European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)
vol. 3, no. 1-2, 2008, pp. 1-16
ISSN 1790–8140

Barycentric Hermite Interpolants for Event Location in Initial-Value Problems[1]

Robert M. Corless[2], Azar Shakoori, D.A. Aruliah, Laureano Gonzalez-Vega

Department of Applied Mathematics,
Faculty of Sciences,
University of Western Ontario,
Ontario, Canada

Received 2 December, 2007; accepted in revised form 20 December, 2007

Abstract:
Continuous extensions are now routinely provided by many IVP solvers, for graphical output, error control, or event location. Recent developments suggest that a uniform, stable and convenient interpolant may be provided directly by value and derivative data (Hermite data), because a new companion matrix for such data allows stable, robust and convenient root-finding by means of (usually built-in) generalized eigenvalue solvers such as `eig` in MATLAB or `Eigenvalues` in MAPLE. Even though these solvers are not as efficient as a special-purpose Hermite interpolant root-finder might be, being $O(d^3)$ in cost instead of $O(d^2)$, for low or moderate degrees d they are efficient enough. More, because all roots are found, the first root (and hence the event) is guaranteed to be found. Further, the excellent conditioning properties (compared to the monomial basis or to divided differences) suggest that the results will be as accurate as possible. The techniques of this paper apply to polynomial or rational interpolants such as the shape-preserving interpolants of Brankin and Gladwell.
We give a sketch of barycentric Hermite interpolation and a sketch of the theory of conditioning of such interpolants. Moreover, we present the construction of the Hermite interpolation polynomial companion matrix pencil and a discussion of scaling and precomputation. We remark that the Bézout matrix can be used used to solve more complicated event location problems involving more than one polynomial, via polynomial eigenvalue problems.

Keywords: Initial-value problem, ordinary differential equation, event location, root-finding, generalized eigenvalues, companion matrix pencil, Hermite interpolation problem, Bézout matrix.

Mathematics Subject Classification: 65L05; 65D05

[1]Published electronically March 31, 2008
[2]E-mail: rcorless@uwo.ca

1 Introduction

We illustrate how some recent improvements in the understanding and techniques of interpolation can be specialized to event location in the numerical solution of initial-value problems (IVPs) for Ordinary Differential Equations (ODEs). Specifically, we show how a new companion matrix pencil can be used to explore different choices of interpolant for event handling in a robust, numerically stable, and convenient way. The technique allows the use of polynomial or rational interpolants in an equally convenient manner. Since we use a generalized eigenvalue solver to find the zeros of the interpolants, we do not claim efficiency, but we have found that our approach seems to be efficient enough to be interesting. Moreover, since all roots are found automatically, there is no difficulty identifying the first root, which defines the event. We sketch some alternative root-finding methods in Section 7, but do not pursue them here.

As is usual in event-handling for IVPs for ODEs, we use the data produced by the solver (which is often Hermite data, i.e., values and some derivatives) to construct an interpolant. The interpolant approximates a function whose zeros define the event. The location of these zeros can be found numerically to isolate the event. Often, more than one event function is considered in the same timestep; in this case, the interpolants are most naturally solved one at a time and the first event deduced from the results. Event location is not as easy as it may appear at first glance: see [25] for a discussion of practical difficulties.

All of these difficulties also pertain to the interpolants of this paper. Our contribution is to show how to use the Hermite basis (which includes the Lagrange basis) consistently; this gives some numerical stability advantages over other representations of the interpolating polynomial or rational function. In particular, we use the barycentric forms. This uniform approach also confers flexibility: it is simple, with our approach, to change interpolants.

Polynomial interpolants have some advantages, but rational interpolants can sometimes give added flexibility for, say, shape preservation [7]. Other parameters, including so-called "tension parameters," can be used for various purposes. These can all be accommodated by the methods of this paper.

2 Barycentric Hermite Interpolation

We assume that the function f to be interpolated is specified by Hermite interpolation data: that is, given n distinct nodes τ_i $(1 \leqslant i \leqslant n)$ and given values and derivative values $\rho_{i,k}$ of $f(t)$ at those nodes, we have

$$\rho_{i,k} = \frac{f^{(k)}(\tau_i)}{k!} = \frac{1}{k!}\frac{d^k}{dt^k}[f(t)]\Big|_{\{t=\tau_i\}} \qquad (1 \leqslant i \leqslant n; 0 \leqslant k \leqslant s_i - 1). \tag{1}$$

In (1), the non-negative integers s_i are the *confluencies* of the nodes that enumerate the number of derivatives known at each node $(1 \leqslant i \leqslant n)$. The factorials are included in the definition of $\rho_{i,k}$ both for scaling and for convenience. The classical theory of Hermite interpolation assures us that there is a unique *polynomial* of degree at most d, where

$$d = -1 + \sum_{i=1}^{n} s_i \,, \tag{2}$$

that fits the data $\rho_{i,k}$, since the τ_i are distinct: $\tau_i = \tau_j \Longleftrightarrow i = j$. At least one s_i must be positive for d to be non-negative.

Explicit constructions of this unique Hermite interpolating polynomial $p(t)$ are given in textbooks usually only if all $s_i = 1$ (which corresponds to Lagrange interpolation). However, there are

many algorithms for constructing $p(t)$ given τ_i and the $\rho_{i,k}$ in the literature (e.g., [19, 16]). In the present work, we use a simple approach from [28] called barycentric Hermite interpolation which also allows *rational* interpolation. Rational interpolation is somewhat more complicated than polynomial interpolation in that there may be unwanted poles and that some data may be "unattainable". The techniques we present require a fixed denominator known *a priori*. This restriction is sufficient for our purposes. Unattainable points are easily detectable with the barycentric approach and can be avoided by choosing a different denominator [28].

Barycentric Lagrange interpolation (a special case of barycentric Hermite interpolation) has been shown to be numerically stable and more efficient than had been previously acknowledged [5, 17]. Barycentric Hermite interpolation seems to share these benefits provided no s_i is too large [28, 24] (although this has not yet been proved). Perhaps the most surprising feature in the derivation of barycentric Hermite interpolants is that it amounts to no more than a partial fraction expansion. For greater details supporting the theoretical results used in this paper, consult [24].

