

## Sparse Interpolation

Interpolation is a classic numeric-symbolic algorithm, used to find polynomial representations of factors. The main source for this lecture is [2]. As application we look at singularities of Stewart-Gough platforms [9].

### 1 Prony's Method

Consider the exponential function

$$F(x) = c_1 e^{\mu_1 x} + c_2 e^{\mu_2 x} \quad (1)$$

for unknown values of the coefficients  $c_1$ ,  $c_2$ , and exponents  $\mu_1$ ,  $\mu_2$ . To determine these four unknowns, we evaluate at  $x = 0, 1, 2, 3$ :

$$\begin{cases} F(0) = c_1 + c_2 = c_1 + c_2 \\ F(1) = c_1 e^{\mu_1} + c_2 e^{\mu_2} = c_1 b_1 + c_2 b_2 \\ F(2) = c_1 e^{2\mu_1} + c_2 e^{2\mu_2} = c_1 b_1^2 + c_2 b_2^2 \\ F(3) = c_1 e^{3\mu_1} + c_2 e^{3\mu_2} = c_1 b_1^3 + c_2 b_2^3 \end{cases} \quad (2)$$

where  $b_1 = e^{\mu_1}$  and  $b_2 = e^{\mu_2}$ . Viewing (2) as a linear system in  $c_1$  and  $c_2$ , observe the Vandermonde matrix. We separate the coefficients  $c_1$  and  $c_2$  from  $b_1$  and  $b_2$  by the introduction of a auxiliary polynomial

$$p(z) = (z - b_1)(z - b_2) = z^2 + \lambda_1 z + \lambda_0. \quad (3)$$

If we know the coefficients of  $p$ , then  $b_1$  and  $b_2$  are found by computing the roots of  $p$ . Once  $b_1$  and  $b_2$  are known, we find  $c_1$  and  $c_2$  by solving a linear system.

Because  $p(b_1) = 0$  and  $p(b_2) = 0$ , we have

$$\begin{cases} c_1 p(b_1) + c_2 p(b_2) = 0 \\ c_1 b_1 p(b_1) + c_2 b_2 p(b_2) = 0 \end{cases} \quad (4)$$

which leads to

$$\begin{cases} c_1 (b_1^2 + \lambda_1 b_1 + \lambda_0) + c_2 (b_2^2 + \lambda_1 b_2 + \lambda_0) = 0 \\ c_1 (b_1^3 + \lambda_1 b_1^2 + \lambda_0 b_1) + c_2 (b_2^3 + \lambda_1 b_2^2 + \lambda_0 b_2) = 0. \end{cases} \quad (5)$$

Collecting terms in  $\lambda_0$  and  $\lambda_1$  gives

$$\begin{cases} \lambda_0 (c_1 + c_2) + \lambda_1 (c_1 b_1 + c_2 b_2) + c_1 b_1^2 + c_2 b_2^2 = 0 \\ \lambda_0 (c_1 b_1 + c_2 b_2) + \lambda_1 (c_1 b_1^2 + c_2 b_2^2) + c_1 b_1^3 + c_2 b_2^3 = 0. \end{cases} \quad (6)$$

As we recognize  $F(0)$ ,  $F(1)$ ,  $F(2)$ , and  $F(3)$ , we write the system in shortened form as

$$\begin{bmatrix} F(0) & F(1) \\ F(1) & F(2) \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \end{bmatrix} = - \begin{bmatrix} F(2) \\ F(3) \end{bmatrix}. \quad (7)$$

The matrix is a Hankel matrix. To find the coefficients and exponents of (1), the method of Prony [5] executes three steps:

1. Solve the Hankel system (7) to find the coefficients  $\lambda_1$  and  $\lambda_0$  of  $p(z)$  in (3).
2. Solve  $p(z) = 0$ . The roots of  $p$  are  $b_1$  and  $b_2$ . So the exponents are  $\mu_1 = \ln(b_1)$  and  $\mu_2 = \ln(b_2)$ .
3. Solve the first two equations of (2) for  $c_1$  and  $c_2$ . For known  $b_1$  and  $b_2$ , this system is linear.

