Journal of Computational and Applied Mathematics 45 (1993) 103–117
North-Holland

CAM 1286

Linear system solvers for boundary value ODEs

Lixin Liu and Robert D. Russell

Department of Mathematics and Statistics, Simon Fraser University, Burnaby, B.C., Canada

Received 11 November 1991
Revised 28 January 1992

Abstract

Liu, L. and R.D. Russell, Linear system solvers for boundary value ODEs, Journal of Computational and Applied Mathematics 45 (1993) 103–117.

We investigate the stability properties of several linear system solvers for solving boundary value ODEs. We consider the compactification algorithm, Gaussian elimination with row partial pivoting, and a QR algorithm applied to linear systems arising from solving BVPs for which the matrix is block-bidiagonal except for bordering along the last n rows and columns. We will particularly compare AUTO's original linear solver (an LU decomposition with partial pivoting) and our implementation of the analogous QR algorithm to AUTO. Two other factors (the underlying continuation strategy and mesh selection strategy) may affect the stability of the linear system solver for ODE continuation codes as well and are also discussed in our numerical investigations.

Keywords: Boundary value problems; continuation; multiple shooting; finite difference; collocation; compactification; Gaussian elimination; QR factorization; stability.

1. Introduction

Most of the widely-used methods for solving *boundary value problems* (BVPs) for ordinary differential equations (ODEs) involve solving block-bidiagonal linear systems of equations. Both the efficiency and the stability of the linear system solvers are crucial in developing BVP codes. Since these nN linear systems have only n^2N nonzero elements, the solution method should minimize fill-in in order to reduce the storage requirements and CPU-time. At the same time, the solvers should to the extent possible also avoid potential instabilities, and recently several methods have been developed with this in mind, viz., Gaussian elimination with complete row pivoting implemented in the code SOLVEBLOK [6], alternate row and column

Correspondence to: Prof. R.D. Russell, Department of Mathematics and Statistics, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6.

elimination implemented in [7], the DECOMP and SOLVE routines from the code PASVA [13] and a stable block factorization [15,16]. The latter ones incorporate nonseparated boundary conditions which destroy the block-bidiagonal structure.

For BVP continuation codes (involving boundary value problems with at least one parameter), one computes a series of solutions on one or several branches rather than one particular solution of a BVP, and in this context an efficient linear system solver becomes even more important. For this type of problem, the discrete system involves a matrix with block-bidiagonal structure except for nonzero blocks bordering the matrix in the last rows and columns. Stable block factorizations for this type of system have recently been investigated in [15,16]. The system solution is only one of several important factors that affect the stability of a method. Moreover, stability of the solver can be strongly influenced by some other factors, e.g., the mesh selection strategy and the continuation strategy. This will be demonstrated here. One of our main objectives is to analyze and improve the stability properties of the linear system solver for the code AUTO [8,9], popular mathematical software for performing bifurcation analysis. AUTO uses a restricted pivoting strategy when solving the linear systems arising from a collocation method. While the emphasis is on the special features of AUTO, our analysis also applies if the fast solvers considered here were used for other continuation codes, such as COLCON [4].

2. Background

In this section, we briefly review some methods for solving the linear two-point boundary value problem

$$y' = A(t)y + q(t), \tag{1}$$

$$B_a y(a) + B_b y(b) = d, \tag{2}$$

where $t \in [a, b]$, y, $q(t)$, $d \in \mathbb{R}^n$ and $A(t)$, B_a, $B_b \in \mathbb{R}^{n\times n}$.

The widely-used methods which have been developed for solving (1) and (2) are the *multiple-shooting* method, the *finite-difference* method and the *collocation* method. All these methods lead to a similar linear system of equations.

2.1. Multiple-shooting method

Given a mesh $a = t_1 < t_2 < \cdots < t_{N+1} = b$, we compute a fundamental solution $Y_i(t) \in \mathbb{R}^{n\times n}$ and a particular solution $v_i(t) \in \mathbb{R}^n$ on each mesh interval $[t_i, t_{i+1}]$, $1 \leqslant i \leqslant N$, such that

$$Y_i' = A(t)Y_i, \qquad Y_i(t_i) = F_i, \tag{3}$$

$$v_i' = A(t)v_i + q(t), \quad v_i(t_i) = e_i. \tag{4}$$

We then find a vector $s_i \in \mathbb{R}^n$, $1 \leqslant i \leqslant N$, such that

$$y(t) = Y_i(t)s_i + v_i(t), \quad t \in [t_i, t_{i+1}], \quad i = 1, \ldots, N. \tag{5}$$

For the standard multiple shooting, we choose $F_i = I$ and $e_i = \mathbf{0}$. By satisfying the continuity conditions at the mesh-points and the boundary conditions to the endpoints, we obtain

$$\begin{bmatrix} Y_1(t_2) & -I & & & \\ & Y_2(t_3) & -I & & \\ & & \ddots & \ddots & \\ & & & Y_{N-1}(t_N) & -I \\ B_a & & & & B_b \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{N-1} \\ s_N \end{bmatrix} = \begin{bmatrix} -v_1(t_2) \\ -v_2(t_3) \\ \vdots \\ -v_{N-1}(t_N) \\ d \end{bmatrix}. \tag{6}$$

The conditioning of the coefficient matrix has been discussed, for example, in [2,12]. It is well conditioned when N is large enough, its condition number being the same order as the conditioning constants of the BVP itself.