2.1 Details and notation

We construct the polynomial

$$w(t) = \prod_{i=1}^{n} (t - \tau_i)^{s_i} \tag{3}$$

which is of degree $d + 1$ with a zero of order s_i at each node τ_i ($1 \leqslant i \leqslant n$). In the rational interpolation problem we wish to solve, $q(t)$ will be the known denominator and $f(t) = p(t)/q(t)$ will be the function to reconstruct from the values (1). Then, for the prescribed polynomial $q(t)$, we compute the partial fraction expansion of the rational function $q(t)/w(t)$:

$$\frac{q(t)}{w(t)} = \sum_{i=1}^{n} \sum_{j=0}^{s_i - 1} \frac{\gamma_{i,j}}{(t - \tau_i)^{j+1}} \,. \tag{4}$$

The partial fraction expansion (4) uniquely defines the complex constants $\gamma_{i,j}$ that we call the *generalized barycentric weights*. These weights can be precomputed once given the denominator $q(t)$, the nodes τ_i and the corresponding confluencies s_i ($1 \leqslant i \leqslant n$). For the important case of polynomial interpolation, $q(t) \equiv 1$, in which case the theory of residues leads to an analytical formula for the leading terms

$$\gamma_{i,s_i-1} = \prod_{\substack{j=1 \\ j \neq i}}^{n} (\tau_i - \tau_j)^{-s_i} \,. \tag{5}$$

In the case where the denominator $q(t) = (t - z_1)(t - z_2) \cdots (t - z_q)$ is given by its poles, again an explicit formula is known in the Lagrange case [4], and can be extended to the leading terms in the Hermite case:

$$\gamma_{i,s_i-1} = \prod_{\substack{j=1 \\ j \neq i}}^{n} (\tau_i - \tau_j)^{-s_i} \prod_{k=1}^{q} (\tau_i - z_k) \,. \tag{6}$$

In [24] we discuss a stable, series-based algorithm, implemented in Maple, for computing the $\gamma_{i,j}$ of equation (4). A Matlab implementation accompanies this present paper. A faster algorithm based on confluent divided differences is provided in [28]. We believe the algorithm of [28] for computing the generalized barycentric weights $\gamma_{i,j}$ is less stable than our Matlab algorithm. For the relatively low degrees we consider at present, neither the speed nor the (minor) instability are significant.

Using all these computed $\gamma_{i,j}$, we may write the unique rational function $f(t) = p(t)/q(t)$ with numerator $p(t)$ of degree at most d that interpolates the data $\rho_{i,k}$ in (1) as

$$f(t) = \frac{p(t)}{q(t)} = w(t) \sum_{i=1}^{n} \sum_{j=0}^{s_i-1} \frac{\gamma_{i,j}}{(t-\tau_i)^{j+1}} \sum_{k=0}^{j} \rho_{i,k}(t-\tau_i)^k, \tag{7a}$$

or, as the second barycentric form

$$\frac{p(t)}{q(t)} = \frac{\displaystyle\sum_{i=1}^{n} \sum_{j=0}^{s_i-1} \frac{\gamma_{i,j}}{(t-\tau_i)^{j+1}} \sum_{k=0}^{j} \rho_{i,k}(t-\tau_i)^k}{\displaystyle\sum_{i=1}^{n} \sum_{j=0}^{s_i-1} \frac{\gamma_{i,j}}{(t-\tau_i)^{j+1}}}. \tag{7b}$$

To deal with removable singularities at the interpolation nodes τ_i, in MATLAB, we use the "`exact`" trick of [5] to evaluate the rational function (7) (Note that there is a typo in the code fragment in [5, p. 510]: the line `exact(xdiff==0) = 1` should be `exact(xdiff==0) = j` to record just which evaluation points are equal to interpolation point(s)). This amounts to replacing the rational function by the exact data value when, and only when, the floating-point evaluation point is bit for bit identical to a node; it is surprising that the second barycentric form is stable enough (if all s_i are 1) that this works. This also appears to work in the Hermite case, though our experiments show some (minor) degradation with increasing confluency. This will be investigated further and reported on in a future paper.

The barycentric forms (7), specialized to the Lagrange case $s_i = 1$ ($1 \leqslant i \leqslant n$), go back to 1945 (at least) and have been recommended in [22], [15, vol. 2], [29], and [28]. More recently, [5] and [17] have shown barycentric Lagrange interpolation to be numerically stable: specifically, Higham shows that the first barycentric form (7a) is backward stable, and that although the second form (7b) is not backward stable, it is forward stable for nodes with a small Lebesgue constant (again, if all $s_i = 1$).

In the more general case of barycentric Hermite interpolation, the second barycentric form (7b) is backward stable in that the right-hand side is a rational function that interpolates $\rho_{i,k}$ at τ_i for any nonzero values of $\gamma_{i,j}$ [29, 28]. This observation explains the often remarkable success of the second form (7b). Regardless, in [24], we argue for accurate computation of the $\gamma_{i,j}$ anyway, and show that the residue method does a good job, and is reasonably efficient. This is in contrast with the symbolic algorithms used in the MAPLE function calls `convert(`$q(t)/w(t)$`,parfrac)` and `convert(`$q(t)/w(t)$`,fullparfrac)`. These MAPLE routines internally convert the rational expressions to the monomial basis centred at the origin, and cannot recover from the resulting numerical instability. The algorithm of [28] also makes an implicit change of basis to the Newton basis and, with a proper ordering of the nodes, is less susceptible to rounding errors than MAPLE's partial fraction routines. However, the method of [28] can be quite unstable under certain orderings of the nodes [29]. In the present work, the examples are sufficiently small that the method by which the weights $\gamma_{i,j}$ are computed is immaterial. For larger problems, we recommend the method of [24].

Having computed the generalized barycentric weights in (4) by some means, we may explicitly write down the Hermite polynomial basis[3] associated with the interpolation problem at hand.

[3] not to be confused with orthogonal Hermite polynomials from mathematical physics

Define

$$H_{I,J}(t) := \frac{1}{J!}\Theta_{I,J}(t) = \frac{\dfrac{1}{J!}\displaystyle\sum_{j=0}^{s_I-J-1} \gamma_{I,j+J}(t-\tau_I)^{-j-1}}{\displaystyle\sum_{i=1}^{n}\sum_{j=0}^{s_i-1} \gamma_{i,j}(t-\tau_i)^{-j-1}} \,. \tag{8}$$

We have

$$H_{I,J}^{(\ell)}(\tau_k) = \delta_{I,k}\delta_{J,\ell} \quad \text{and} \quad \Theta_{I,J}^{(\ell)}(\tau_k) = J!\delta_{I,k}\delta_{J,\ell} \tag{9}$$

for $1 \leqslant I,k \leqslant n$ and $0 \leqslant J \leqslant s_I - 1$, $0 \leqslant \ell \leqslant s_k - 1$. The formula (8) parallels the explicit barycentric form of the Lagrange basis polynomials.

