Making minimal solvers inverse-free using null space computation

Supplementary Material

In this supplementary part, the next section presents the connection between realization theory and the proposed method. In Section 2, we discuss the effect of spurious solutions on the null space computation process by using an example. Finally, in Section 4, we provide additional error histograms and a table showing the sizes of the coefficient matrices for the minimal problems.

1. Realization theory and sparse reultant-based solvers

Our proposed method is considered to be part of sparse resultant-based methods. As stated earlier, in [1], it was proven that sparse resultant and action-based solvers are equivalent under certain conditions. Realization theory is another approach for solving polynomial systems of equations [2, 3], particularly in control theory applications. It can also be shown that realization theory and sparse resultant-based methods are equivalent under certain conditions.

In realization theory, the properties of multivariate Vandermonde vectors are used to convert the problem into a generalized eigenvalue problem. Notably, this approach does not require the system to have a square coefficient matrix.

A multivariate Vandermonde vector is an extension of the univariate Vandermonde vectors. Suppose $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ is a vector of variables. Then a multivariate Vandermonde vector is defined as follows,

$$V(\mathbf{x}) = \begin{bmatrix} \mathbf{x}^{\alpha^{(1)}} \\ \mathbf{x}^{\alpha^{(2)}} \\ \vdots \\ \mathbf{x}^{\alpha^{(m)}} \end{bmatrix} \tag{1}$$

where each $\alpha^{(i)} = (\alpha_1^{(i)}, \ldots, \alpha_n^{(i)})$ is the exponent vector of a monomial and $\mathbf{x}^{\alpha^{(i)}} = x_1^{\alpha_1^{(i)}} x_2^{\alpha_2^{(i)}} \cdots x_n^{\alpha_n^{(i)}}$ is the corresponding monomial. Based on this definition, the eigenvectors encountered in resultant-based methods are each a multivariate Vandermonde vector.

The method based on realization theory relies on the observation that if we multiply the Vandermonde vector of a solution by one component of that solution, some of the monomials presented in the original Vandermonde vector will also appear in the product. For simplicity, suppose we have two variables x, and y, and the system degree is two, then if we mutiply the Vandermonde vector by x, we have

$$\begin{bmatrix} 1 \\ y \\ y^2 \\ x \\ xy \\ x^2 \end{bmatrix} x = \begin{bmatrix} x \\ xy \\ xy^2 \\ x^2 \\ x^2y \\ x^3 \end{bmatrix} \tag{2}$$

Obviously, the first, second and fourth elements of the resulting vector also appear in the original Vandermonde vector. This relationship can be expressed using the following equation:

$$P_1 V \sigma = P_2 V \tag{3}$$

where V represents the Vandermonde vector of the solution, σ denotes a component of the solution such as x or y, which is also referred to as the shift function. The matrices P_1 and P_2 are row selection matrices.

In this formulation, V is replaced by a matrix formed from the null space of the Macaulay matrix, multiplied by a basis transformation matrix. As a result, the problem reduces to solving a generalized eigenvalue problem. This method has also been applied in a tensor context, where the tensor CP decomposition was used to extract the solutions of the polynomial system. [4].

In the context of sparse reultant-based solvers, suppose we introduce an auxiliary equation $x - \lambda$ where λ is a new variable. Then, if we replace x that was multiplied into the Vandermonde vector in the left part of Equation (2) with λ, then Equation (2) is actually consisted of polynomial $x - \lambda$ multiplied by six monomials.

Therefore, for the sparse resultant-based and realization theory methods to be equivalent, it suffices that the first, second and fourth equations of Equation (2) are present in the expanded system of polynomials used in the sparse resultant method. Based on this observation, we can state the following proposition.

