Performance Survey of Robust Pole Placement Methods

Amit Pandey, Robert Schmid, Thang Nguyen, Yaguang Yang, Vasile Sima and André L. Tits

Abstract—The classic problem of robust pole placement for linear time invariant systems via state feedback has been studied for several decades, and involves obtaining a gain matrix that will assign a certain desired set of closed-loop poles, while also providing a robust eigenstructure that is insensitive to uncertainties in the system matrices. There are several ways of measuring the robustness of the eigenstructure, and numerous methodologies have appeared in the literature to address the problem. In this paper, results from extensive experiments comparing the performance of a number of methods—including variations on two methods previously proposed by the present authors—against a variety of robustness measures are reported.

The size of the matrix gain, runtime and accuracy of the pole placement achieved by each method are also compared. The results show some notable differences between the methods surveyed.

I. INTRODUCTION

The problem of robust pole placement for LTI systems considers systems in state space form

$$\dot{x}(t) = Ax(t) + Bu(t),$$

(1)

where, for all $t \in \mathbb{R}$, $x(t) \in \mathbb{R}^n$ is the state, and $u(t) \in \mathbb{R}^m$ is the control input. We assume that $B$ has full column rank. We let $\mathcal{L} = \{\lambda_1, \ldots, \lambda_\nu\}$ be a self-conjugate set of $\nu$ complex numbers in the LHP, with associated algebraic multiplicities $M = \{m_1, \ldots, m_\nu\}$ satisfying $m_1 + \cdots + m_\nu = n$, $m_i \leq m$ for all $i \in \{1, \ldots, \nu\}$. For pairs of complex conjugate poles with $\lambda_i = \bar{\lambda}_i$, we require $m_i = m_\nu$. The classic problem of nondefective exact pole placement by state feedback (EPP) is that of finding a gain matrix $F$ such that the closed-loop matrix $A + BF$ is non-defective and satisfies

$$(A + BF)X = AX,$$

(2)

where $\Lambda$ is an $n \times n$ diagonal matrix with spectrum specified by $(\mathcal{L}, M)$, and $X$ is a non-singular matrix of associated closed-loop eigenvectors of unit length. If $(A, B)$ has any uncontrollable modes, they are assumed to be included within the set $\mathcal{L}$. For single-input systems, there exists a unique matrix $F$ to yield the desired poles. However, for multiple-input systems, $F$ is not unique, and this naturally invites the selection of $F$ that also possesses other desirable control characteristics. The robust exact pole placement problem by state feedback (REPP) involves solving the EPP problem and also obtaining $F$ that renders these closed-loop eigenvalues as insensitive to perturbations in $A + BF$ as possible.

In the following we will use $I_n$ to denote the identity matrix of dimension $n$, and for any matrix $M$, we use $M^*$ to denote its conjugate transpose, $M^T$ to denote its transpose, and $\ker(M)$ denotes its kernel. For any square matrix $M$, we use $\sigma(M)$ to denote its spectrum.

For square matrices, a variety of measures have been proposed for the robustness of their eigenstructure. When all the eigenvalues are simple, the first order sensitivity of each individual $\lambda_i$ to uncertainty in $M$ is given by the eigenvalue condition number [1]

$$c_i(X) := \frac{\|y_i\|_2\|x_i\|_2}{\|y_i^T x_i\|_2},$$

(3)

where $y_i$ and $x_i$ are the left and right eigenvectors of $M$ associated with $\lambda_i$ and $X$ is the matrix of right eigenvectors; $c_i(X)$ is the Frobenius norm of the gradient of $\lambda_i(M)$ with respect to $M$ under the (natural) trace inner product. We use

$$c_\infty(X) := \max_i c_i(X)$$

(4)

to denote the worst-case eigenvalue condition number. In the numerical experiments presented in this survey, we will only consider systems with simple closed-loop eigenvalues.