2.2 Differentiation of Hermite Interpolants

The work of [28] gives a formula for computing derivatives of the rational Hermite interpolant based on confluent divided differences. As an alternative, we have generalized the result presented in [5] for the Lagrange case to the general rational Hermite case; we conjecture that the formulas below are better suited to our root-finding applications because they allow precomputation and also may be more numerically stable.

We wish to compute derivatives. In particular, knowledge of $H_{I,J}^{(s_L)}(\tau_k)$ $(1 \leqslant I,J,L,k \leqslant n,)$ allows us to differentiate any rational function given by Hermite data. This follows because we can compute $p^{(s_k)}(\tau_k)$ at each node by the expression

$$p^{(s_k)}(\tau_k) = \sum_{i=1}^{n}\sum_{j=0}^{s_i-1} p^{(j)}(\tau_i) H_{i,j}^{(s_k)}(\tau_k)\,, \tag{10}$$

and then we know $p'(\tau_k), p''(\tau_k), \ldots, p^{(s_k)}(\tau_k)$ for $1 \leqslant k \leqslant n$; hence

$$p'(t) = \sum_{i=1}^{n}\sum_{j=0}^{s_i-1} p^{(j+1)}(\tau_i) H_{i,j}(t)\,. \tag{11}$$

That is, by finding $H_{I,J}^{(s_L)}(\tau_k)$ $(1 \leqslant I,J,L,k \leqslant n,)$, we have constructed the nontrivial entries of the differentiation matrix for the Hermite nodes. This process can be iterated to get $p''(t)$ at the nodes (this amounts to squaring the differentiation matrix). Once the values of the derivatives at the nodes are computed, we may use the same barycentric form (7b) to evaluate $p'(t)$ for any t in the interval.

Theorem 2.1

$$H_{I,J}^{(s_L)}(\tau_L) = \frac{s_L!}{J!\gamma_{L,s_L-1}} \sum_{j=0}^{s_I-J-1} \gamma_{I,j+J}(\tau_L - \tau_I)^{-j-1} \tag{12}$$

for all $I \neq L$, and all $0 \leqslant J \leqslant s_I - 1$. Also,

$$H_{L,\mu}^{(s_L)}(\tau_L) = -\sum_{\substack{i=1\\ i\neq L}}^{n} \sum_{j=0}^{\min(\mu,s_i-1)} \frac{1}{(\mu-j)!}(\tau_i - \tau_L)^{\mu-j} H_{i,j}^{(s_L)}(\tau_L)\,, \tag{13}$$

for all $0 \leqslant \mu \leqslant s_L - 1$.

Proof See [24].

Table 1: MATLAB code for an extreme example

```
function ext=derzeros3(n)
  tau  = cos( (0:n)*pi/n ); % nodes
  t    = linspace( -1, 1, 20*n+1 );
  rho  = [ones(1,n+1); zeros(1,n+1); zeros(1,n+1)]; % data
  r    = rho(:);
  r(4) = 1.5; % bump in curve
  [w,D]   = genbarywts( tau, 3 ); % get barycentric weights
  % Now differentiate at nodes and set up companion pencil for p'
  [C0,C1] = gcmp( D*r/norm(D*r,inf), tau, 3, w/norm(w,inf) );
  ext  = eig( C0, C1 ); % locate extrema
  yext = hermiteval( r, ext, tau, 3, w, D ); % extremal heights
  [y,yp]  = hermiteval( r, t, tau, 3, w, D ); % evaluate interpolant
  plot( tau,r(1:3:3*n+1),'bo', t,y, ext,yext,'r+' );
  axis( [-1,1,0,2] );
```

2.3 An extreme example

Consider a set of n distinct nodes τ_i $(1 \leqslant i \leqslant n)$ and the polynomial $p(t)$ such that

- $p(\tau_k) = 1$ for $1 \leqslant k \leqslant n-2$ and $k = n$
- $p'(\tau_i) = p''(\tau_i) = 0$ for $1 \leqslant i \leqslant n$
- $p(\tau_{n-1}) = 1.5$.

That is, $p(t)$ is a perturbation from the constant polynomial $p(t) = 1$ (although its derivatives are unchanged at $\tau = \tau_{n-1}$). Polynomials do not 'like' this data well, for high degrees: the single perturbation (correctly) induces large oscillations. However, this wiggly polynomial can be evaluated accurately, and the zeros of its derivative found easily, by the methods of this paper. To be specific, in MATLAB, the call `derzeros3(34)` (see Table 1) produces the graph in Figure 1.

3 Conditioning

The condition number for evaluating $p(t)$ at $t = \lambda$ (or the polynomial evaluation condition number) is a measure of the sensitivity of $p(\lambda)$ to changes in the values of the coefficients of $p(t)$; this quantity must obviously depend on the basis used to represent $p(t)$. Similarly, the condition number for finding a simple root λ (the root-finding condition number) turns out to be the polynomial evaluation condition number of $p(t)$ at $t = \lambda$ apart from a factor $1/|p'(\lambda)|$ which is independent of the basis. The theory of how condition numbers in one basis are related to those in another is elegantly explained in [12] and slightly extended in [9].