## 2 Singularities of Stewart-Gough platforms

A Stewart-Gough platform consists of a moving platform supported above a stationary base by six legs. One end of the  $i$ -th leg connects to the base via a ball joint centered on point  $\mathbf{b}_i$  (given in the base coordinate system) and the other end connects to the platform via a ball joint at point  $\mathbf{a}_i$  (given in platform coordinates). The length of the leg,  $L_i$ , is controlled by a linear actuator. In designing the platform, an understanding of its singularities is crucial; see [6, 7].

Here we follow [9]. The condition for singularity can be derived as follows. We represent the position of the platform by  $\mathbf{p} \in \mathbb{C}^3$  and the orientation by a quaternion  $\mathbf{q} \in \mathbb{P}^3$ . Letting  $\mathbf{v}_i$  be the vector, in base coordinates, from base point  $\mathbf{b}_i$  to the corresponding platform point  $\mathbf{a}_i$ , we have

$$\mathbf{v}_i := -\mathbf{b}_i + \mathbf{p} + R(\mathbf{q})\mathbf{a}_i = -\mathbf{b}_i + \mathbf{p} + \mathbf{q}\mathbf{a}_i\mathbf{q}'/\mathbf{q}\mathbf{q}', \quad (8)$$

where  $R(\mathbf{q})$  is the  $3 \times 3$  rotation matrix giving the same rotation to a vector  $\mathbf{w}$  as the quaternion operation  $\mathbf{q}\mathbf{w}\mathbf{q}'/\mathbf{q}\mathbf{q}'$ . The squared length of leg  $i$  is  $L_i^2 = \mathbf{v}_i \cdot \mathbf{v}_i$ , so

$$L_i \dot{L}_i = \mathbf{v}_i \cdot \dot{\mathbf{v}}_i = \mathbf{v}_i \cdot (\dot{\mathbf{p}} + \boldsymbol{\omega} \times (R\mathbf{a}_i)), \quad (9)$$

where “ $\cdot$ ” is the vector inner product,  $\times$  is the vector cross product, and  $\boldsymbol{\omega}$  is the angular velocity vector. Rewriting  $R = \widehat{R}/Q$  with

$$Q := \mathbf{q}\mathbf{q}' = q_0^2 + q_1^2 + q_2^2 + q_3^2, \quad (10)$$

and substituting from (8), we may transform (9) into

$$QL_i \dot{L}_i = (Q(\mathbf{p} - \mathbf{b}_i) + \widehat{R}\mathbf{a}_i) \cdot \dot{\mathbf{p}} + ((\widehat{R}\mathbf{a}_i) \times (\mathbf{p} - \mathbf{b}_i)) \cdot \boldsymbol{\omega}. \quad (11)$$

Letting  $\mathbf{J}$  be the matrix whose  $i$ -th column is

$$\mathbf{J}_i = \begin{bmatrix} Q(\mathbf{p} - \mathbf{b}_i) + \widehat{R}\mathbf{a}_i \\ (\widehat{R}\mathbf{a}_i) \times (\mathbf{p} - \mathbf{b}_i) \end{bmatrix}, \quad i = 1, 2, \dots, 6, \quad (12)$$

the singularity condition is  $0 = \mathbf{J}\mathbf{w}$  for some  $\mathbf{w} \neq 0$ , or equivalently,  $\det(\mathbf{J}) = 0$ . The expanded  $\det(\mathbf{J})$  defines the polynomial we will factor.

Since the elements of  $\widehat{R}$  and  $Q$  are quadratic polynomials in  $\mathbf{q}$ , one sees that  $\det(\mathbf{J})$  is a polynomial in  $\mathbf{p}$ ,  $\mathbf{q}$ ,  $\mathbf{a}_i$ ,  $\mathbf{b}_i$ . Taking all of these as variables, the first three rows of  $\mathbf{J}$  are cubic and the last three are quadric, so  $\det(\mathbf{J})$  is a homogeneous polynomial of degree 1728 in 42 variables. Considerable insight can be gained by studying cases where some variables are taken as given, as has been done in [6, 8].