Proposition 1 *If $P_1 V \lambda = P_2 V$ is present in the set of muliples of the extra equation $x - \lambda$, then the sparse reultant-based and relization theory methods are equivalent.*

2. The role of spurious solutions in null space computation

Spurious solutions are a known issue in resultant-based methods. This typically indicates that the coefficient matrix is larger than it should be. While extraneous solutions are generally considered a nuisance, in the context of null

Method	Sparse Resultant-based	GAPS
problem_9pt2radial	90×117	165×200
problem_relpose_6p_focal	12×30	31×50
problem_relpose_7p_fr_1s_partial_elim	22×41	52×71
problem_opt_pnp_hesch	90×117	88×115
problem_three-view_geometry_rad.dist.	146×178	190×222
problem_rel_pose_E+f_6pt	27×35	26×34
problem_p4p_optimal_abs_pos	339×371	359×392
problem_pc_relpose_5p_nulle_ne_simple	10×20	10×20
problem_triangulation_satellite	93×120	88×115
problem_unsync_relpose_diff_focal	120×166	104×152
problem_8ptF_radial_1s	11×20	11×20
problem_relpose_6p_focal_elim	21×36	21×36

Table 1. Sizes of the coefficient matrices for minimal problems using sparse resultant-based methods (Proposed and SRBM) and GAPS.

space calculations, they can sometimes be beneficial. For more details, see the next example.

Example 1 *In order to demonstrate the a key note in null space computation, consider the following system in two variables:*

$$\begin{aligned} f_1(x,y) &= a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6 \\ f_2(x,y) &= b_1x^2 + b_2xy + b_3y^2 + b_4x + b_5y + b_6 \end{aligned} \tag{4}$$

By adding the extra equation $f_3(x,y) := x - \lambda$ where λ is a new and hidden variable, we can construct the Macaulay matrix as follows:

$$\begin{pmatrix} b_6 & b_5 & b_3 & 0 & b_4 & b_2 & 0 & b_1 & 0 \\ 0 & b_6 & b_5 & b_3 & 0 & b_4 & b_2 & 0 & b_1 \\ a_6 & a_5 & a_3 & 0 & a_4 & a_2 & 0 & a_1 & 0 \\ 0 & a_6 & a_5 & a_3 & 0 & a_4 & a_2 & 0 & a_1 \\ -\lambda & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\lambda & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\lambda & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\lambda & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\lambda & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ y \\ y^2 \\ y^3 \\ x \\ xy \\ xy^2 \\ x^2 \\ x^2y \end{pmatrix} = 0 \tag{5}$$

We consider the transpose of the rows of the Macaulay matrix that corresponds to the multiplicands of equations f_1 and f_2 as A^t. We are intrested in the null space of this matrix.

To facilitate this, we apply a permutation matrix to rearrange the rows of A^t in descending order based on the number of non-zero elements. This reordering can improve the efficiency and stability of subsequent symbolic or numerical computations.

$$A^t = \begin{pmatrix} b_6 & 0 & a_6 & 0 \\ b_5 & b_6 & a_5 & a_6 \\ b_3 & b_5 & a_3 & a_5 \\ 0 & b_3 & 0 & a_3 \\ b_4 & 0 & a_4 & 0 \\ b_2 & b_4 & a_2 & a_4 \\ 0 & b_2 & 0 & a_2 \\ b_1 & 0 & a_1 & 0 \\ 0 & b_1 & 0 & a_1 \end{pmatrix} \Rightarrow PA^t = \begin{pmatrix} b_6 & 0 & a_6 & 0 \\ b_4 & 0 & a_4 & 0 \\ b_1 & 0 & a_1 & 0 \\ 0 & b_3 & 0 & a_3 \\ 0 & b_2 & 0 & a_2 \\ 0 & b_1 & 0 & a_1 \\ b_5 & b_6 & a_5 & a_6 \\ b_3 & b_5 & a_3 & a_5 \\ b_2 & b_4 & a_2 & a_4 \end{pmatrix} \tag{6}$$

As we can see, there are six rows with only two non-zero elements, and three of them have the same structure. This fact is helpful for making some rows zero and making other ones containing only one non-zero element.