Suppose $p(t)$ is expressed in the basis $\phi_k(t)$, $0 \leqslant k \leqslant d$, as

$$p(t) = \sum_{k=0}^{d} c_k \phi_k(t)\,, \tag{14}$$

and that a perturbed polynomial

$$(p + \Delta p)(t) = \sum_{k=0}^{d} (c_k + \Delta c_k)\phi_k(t) \tag{15}$$

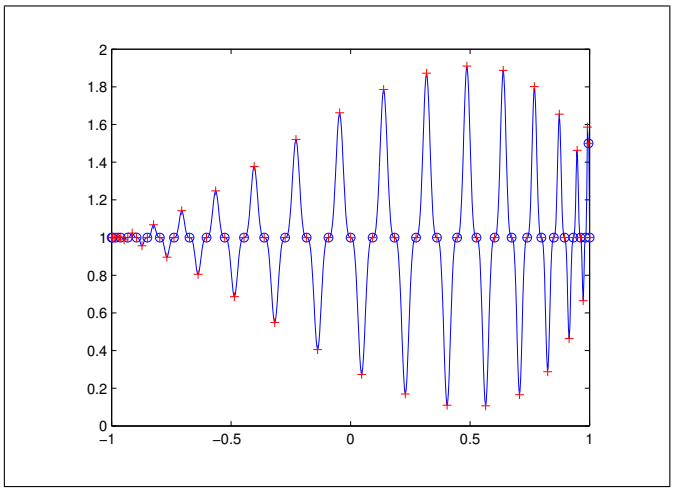

Figure 1: A deliberately wiggly high-degree polynomial specified by Hermite data. The methods of the paper accurately evaluate and differentiate the polynomial, and locate its extrema.

has coefficients slightly different, with

$$|\Delta c_k| \leqslant \alpha_k \varepsilon \tag{16}$$

for some as yet unspecified weights $\alpha_k \geqslant 0$, $0 \leqslant k \leqslant d$, not all zero, and some presumably small $\varepsilon > 0$. Then Hölder's inequality gives

$$|\Delta p(t)| \leqslant \left(\sum_{k=0}^{d} \alpha_k |\phi_k(t)| \right) \varepsilon \,. \tag{17}$$

This gives an absolute condition number for evaluation of $p(t)$. Put

$$B(t) = \sum_{k=0}^{d} \alpha_k |\phi_k(t)| \,. \tag{18}$$

Now, assuming ε small, suppose that λ is a root of $p(t)$ and $\lambda + \Delta\lambda$ is the corresponding root of $(p + \Delta p)(t)$. Then, to first-order, we have

$$\Delta\lambda = \frac{-\Delta p(\lambda)}{p'(\lambda)} + O(\Delta^2) \,, \quad \text{and hence } |\Delta\lambda| \lessapprox \frac{B(\lambda)\varepsilon}{|p'(\lambda)|} \,. \tag{19}$$

This can be extended to the following result on pseudozeros (itself a special case of the result for pseudospectra of [1, 8]): define the pseudozero set as

$$\Delta_\varepsilon(p) = \{ z : (p + \Delta p)(z) = 0 \ni |\Delta c_k| \leqslant \alpha_k \varepsilon \} \,. \tag{20}$$

Then this may also be characterized as

$$\Delta_\varepsilon(p) = \{ z : |p^{-1}(z)| \geqslant (B(z)\varepsilon)^{-1} \} \,, \tag{21}$$

where now no assumption is made about the size of ε. This result includes multiple roots. It should be clear that small $B(z)$ is desirable (essentially shrinking ε, or at least not growing it).

If all α_k are taken to be 1, then the Δc_k are absolute perturbations, and $B(\lambda)$ becomes the familiar Lebesgue function from interpolation theory (see, e.g., [21]). This is pursued briefly in [8] but is not needed here. More often, we take $\alpha_k = |c_k|$ so that the Δc_k are small relative perturbations in the coefficients. If this is done, then the theory of [12] shows that the Bernstein basis is optimal among all bases non-negative on $[0, 1]$ (and by translation, on any real interval). The extension of [9] shows that Lagrange bases are optimal over all bases non-negative on the nodes, and are then optimal on a set with non-empty interior containing the nodes. This set can be surprisingly large in $\mathbb{C}$ and need not be a real interval. This result explains why the Lagrange basis can be orders of magnitude better conditioned than even the Bernstein basis.

This result has not been extended to the case $s_k > 1$, the Hermite case. Nonetheless, experience shows that Hermite bases can be very well-conditioned. Properly interpreted, this result supports the observation of [3] that showed the superior conditioning of Hermite interpolation bases and of monomial bases for the numerical solutions of boundary value problems (BVP). In the solution of BVPs, when piecewise polynomials are used, the continuity conditions that are imposed in effect feed information into each subinterval from its endpoints. Thus, for BVPs, the use of monomial bases is akin to using Hermite bases, with a node at each end of the subinterval and with high confluency.

The main practical observation is that $B(t)$ is usually smaller for Hermite interpolation bases than it is for monomial bases that are based at one endpoint of the interval or for Newton bases ordered from one end to the other [24]. Note that this observation is in some sense intuitively plausible. If all our information about $p(t)$ is concentrated at $t = 0$, then good predictions about the behaviour of the interpolant $p(t)$ at $t = 1$ might be harder than if $p(t)$ were known more uniformly in $[0, 1]$ instead.

3.1 On the sources of errors

In the present work, we scale the integration step $[t_n, t_{n+1}]$ to the unit interval: $t = t_n + (t_{n+1} - t_n)\theta$, $0 \leqslant \theta \leqslant 1$. This choice of scaling multiplies derivatives by multiples of $h_n = t_{n+1} - t_n$ and further implies that errors in y will be compared against errors in (e.g.) $h_n \dot{y}$. Perturbations in the locations of the nodes τ_i themselves will not be considered here, but see [24] for expressions and recurrence relations for $\dfrac{\partial \gamma_{i,j}}{\partial \tau_k}$.

4 Rootfinding without changing basis

If we wish to find λ such that $p(\lambda) = 0$ (more specifically, $\lambda \in [t_n, t_{n+1}] \ni p(\lambda) = 0$), then there are many numerical methods available. We sketch some in Section 7. However, most numerical methods find only one root at a time and require good initialization. On the other hand, most methods that use the fact that $p(t)$ is polynomial in t find all roots but usually require that a monomial basis expression for $p(t)$ be computed first. We wish to avoid this step because, as is well-known, the root-finding problem associated with the resulting expression can be (and often is) ill-conditioned. Instead, we develop a special-purpose method that works directly from Hermite interpolation data. Such an approach would need to preserve the good conditioning properties of the Hermite basis itself and would need to be reasonably efficient, but most of all, would need to be flexible.

It turns out that there is a companion matrix pencil that linearizes—that is, converts to a generalized eigenvalue problem—any polynomial $p(t)$ written in a Hermite interpolation basis. This extends work of [2].