2.2. Finite-difference method

We consider a one-step finite-difference method, choosing the mesh as in the multiple-shooting method. Letting $h_i = t_{i+1} - t_i$ and $t_{i+1/2} = t_i + \frac{1}{2}h_i$, the BVP can be approximated by

$$\begin{bmatrix} S_1 & R_1 & & & \\ & S_2 & R_2 & & \\ & & \ddots & \ddots & \\ & & & S_N & R_N \\ B_a & & & & B_b \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \\ y_{N+1} \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \\ d \end{bmatrix}, \tag{7}$$

where S_i, R_i are $n \times n$ matrices. For the trapezoidal scheme,

$$S_i = -h_i^{-1}I - \tfrac{1}{2}A(t_i), \qquad R_i = -h_i^{-1}I - \tfrac{1}{2}A(t_{i+1}), \qquad q_i = \tfrac{1}{2}[q(t_{i+1}) + q(t_i)], \tag{8}$$

and for the midpoint scheme,

$$S_i = R_i = -h_i^{-1}I - \tfrac{1}{2}A(t_{i+1/2}), \qquad q_i = q(t_{i+1/2}). \tag{9}$$

2.3. Collocation method

Since our main purpose is to examine the code AUTO, we only discuss the collocation method as it is implemented therein with Lagrange basis functions. We try to find

$$p_j(t) = \sum_{i=0}^{m} w_{ji}(t) y_{j+i/m}, \tag{10}$$

where

$$w_{ji}(t) = \prod_{k=0,k\neq i}^{m} \frac{t - t_{j+k/m}}{t_{j+i/m} - t_{j+k/m}}, \qquad t_{j+i/m} = t_j + \frac{i}{m}(t_{j+1} - t_j), \tag{11}$$

such that

$$p_j'(z_{ji}) = A(z_{ji})p_j(z_{ji}) + q(z_{ji}), \tag{12}$$

$$B_a y_1 + B_b y_{N+1} = d, \tag{13}$$

with z_{ji}, $1 \leqslant i \leqslant m$, the Gauss points on the interval $[t_j, t_{j+1}]$, $1 \leqslant j \leqslant N$. The above formulation results in $(mN+1)n$ linear equations. By using the *condensation* method, we can eliminate the unknowns at non-mesh-points, and the linear system is reduced to

$$\begin{bmatrix} A_1 & C_1 & & & \\ & A_2 & C_2 & & \\ & & \ddots & \ddots & \\ & & & A_N & C_N \\ B_a & & & & B_b \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \\ y_{N+1} \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \\ d \end{bmatrix}. \tag{14}$$

2.4. Solution of linear systems

The three linear system (6), (7) and (14) have similar almost block-bidiagonal structure. The conditioning of the systems (7) and (14) is quite similar to that for the multiple-shooting case when N is large [2]. Thus (14) can be reliably solved with a standard linear system solver such as Gaussian elimination with partial pivoting or the QR factorization. However, to preserve the sparsity of the system, special linear system solvers are used. We discuss several of these below.

Compactification

The compactification algorithm is a well-known but unstable method for solving block-bidiagonal linear systems formed when solving the two-point boundary value problem. To study this method, we consider the linear system generated by the multiple-shooting method. We compute a discrete fundamental solution $\{\Phi_i\}_{i=1}^{N+1}$ and particular solution $\{p_i\}_{i=1}^{N+1}$ by

$$p_{i+1} = \Phi_i p_i + v_i(t_{i+1}), \quad \Phi_{i+1} = Y_i(t_{i+1})\Phi_i, \qquad i = 1, \dots, N,$$

with the initial condition

$$p_1 = 0, \qquad \Phi_1 = I,$$

and form

$$s_i = \Phi_i s_1 + p_i, \quad i = 2, \dots, N,$$

where s_1 satisfies

$$[B_a + B_b \Phi_{N+1}] s_1 = d - B_b p_{N+1}.$$

The compactification algorithm, like the *single-shooting* method, suffers from instability because the fastest growing (fundamental solution) mode dominates the others. A similar compacification technique can be applied for other methods by using

$$y_{i+1} = \tilde{q}_i - \tilde{A}_i y_i, \quad \tilde{q}_i = C^{-1} q_i, \quad \tilde{A}_i = C^{-1} A_i, \qquad i = 1, \dots, N,$$

to eliminate all the interior mesh-point unknowns $y_2, \dots, y_N$ and reduce the linear system to a $2n \times 2n$ dense system in the unknowns y_1 and y_{N+1}.

LU decomposition

Gaussian elimination with various pivoting strategies has been used for solving the linear system (14). Although it is well known that the complete row pivoting Gaussian elimination method is stable in practice, it is also known that the worst-case error growth is $O(2^{(N+1)n})$. For the separated boundary conditions case, the linear system is block-bidiagonal and Gaussian elimination with row pivoting can be used to efficiently solve the system

$$\begin{bmatrix} B_a & & & & \\ A_1 & C_1 & & & \\ & \ddots & \ddots & & \\ & & A_N & C_N & \\ & & & B_b & \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \\ y_{N+1} \end{bmatrix} = \begin{bmatrix} d_a \\ q_1 \\ \vdots \\ q_N \\ d_b \end{bmatrix}. \tag{15}$$

This method has been widely used (e.g., software SOLVEBLOK [6] used in COLSYS [1] and COLNEW [3]), and it is generally very stable. In fact the worst-case error growth is exponential only in the bandwidth [5]. However, the nonseparated boundary condition case is more delicate and more important to us because our purpose is to modify AUTO's linear system solver. The well-conditioned BVP has ordinary or exponential dichotomy. For the latter, the pivoting strategy should properly "decouple" the exponentially increasing and exponentially decreasing modes of the fundamental solution. With this idea in mind, Mattheij [15,16] develops a stable block LU decomposition based on Gaussian elimination with row partial pivoting. However, this method is not always practical, because it requires knowing the exact number of increasing modes, and this may not be realistic to assume. Recently, Wright [19] considers applying Gaussian elimination to the columns of the reordered system

$$\begin{bmatrix} C_1 & & & & A_1 \\ A_2 & C_2 & & & \\ & \ddots & \ddots & & \\ & & A_N & C_N & \\ & & & B_b & B_a \end{bmatrix} \begin{bmatrix} y_2 \\ \vdots \\ y_N \\ y_{N+1} \\ y_1 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \\ d \end{bmatrix}. \tag{16}$$