If the extended system contained one more extraneous solution, we get

$$\begin{pmatrix} 0 & b_6 & b_5 & b_3 & 0 & b_4 & b_2 & 0 & b_1 & 0 \\ 0 & 0 & 0 & 0 & b_6 & b_5 & b_3 & b_4 & b_2 & b_1 \\ a_6 & a_5 & a_3 & 0 & a_4 & a_2 & 0 & a_1 & 0 & 0 \\ 0 & a_6 & a_5 & a_3 & 0 & a_4 & a_2 & 0 & a_1 & 0 \\ -\lambda & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\lambda & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\lambda & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\lambda & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\lambda & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ y \\ y^2 \\ y^3 \\ x \\ xy \\ xy^2 \\ x^2 \\ x^2y \\ x^3 \end{pmatrix} \tag{7}$$

Repeating the previous process we get,

$$B^t = \begin{pmatrix} 0 & 0 & a_6 & 0 \\ b_6 & 0 & a_5 & a_6 \\ b_5 & 0 & a_3 & a_5 \\ b_3 & 0 & 0 & a_3 \\ 0 & b_6 & a_4 & 0 \\ b_4 & b_5 & a_2 & a_4 \\ b_2 & b_3 & 0 & a_2 \\ 0 & b_4 & a_1 & 0 \\ b_1 & b_2 & 0 & a_1 \\ 0 & b_1 & 0 & 0 \end{pmatrix} \Rightarrow PB^t = \begin{pmatrix} 0 & b_1 & 0 & 0 \\ 0 & 0 & a_6 & 0 \\ 0 & b_4 & a_1 & 0 \\ 0 & b_6 & a_4 & 0 \\ b_3 & 0 & 0 & a_3 \\ b_2 & b_3 & 0 & a_2 \\ b_1 & b_2 & 0 & a_1 \\ b_5 & 0 & a_3 & a_5 \\ b_6 & 0 & a_5 & a_6 \\ b_4 & b_5 & a_2 & a_4 \end{pmatrix} \tag{8}$$

Evidently, the sructure of this matrix make the process of Gaussian Fraction-Free Elimination easier to do, because we have two one-element rows and three two-element rows.

As demonstrated in the previous example, working with a larger matrix—potentially containing extraneous solutions in the extended version of the system—can actually facilitate the computation of the null space. This is especially beneficial when the matrix includes single-element rows and elements with shorter algebraic expressions, which simplify symbolic elimination and make it easier to identify the null space. Notably, this approach is particularly advantageous when employing fraction-free Gaussian elimination, as it helps preserve the sparsity of the matrix in the resulting null space. We should note that while it is beneficial with bigger matrices, the esistance of extraneous solutions can have a effect on the final accuracy of the method, and they should be eliminate via for example association with really large eigenvalues.

3. Extracting null space from rows of transformation matrix

Given that the matrices for which we compute the null space are typically tall (i.e., have many rows), a more efficient strategy is to consider the transpose of the matrix. By applying fraction-free Gaussian elimination to the transposed matrix, we can zero out its last rows. The corresponding rows in the transformation matrix—those associated with the zero rows—can then be used to construct the null space.

More formally, for matrix A with size $k \times p$, we have a transformation matrix C such that

$$CA^t = \begin{bmatrix} A_1 \\ 0 \end{bmatrix} \tag{9}$$

where the number of zero rows is the same as the size of null space of A, which is $p - k$. The last $p - k$ rows of C denoted by Z span the null space of A.

Similar to Step 5 of Algorithm 1, this way is based on the fact that the null space of A is actually the orthogonal complement of the row space of A and therefore we can say that the null space is the orthogonal complement of the column space of A^t.

The main merit of this approach is that it eliminates the need for back-substitution, as the null space is directly extracted from the Fraction-free Gaussian elimination process. We can see from Algorithm 1 that this approach is mainly consisted of fraction-free Gaussian elimination.

Based on our experince, back-substitution approach is slower but more stable than the approach using transformation matrix. This makes the back-substitution approach more suitable for sparser matrices.

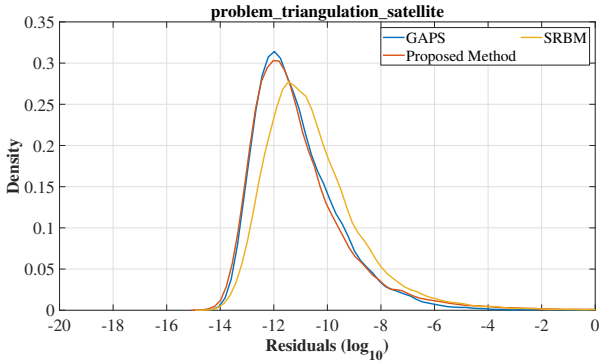
Algorithm 1: Compute Null Space via Transpose and Fraction-Free Elimination