Theorem 4.1 *[23, 24] Let*

$$(\underbrace{\tau_1,\ldots,\tau_1}_{s_1\ times},\underbrace{\tau_2,\ldots,\tau_2}_{s_2\ times},\ldots,\underbrace{\tau_n,\ldots,\tau_n}_{s_n\ times}) \tag{22}$$

be the confluent nodes with s_i being the confluency of the node τ_i $(1 \leqslant i \leqslant n)$. With the convention

$$p_i^{(k)} = p^{(k)}(\tau_i) \quad (0 \leqslant k \leqslant s_i - 1)\,, \tag{23}$$

we assume

$$(p_1^{(0)}, p_1^{(1)}, \ldots, p_1^{(s_1-1)}, \ldots, p_n^{(0)}, \ldots, p_n^{(s_n-1)}) \tag{24}$$

are the values of the polynomial $p(t) \in \mathbb{P}_d$ and the values of its derivatives at the given nodes. Then, a companion pencil $(\mathbf{C}_0,\, \mathbf{C}_1) \in [\mathbb{C}^{(d+2)\times(d+2)}]^2$ in the Hermite basis which satisfies $\det(t\mathbf{C}_1 - \mathbf{C}_0) = p(t)$ is

$$\mathbf{C}_0 = \begin{bmatrix} \mathbf{J}_1^T(\tau_1) & & & & \boldsymbol{\Pi}_1 \\ & \mathbf{J}_2^T(\tau_2) & & & \boldsymbol{\Pi}_2 \\ & & \ddots & & \vdots \\ & & & \mathbf{J}_n^T(\tau_n) & \boldsymbol{\Pi}_n \\ -\boldsymbol{\Gamma}_1 & -\boldsymbol{\Gamma}_2 & \ldots & -\boldsymbol{\Gamma}_n & 0 \end{bmatrix}, \quad \mathbf{C}_1 = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 0 \end{bmatrix}, \tag{25a}$$

$$\mathbf{J}_i(\tau_i) = \begin{bmatrix} \tau_i & 1 & & \\ & \tau_i & 1 & \\ & & \ddots & 1 \\ & & & \tau_i \end{bmatrix} \in \mathbb{C}^{s_i\times s_i}\,, \qquad (1 \leqslant i \leqslant n) \tag{25b}$$

$$\boldsymbol{\Pi}_i = \begin{bmatrix} \frac{p_i^{(0)}}{0!}, & \ldots, & \frac{p_i^{(s_i-1)}}{(s_i-1)!} \end{bmatrix}^T \in \mathbb{C}^{s_i\times 1}\,, \qquad (1 \leqslant i \leqslant n) \tag{25c}$$

$$-\boldsymbol{\Gamma}_i = \begin{bmatrix} -\gamma_{i,0}, & \ldots, & -\gamma_{i,s_i-1} \end{bmatrix} \in \mathbb{C}^{1\times s_i} \qquad (1 \leqslant i \leqslant n). \tag{25d}$$

Thus, generalized eigenvalues of the pencil $(\mathbf{C}_0, \mathbf{C}_1)$ are roots of $p(t)$. The matrices are of dimension $(d+2)\times(d+2)$, so there are two spurious roots at infinity, but these do not bother us (they are a slight annoyance, no more). There is (in the Lagrange case) a $d \times d$ pencil, and it seems that this can be extended to the Hermite case, but because it singles out two nodes for special treatment, there is some potential instability; for our purpose, the present pencil is better.

It is easy to construct the matrices (25a). It is similarly easy to compute the generalized eigenvalues by calling built-in routines such as `eig` in MATLAB. We then discard all λ not of interest (including the spurious infinite eigenvalues). Identification of the first root, which defines the event, is straightforward.

Remark: Hermite interpolants are well-conditioned only in a small region near the set of interpolation points [9, 8]. Computed roots much outside an interval containing the nodes (say, two or three units away) are inaccurate. Thus it is easy to discard the spurious infinite roots. *However, scaling of the generalized barycentric weights and the polynomial values is critical to the success of the generalized eigensolver.* We recommend scaling the last block row and block column of $\mathbf{C}_0$ so that $\|\boldsymbol{\Pi}\|_\infty = \|\boldsymbol{\Gamma}\|_\infty = 1$ (*cf* Table 1). From the second barycentric form (7b), both these vectors may be multiplied by any nonzero constant without changing the generalized eigenvalues.

The use of eigenvalue problems of companion matrices of polynomials expressed in the monomial basis has been widely studied. As a polynomial root-finding strategy, it is believed to be numerically stable in all but very exceptional circumstances [20, 27, 11]. Of course, the poor conditioning of the

monomial basis means that even though the solving step is stable, the results might be poor: it is not a good idea to convert to the monomial basis in order to use a standard (Frobenius) companion matrix as input to an eigenvalue solver. But that is precisely what we avoid by using the Hermite companion pencil. We have observed in our experiments that the computation of the eigenvalues of the Hermite companion pencil seems to be just as numerically stable as the standard companion matrix is, and we expect that the arguments of [11] might well be extendable to this "new" basis. As of time of writing, this numerical stability of root-finding of Hermite polynomial interpolants remains a plausible conjecture. The consequence is, since the Hermite basis is well-conditioned, that *the computed roots should be as accurate as the data generated by the IVP solver.*

It is true that this approach is more computationally expensive than it needs to be. The cost of solving a generalized eigenproblem is $O(d^3)$ and one would expect $O(d^2)$ methods to be constructible, or even $O(d)$ if we restrict attention to the smallest root in the interval $[t_n, t_{n+1}]$. We sketch some such approaches to faster, special-purpose methods in Section 7. However, for low or moderate degrees d this expense does not seem too onerous.

Finally, the eigenvalue approach easily generalizes via tensor products to the matrix polynomial case. This allows bivariate problems to be solved via resultant-like matrices, e.g., the Bezoutian. See [23] and [24] for details of this approach. The use of Bezoutians might be of interest for bifurcation problems for IVP or for DAE; we do not report on this here.

5 Examples

5.1 Cubic Hermite Interpolant

As an important example, the generalized eigenproblem for the cubic Hermite interpolant satisfying

$$p(0) = \rho_{1,0}\,, \quad p(1) = \rho_{2,0} \tag{26a}$$

$$\frac{dp}{d\theta}(0) = \rho_{1,1}\,, \quad \frac{dp}{d\theta}(1) = \rho_{2,1}\,. \tag{26b}$$

is

$$\mathbf{C}_0 = \begin{bmatrix} 0 & & & & \rho_{1,0} \\ 1 & 0 & & & \rho_{1,1} \\ & & 1 & & \rho_{2,0} \\ & & 1 & 1 & \rho_{2,1} \\ -2 & -1 & 2 & -1 & 0 \end{bmatrix}, \quad \mathbf{C}_1 = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 0 \end{bmatrix}, \tag{27}$$

and inspection shows

$$\det\left(\theta\mathbf{C}_1 - \mathbf{C}_0\right) = p(\theta)\,. \tag{28}$$

The (negative) generalized barycentric weights $-2, -1, 2$, and -1 arise because

$$\frac{1}{\theta^2(\theta-1)^2} = \frac{2}{\theta} + \frac{1}{\theta^2} - \frac{2}{\theta-1} + \frac{1}{(\theta-1)^2} \tag{29}$$

as can be easily verified.