The cases that were studied in [9] are:

1. General platform, fixed position: In this case,  $\det(\mathbf{J})$  is a homogeneous polynomial of degree 12 with 910 terms in  $\mathbf{q} = \{q_0, q_1, q_2, q_3\}$ . The factorization is  $\det(\mathbf{J}) = F_1(\mathbf{q})Q^3$ , where  $F_1(\mathbf{q})$  is a sextic. The factor of  $Q^3 = 0$  is not of physical significance.
2. Planar base and platform, fixed position:  $\det(\mathbf{J})$  is still homogeneous of degree 12, and still factors in two: one irreducible single factor of degree six, and the quadratic factor having multiplicity three.
3. Planar base and platform, parallel planes: In this case, which was studied in [6, 8],  $\det(\mathbf{J})$  becomes cubic in  $\mathbf{p}$ , homogeneous of degree 12 in  $(q_0, q_3)$ , and degree 15 in  $(\mathbf{p}, \mathbf{q})$  together. This polynomial is much sparser: only 24 terms. The computed factorization is

$$\det(\mathbf{J}) = ap_3^3(q_0 + bq_3)(q_0 + cq_3)(q_0 + iq_3)^5(q_0 - iq_3)^5, \quad (13)$$

where  $a, b, c$  are constants that depend on the choice of  $\mathbf{a}_i$ ,  $\mathbf{b}_i$ . This result is in agreement with [8], when we consider that, over the complex numbers, any homogeneous polynomial in two variables breaks into linear factors.

### 3 The QZ Method for $A\mathbf{x} = \lambda B\mathbf{x}$

A generalization of the eigenvalue problem is  $A\mathbf{x} = \lambda B\mathbf{x}$ ,  $A, B \in \mathbb{C}^{n \times n}$ , where  $\lambda$  is a scalar and  $\mathbf{x}$  an  $n$ -vector. We look for eigenvalues  $\lambda \in \mathbb{C}$  so that  $(A - \lambda B)\mathbf{x} = \mathbf{0}$  admits an eigenvector  $\mathbf{x} \in \mathbb{C}^n$ ,  $\mathbf{x} \neq \mathbf{0}$ . If  $B$  has an inverse  $B^{-1}$ , then we could just solve  $B^{-1}A\mathbf{x} = \lambda\mathbf{x}$ . However, even if  $B^{-1}$  would exist, an ill conditioned matrix  $B$  will prevent the accurate computation of well conditioned eigenvalue-eigenvector pairs.

Instead of solving  $B^{-1}A\mathbf{x} = \lambda\mathbf{x}$ , we first compute the Schur decomposition of  $A$  and  $B$ :

$$\begin{cases} Q^H A Z = T, & Q^H Q = I, Z^H Z = I, \\ Q^H B Z = S, & T \text{ and } S \text{ are upper triangular.} \end{cases} \quad (14)$$

$I$  represents the identity matrix. By  $\cdot^H$  we denote the conjugate transposition:  $(a_{ij})^H = \bar{a}_{ji}$ . Applying the transformations to the problem  $A\mathbf{x} = \lambda B\mathbf{x}$  gives  $Q^H(A - \lambda B)Z = Q^H A Z - \lambda Q^H B Z = T - \lambda S$ , reducing  $A\mathbf{x} = \lambda B\mathbf{x}$  to  $T\mathbf{x} = \lambda S\mathbf{x}$ .

Orthogonal (or unitary) matrices to introduce zeroes are defined via Givens rotations:

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} z \\ 0 \end{bmatrix}, \quad z = \sqrt{x^2 + y^2}, \quad c = \frac{x}{z}, \quad s = -\frac{y}{z}, \quad (15)$$

where  $c$  and  $s$  are respectively  $\cos(\theta)$  and  $\sin(\theta)$  for the rotation angle  $\theta$ .

The first step to obtain a Schur decomposition is to compute a Hessenberg-triangular reduction. A matrix  $A = (a_{ij})$  is upper Hessenberg when  $a_{ij} = 0$  for all  $i > j + 1$ . In Algorithm 3.1 we outline the computation of the Hessenberg-triangular reduction of  $(A, B)$ .

#### Algorithm 3.1 Hessenberg-Triangular Reduction of $(A, B)$

Input:  $A, B \in \mathbb{C}^{n \times n}$ .

Output:  $Q, Z, T, S \in \mathbb{C}^{n \times n}$ :  $Q^H Q = I, Z^H Z = I,$

$Q^H A Z = T, T$  is upper Hessenberg,

$Q^H B Z = S, S$  is upper triangular.