The serial version of this method starts to use Gaussian elimination with row pivoting for the first $2n$ rows of (16) such that

$$\tilde{L}_{1,n}P_{1,n} \cdots \tilde{L}_{1,2}P_{1,2}\tilde{L}_{1,1}P_{1,1}\begin{bmatrix} C_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} U_1 \\ 0 \end{bmatrix},$$

where $P_{1,i} \in \mathbb{R}^{2n\times 2n}$, $i = 1,\dots,n$, are permutation matrices, $L_{1,i} \in \mathbb{R}^{2n\times 2n}$, $i = 1,\dots,n$, are Gauss transformations, and U_1 is upper triangular. The transformed coefficient matrix of (16) becomes

$$\begin{bmatrix} U_1 & E_1 & & & & G_1 \\ & \tilde{C}_2 & E_2 & & & \tilde{G}_2 \\ & A_3 & C_3 & & & \\ & & \ddots & \ddots & & \\ & & & A_N & C_N & \\ & & & & B_N & B_0 \end{bmatrix}. \tag{17}$$

If we apply a similar process to the blocks $[{}^{\tilde{C}_k}_{A_{k+1}}]$ for $k = 2, 3, \ldots, N-1$, we obtain the coefficient matrix

$$\begin{bmatrix} U_1 & E_1 & & & & G_1 \\ & U_2 & E_2 & & & G_2 \\ & & \ddots & \ddots & & \vdots \\ & & & U_{N-1} & E_{N-1} & G_{N-1} \\ & & & & \tilde{C}_N & \tilde{A}_N \\ & & & & B_N & B_0 \end{bmatrix}. \tag{18}$$

It is then easy to solve for y_i, $i = 1, \ldots, N+1$, from the above formulation. Wright [19] proves that under certain assumptions this method is equivalent to Mattheij's block LU decomposition and that it is stable. However, it is not clear how often the required assumptions are satisfied. In fact, if no pivoting occurs between the blocks C_i and A_{i+1}, then this method is equivalent to the compactification algorithm. The worst-case error growth $O(2^{(N+1)n})$ can happen.

QR factorization

Wright [18] also proposes applying a Householder QR decomposition procedure to solve the above linear system (16). The process is similar; instead of applying LU decompositions, we use QR decompositions to obtain the coefficient matrix (18). Wright has proved the stability of this algorithm. The serial version of the LU decomposition is about twice as fast as the QR decomposition, but both the LU decomposition and the QR decomposition have the advantage that they can be easily parallelized [18,19].

To compare the stability properties of these methods, we now study two examples. In our numerical results, the multiple-shooting method uses the ODE initial-value code DVERK from *netlib* and the tolerance in this routine is set to 10^{-10}; the finite-difference code uses the midpoint scheme.

Example 1.

$$y''' = 20y'' + y' - 20y,$$

$$y(0) = 0.1\ e^{-T} + e^{-20T} + 0.1, \qquad y(T) = 1.1 + 0.1\ e^{-T}, \qquad y'(T) = 20.1 + 0.1\ e^{-T}.$$

The exact solution is $y(t) = 0.1\ e^{t-T} + e^{20(t-T)} + 0.1\ e^{-t}$, and the numerical results are given in Table 1.

Table 1
Maximum absolute error for Example 1 (50 mesh intervals)

Time	Multiple-shooting method			Finite-difference method		
	Compact	LU method	QR method	Compact	LU method	QR method
$T=1$	$3.7357\cdot10^{-10}$	$3.4657\cdot10^{-10}$	$3.4657\cdot10^{-10}$	$4.8981\cdot10^{-3}$	$4.8981\cdot10^{-3}$	$4.8981\cdot10^{-3}$
$T=2$	$7.4008\cdot10^{-3}$	$3.1327\cdot10^{-9}$	$3.1326\cdot10^{-9}$	$5.4491\cdot10^{-2}$	$2.0758\cdot10^{-2}$	$2.0758\cdot10^{-2}$
$T=5$	$4.2072\cdot10^{+23}$	$3.8738\cdot10^{-9}$	$3.8738\cdot10^{-9}$	Fails	$1.3534\cdot10^{-1}$	$1.3534\cdot10^{-1}$
$T=10$	$6.8387\cdot10^{+63}$	$1.2815\cdot10^{-9}$	$1.2815\cdot10^{-9}$	$2.6306\cdot10^{+4}$	$3.5170\cdot10^{-1}$	$3.5170\cdot10^{-1}$

Table 2
Maximum absolute error for Example 2 ($T=10$)

Mesh	Multiple-shooting method			Finite-difference method		
	Compact	LU method	QR method	Compact	LU method	QR method
$N=50$	$1.0000\cdot10^{+0}$	$4.3887\cdot10^{-9}$	$4.3887\cdot10^{-9}$	$1.0000\cdot10^{+0}$	$7.0126\cdot10^{-2}$	$7.0126\cdot10^{-2}$
$N=150$	$1.0000\cdot10^{+0}$	$5.9750\cdot10^{-9}$	$5.6750\cdot10^{-9}$	$1.0000\cdot10^{+0}$	$6.8273\cdot10^{-3}$	$6.8273\cdot10^{-3}$
$N=200$	$1.0000\cdot10^{+0}$	$1.0000\cdot10^{+0}$	$4.6708\cdot10^{-9}$	$3.9188\cdot10^{+5}$	$1.0000\cdot10^{+0}$	$3.8006\cdot10^{-3}$
$N=500$	$1.0000\cdot10^{+0}$	$1.0000\cdot10^{+0}$	$3.1246\cdot10^{-11}$	$1.0000\cdot10^{+0}$	$1.0000\cdot10^{+0}$	$6.0204\cdot10^{-4}$

This artificial example is constructed in [12] to show that the compactification algorithm can easily fail, which it does around $T=2$. However, both the LU and QR algorithms, kindly supplied by Wright, solve the problem without any difficulty for much larger values of T.