The differentiation matrix is

$$\mathbf{D} = \begin{bmatrix} & 1 & & \\ -6 & -4 & 6 & -2 \\ & & & 1 \\ 6 & 2 & -6 & 4 \end{bmatrix} \tag{30}$$

and this gives derivatives with respect to θ on the nodes.

5.2 Companion matrix pencil for rational functions

We can construct a companion pencil for the numerator of a rational function $f(t) = p(t)/q(t)$ directly from the Hermite data for the rational function f together with the Hermite data for the denominator q. We first construct the partial fraction expansion of the function $1/w(t)$, by means of equation 4 with $q(t)$ replaced by 1:

$$\frac{1}{w(t)} = \sum_{i=1}^{n} \sum_{j=0}^{s_i-1} \frac{\gamma_{i,j}}{(t-\tau_i)^{j+1}} \,. \tag{31}$$

This defines the $\gamma_{i,j}$ uniquely, and the algorithm of [24] suffices to find them stably. Next, determine Hermite data for the proposed denominator $q(t)$, i.e., $\sigma_{i,k} = q^{(k)}(\tau_i)/k!$. Then the barycentric form of the Hermite interpolant of $q(t)$ can be rewritten by interchanging the order of summation:

$$\frac{q(t)}{w(t)} = \sum_{i=1}^{n} \sum_{j=0}^{s_i-1} \sum_{k=0}^{j} \gamma_{i,j}\sigma_{i,k}(t-\tau_i)^{k-j-1} \tag{32a}$$

$$= \sum_{i=1}^{n} \sum_{j=0}^{s_i-1} \frac{\hat{\gamma}_{i,j}}{(t-\tau_i)^{j+1}} \,, \quad \text{with} \tag{32b}$$

$$\hat{\gamma}_{i,j} := \sum_{k=0}^{s_i-1-j} \gamma_{i,k+j}\sigma_{i,k} \tag{32c}$$

The new generalized barycentric weights $\hat{\gamma}_{i,j}$ resulting from (32c) depend on the data for $q(t)$. In fact, it is these weights that are used in (7), but the hats were dropped in that section, because the computation of the $\gamma_{i,j}$ for the denominator 1 is really just an intermediate computation. Keeping the hats in this section for clarity, however, the companion matrix pencil for the numerator polynomial $p(t)$ is as in Theorem 4.1, i.e.,

$$\mathbf{C}_0 = \begin{bmatrix} \mathbf{J}_1^T(\tau_1) & & & & \mathbf{\Pi}_1 \\ & \mathbf{J}_2^T(\tau_2) & & & \mathbf{\Pi}_2 \\ & & \ddots & & \vdots \\ & & & \mathbf{J}_n^T(\tau_n) & \mathbf{\Pi}_n \\ -\hat{\mathbf{\Gamma}}_1 & -\hat{\mathbf{\Gamma}}_2 & \dots & -\hat{\mathbf{\Gamma}}_n & 0 \end{bmatrix}, \quad \mathbf{C}_1 = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 0 \end{bmatrix}, \tag{33a}$$

where

$$\hat{\mathbf{\Gamma}}_i = \begin{bmatrix} \hat{\gamma}_{i,0}, & \dots, & \hat{\gamma}_{i,s_i-1} \end{bmatrix} . \tag{33b}$$

The blocks $\mathbf{J}_i$ and $\mathbf{\Pi}_i$ in (33a) are as in (25) with the values $\rho_{i,k}$ giving values of $f(t) = p(t)/q(t)$ as in (1) ($1 \leqslant i \leqslant n, 0 \leqslant k \leqslant s_i - 1$). We have that $\det(t\mathbf{C}_1 - \mathbf{C}_0) = p(t)$ where

$$p(t) = f(t)q(t) = w(t) \sum_{i=1}^{n} \sum_{j=0}^{s_i-1} \frac{\hat{\gamma}_{i,j}}{(t-\tau_i)^{j+1}} \sum_{k=0}^{j} \rho_{i,k}(t-\tau_i)^k \,.$$

5.3 Use with shape-preserving interpolants

In the notation of [7], we construct a cubic rational interpolant according to

$$s(x) = \frac{P(\theta)}{Q(\theta)}\,, \tag{34a}$$

$$\theta = \frac{(x - x_n)}{h_n}\,, \text{and} \tag{34b}$$

$$Q(\theta) = 1 + (r-3)\theta(1-\theta)\,. \tag{34c}$$

The numerator $P(\theta)$ is chosen to satisfy the conditions

$$\begin{aligned} s(x_n) &= y_n\,, \quad s(x_{n+1}) = y_{n+1} \\ s'(x_n) &= f_n\,, \quad s'(x_{n+1}) = f_{n+1}\,. \end{aligned} \tag{34d}$$

The "shape parameter" r is chosen on each interval in such a way as to guarantee monotonicity of the interpolant (if the data suggest $y(x)$ is monotone) or convexity (if the data suggest convexity instead); the interpolant can be constructed to guarantee both. For various reasons, it may be desirable to use this for event location and not just plotting. Further, higher-order shape-preserving interpolants may also be of interest [10]. The companion matrix method allows a simple zero-finding algorithm valid for all such. We demonstrate with the Brankin & Gladwell interpolant.