Input: Symbolic matrix $A \in \mathbb{R}^{m \times n}$ with $m > n$
Output: Matrix N whose rows form a basis for null(A)

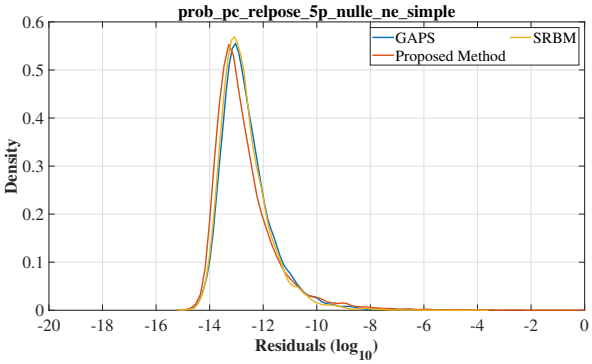
$A^T \leftarrow$ transpose of A;
$C \leftarrow I_m$ (identity matrix of size m);
$R \leftarrow A^T$;
$row \leftarrow 1$;
for $col = 1$ **to** n **do**
  Find the first nonzero entry in column col from row row downward;
  if *pivot found at row* k **then**
    if $k \neq row$ **then**
      Swap rows row and k in both R and C;
    for $i = row + 1$ **to** m **do**
      if $R[i, col] \neq 0$ **then**
        $R[i, :] \leftarrow R[row, col] \cdot R[i, :] - R[i, col] \cdot R[row, :]$;
        $C[i, :] \leftarrow R[row, col] \cdot C[i, :] - R[i, col] \cdot C[row, :]$;
    $row \leftarrow row + 1$;
rank $\leftarrow$ number of nonzero rows in R;
$N \leftarrow$ rows (rank $+ 1$) to m of C;
return N

4. Error histograms and coefficient matrix sizes

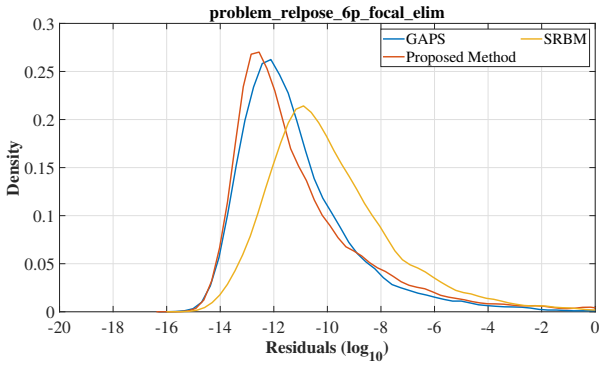
We have included additional plots of error histograms and the sizes of coefficient matrices in Figure 1 and Table 1. As can be seen from Table 1, the proposed method generally performs better in most cases when the size of the coefficient matrix is large.



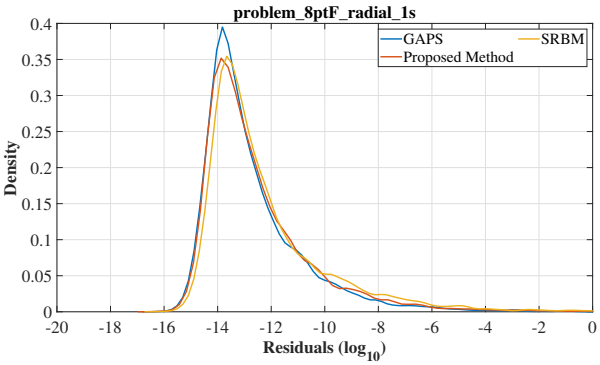
(a) Histogram of the problem_triangulation_satellite.