The Bauer-Fike theorem (e.g. [2]) establishes that $c_\infty(X)$ is upper-bounded by the spectral condition number of the matrix of eigenvectors

$$\kappa_2(X) := \|X\|_2^2 \|X^{-1}\|_2,$$

(5)

and hence $\kappa_2(X)$ is often used as a robustness measure. The Frobenius condition number of $X$ is given by

$$\kappa_{\text{fro}}(X) := \|X\|_{\text{fro}} \|X^{-1}\|_{\text{fro}} \geq \kappa_2(X).$$

(6)

$\kappa_{\text{fro}}(X)$ provides a more conservative bound on the eigenvalue sensitivity than $\kappa_2(X)$, but enjoys the virtue of being differentiable, and hence is often also used as a robustness measure. Minimizing the measures $c_\infty(X)$, $\kappa_2(X)$ and $\kappa_{\text{fro}}(X)$ corresponds to superior robustness, with perfect robustness being achieved only when the eigenvector matrix is unitary, i.e. when $M$ is normal.

The REPP problem with state feedback has been studied for several decades, and numerous methods have been proposed to obtain a robust eigenstructure for $A + BF$. Kautsky et al [3] developed their Method 0 to seek an eigenvector matrix that is approximately unitary. Two of the authors of
the present paper proved in [4] that this method is equivalent to a sequential maximization of

$$|\det(X)| = \sqrt{\det(X^*X)},$$

which is the volume of the box spanned by the (unit length) column vectors of $X$ and is a good measure of orthogonality, which may be used as the robustness measure.

In addition to the robustness of the closed-loop eigenstructure, other important considerations are (i) the norm of the gain matrix $F$, since it is related to the control amplitude or energy required to achieve any desired closed loop performance objective; (ii) the accuracy of the closed-loop assignment (distance between the eigenvalues of $A + BF$ and the corresponding $\lambda_i$ in $L$); and (iii) the computational cost, which may be an overriding factor when solutions are required to be computed online.

Over the past few decades numerous authors have addressed the robust pole placement problem, offering methodologies to optimise one or another of these performance measures. Notable early contributions include Kautsky et al [3], which gave a method of selecting an initial candidate set of closed-loop eigenvectors and then used a variety of heuristic methods to make these vectors more orthonormal. Subsequently two of the present authors revisited the heuristic methods of [3] and offered a range of improvements [4]. Byers and Nash [5], Tam and Lam [6] and Varga [7] took $\kappa_F(X)$ as their robustness measure and cast the REPP problem as an unconstrained nonlinear optimization problem, to be solved by gradient iterative search methods. Recent contributors in this area include Li et al [8], who introduced a method for minimizing the ‘departure from normality’ robustness measure, and Ait Rami et al [9], who introduced a global constrained nonlinear optimal problem for the minimisation of the Frobenius condition number and showed that the solution could be approximated by a global optimization problem under LMI constraints, for which the authors gave an LMI-based algorithm.

Very recently Le and Wang [10] used a ‘neurodynamics’ optimization approach using recurrent neural networks, taking the spectral condition number as their robustness measure. The recent work [11] by three of the present authors adapted the classic pole placement method of Moore [12] to obtain a parametric formula for both $X$ and $F$, and addressed the robustness problem by minimizing the Frobenius condition number of $X$ via gradient search methods. The approach resembled that of [5], but with a different parametric formulation for the pole-placing gain matrix.

A related optimal control problem is the minimum gain exact pole placement problem (MGEPP), which involves solving the EPP problem and also obtaining the feedback matrix $F$ that has the least gain (smallest matrix norm), which gives a measure of the control amplitude or energy required by the control action. Recent papers addressing the MGEPP with minimum Frobenius norm for $F$ include [13] and [14]. The methods of [7] and [11] can be adapted to address the MGEPP problem. Lastly we note the paper of [15], which gave a method for the solution of the EPP but did not address any optimal control problem; the authors argued that the computational efficiency of their method facilitated its application to large scale systems.

All of the above mentioned papers were claimed by their authors to offer some advantages over their predecessors, using applications to some sample systems to show their ability to deliver better performance on one or more of the performance measures (4)-(7). However, testing on just a few low-dimensional examples does not allow any general conclusions to be drawn as to relative merits of each method over the alternatives. To date there have been very few papers offering performance comparisons drawn from testing on large numbers of sample systems.