Example 2.

$$y'=\begin{bmatrix}-1 & 6\\ 6 & -1\end{bmatrix}y,\quad y_1(0)=1+e^{-5T},\quad y_2(T)=1-e^{-7T}.$$

The exact solution is

$$y(t)=\begin{bmatrix}e^{5(t-T)}+e^{-7T}\\ e^{5(t-T)}-e^{-7T}\end{bmatrix}.$$

This example is constructed by Wright to show difficulties for the LU algorithm applied to (16). If we compute the solution for $T=10$ and use an equal-spaced mesh, the compactification algorithm fails for all N, and the LU method fails for both multiple-shooting and finite-difference methods when the number of mesh points is big enough ($N>177$). The QR method performs fine (see Table 2). The failure of the LU decomposition is because there is no pivoting between blocks C_i and A_{i+1}.

3. Linear system solver for AUTO

In this section we study the stability properties of the various system solvers when implemented in AUTO. There are several other factors which affect the stability of the resulting codes, for example, the mesh selection and continuation strategies. We will investigate the role of these factors with a few examples.

AUTO solves ODEs in the form

$$u'(t)=f(u(t),\lambda),\quad t\in[0,1],\quad u,f\in\mathbb{R}^n,\quad \lambda\in\mathbb{R}^{n_\lambda},\tag{19}$$

and subject to the nonseparated boundary condition

$$b(u(0),u(1),\lambda)=0,\quad b\in\mathbb{R}^{n_b},\tag{20}$$

and integral constraints

$$\int_0^1 q(u(t),\lambda)\,dt=0,\quad q\in\mathbb{R}^{n_q},\tag{21}$$

where $n_\lambda = n_b + n_q - n + 1$. The continuation strategy in AUTO uses the pseudo-arclength continuation equation [11]

$$\theta_u^2 \int_0^1 (u(t) - u_0(t))^T \dot{u}_0(t)\,dt + \theta_\lambda^2 (\lambda - \lambda_0)^T \dot{\lambda}_0(t) - \delta s = 0, \tag{22}$$

where (u_0, λ_0) is the previously computed point on the solution branch, $(\dot{u}_0, \dot{\lambda}_0)$ is the normalized direction of the branch at that point, δs is the continuation stepsize, and θ_u and θ_λ are user-defined scaling parameters. The system of ODEs is approximated in AUTO with spline collocation using m Gauss points per subinterval and Lagrange basis polynomials as we discussed in the previous section. The differential equations are approximated by

$$p_j'(z_{ji}) = f(p_j(z_{ji}), \lambda), \quad i = 1, \dots, m, \; j = 1, \dots, N+1; \tag{23}$$

the boundary conditions become

$$b(u_1, u_{N+1}, \lambda) = 0; \tag{24}$$

the integral equations are approximated by a Gaussian quadrature

$$\sum_{j=1}^{N} \sum_{i=0}^{m} \omega_{ji} q(u_{j+i/m}, \lambda) = \mathbf{0}; \tag{25}$$

and the discretization of the pseudo-arclength equation becomes

$$\theta_u^2 \sum_{j=1}^{N} \sum_{i=0}^{m} \omega_{ji} \left[u_{j+i/m} - (u_0)_{j+i/m}\right]^T (\dot{u}_0)_{j+i/m} + \theta_\lambda^2 (\lambda - \lambda_0)^T \dot{\lambda} - \delta s = 0. \tag{26}$$

The above discretization gives $mnN + n_b + n_q + 1$ nonlinear algebraic equations which are then solved by a Newton iteration. When solving the linearized system, all of the local variables are eliminated by a condensation of parameters algorithm, which leaves a linear system of the following form:

$$A\begin{bmatrix} u \\ \lambda \end{bmatrix} = \begin{bmatrix} A_1 & C_1 & & & & D_1 \\ & A_2 & C_2 & & & D_2 \\ & & \ddots & \ddots & & \vdots \\ & & & A_N & C_N & D_N \\ B_0 & B_1 & \cdots & B_{N-1} & B_N & E \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N+1} \\ \lambda \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{N+1} \\ q_\lambda \end{bmatrix}, \tag{27}$$

where $A_i, C_i \in \mathbb{R}^{n \times n}$, $B_i \in \mathbb{R}^{n_r \times n}$, $D_i \in \mathbb{R}^{n \times n_\lambda}$, $E \in \mathbb{R}^{n_r \times n_\lambda}$, $n_r = n_b + n_q + 1$.

This linear system is solved in AUTO by a Gaussian elimination with (row) partial pivoting algorithm similar to Wright's LU decomposition. The pivoting is only applied to the first nN equations and the blocks of B_k, $1 \leqslant k \leqslant N-1$, are eliminated without any pivoting. One advantage of this linear system solver is that the Floquet multipliers of the periodic solutions can be obtained with little extra work. However as we have seen, this type of linear solver has potential instability. An alternative is to modify Wright's QR algorithm to replace the current linear system solver. It is also easy to perform partial pivoting during the condensation of parameters. While we have experimented extensively with this modification to the condensation process, it is usually not critical, and not important to us here, so we will not discuss it further.