$[Q^H, B] := QR(B);$

*make B upper triangular via QR*

$A := Q^H A; Z := I;$

*perform same transformation on A*

for  $j$  from 1 to  $n - 1$  do

*work on column j of A*

  for  $i$  from  $n + 1 - j$  to  $j + 2$  do

*make (i, j)th element of A zero*

$A := Q_{ij}^H A; B := Q_{ij}^H B; Q := Q_{ij}^H Q;$

*Givens rotation  $Q_{ij}$  makes 0 in A*

$A := A Z_{ij}; B := B Z_{ij}; Z := Z Z_{ij};$

*Givens rotation  $Z_{ij}$  restores 0 in B*

return  $(Q, Z, A, B)$ .

If there would be a zero on the diagonal of  $B$ , then via Givens rotations  $Q^H$  and  $Z$  on both  $A$  and  $B$  that zero can be push down to the  $(n, n)$ th position of  $B$ . This process of pushing down zeroes is called deflation.

We can now describe one QZ step. Let  $(A, B)$  be in Hessenberg-triangular form and assume  $B^{-1}$  exists, then  $C = AB^{-1}$  is upper Hessenberg.

1. Take  $\mathbf{c}$  as the first column of  $C$ . Denote  $\mathbf{e}_1 = [1 \ 0 \ 0 \ \dots \ 0]^T$  and let  $\mathbf{u} = \mathbf{c} \pm \|\mathbf{c}\|_2 \mathbf{e}_1$ , choosing  $+$  or  $-$  so  $u_1 > 0$ . Then the matrix  $H = \left( I - 2 \frac{\mathbf{u}\mathbf{u}^T}{\mathbf{u}^T \mathbf{u}} \right)$  defines a Householder reflection:  $H\mathbf{c} = \pm \|\mathbf{c}\|_2 \mathbf{e}_1$ .
2. Determine unitary matrices  $Q$  and  $Z$  with  $Q\mathbf{e}_1 = \mathbf{e}_1$  so that  $(Q^H(HA)Z, Q^H(HB)Z)$  is in Hessenberg-triangular form.

As a result of these steps, we can expect one of the subdiagonal elements of  $A$  to approach zero or one of the diagonal elements of  $B$  to converge to zero, which can then be deflated.

Our introduction to the QZ algorithm follows [4]. Its running time is cubic in the dimension of the matrices. MATLAB and Octave provide implementation of the QZ algorithm.

## 4 Reformulating Prony's Method

The three steps in Prony's algorithm all suffer numerical problems:

- (1) a Hankel matrix might be very ill-conditioned;
- (2) the roots are often very sensitive to small changes in the coefficients;
- (3) Vandermonde matrices are also typically ill-conditioned.

Therefore, observe the following decomposition:

$$\begin{aligned} \begin{bmatrix} 1 & 1 \\ b_1 & b_2 \end{bmatrix} \underbrace{\begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix}} \begin{bmatrix} 1 & b_1 \\ 1 & b_2 \end{bmatrix} &= \underbrace{\begin{bmatrix} F(0) & F(1) \\ F(1) & F(2) \end{bmatrix}} \\ \begin{bmatrix} 1 & 1 \\ b_1 & b_2 \end{bmatrix} \underbrace{\begin{bmatrix} c_1 & c_1 b_1 \\ c_2 & c_2 b_2 \end{bmatrix}} &= \underbrace{\begin{bmatrix} c_1 + c_2 & c_1 b_1 + c_2 b_2 \\ c_1 b_1 + c_2 b_2 & c_1 b_1^2 + c_2 b_2^2 \end{bmatrix}}, \end{aligned} \quad (16)$$

abbreviated as  $VDV^T = H_0$ . Multiply the diagonal matrix  $D$  with another diagonal matrix that has  $b_1$  and  $b_2$  on its diagonal:

$$\begin{aligned} \begin{bmatrix} 1 & 1 \\ b_1 & b_2 \end{bmatrix} \underbrace{\begin{bmatrix} c_1 b_1 & 0 \\ 0 & c_2 b_2 \end{bmatrix}} \begin{bmatrix} 1 & b_1 \\ 1 & b_2 \end{bmatrix} &= \underbrace{\begin{bmatrix} F(1) & F(2) \\ F(2) & F(3) \end{bmatrix}} \\ \begin{bmatrix} 1 & 1 \\ b_1 & b_2 \end{bmatrix} \underbrace{\begin{bmatrix} c_1 b_1 & c_1 b_1^2 \\ c_2 b_2 & c_2 b_2^2 \end{bmatrix}} &= \underbrace{\begin{bmatrix} c_1 b_1 + c_2 b_2 & c_1 b_1^2 + c_2 b_2^2 \\ c_1 b_1^2 + c_2 b_2^2 & c_1 b_1^3 + c_2 b_2^3 \end{bmatrix}}, \end{aligned} \quad (17)$$