One such survey was given by three of the present authors in [16], which surveyed the relative performance of place and rohpole on sets of up to 45 sample systems. rohpole was seen to offer superior robustness to place, with respect to the $\kappa_F(X)$ robustness measure. The paper [11] by the first three of the present authors conducted extensive numerical testing involving sets of 500 sample systems, and compared the methods of [3], [4], [5], [7] and [9] against the proposed method of that paper. The methods of [3], [4] and [7] were implemented in the forms of the MATLAB® toolboxes place, rohpole and sylvplace respectively. MATLAB® toolboxes were developed for each of [5], [9] and the proposed method, and these were referred to as byersnash, rfbt and span. Methods byersnash, sylvplace and span were shown to offer the best robustness performance with respect to the Frobenius condition number, method span of [11] was able to achieve this performance with smaller gain than byersnash, and better accuracy than sylvplace. Substantial differences in the runtimes were also observed, with place and rohpole the quickest by orders of magnitude.

As the Frobenius condition number is a conservative upper bound for the eigenvalue sensitivity, in this paper we revisit the numerical experiments of [11], using instead $c_\infty(X)$ and $|\det(X)|$ as the robustness measures. Indeed, for simple eigenvalues, $c_\infty(X)$ is a more fundamental measure of robustness, and $|\det(X)|$ is a natural geometric measure of closeness to orthogonality. We also compare the accuracy and runtime of these methods, and the size of the matrix gain they produce. All the methods considered in [11] will again be included in the present survey, except for rfbt [9], as the experiments in [11] showed that this method gives inferior performance on all the criteria discussed in that paper. We also offer some improvements to the robust pole placement methods of [4] and [11]; they are implemented within the revised MATLAB® toolboxes rohpole 1.5.1 and span2.

We conclude by comparing and contrasting the differences of these methods across the various performance criteria. The large number of sample systems considered permits some general assessments to be made. In particular, we note that the results for the robustness performance criteria $c_\infty(X)$ and $|\det(X)|$ are quite different from the results observed in [11] with respect to $\kappa_F(X)$. Thus the paper offers further insights into the relative performance of the methods surveyed, according to these alternative performance measures.
II. REVIEW OF ROBUST POLE PLACEMENT METHODS

In this section we briefly summarise the pole placement methods to be compared in this paper, and their implementation within the MATLAB® toolboxes we will use to compare their performance. All the methods surveyed in this paper can obtain a suitable feedback matrix $F$ to solve the EPP problem (2) for the case where the set of desired closed-loop poles $\mathcal{L}$ contains complex conjugate pairs, and poles may be repeated with multiplicity up to at most $m = \text{rank}(B)$. For simplicity and brevity, the presentation given below makes the following additional assumptions:

1) the pair $(A, B)$ is controllable;
2) the desired poles in $\mathcal{L}$ are all real and distinct ($\nu = n$).

A. Method 0 of Kautsky, Nichols and van Dooren [3]

Theorem 2.1: ([3, see Theorem 3]) $F$ exists if and only if

$$U_1^T (AX - XA) = 0,$$

where

$$B = [U_0 \ U_1] \begin{bmatrix} Z \\ 0 \end{bmatrix},$$

with $U := [U_0 \ U_1]$ orthogonal and $Z$ nonsingular. Then $F$ is given by

$$F := Z^{-1} U_0^T (XAX^{-1} - A).$$

The authors extended the work of Kautsky et al to accommodate the assignment of complex eigenvalues. Together with V. Sima, they developed their Method 5.1a into the MATLAB® toolbox robpole. In the present paper an updated version of robpole is used, with version number robpole 1.5.1. In earlier versions the search was always initialized according to a default option, while in this version the user can provide initialization input with random weights.

D. The method of Varga [7]

The author used the parametric form for $X$ and $F$ solving (2) that appeared in [17]:

Theorem 2.3: ([17, see Main Theorem]) Let $\tilde{A}$ be a diagonal matrix such that $\sigma(A) = \mathcal{L}$, and assume $\sigma(\tilde{A}) \cap \sigma(A) = \emptyset$. Let $G$ and $X$ satisfy

$$AX - X\tilde{A} + BG = 0.$$  

Then $F$ is given by $F := GX^{-1}$. Similarly to the method of Byers and Nash, the author introduced the unconstrained optimisation problem of minimising the objective function

$$J(G) := \kappa_{\text{info}}(X(G))$$

in terms of the parameter matrix $G$. The paper offers a simple formula for the gradient of $J$ in terms of $G$, and used gradient search methods to find local minima. The method was developed into the MATLAB® toolbox known as sylvplace.