We now give a brief description of our implementation of Wright's Householder QR algorithm. Let the Householder transformation $\hat{Q}_1 \in \mathbb{R}^{2n \times 2n}$ satisfy

$$\hat{Q}^{\mathrm{T}}\begin{bmatrix} A_1 & C_1 & & D_1 \\ & A_2 & C_2 & D_2 \end{bmatrix} = \begin{bmatrix} G_1 & R_1 & F_1 & H_1 \\ \tilde{A}_2 & & \tilde{C}_2 & \tilde{D}_2 \end{bmatrix}, \tag{28}$$

where $R_1 \in \mathbb{R}^{n \times n}$ is upper triangular. If $I_M \in \mathbb{R}^{M \times M}$ where $M = (N-2)n + n_r$, then

$$QA = \begin{bmatrix} \hat{Q}^{\mathrm{T}} & \\ & I_M \end{bmatrix} A = \begin{bmatrix} G_1 & R_1 & F_1 & & H_1 \\ \tilde{A}_2 & & \tilde{C}_2 & & \tilde{D}_2 \\ & & A_3 & C_3 & D_3 \\ & & \ddots & \ddots & \vdots \end{bmatrix}. \tag{29}$$

Similarly, let $\hat{Q}_i \in \mathbb{R}^{2n \times 2n}$ satisfy

$$\hat{Q}_i^{\mathrm{T}}\begin{bmatrix} \tilde{A}_i & \tilde{C}_i & & \tilde{D}_i \\ & A_{i+1} & C_{i+1} & D_{i+1} \end{bmatrix} = \begin{bmatrix} G_i & R_i & F_i & H_i \\ \tilde{A}_{i+1} & & \tilde{C}_{i+1} & \tilde{D}_{i+1} \end{bmatrix}, \tag{30}$$

where $1 \leqslant i \leqslant N-1$ and R_i is upper triangular. Let

$$Q_i = \begin{bmatrix} I_{M_1} & & \\ & \hat{Q}_i & \\ & & I_{M_2} \end{bmatrix}, \quad i = 1, \ldots, N-1, \tag{31}$$

where I_{M_1} and I_{M_2} are $(i-1)n \times (i-1)n$ and $(N-i-1)n + n_r \times (N-i-1)n + n_r$ identity matrices. Applying these Householder transformations to A, we obtain

$$Q_{N-1}^{\mathrm{T}} \cdots Q_1^{\mathrm{T}} A = \begin{bmatrix} G_1 & R_1 & F_1 & & & & H_1 \\ G_2 & & R_2 & F_2 & & & H_2 \\ \vdots & & & \ddots & \ddots & & \vdots \\ G_{N-1} & & & & R_{N-1} & F_{N-1} & H_{N-1} \\ \tilde{A}_N & & & & & \tilde{C}_N & \tilde{D}_N \\ B_0 & B_1 & \cdots & \cdots & B_{N-1} & B_N & E \end{bmatrix}. \tag{32}$$

After eliminating blocks $B_1, B_2, \ldots, B_{N-1}$, we obtain the same coefficient matrix structure as does AUTO's original linear system solver. For a periodic solution, it has been shown that the Floquet multipliers are the eigenvalues of the matrix $-\tilde{C}_N^{-1}\tilde{A}_N$ [14].

The original linear system solver in AUTO is virtually the same as the LU algorithm we discussed in the previous section and will thus be referred to below as simply the LU method. As previously mentioned, in the serial version, which is all that has been implemented in AUTO, the QR method roughly doubles the cost of the LU method. In practice, however, because the condensation of parameters takes most of the CPU-time, the actual increase is not very significant. It is worth noting that modification of AUTO's system solver was a nontrivial undertaking, as the code's construction is far from structured, and it is a major undertaking to replace its linear system solver.

Table 3
Maximum absolute error for Example 1 (NTST = 50, NCOL = 2)

Time	AUTO without adaptive mesh (IAD = 200)		AUTO with adaptive mesh (IAD = 1)	
	LU decomposition	QR factorization	LU decomposition	QR factorization
$T=1$	$4.9434\cdot10^{-5}$	$4.9434\cdot10^{-5}$	$2.6954\cdot10^{-6}$	$2.6954\cdot10^{-6}$
$T=2$	$5.9528\cdot10^{-4}$	$5.9528\cdot10^{-4}$	$1.1513\cdot10^{-4}$	$1.1513\cdot10^{-4}$
$T=5$	$1.0737\cdot10^{-2}$	$1.0737\cdot10^{-2}$	$1.6647\cdot10^{-4}$	$1.6647\cdot10^{-4}$
$T=10$	$5.8607\cdot10^{-2}$	$5.8607\cdot10^{-2}$	$6.2498\cdot10^{-4}$	$6.2498\cdot10^{-4}$

To analyze the stability of the linear system solvers, we first test them on Examples 1 and 2 from the previous section. Somewhat surprisingly, not only the QR solver but *also* the LU solver computes the solutions without any difficulty (see Tables 3 and 4). In these tests, we use both fixed and adaptive meshes. One possible explanation for the success of AUTO (with the LU solver) for Example 2 is that the addition of the continuation parameter equation appears to help stabilize the problem.

In order to further challenge the LU method for AUTO, we extend Example 2 to the following example.

