

COMPUTING THE NULL SPACE OF FINITE ELEMENT PROBLEMS

GIL SHKLARSKI AND SIVAN TOLEDO

We present a novel method to efficiently compute the null space of symmetric positive semidefinite finite-element matrices, possibly with additional linear equality constraints. Our method is strictly algebraic and does not use the geometry or material properties of the underlying model; the only information needed is the model in elemental form and a representation of the linear constraints. Our method works for any model with symmetric semidefinite elements. It works both when the constraints are symmetrically added to the finite-element matrix and when the constraints are appended to the symmetric finite-element matrix to create a rectangular matrix.

The computation of the null space of finite element matrices is an important task in many industrial applications. In structural mechanics the null space of a finite-element matrix corresponds to the zero energy modes of the structure. Such zero energy modes are important in static and dynamic analysis of floating structures, for example, where the computed null space is used to solve a consistent but rank-deficient system of linear equations. Certain domain decomposition methods also require the computation of the null space of floating substructures. This is the case for FETI, for the balancing domain decomposition method, and for some direct flexibility methods. Finally, algebraic multigrid methods such as smooth aggregation also benefit from an efficient computation of the null space of semidefinite finite-element matrices.

Let $K = \sum_e K_e$ be an n -by- n finite-element matrix, where all the K_e s are symmetric positive semidefinite element matrices. Let C be a c -by- n row block of linear constraints (C may be empty). Our ultimate goal is to compute the null space of the rectangular matrix

$$K_C = \begin{bmatrix} K \\ C \end{bmatrix},$$

or of the symmetric square matrix

$$\begin{bmatrix} K & C^T \\ C & \end{bmatrix}.$$

We compute the null space using a two phase process. In the first phase we efficiently compute an $(n + \ell)$ -by- $(n + \ell)$ matrix $\mathcal{F}(K)$ called the *Forest-Fretsaw Extension* of K . The matrix $\mathcal{F}(K)$ is an easy-to-factor approximation of K that preserves its null space (in a specific technical sense: the null space of K equals the null space of the Schur complement of $\mathcal{F}(K)$). The

theory behind the approximation $\mathcal{F}(K)$ was developed in an earlier paper and in this work linear time algorithms were developed and implemented.

In the second phase we compute the null space of

$$\hat{K} = \begin{bmatrix} \mathcal{F}(K) \\ \hat{C} \end{bmatrix},$$

where \hat{C} is the matrix C padded with ℓ zero columns. We then extract the null space of K from the computed null space of \hat{K} . Any reliable method for computing the null space of \hat{K} is appropriate here.

The key insight in our method is the construction of the matrix \hat{K} such that factoring it is significantly easier than factoring K_C . Once a matrix is factored using a backward-stable factorization, computing its null space is relatively easy using inverse power iterations. This technique works well even for rectangular matrices. Unfortunately, factoring K_C (or even just K) is expensive for three-dimensional models. Our method addresses this by replacing K by $\mathcal{F}(K)$, which preserves the null space but is easier to factor (even though its dimension is higher).

Theoretical considerations show that under certain conditions, both the amount of computation and the amount of memory required by our method scale linearly with model size; memory scales linearly but computation scales quadratically with the dimension of the null space.

Our experiments confirm this: the method scales extremely well on 3-dimensional model problems, leading to work and memory requirements that are linear in the size of the model. On this model problem, the new method outperforms two established methods, one based on inverse iteration the factors of the matrix, and the other based on a preconditioned iterative eigensolver (LOB-PCG with a Schwarz preconditioner).

In general, large industrial models do not satisfy all the conditions that the theoretical results assume; however, experimentally the method performs well and outperforms an established method on industrial models, including models with many equality constraints.

The accuracy of the computed null vectors is acceptable, but the computed null vectors are 2–5 decimal digits less accurate than those computed by the established methods (which are computationally much more expensive). Both the new method and one of the established methods are somewhat sensitive to a threshold parameter in the iterative solver that we use, which is based on inverse iteration.