abbreviated as  $VDBV^T = H_1$ . In [3], the two Hankel matrices  $H_0$  and  $H_1$  are diagonalized simultaneously:

$$V^{-1}H_0V^{-T} = D \quad (18)$$

$$V^{-1}H_1V^{-T} = DB. \quad (19)$$

Therefore, we obtained the generalized eigenvalue problem

$$(H_1 - \lambda H_0)\mathbf{v} = \mathbf{0}. \quad (20)$$

This reformulation of Prony's method as a generalized eigenvalue problem avoids the computation of the coefficients of the auxiliary polynomial and the corresponding root finding problem.

To improve the conditioning of the sparse interpolation problem, the authors of [2] propose to sample at roots of unity. In [2] estimates of the condition number of the matrices  $V$  are given and the results of computational experiments are listed.

## 5 the qd algorithm

To determine a bound on the number of samples, the application of the quotient-difference algorithm (or qd-algorithm) is explained in [5] (and further elaborated in [1]) The qd-algorithm is an iterative numerical scheme to determine the number of poles of a meromorphic function from its Taylor series.

The problem of sparse interpolation is to determine the number of monomials in the polynomial

$$f(\mathbf{x}) = \sum_{\mathbf{a} \in A} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}}, \quad m = \#A. \quad (21)$$

Given is a blackbox function to evaluate  $f$  at roots of unity  $\omega_k$  yielding values  $v_k = f(\omega_k)$ ,  $k = 0, 1, \dots, 2m-1$ . The problem is to determine  $m$ .

If  $m = 3$ , then we construct the polynomial  $p(x) = x^3 + \lambda_2 x^2 + \lambda_1 x + \lambda_0$  where

$$\begin{bmatrix} v_0 & v_1 & v_2 \\ v_1 & v_2 & v_3 \\ v_2 & v_3 & v_4 \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = - \begin{bmatrix} v_3 \\ v_4 \\ v_5 \end{bmatrix}. \quad (22)$$

Moving the righthandside of the above equation to the left and adding  $\lambda_0 + \lambda_1 x + \lambda_2 x^2 + x^3 = 0$  gives the matrix equation

$$\begin{bmatrix} v_0 & v_1 & v_2 & v_3 \\ v_1 & v_2 & v_3 & v_4 \\ v_2 & v_3 & v_4 & v_5 \\ 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (23)$$

The determinant of this matrix equation is the polynomial denoted by  $H_3^{(0)}(x)$ , where its leading coefficient is the determinant of the Hankel matrix  $H_3^{(0)}$ . Observe that, if  $m = 2$ , then  $\det H_3^{(0)} = 0$ .

A  $k$ -by- $k$  Hankel matrix  $H_k^{(j)}$  is defined by

$$H_k^{(j)} = \begin{bmatrix} v_j & v_{j+1} & \cdots & v_{j+k-1} \\ v_{j+1} & v_{j+2} & \cdots & v_{j+k} \\ \vdots & \vdots & \ddots & \vdots \\ v_{j+k-1} & v_{j+k} & \cdots & v_{j+2k-2} \end{bmatrix}. \quad (24)$$

Neighboring Hankel determinants satisfy

$$\left(\det H_k^{(j)}\right)^2 = \det H_k^{(j-1)} \det H_k^{(j+1)} - \det H_{k+1}^{(j-1)} \det H_{k-1}^{(j+1)}. \quad (25)$$

The qd algorithm bypasses the computation of Hankel determinants.