The authors adapted the classic pole placement algorithm of Moore [12] to obtain a simple parametric formula for matrices $X$ and $F$ that solve (2) as follows:

Theorem 2.4: ([11, see Proposition 2.1]) For each $i \in \{1, \ldots, n\}$, define the matrix pencil

$$S(\lambda_i) := [A - \lambda_i I_n \ B].$$

Let $T(\lambda_i)$ be a basis matrix for the nullspace of $S(\lambda_i)$; then $\text{rank}(T) = m$, due to the controllability of $(A, B)$. Let $M$ be an $(n + m) \times n$ matrix given by

$$M := [T(\lambda_1) | \ldots | T(\lambda_\nu)]K,$$
where $K$ is a real block-diagonal matrix with $i$th block of size $m \times m_i$, and let

$$X := P_{1:n}(M),$$

(19)

$$W := P_{n+1:n+m}(M),$$

(20)

where $P_{1:n}(M)$ and $P_{n+1:n+m}(M)$ denote the first $n$, and the last $m$, rows of the matrix $M$, respectively. If $X$ is non-singular, then $F$ is given by

$$F := WX^{-1}.$$  

(21)

Similarly to the method of Byers and Nash, three of the present authors introduced the unconstrained optimisation problem of minimising the objective function

$$f(K) = \kappa_{\text{fo}}(X(K))$$

(22)

in terms of the parameter matrix $K$, and used gradient search methods to find local minima. The method was developed into a MATLAB® toolbox known as span.

In this paper we introduce a modification to the method of [11], to consider the alternative objective function

$$g(K) := c_{\infty}(X(K)).$$

(23)

Minimising this worst-case eigenvalue sensitivity measure introduces an unconstrained minimax problem, requiring gradient information on $c_{\infty}(X)$ in (3) in terms of the parameter $K$; this can be obtained in terms of the derivatives with respect to $K$ of the eigenvector matrix $X$ and its inverse. Our MATLAB® toolbox for this modified method employs the fminimax command and is called span2.

III. PERFORMANCE COMPARISON OF POLE PLACEMENT METHODS

In this section we compare the performance of the methods discussed in Section II on randomly generated systems, using the MATLAB® testing suite that was developed for the survey given in [11]. Using randomly generated systems will enable us to readily generate large collections of systems of specified dimensions, and applying the ‘multiplicative averaging’ technique defined in (26) below yields comparative performance indices. We now describe how we formulated our numerical experiments to best reveal the particular merits (and drawbacks) of each method in our survey.

A. The Role of Initial Points and Stopping Conditions

A common feature of all the methods introduced above is that they utilise a parametric formula that generates all possible eigenvector matrices $X$ and gain matrices $F$ solving (2). All the algorithms begin with an initial point $X_0$ and then use a variety of methods to find better solutions $X$ with improved robustness. Thus place commences with an initial eigenvector matrix $X_0$ with each column vector $x_i$ of $X_0$ such that $x_i \in S_i$ is a unit vector. For robpole 1.5.1, an orthonormal basis matrix $\Sigma_i$ is firstly obtained for $S_i$. Then the columns of the initial eigenvector matrix $X_0$ are obtained by first generating $\Xi_i = \text{rand}(m,1)$, then using normalization for $x_i = \Sigma_i \Xi_i$ of (12) to obtain the initial point $X_0$. byersnash uses an initial parameter matrix $\xi$ to generate $X_0$ via (12)-(13). sylvplace uses an initial parameter matrix $G$ to obtain $X_0$ from (15). Finally span2 uses an initial parameter matrix $K$ to generate $X_0$ via (18)-(19). In the case of byersnash, sylvplace and span2, the initial parameter matrices used to obtain the initial point $X_0$ are randomly generated from the standard uniform distribution on the open interval $(0, 1)$ with the MATLAB® rand command.

For byersnash and sylvplace, the gradient search is implemented using the MATLAB® command fminunc, while span2 uses the MATLAB® command fminimax. The gradient search ceases when either the objective function or the step size increment changes by less than $10^{-6}$. For robpole 1.5.1, the algorithm stops when a user-specified number of iterations, or sweeps, of the algorithm have been completed. The default option for this has been set at 5 sweeps. The output from each algorithm is a gain matrix $F$ and normalised eigenvector matrix $X$.