Example 3.

$$y' = \begin{bmatrix} -1 & 6 & & \\ 6 & -1 & & \\ & & -1 & 8 \\ & & 8 & -1 \end{bmatrix} y,$$

$$y_1(0) = 1 + e^{-5T}, \qquad y_2(T) = 1 - e^{-7T}, \qquad y_3(0) = 1 + e^{-7T}, \qquad y_4(T) = 1 - e^{-9T}.$$

The exact solution is

$$y = \begin{bmatrix} e^{5(t-T)} + e^{-7t} \\ e^{5(t-T)} - e^{-7t} \\ e^{7(t-T)} + e^{-9t} \\ e^{7(t-T)} - e^{-9t} \end{bmatrix}.$$

Not surprisingly, using the multiple-shooting and finite-difference methods as in Example 2, the only stable linear system solver is the QR decomposition method (see Table 5). We also run

Table 4
Maximum absolute error for Example 2 ($T = 10$, NCOL = 2)

Mesh	AUTO without adaptive mesh (IAD = 50)		AUTO with adaptive mesh (IAD = 1)	
	LU decomposition	QR factorization	LU decomposition	QR factorization
NTST = 50	$3.7409\cdot10^{-3}$	$3.7409\cdot10^{-3}$	$9.6359\cdot10^{-5}$	$9.6359\cdot10^{-5}$
NTST = 150	$8.7258\cdot10^{-5}$	$8.7258\cdot10^{-5}$	$2.1765\cdot10^{-6}$	$2.1765\cdot10^{-6}$
NTST = 200	$3.0056\cdot10^{-5}$	$3.0056\cdot10^{-5}$	$1.0281\cdot10^{-6}$	$1.0704\cdot10^{-6}$
NTST = 500	$8.9945\cdot10^{-7}$	$8.9945\cdot10^{-7}$	$1.0833\cdot10^{-7}$	$1.0833\cdot10^{-7}$

Table 5
Maximum absolute error for Example 3 (multiple-shooting and finite-difference methods, $N = 300$)

Time	Multiple-shooting method			Finite-difference method		
	Compact	LU method	QR method	Compact	LU method	QR method
$T=2$	$7.1638 \cdot 10^{-11}$	$5.0653 \cdot 10^{-10}$	$2.1455 \cdot 10^{-13}$	$1.1039 \cdot 10^{-4}$	$1.1039 \cdot 10^{-4}$	$1.1039 \cdot 10^{-4}$
$T=5$	$4.1914 \cdot 10^{-2}$	$1.3137 \cdot 10^{-1}$	$4.6527 \cdot 10^{-11}$	$7.1832 \cdot 10^{-2}$	$1.3137 \cdot 10^{-1}$	$6.9059 \cdot 10^{-4}$
$T=8$	$1.0000 \cdot 10^{+0}$	$1.0000 \cdot 10^{+0}$	$6.6350 \cdot 10^{-10}$	$1.1553 \cdot 10^{+8}$	$2.3920 \cdot 10^{+0}$	$1.7756 \cdot 10^{-3}$
$T=10$	$1.0000 \cdot 10^{+0}$	$1.0000 \cdot 10^{+0}$	$2.1705 \cdot 10^{-9}$	$1.3333 \cdot 10^{+14}$	$1.2665 \cdot 10^{+0}$	$2.7725 \cdot 10^{-3}$

AUTO, using T as the continuation parameter and starting to compute the solution from $T = 1$. As before, we turn off the adaptive mesh selection (this is done by setting the parameter IAD to a large number). AUTO successfully computes the solution for at least $T = 20$ when using a small number of mesh intervals. However, when we increase the mesh number (parameter NTST) to 300, AUTO fails when $T = 8.9893$. On the other hand, the QR solver is successful in all cases at least until $T = 20$ (see Table 6). This seems to demonstrate that the continuation parameter equation in AUTO helps to stabilize the problem when there is only one increasing mode, but with two increasing modes, instability can occur.

Interestingly, when we turn on the mesh selection, then in all of the above cases, AUTO successfully computes the solution with the LU method. This shows how a good mesh selection strategy can help to stabilize the problem. Presumably, this stabilizing effect of the adaptive mesh selection would carry over for many other BVP solvers.

Finally, we give a practical example to show how the QR solver can succeed when the LU solver fails.

Example 4 (*Kuramoto–Sivashinsky equation* [10]).

$$\frac{\partial u}{\partial t} + 4\frac{\partial^4 u}{\partial x^4} + \alpha\left[\frac{\partial^2 u}{\partial x^2} + u\frac{\partial u}{\partial x}\right] = 0, \quad (x, t) \in \mathbb{R} \times \mathbb{R}^+,$$

$$u(x, t) = u(x + 2\pi, t), \qquad u(x, t) = -u(2\pi - x, t).$$

Following [17], we use a spectral method to discretize in x, use a traditional Galerkin method to form a system of ODEs for the resulting series coefficients, and solve these resulting ODEs for periodic solutions with AUTO. The resulting bifurcation analysis of this problem has

Table 6
Maximum absolute error for Example 3 (use AUTO, NTST = 300, NCOL = 2)

Time	AUTO without adaptive mesh (IAD = 50)		AUTO with adaptive mesh (IAD = 1)	
	LU decomposition	QR factorization	LU decomposition	QR factorization
$T=2$	$4.7525 \cdot 10^{-8}$	$4.7525 \cdot 10^{-8}$	$3.9444 \cdot 10^{-8}$	$3.9444 \cdot 10^{-8}$
$T=5$	$1.1764 \cdot 10^{-6}$	$1.1764 \cdot 10^{-6}$	$5.8461 \cdot 10^{-9}$	$5.8461 \cdot 10^{-9}$
$T=8$	$7.2077 \cdot 10^{-6}$	$7.2077 \cdot 10^{-6}$	$1.0866 \cdot 10^{-8}$	$1.0866 \cdot 10^{-8}$
$T=10$	Fails at $T = 8.9893$	$1.6817 \cdot 10^{-5}$	$3.1209 \cdot 10^{-8}$	$3.1209 \cdot 10^{-8}$