The initialization of the qd algorithm goes as

$$e_0^{(j)} = 0, j = 0, 1, \dots \quad q_1^{(j)} = \frac{v_{j+1}}{v_j}, j = 1, 2, \dots \quad (26)$$

and

$$q_k^{(0)} = \frac{\det H_{k-1}^{(0)} \det H_k^{(1)}}{\det H_k^{(0)} \det H_{k-1}^{(1)}}, \quad e_k^{(0)} = \frac{\det H_{k+1}^{(0)} \det H_{k-1}^{(1)}}{\det H_k^{(0)} \det H_k^{(1)}}, \quad k = 1, 2, \dots \quad (27)$$

Then the progressive qd algorithm uses the formulas

$$q_k^{(j+1)} = e_{k-1}^{(j+1)} - e_k^{(j)} + q_k^{(j)}, \quad e_k^{(j+1)} = \left( \frac{q_{k+1}^{(j)}}{q_k^{(j+1)}} \right) e_k^{(j)}, \quad \text{for } k = 1, 2, \dots, j = 0, 1, \dots \quad (28)$$

The qd algorithm computes both the roots  $z_k$  of the polynomial  $p$  and its degree  $m$ . The key properties are  $\lim_{j \rightarrow \infty} q_k^{(j)} = z_k$  and  $e_m^{(j)} = 0$  for all  $j = 1, 2, \dots$

## 6 Exercises

1. Suppose a polynomial in one variable is given as a blackbox function: evaluation is cheap, but we do not know its algebraic representation. Explain by example(s) how divided differences and Newton interpolation lead to the degree of the polynomial.

2. Generate Vandermonde matrices of increasing dimension (using MATLAB or Octave) and compute their condition number.

Take interpolation points in  $[-1, +1]$  and compare the condition numbers for (1) equidistant interpolation points; (2) randomly chosen points, from a uniform distribution; and (3) for interpolation points as the roots of the Chebyshev polynomials.

3. Verify  $H\mathbf{c} = \pm\|\mathbf{c}\|_2\mathbf{e}_1$ , for  $\mathbf{e}_1 = [1\ 0\ 0\ \cdots\ 0]^T$ ,  $\mathbf{u} = \mathbf{c} \pm \|\mathbf{c}\|_2\mathbf{e}_1$ , and  $H = \left(I - 2\frac{\mathbf{u}\mathbf{u}^T}{\mathbf{u}^T\mathbf{u}}\right)$ .
4. Use the QZ algorithm as implemented in MATLAB or Octave to verify an instance of (16).
5. Consider  $f(x, y) = x^5y + 0.1xy^{13} - 0.5xy + 2.2x^4y^4$ . Describe how to recover the exponents and coefficients, knowing that  $f$  has four terms.

## References

- [1] A. Cuyt and W.-s. Lee. A new algorithm for sparse interpolation of multivariate polynomials. *Theoretical Computer Science*, 409(2):180–185, 2008. Special Issue on Symbolic-Numeric Computations edited by D. Bini, V.Y. Pan, and J. Verschelde.
- [2] M. Giesbrecht, G. Labahn, and W.-s. Lee. Symbolic-numeric sparse interpolation of multivariate polynomials. In J.-G. Dumas, editor, *Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation (ISSAC 2006)*, pages 116–123. ACM, 2006.
- [3] G.H. Golub, P. Milanfar, and J. Varah. A stable numerical method for inverting shape from moments. *SIAM J. Sci. Comput.*, 21(4):1222–1243, 1999.
- [4] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, first edition, 1983.
- [5] W.-s. Lee. From quotient-difference to generalized eigenvalues and sparse polynomial interpolation. In J. Verschelde and S.M. Watt, editors, *SNC'07. Proceedings of the 2007 International Workshop on Symbolic-Numeric Computation*, pages 110–116. ACM, 2007.
- [6] B. Mayer St-Onge and C.M. Gosselin. Singularity analysis and representation of the general Gough-Stewart platform. *Int. J. Robotics Research*, 19(3):271–288, 2000.
- [7] J.P. Merlet. Singular configurations of parallel manipulators and Grassmann geometry. *Int. J. Robotics Research*, 8(5):45–56, 1989.
- [8] F. Pernkopf and M.L. Husty. Singularity analysis of spatial Stewart-Gough platforms with planar base and platform. In *Proc. ASME Design Eng. Tech. Conf.*, Montreal, Canada, Sept. 30–Oct. 2, 2002.
- [9] A.J. Sommese, J. Verschelde, and C.W. Wampler. Numerical factorization of multivariate complex polynomials. *Theoretical Computer Science*, 315(2-3):651–669, 2004. Special Issue on Algebraic and Numerical Algorithms edited by I.Z. Emiris, B. Mourrain, and V.Y. Pan.