In our notation, we compute

$$\frac{Q(\theta)}{\theta^2(\theta-1)^2} = \frac{1}{\theta^2} + \frac{r-1}{\theta} + \frac{1-r}{\theta-1} + \frac{1}{(\theta-1)^2} \tag{35}$$

which gives the generalized barycentric weights

$$[\hat{\gamma}_{1,0}, \hat{\gamma}_{1,1}, \hat{\gamma}_{2,0}, \hat{\gamma}_{2,1}], = [r-1, 1, 1-r, 1]\,. \tag{36}$$

Then, the nontrivial matrix in the companion pencil is

$$\mathbf{C}_0 = \begin{bmatrix} 0 & & & & y_n \\ 1 & 0 & & & h_n f_n \\ & & 1 & & y_{n+1} \\ & & 1 & 1 & h_n f_{n+1} \\ 1-r & -1 & r-1 & -1 & 0 \end{bmatrix}. \tag{37}$$

We have $\det(\theta\mathbf{C}_1 - \mathbf{C}_0) = P(\theta)$, where $s(x) = P(\theta)/Q(\theta)$, and hence all zeros of $s(x)$ are shifted and scaled generalized eigenvalues of the pencil $(\mathbf{C}_0, \mathbf{C}_1)$. This can be extended to any rational interpolant with denominator of fixed form.

By our MATLAB program (interpolating numerical r) we find that the differentiation matrix is

$$\mathbf{D} = \begin{bmatrix} & 1 & & \\ -2r & 2-2r & 2r & -2 \\ & & & 1 \\ 2r & 2 & -2r & -2+2r \end{bmatrix} \tag{38}$$

and this gives derivatives with respect to θ on the nodes.

5.4 A higher-order example

For higher-order one-step methods, several approaches yield accurate local interpolants. In [26], for example, we find a discussion of inexpensive and accurate local interpolants that are of the Hermite type studied here. We take just one example, the quartic Hermite interpolant given by value and derivative data at each endpoint, and the value (only) at some interior point. For example, some methods make the value at the midpoint available as a natural byproduct of the computation. Thus our confluency vector is $[2, 1, 2]$.

To add an indicative complication purely as motivation, we suppose further that we wish to use the Brankin-Gladwell rational interpolant as a model, and insist on the denominator (35). Then, the generalized barycentric weights are

$$[\hat{\gamma}_{1,0}, \hat{\gamma}_{1,1}, \hat{\gamma}_{2,0}, \hat{\gamma}_{3,0}, \hat{\gamma}_{3,1}], = [-8 - 2(r-3), -2, 16 + 4(r-3), -8 - 2(r-3), 2] \,. \tag{39}$$

The differentiation matrix is (by MAPLE, and note that entries are nonlinear in r this time)

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -6\,r-4 & -2-2\,r & 8\,r+8 & -2\,r-4 & 2 \\ -\frac{r+3}{1+r} & -(1+r)^{-1} & 0 & \frac{r+3}{1+r} & -(1+r)^{-1} \\ 0 & 0 & 0 & 0 & 1 \\ -2\,r-4 & -2 & 8\,r+8 & -6\,r-4 & 2\,r+2 \end{bmatrix} \tag{40}$$

and the nontrivial companion matrix is

$$\mathbf{C}_0 = \begin{bmatrix} 0 & & & & & y_n \\ 1 & 0 & & & & h_n f_n \\ & & 1/2 & & & y_{n+1/2} \\ & & & 1 & & y_{n+1} \\ & & & 1 & 1 & h_n f_{n+1} \\ -8-2(r-3) & -2 & 16+4(r-3) & -8-2(r-3) & 2 & 0 \end{bmatrix} . \tag{41}$$

6 Implementations

Both MAPLE and MATLAB implementations of the methods of this paper have been coded. For construction of low-order fixed interpolants, code in MAPLE is scarcely needed: the built-in facilities for constructing partial fractions, differentiation, and manipulation suffice. Nonetheless, the series algorithm of [24] was so coded, and is available on the ORCCA technical report web site. This code is especially convenient for construction of interpolants containing symbolic parameters for shape-preservation. See `www.orcca.on.ca/TechReports/2007/TR-07-05.html`

Evaluation of fixed low-order interpolants and their derivatives in MATLAB is similarly easy (especially if they were constructed in MAPLE, because it is a simple matter to use the `Codegeneration [Matlab]` routine to export the formulae). Nonetheless, an explicit numerical version of the (fully general) series algorithm of [24] was also coded, and is also available on the ORCCA technical report web site. Linear parameters can be handled relatively easily by interpolation, as was done in the examples in this present paper.

7 Alternative root-finding methods

Using generalized eigenvalue solvers may strike the reader as being inefficient. Even using simple eigenvalue solvers seems extravagant at first glance, but see [14] for a study of comparative effi-

ciency, which is generally favorable to the eigenvalue approach. Still, generalized eigenvalue solvers can be five times slower than simple eigenvalue solvers, so alternatives may be of interest.

7.1 Simultaneous Iterations

Recent work by Bini, Gemignani and coworkers on structured QR iteration may prove useful here [6, 13]. Victor Pan (private communication) suggests Weierstrass iteration [18], and this may prove eventually to be the method of choice.

7.2 One root at a time

One may wish faster methods, if only one root is to be found, in the case where it is known that there is only one. This section sketches some possibly interesting methods. Given

$$\frac{p(t)}{q(t)} = \frac{\displaystyle\sum_{i=1}^{n}\sum_{j=0}^{s_i-1}\sum_{k=0}^{j}\gamma_{i,j}\rho_{i,k}(t-\tau_i)^{k-j-1}}{\displaystyle\sum_{i=1}^{n}\sum_{j=0}^{s_i-1}\gamma_{i,j}(t-\tau_i)^{-j-1}}\,, \tag{42}$$

note that the roots of $p(t)$ are the same as that of the rational function

$$r(t) = \sum_{i=1}^{n}\sum_{j=0}^{s_i-1}\sum_{k=0}^{j}\gamma_{i,j}\rho_{i,k}(t-\tau_i)^{k-j-1} \tag{43}$$

which has the advantage that its derivative is simple:

$$r'(t) = \sum_{i=1}^{n}\sum_{j=0}^{s_i-1}\sum_{k=0}^{j}(k-j-1)\gamma_{i,j}\rho_{i,k}(t-\tau_i)^{k-j-2} \tag{44}$$

and so Newton's method becomes attractive. We remark that $p(t) = w(t)r(t)$, where $w(t)$ is as in Equation (3) and so

$$\frac{w'(t)}{w(t)} = \sum_{i=1}^{n}\frac{s_i}{t-\tau_i}\,, \tag{45}$$

whence

$$p'(t) = w(t)\left(\sum_{i=1}^{n}\frac{s_i}{t-\tau_i}\right)r(t) + w(t)r'(t) \tag{46}$$

so $p'(t)$ is not much more complicated. In all cases, we must avoid $t = \tau_i$ for any node. Newton's formula for a root of $r(t)$ is

$$\lambda_{m+1} = \lambda_m - \frac{r(\lambda_m)}{r'(\lambda_m)} \tag{47}$$

whereas that for $p(t)$ is

$$\mu_{m+1} = \mu_m - \frac{r(\mu_m)}{r'(\mu_m) + r(\mu_m)\displaystyle\sum_{i=1}^{n}\frac{s_i}{\mu_m-\tau_i}} \tag{48}$$

and one may switch between formulas, if desired. The secant method is also potentially attractive, but we do not pursue this here.