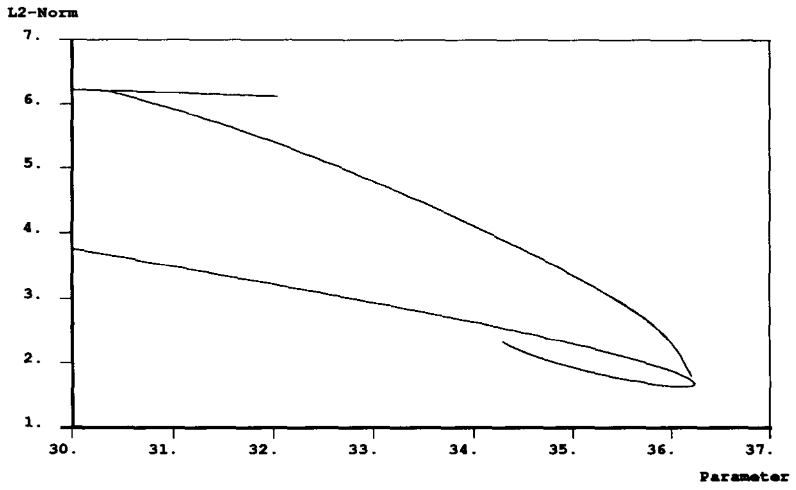

Fig. 1. Use of the LU method for solving the KS equation.

been discussed by many authors [10,17]. There is a Hopf bifurcation point around $\alpha = 30.34522$ which lies on a secondary steady-state solution branch (the upper curve in Figs. 1 and 3). We use a 12-mode traditional Galerkin method with ten mesh intervals and four collocation points per interval to follow the periodic branch starting from this Hopf bifurcation point (the middle curve). Both the LU and QR methods fail after 115 continuation steps when $\alpha = 35.97086$. After increasing the number of mesh intervals to 40 and restart from $\alpha = 35.37104$, the LU method fails again after only 39 continuation steps when it reaches the turning point at $\alpha = 36.19939$. With the QR decomposition method, AUTO continues following this branch for

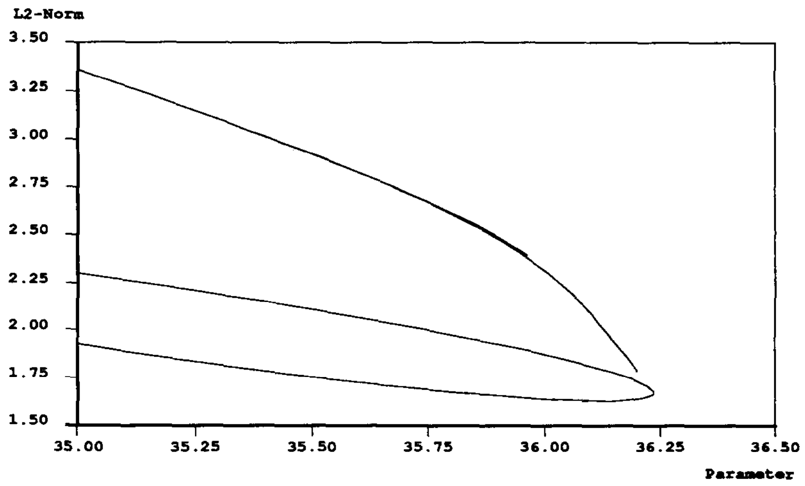

Fig. 2. Use of the LU method for solving the KS equation (blowup).

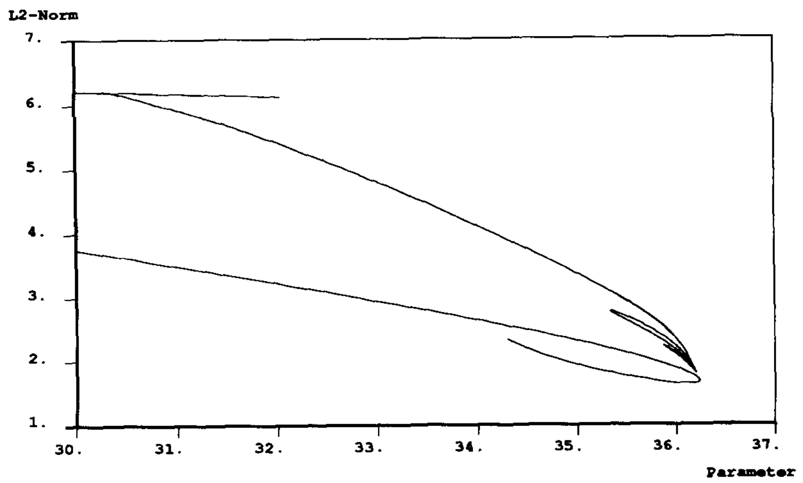

Fig. 3. Use of the QR method for solving the KS equation.

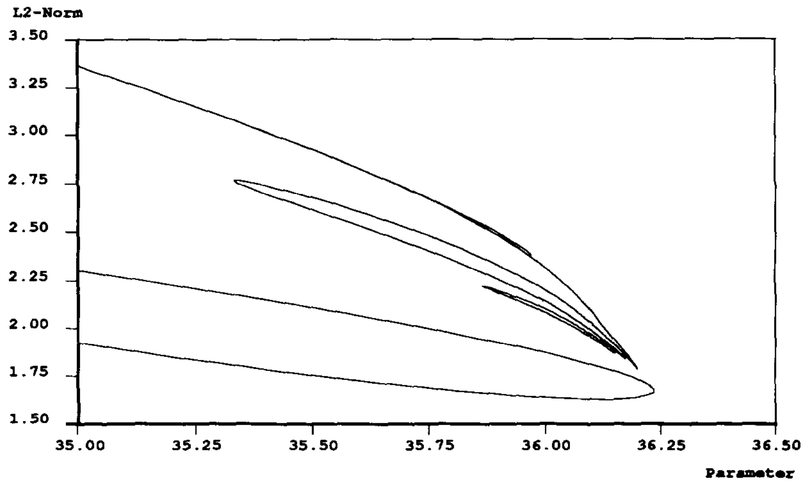

Fig. 4. Use of the QR method for solving the KS equation (blowup).

another 305 steps without any difficulties (see Figs. 1–4). In these figures, the lower curve is another steady-state solution branch which is not important to our discussion here.