B. The Core Testing Suite

The paper [11] introduced a suite of performance testing methods known as core, and this suite will again be used in the present study. core consists of three sets of 500 randomly generated sample systems $(A, B)$ with closed-loop eigenvalues $\mathcal{L}$, all of state dimension $n = 20$, and with control input dimensions of $m = 2$, $m = 4$ and $m = 8$. The pole positions $\mathcal{L}$ for each system are all distinct, and do not coincide with the open-loop poles. They include at least one and at most five complex conjugate pairs; the number of such pairs is generated by MATLAB®’s randi command. The entries of $A$, $B$ and $\mathcal{L}$ take uniformly distributed values within the interval $[-2, 2]$.

In order to test the methods in a manner that takes account of their computational complexity, we chose to grant each method 20 seconds of runtime. The methods were coded to randomly generate $X_0$ as outlined above and then apply their optimisation methods until the runtime expired; the outputs were selected to be the $F$ and $X$ that gave the best values for the user-specified performance measure. For systems with $n = 20$ states and $m = 2$ inputs, we observed the number of different initial conditions executed by each method were as follows: byersnash: 9, robpole 1.5.1: 44, sylvplace: 240 and span2: 4. These numbers show the execution speed of the methods vary considerably; granting them all an equal amount of runtime and then selecting the best performing gain matrix thus provides an equitable basis for comparison that rewards the more computationally efficient methods.

To compare the robustness of each method, we will use the $c_{\infty}(X)$ measure, and also introduce

$$\Gamma(X) := 1 - \log(|\det(X)|),$$

(24)

which carries the same information as $|\det(X)|$. The use of $\Gamma(X)$ is preferred for consistency with $c_{\infty}(X)$ in that smaller values correspond to superior robustness. Since computation of $\log(|\det(X)|)$ is not numerically reliable in MATLAB®, in our implementation we instead computed

$$\Gamma(X) = 1 - \sum_{i} n \log(s_i),$$

(25)
where \( s_i \) denotes the \( i \)-th singular value of \( X \). We also considered the accuracy and matrix gain of each method. Using \( \star \) to denote any of the methods \textit{place}, \textit{robpole 1.5.1}, \textit{byersnash}, \textit{sylvplace}, \textit{span2}, we computed, for each system \( j \in \{1, \ldots, 500\} \) and each method \( \star \),

- \( c_\infty(\star, j) \): the \( c_\infty(X) \) condition number of method \( \star \) for the \( j \)-th system;
- \( \Gamma(\star, j) \): the value of \( 1 - \log(\|\det(X)\|) \) for method \( \star \) for the \( j \)-th system;
- \( \Delta(\star, j) \): the accuracy of method \( \star \) on the \( j \)-th system, equal to \( \max\{|\text{eig}(A+BF)-\lambda_j| : i \in \{1, \ldots, n\}\} \), the largest absolute value difference between each eigenvalue of \( A+BF \) and the corresponding \( \lambda_j \) in \( \mathcal{L} \); and
- \( F(\star, j) \): the spectral norm \( \|F\|_2 \) of \( F \) from method \( \star \) on system \( j \).

Noting that \textit{place} is the industry standard for the REPP, \textit{core} compares all the other methods according to their ability to improve upon \textit{place}. We computed comparative performance index functions \( \text{ind}(\star, \bullet) \) relative to \textit{place} for each method \( \star \), and for each performance criterion \( \bullet \in \{c_\infty, \Gamma, \Delta, F\} \), as follows:

\[
(1 - \text{ind}(\star, \bullet))^{500} = \prod_{j=1}^{500} \frac{\bullet(\star, j)}{\bullet(\text{place}, j)}. \tag{26}
\]

For example, if \( \text{ind}(\textit{robpole 1.5.1}, F) = 0.1 \), then method \textit{robpole 1.5.1} gives values of \( \|F\|_2 \) that are on average 10\% smaller than \textit{place}. Larger indices imply greater improvement on \textit{place}, and negative indices indicate performance inferior to \textit{place}.

