. Convergence rates for examples 1, 2, 3

<table>
<thead>
<tr>
<th>Iterate</th>
<th>G(1)</th>
<th>G(2) (First)</th>
<th>G(2) (Second)</th>
<th>G(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.13424090D 00</td>
<td>.46864343D-01</td>
<td>.99067948D-01</td>
<td>.25796734D 00</td>
</tr>
<tr>
<td>1</td>
<td>.70656904D-09</td>
<td>.29514342D-03</td>
<td>.13416748D-03</td>
<td>.47762329D-01</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>.43146141D-07</td>
<td>.89165205D-08</td>
<td>.11373783D-01</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>.13659226D-15</td>
<td>.24686947D-02</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>.24554665D-03</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>.29503517D-05</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>.42852478D-09</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>.27782667D-16</td>
</tr>
</tbody>
</table>

Table 5. Actual angular velocities, p = 500, k = 3, G = 1.4 \times 10^{-9}

<table>
<thead>
<tr>
<th>t</th>
<th>x_1(t)</th>
<th>x_2(t)</th>
<th>x_3(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.900000000009D 00</td>
<td>.600000000000D 00</td>
<td>.149999999936D 00</td>
</tr>
<tr>
<td>2</td>
<td>.872414246872D 00</td>
<td>.629817462704D 00</td>
<td>-.128016902260D 00</td>
</tr>
<tr>
<td>4</td>
<td>.112781834094D 01</td>
<td>-.116272713056D 00</td>
<td>-.283278619053D 00</td>
</tr>
<tr>
<td>6</td>
<td>.812593113650D 00</td>
<td>-.687218541221D 00</td>
<td>-.615503538527D-01</td>
</tr>
<tr>
<td>8</td>
<td>.993580492174D 00</td>
<td>-.476548375618D 00</td>
<td>.211306114926D 00</td>
</tr>
<tr>
<td>10</td>
<td>.104095297556D 01</td>
<td>.393462421900D 00</td>
<td>.238113181131D 00</td>
</tr>
<tr>
<td>12</td>
<td>.797607128279D 00</td>
<td>.700262201798D 00</td>
<td>-.277874327246D-01</td>
</tr>
<tr>
<td>14</td>
<td>.110458211910D 01</td>
<td>.228962349593D 00</td>
<td>-.271592906781D 00</td>
</tr>
<tr>
<td>16</td>
<td>.914797306578D 00</td>
<td>-.582974626747D 00</td>
<td>-.160800385511D 00</td>
</tr>
<tr>
<td>18</td>
<td>.859870897543D 00</td>
<td>-.642624718677D 00</td>
<td>.116928375741D 00</td>
</tr>
<tr>
<td>20</td>
<td>.113225423267D 01</td>
<td>.774613757148D-01</td>
<td>.285481971051D 00</td>
</tr>
</tbody>
</table>
### Table 6. Estimated angular velocities using noisy measurements

<table>
<thead>
<tr>
<th>t</th>
<th>$x_1(t)$</th>
<th>$x_2(t)$</th>
<th>$x_3(t)$</th>
<th>$\lambda_1(t^+)$</th>
<th>$\lambda_2(t^+)$</th>
<th>$\lambda_3(t^+)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exact initial values $0_x = (0.9, 0.6, 0.15)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.9287D 00</td>
<td>0.5861D 00</td>
<td>0.1410D 00</td>
<td>-0.2875D-01</td>
<td>0.1386D-01</td>
<td>0.9020D-02</td>
</tr>
<tr>
<td>2</td>
<td>0.8120D 00</td>
<td>0.6206D 00</td>
<td>-1.052D 00</td>
<td>0.2930D-01</td>
<td>-0.1800D-02</td>
<td>0.1896D-02</td>
</tr>
<tr>
<td>4</td>
<td>0.141D 01</td>
<td>-0.1264D 00</td>
<td>-0.3131D 00</td>
<td>0.2190D-01</td>
<td>0.2221D-01</td>
<td>0.1523D-01</td>
</tr>
<tr>
<td>6</td>
<td>0.7467D 00</td>
<td>-0.7344D 00</td>
<td>-0.3891D-01</td>
<td>-0.5147D-01</td>
<td>-0.1925D-02</td>
<td>-0.4047D-01</td>
</tr>
<tr>
<td>8</td>
<td>0.1003D 01</td>
<td>-0.4578D 00</td>
<td>0.2234D 00</td>
<td>-0.1612D-01</td>
<td>0.8673D-02</td>
<td>-0.6475D-02</td>
</tr>
<tr>
<td>10</td>
<td>0.1026D 01</td>
<td>0.5000D 00</td>
<td>-0.2372D 00</td>
<td>0.4466D-02</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>12</td>
<td>0.8038D 00</td>
<td>0.7166D 00</td>
<td>-0.4453D-01</td>
<td>0.5487D-02</td>
<td>0.336D-02</td>
<td>0.7706D-02</td>
</tr>
<tr>
<td>14</td>
<td>0.1103D 01</td>
<td>0.1600D 00</td>
<td>-0.3010D 00</td>
<td>0.1017D-01</td>
<td>0.2451D-02</td>
<td>-0.1324D-01</td>
</tr>
<tr>
<td>16</td>
<td>0.7867D 00</td>
<td>-0.6657D 00</td>
<td>-0.1771D 00</td>
<td>0.8798D-02</td>
<td>-0.3810D-02</td>
<td>-0.283D-02</td>
</tr>
<tr>
<td>18</td>
<td>0.6237D 00</td>
<td>-0.7627D 00</td>
<td>0.7565D-01</td>
<td>0.9886D-02</td>
<td>-0.6799D-02</td>
<td>-0.1231D-01</td>
</tr>
<tr>
<td>20</td>
<td>0.1022D 01</td>
<td>-0.2022D 00</td>
<td>0.336D-00</td>
<td>0.1890D-01</td>
<td>-0.2776D-02</td>
<td>-0.2892D-01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inexact initial values $0_x = (0.9, 0.1, 0.1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>20</td>
</tr>
</tbody>
</table>

### Table 7. Convergence rates for example 4

<table>
<thead>
<tr>
<th>Iterate</th>
<th>$G$ (exact measurements)</th>
<th>$G$ (noisy measurements)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$0_x$ exact</td>
<td>$0_x$ exact</td>
</tr>
<tr>
<td>0</td>
<td>0.91623875D-01</td>
<td>0.95230490D-01</td>
</tr>
<tr>
<td>1</td>
<td>0.48239491D-01</td>
<td>0.57285444D-01</td>
</tr>
<tr>
<td>2</td>
<td>0.17552647D-01</td>
<td>0.76232704D-02</td>
</tr>
<tr>
<td>3</td>
<td>0.20318327D-02</td>
<td>0.62961241D-03</td>
</tr>
<tr>
<td>4</td>
<td>0.79779744D-04</td>
<td>0.48461517D-05</td>
</tr>
<tr>
<td>5</td>
<td>0.10908930D-06</td>
<td>0.48796026D-09</td>
</tr>
<tr>
<td>6</td>
<td>0.11362710D-12</td>
<td></td>
</tr>
</tbody>
</table>

Journal of Computational and Applied Mathematics, volume 6, no 2, 1980. 143