4. Conclusion

The linear system of equations generated by the multiple-shooting method, the finite-difference method and the collocation method for solving boundary value problems of ODEs can have a similar structure and conditioning. Specifically, this linear system is well-conditioned if the BVP is well-conditioned and if the number of mesh intervals N is large enough. However, instability can arise from the linear system solver. Compactification is one of the well-known examples of an unstable linear solver of this type. When applying Gaussian elimination with partial pivoting to solve this linear system, it is generally stable if the pivoting starts in blocks associated with the boundary conditions at the left endpoint, but instability can more easily happen if the boundary conditions are not involved in the pivoting process until the end. This type of linear solver is used in AUTO in order to obtain the Floquet multipliers of the periodic solution without significant extra cost. In the continuation code, the equations for the continuation parameters can help stabilize the linear solver, but the potential instability still exists, as we have shown in Example 3. To overcome the instability problems of Gaussian elimination and still preserve the structure of the Jacobian in AUTO to obtain the Floquet multipliers with little extra cost, we have used a QR factorization to replace the current LU decomposition algorithm in AUTO. The extra cost of the QR factorization versus the LU decomposition is not very significant in AUTO, since the linear solver usually takes less than 15% of the total CPU-time in the serial version. A good mesh selection strategy can also help to stabilize the linear system solver. Investigations into why the continuation and mesh selection strategies can stabilize the linear system solvers will be carried out in the future.

References

[1] U. Ascher, J. Christiansen and R.D. Russell, Collocation software for boundary value ODE's, *ACM. Trans. Math. Software* **7** (1981) 209–222.

[2] U. Ascher, R.M.M. Mattheij and R.D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations* (Prentice-Hall, Englewood Cliffs, NJ, 1988).

[3] G. Bader and U. Ascher, A new basis implementation for a mixed order boundary value solver, *SIAM J. Sci. Statist. Comput.* **8** (1987) 483–500.

[4] G. Bader and P. Kunkel, Continuation and collocation for parameter-dependent boundary value problems, *SIAM J. Sci. Statist. Comput.* **10** (1989) 72–88.

[5] Z. Bohte, Bounds for rounding errors in Gaussian elimination for band systems, *J. Inst. Math. Appl.* **16** (1975) 790–805.

[6] C. de Boor and R. Weiss, SOLVEBLOK: A package for solving almost block diagonal linear systems, *ACM Trans. Math. Software* **6** (1980) 80–87.

[7] J.C. Diaz, G. Fairweather and P. Keast, FORTRAN packages for solving certain almost block diagonal linear systems by modified alternate row and column elimination, *ACM Trans. Math. Software* **9** (1983) 358–375.

[8] E.J. Doedel, AUTO: a program for the automatic bifurcation analysis of autonomous systems, *Congr. Numer.* **30** (1981) 265–284.

[9] E.J. Doedel and J.P. Kernevez, AUTO: software for continuation and bifurcation problems in ordinary differential equations, Technical Report, Appl. Math., California Inst. Techn., Pasadena, CA, 1986.

[10] M.S. Jolly, I.G. Kevrekidis and E.S. Titi, Approximate inertial manifolds for the Kuramoto–Sivashinsky equation: analysis and computations, *Phys. D* **44** (1990) 38–60.

[11] H.B. Keller, Numerical solution of bifurcation and nonlinear eigenvalue problems, in: P.H. Rabinowitz, Ed., *Proc. Applications of Bifurcations Theory* (Academic Press, New York, 1977) 359–384.

[12] M. Lentini, M.R. Osborne and R.D. Russell, The close relationships between methods for solving two-point boundary value problems, *SIAM J. Numer. Anal.* **22** (1985) 280–309.

[13] M. Lentini and V. Pereyra, An adaptive finite difference solver for nonlinear two-point boundary value problems with mild boundary layers, *SIAM J. Numer. Anal.* **14** (1977) 91–111.

[14] L. Liu and R.D. Russell, Boundary value ODE algorithms for bifurcation analysis, in: R. Vichnevetsky and J.J.H. Miller, Eds., *Proc. 13th IMACS World Congress on Comput. Appl. Math.*, Dublin, Ireland, 1991 (BAIL Publishers, Dublin) 302–303.

[15] R.M.M. Mattheij, Stability of block LU-decomposition of matrices arising from BVP, *SIAM J. Algebraic Discrete Methods* **5** (1984) 314–331.
[16] R.M.M. Mattheij, Decoupling and stability of algorithms for boundary value problems, *SIAM Rev.* **27** (1985) 1–44.
[17] R.D. Russell, D.M. Sloan and M.R. Trummer, Some numerical aspects of computing inertial manifolds, *SIAM J. Sci. Statist. Comput.*, to appear.
[18] S.J. Wright, Stable parallel algorithms for two-point boundary value problems, *SIAM J. Sci. Statist. Comput.* **13** (1991) 742–764.
[19] S.J. Wright, Stable parallel elimination for boundary value ODE's, *SIAM J. Sci. Statist. Comput.*, to appear.
[20] S.J. Wright, A collection of problems for which Gaussian elimination with partial pivoting is unstable, *SIAM J. Sci. Statist. Comput.*, to appear.