C. Results of the Numerical Experiments

All algorithms discussed in Section II were implemented on an Intel® Core™ Quad CPU, Model Q9400 at 2.66 GHz with 3326 MB of RAM running Windows® XP and MATLAB® 2012a. The results of our numerical experiments appear in Tables I-III. Each method was given 20 seconds of runtime per sample system, and the best performing gain matrix was then chosen, according to each of the robustness measures \( c_\infty(X) \) and \( \Gamma(X) \). The corresponding values of \( \|F\|_2 \) and \( \Delta \) delivered by the best-performing matrix were also computed. The improvement indices, relative to \textit{place}, were then computed and these appear in the table.

Thus for systems with \( m = 2 \) (Table 1), when using method \textit{robpole 1.5.1} and selecting the gain matrix that produced the best \( c_\infty(X) \) condition number, an improvement of 16.48\% over \textit{place} was observed. The matrices giving the best \( c_\infty(X) \) performance averaged a 4.086\% improvement over \textit{place} in the gain \( \|F\|_2 \), and a 28.08\% improvement in the accuracy \( \Delta \) of their pole placement. Similarly, when using method \textit{robpole 1.5.1} for systems with \( m = 2 \) and selecting the gain matrix that produced the best \( \Gamma(X) \) condition number, an improvement of 0.7481\% over \textit{place} is observed. The matrices giving the best \( \Gamma(X) \) performance averaged a 4.242\% improvement over \textit{place} in the gain \( \|F\|_2 \), and a 22.57\% improvement in the accuracy \( \Delta \) of their pole placement.

The results show some interesting differences in the performance of the methods across the various performance criteria. We note that all the methods were able to offer some improvement over \textit{place}, on at least one of the robustness measures. For the \( c_\infty(X) \) condition number and for systems with \( m = 2 \) inputs, \textit{byersnash}, \textit{sylvplace}, \textit{robpole 1.5.1} and \textit{span2} all improved upon \textit{place}, with the largest improvement of 70.54\% beng achieved by \textit{span2} in Table I. These results may be compared with those in [11] where, \textit{byersnash}, \textit{sylvplace} and \textit{span} all achieved roughly a 60\% improvement for the \( c_\infty(X) \) over \textit{place}; we observe that \textit{span2} was able to achieve a further 10\% improvement over \textit{span} on this measure. When we considered the eigenvector orthonormality measure \( \Gamma(X) \), a different story emerged. Here \textit{place} and \textit{robpole 1.5.1} both offered superior performance to the gradient search methods \textit{byersnash}, \textit{sylvplace} and \textit{span2}.

The relative performances noted above in Table I carried through to the different system dimensions considered in Tables II and III. However, the extent of the improvements observed relative to \textit{place} was smaller for the systems with more control inputs. This reduced improvement for larger \( m \) was also observed in [11], and may be attributed to the improved performance of \textit{place} when more control inputs are available. An exception to this general observation was observed for the \( \Gamma(X) \) measure, where \textit{robpole 1.5.1} showed its greatest improvement of 7.822\% over \textit{place} for systems with \( m = 8 \) inputs. The improvement of \textit{robpole 1.5.1} over \textit{place} of about 10\% is consistent with the results given in the numerical study [16] that compared the performance of \textit{robpole} and \textit{place} with respect to the \( \kappa_2(X) \) measure.

Lastly, considering the secondary performance measures of the size of the matrix gain \( \|F\|_2 \) and accuracy \( \Delta \), we observed that methods \textit{byersnash}, \textit{sylvplace} and \textit{span2} all used considerably less gain than \textit{place} and \textit{robpole 1.5.1}, while \textit{byersnash} and \textit{span2} were both considerably more accurate than \textit{place}, \textit{robpole 1.5.1} and \textit{sylvplace}. However, all methods achieved high accuracy in absolute terms, with the worst-case eigenvalue errors being of order \( O(10^{-10}) \), \( O(10^{-13}) \) and \( O(10^{-14}) \) for systems with \( m = 2, m = 4 \) and \( m = 8 \) inputs, respectively.