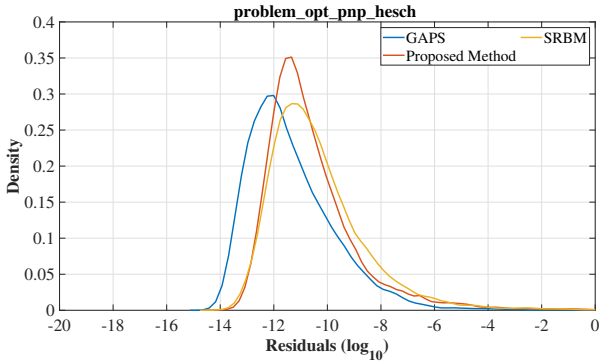
(b) Histogram of the prob_pc_relpose_5p_nulle_ne_simple.



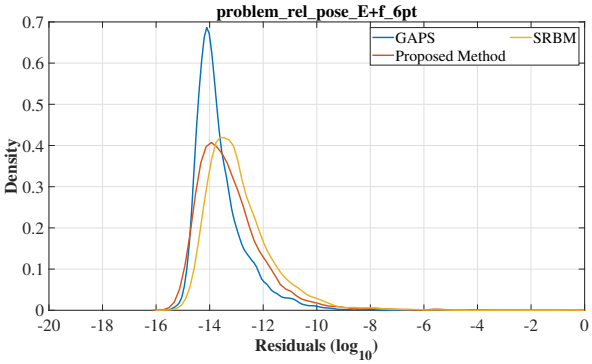
(c) Histogram of the problem_relpose_6p_focal_elim



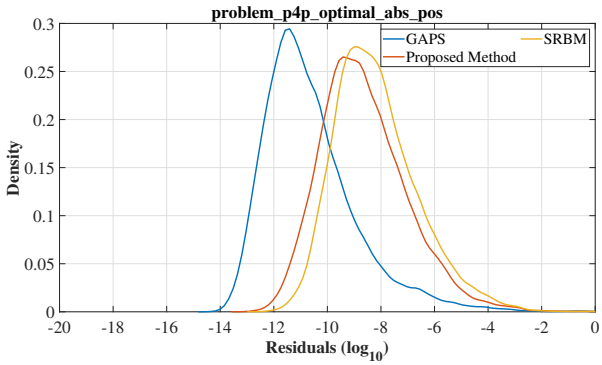
(d) Histogram of the problem_8ptF_radial_1s.



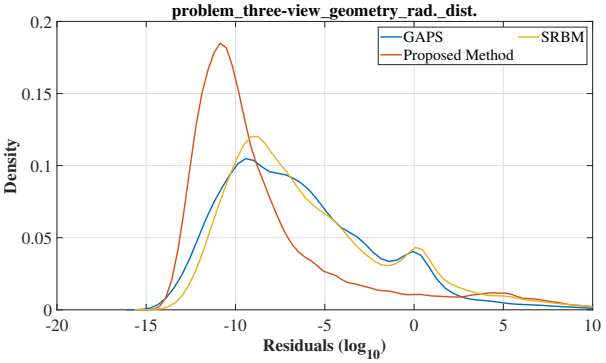
(e) Histogram of the problem_opt_pnp_hesch.



(f) Histogram of the problem rel pose E+f 6pt.



(g) Histogram of the problem_p4p_optimal_abs_pos.



(h) Histogram of the problem_three-view_geometry_rad._dist.

Figure 1. Histograms of Log_{10} residual errors for eight problems.

References

[1] Snehal Bhayani, Janne Heikkilä, and Zuzana Kukelova. Sparse resultant-based minimal solvers in computer vision and their connection with the action matrix. *Journal of Mathematical Imaging and Vision*, 66(3):335–360, 2024. 1

[2] Philippe Dreesen and Bart De Moor. Polynomial optimization problems are eigenvalue problems. In *Model-Based Control: Bridging Rigorous Theory and Advanced Technology*, pages 49–68, 2009. 1

[3] Philippe Dreesen, Kim Batselier, and Bart De Moor. Back to the roots: Polynomial system solving, linear algebra, systems theory. *IFAC Proceedings Volumes*, 45(16):1203–1208, 2012. 1

[4] Jeroen Vanderstukken, Patrick Kürschner, Ignat Domanov, and Lieven De Lathauwer. Systems of polynomial equations, higher-order tensor decompositions, and multidimensional harmonic retrieval: A unifying framework. part ii: The block term decomposition. *SIAM Journal on Matrix Analysis and Applications*, 42(2):913–953, 2021. 1