Experimentation is needed to decide which of these alternatives, if any, is to be preferred; in the meantime, the companion matrix method allows experimentation. Initial guesses may be obtained in many cases by examining the end point data. In the case of cubic Hermite polynomial interpolation, a comprehensive scheme may be worked out to decide if roots are guaranteed to exist on $[0, 1]$ or not. For example, if y_n and y'_n are positive, but $y_{n+1} > 0$ and $y'_{n+1} < 0$, there can be no real root in the time step (because otherwise there would be three extrema, and there can be only two). Again we do not pursue this further, here.

8 Concluding Remarks

We have presented a uniform method for using general rational Hermite interpolants for various purposes in IVP codes for ODE, including event location for polynomial events. This method relies on the good conditioning of rational Hermite interpolants, on the apparent numerical stability of their construction, and on the apparent numerical stability of the solution of a recent generalized companion matrix pencil. All three of these "legs" on which this paper stands need further investigation and formal proof, but our experiments indicate that these should be successful.

Acknowledgment

The authors wish to thank Larry Shampine for his encouragement of this paper, and Nick Trefethen & Jean-Paul Berrut for comments on a draft. This paper was written while the first author was a Visiting Fellow at the John Curtin School of Medical Research at Australian National University, and was supported by NSERC. The anonymous reviewers provided helpful and very timely comments.

References

[1] A. Amiraslani. *Algorithms for Matrices, Polynomials, and Matrix Polynomials*. PhD thesis, University of Western Ontario, London, Ontario, May 2006.

[2] A. Amiraslani, Robert M. Corless, and Peter Lancaster. Linearization of matrix polynomials expressed in polynomial bases. *IMA J. Numerical Analysis*, accepted 2007.

[3] U. Ascher, S. Pruess, and R. D. Russell. On spline basis selection for solving differential equations. *SIAM Journal on Numerical Analysis*, 20:121–142, February 1983.

[4] J.-P. Berrut. The barycentric weights of rational interpolation with prescribed poles. *J. Comput. Appl. Math.*, 86:45–52, 1997.

[5] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004.

[6] Dario A. Bini, Luca Gemignani, and Victor Y. Pan. Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations. *Numer. Math.*, 100:373–408, 2005.

[7] R. W. Brankin and I. Gladwell. Shape-preserving local interpolation for plotting solutions of ODEs. *IMA Journal of Numerical Analysis*, 9:555–566, October 1989.

[8] Robert M. Corless, N. Rezvani, and A. Amiraslani. Pseudospectra of matrix polynomials that are expressed in alternative bases. *Mathematics in Computer Science*, 2007.

[9] Robert M. Corless and S. M. Watt. Bernstein bases are optimal, but, sometimes, Lagrange bases are better. In *Proceedings of SYNASC*, pages 141–153, Timisoara, Romania, September 2004. Mirton Press.

[10] Isabella Cravero and Carlo Manni. Shape-preserving interpolants with high smoothness. *Journal of Computational and Applied Mathematics*, 157:383–405, 2003.

[11] A. Edelman and H. Murakami. Polynomial roots from companion matrix eigenvalues. *Mathematics of Computation*, 64:763–776, 1995.

[12] R. T. Farouki and T. N. T. Goodman. On the optimal stability of the Bernstein basis. *Mathematics of Computation*, 65:1553–1566, 1996.

[13] Luca Gemignani. Structured matrix methods for polynomial root-finding. In *ISSAC '07: Proceedings of the 2007 international symposium on Symbolic and algebraic computation*, pages 175–180, New York, NY, USA, 2007. ACM.

[14] S. Goedecker. Remark on algorithms to find roots of polynomials. *SIAM Journal of Scientific Computing*, 15(5):1059–1063, 1994.

[15] P. Henrici. *Applied and Computational Complex Analysis*. John Wiley & Sons, New York, 1974.

[16] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, 1996.

[17] N. J. Higham. The numerical stability of barycentric Lagrange interpolation. *IMA Journal of Numerical Analysis*, 24(4):547–556, 2004.

[18] T. E. Hull and R. Mathon. The mathematical basis and a prototype implementation of a new polynomial rootfinder with quadratic convergence. *ACM TOMS*, 22(3):261–280, 1996.

[19] Louis M. Milne-Thomson. *The Calculus of Finite Diffrences*. Macmillan, London, 1933.

[20] C. Moler. Roots of polynomials. *Cleve's Corner, the Mathematics Newsletter*, 5:8–9, 1991.

[21] T. J. Rivlin. *Chebyshev Polynomials*. Wiley-Interscience, 1990.

[22] H. Rutishauser. *Lectures on Numerical Mathematics*. Birkhauser, 1990.

[23] A. Shakoori. *Bivariate Polynomial Solver by Values*. PhD thesis, The University of Western Ontario, 2007.

[24] A. Shakoori, D. A. Aruliah, Robert M. Corless, and L. Gonzalez-Vega. Bézoutians and companion matrix pencils for barycentric Hermite interpolants. To be submitted.

[25] L. F. Shampine, I. Gladwell, and R. W. Brankin. Reliable solution of special event location problems for ODEs. *ACM Trans. Math. Softw.*, 17(1):11–25, 1991.

[26] Lawrence F. Shampine. Interpolation for Runge-Kutta methods. *SIAM Journal on Numerical Analysis*, 22(5):1014–1027, 1985.

[27] K.-C. Toh and Lloyd N. Trefethen. Pseudozeros of polynomials and pseudospectra of companion matrices. *Numerische Mathematik*, 68:403–425, 1994.

[28] W. Werner and C. Schneider. Hermite interpolation: The barycentric approach. *Computing*, 46(1):35–51, 1990.

[29] Wilhelm Werner. Polynomial interpolation: Lagrange versus Newton. *Mathematics of Computation*, 43(167):205–217, 1984.