IV. Conclusion

We have continued the numerical experiments begun in [11] to investigate the relative performance of five prominent methods for the classic problem of robust pole placement. Enhancements to the methods of \textit{robpole} [4] and \textit{span} [11] were introduced, and the robustness performance measures used were the eigenvalue condition number \( c_\infty(X) \), and the orthonormality measure \( \Gamma(X) \). The accuracy and size of the matrix gain used by each method were also compared.

The results showed some notable differences between the various methods across the range of performance measures. Methods \textit{byersnash}, \textit{sylvplace} and \textit{span2} were able to achieve substantially better eigenvalue condition numbers than \textit{place} and \textit{robpole 1.5.1}, replicating the performance results with respect to the Frobenius condition number given in the survey in [11]. For the eigenvector orthonormality
measure $\Gamma(X)$, methods place and robpole 1.5.1 were able to achieve superior robustness than byersnash, sylvplace and span2. Methods byersnash and span2 were able to achieve their robust eigenstructures with the least matrix gain. Methods byersnash and span2 also achieved the best accuracy.

The results of this survey further extend the insights provided in [11] regarding the relative performance of the methods surveyed. A novel feature of the present survey was that all methods were granted a common amount of computational resources (runtime). The results showed that no single method can claim to offer the best performance across all the robustness performance measures. Differences were also noted in the accuracy, gain and computational efficiency of the methods. The results enable the control designer to make a better informed choice as to the most suitable pole placement methods, according to their priorities across a range of performance criteria.

Future work will consider systems of larger dimension, where it is anticipated that some of the methods will become computationally impractical.

REFERENCES


TABLE I

SURVEY 1: 500 SYSTEMS WITH $n = 20$, $m = 2$

<table>
<thead>
<tr>
<th>Method</th>
<th>$\text{ind}(c_{\infty})$</th>
<th>$\text{ind}(F)$</th>
<th>$\text{ind}(\Delta)$</th>
<th>$\text{ind}(c_{\infty})$</th>
<th>$\text{ind}(F)$</th>
<th>$\text{ind}(\Delta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>span2</td>
<td>70.54</td>
<td>28.18</td>
<td>62.60</td>
<td>16.48</td>
<td>4.086</td>
<td>28.08</td>
</tr>
<tr>
<td></td>
<td>-14.15</td>
<td>21.38</td>
<td>54.94</td>
<td>0.7481</td>
<td>4.242</td>
<td>22.57</td>
</tr>
<tr>
<td>robpole 1.5.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE II

SURVEY 2: 500 SYSTEMS WITH $n = 20$, $m = 4$

<table>
<thead>
<tr>
<th>Method</th>
<th>$\text{ind}(c_{\infty})$</th>
<th>$\text{ind}(F)$</th>
<th>$\text{ind}(\Delta)$</th>
<th>$\text{ind}(c_{\infty})$</th>
<th>$\text{ind}(F)$</th>
<th>$\text{ind}(\Delta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>span2</td>
<td>63.29</td>
<td>19.98</td>
<td>40.72</td>
<td>18.63</td>
<td>6.428</td>
<td>19.40</td>
</tr>
<tr>
<td></td>
<td>-16.98</td>
<td>18.08</td>
<td>41.04</td>
<td>2.554</td>
<td>6.675</td>
<td>18.44</td>
</tr>
<tr>
<td>byersnash</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sylvplace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE III

SURVEY 3: 500 SYSTEMS WITH $n = 20$, $m = 8$

<table>
<thead>
<tr>
<th>Method</th>
<th>$\text{ind}(c_{\infty})$</th>
<th>$\text{ind}(F)$</th>
<th>$\text{ind}(\Delta)$</th>
<th>$\text{ind}(c_{\infty})$</th>
<th>$\text{ind}(F)$</th>
<th>$\text{ind}(\Delta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>span2</td>
<td>39.48</td>
<td>-0.8838</td>
<td>15.88</td>
<td>16.37</td>
<td>5.471</td>
<td>4.535</td>
</tr>
<tr>
<td></td>
<td>-5.636</td>
<td>16.12</td>
<td>7.822</td>
<td>5.703</td>
<td>1.971</td>
<td></td>
</tr>
<tr>
<td>byersnash</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sylvplace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ACKNOWLEDGEMENTS

The authors would like to thank Andreas Varga for providing them with a copy of the sylvplace toolbox.