Article

Complex Eigenvalue Analysis of Multibody Problems via Sparsity-Preserving Krylov-Schur Iterations

Dario Mangoni [1,†,‡,*], Alessandro Tasora [2,†,‡] and Peng Chao [3,†,‡]

[1] dario.mangoni@unipr.it
[2] alessandro.tasora@unipr.it
[3] chao.peng@unipr.it
[*] Correspondence: dario.mangoni@unipr.it
[†] University of Parma, Dpt. of Engineering and Architecture, Parco Area delle Scienze 181/A, 43124 Parma (PR) - Italy.
[‡] These authors contributed equally to this work.

Abstract: In this work we discuss the numerical challenges involved in the computation of the complex eigenvalues of damped multi-flexible-body problems. Aiming at the highest generality, the candidate method must be able to deal with arbitrary rigid body modes (free-free mechanisms), arbitrary algebraic constraints, and must be able to exploit the sparsity pattern of Jacobians of large systems. We propose a custom implementation of the Krylov-Schur method, proving its robustness and its accuracy in a variety of different complex test cases.

Keywords: modal analysis; eigenvalues; multibody; damped modes; sparse eigenproblem

1. Introduction

Equations of motion of multibody systems are highly non-linear in general, but there are cases where one is interested in a linearization of such equations as a way to study the effects of perturbations around a given configuration. To this end, being able to compute the eigenvalues and eigenvectors of the linearized model is of fundamental importance, and it is not limited to the conventional methods of modal analysis.

For instance, among other applications, eigenvectors can be used to perform a component mode synthesis, also known as *modal reduction*, that is an effective approach which turns a complex system into a surrogate model with a smaller set of coordinates, hence obtaining faster simulations [1] [2] [3] . Another application that require the computation of eigenpairs is the stability analysis of dynamic systems, for instance the aeroelastic stability of helicopter blades, wind turbines and other slender structures. In this case, one needs to implement a complex-valued eigenvalue problem, where the imaginary and real parts of the eigenvalues give an indication of the damping factor and, consequently, an indication about the impending instability [4] [5] . Finally we can mention that, in the field of control theory, often a state space representation of the linearized system is required, and this is another problem that motivates the research of efficient methods to recover the eigenvalues of the multibody system [6] [7] .

Motivated by the above mentioned applications, in this paper we discuss the numerical difficulties related to the computation of eigenvalues and eigenvectors in multi-flexible-body systems under the most general assumptions: we assume that the system can present singular modes (also called rigid body or free-free modes), we consider the optional presence of damping, hence leading to complex-valued eigenpairs, we consider an arbitrary number of parts and constraints, and we assume that the size of the system could be arbitrarily large. In particular this last requirement imposes some limitations on the type of solver, that must preserve the sparsity of the matrices for the sake of acceptable computational performance, and that should be able to output just a small subset of eigenvalues, either the lowest ones or those clustered around a frequency of interest.

The problem of eigenvalue computation in multibody systems is discussed by various authors in literature, although the topic is more common in the field of Finite Element Analysis (FEA). A difficulty of multibody systems with respect to conventional FEA is that constraints are ubiquitous and often described by algebraic equations and Lagrange multipliers. A classical approach is to remove constraints by means of an orthogonal complement that reduces the generalized coordinates to the lowest amount possible, as discussed for instance in [8] [9] . This idea has the benefit that the linearized equations are those of an unconstrained system, thus a conventional eigenvalue solver can be applied. However, there are also drawbacks that will be discussed in the next paragraphs.

Alternatively, one can solve an eigenvalue problem paired with constraints, thus leading to matrices that are larger but sparser. This approach is shown, for example, in [10] [11] . Despite the increment in the number of unknown eigenvalues and the increment in the dimension of eigenvectors, we experienced that this approach leads to a simpler formulation. Most important, we noticed that this method preserves the sparsity of the matrices, so that we could design a solver that can leverage this useful property.

An eigenvalue solver that achieved a big popularity in the past years is the Implicit Restarted Arnoldi Method (IRAM) [12]. In fact, this is the method implemented in ARPACK, a widespread Fortran77 library that can solve generalized eigenvalue problems, with both sparse or structured matrices [13]. As such, IRAM would be sufficient to satisfy our requirements, however we experienced that it fails to provide good convergence in some difficult cases, so we pointed our attention to the more recent Krylov-Schur method.

The Krylov-Schur method has been presented in [14] as an improvement over previous Krylov subspace methods such as IRAM and Lanczos. Because of an efficient and robust restarting strategy, it is often able to converge even in cases where IRAM stalls, and in general it exhibits superior robustness and faster convergence [15]. For these reasons the Krylov-Schur method has become the default for MATLAB `eigs` command, and it is also available in the SLEPc library [16] , an extension of the PETSc linear algebra library, as well as in the TRILINOS library [17] . However both are large libraries that target supercomputing and require complex build toolchains. On the other hand there are efforts like the SPECTRA C++ library [18], that are lightweight but might not offer all the desired functionalities; for instance SPECTRA contains the Krylov-Schur method in a partially implemented form, making it usable only for symmetric matrices (hence the complex eigenvalue problem is out of reach, making it unuseful for damped eigenmode computation at the moment of writing). The lack of reliable, complete and lightweight open source libraries for computing eigenvalues with the Krylov-Schur method motivated us to develop our C++ version of it, which is described in the following pages.

In the next session we will discuss how to obtain the needed matrices from a linearization of the multibody system, then we will review different formulations for expressing the eigenvalue problem, with or without constraints, with or without damping, then we will discuss some computational aspects related to the implementation of the sparsity-preserving Krylov-Schur solver, and finally we will show some applications and benchmarks.

2. Linearization of Multibody Structures

We introduce the semi-explicit Differential Algebraic Equations (DAE) of a generic, nonlinear multi-flexible body system with generalized coordinates $\boldsymbol{q} \in \mathbb{R}^n$:

$$\begin{cases} M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C_q(\boldsymbol{q},t)^T\gamma = \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t) - \boldsymbol{f}_g(\dot{\boldsymbol{q}}) & (1) \\ \boldsymbol{C}(\boldsymbol{q},t) = \boldsymbol{0} & (2) \end{cases}$$

where $\boldsymbol{C}(\boldsymbol{q},t) = \boldsymbol{0}$ is a vector of m holonomic-rheonomic constraints with a $m \times n$ sparse jacobian $C_q(\boldsymbol{q},t) = \frac{\partial \boldsymbol{C}(\boldsymbol{q},t)}{\partial \boldsymbol{q}}$. Also, $\boldsymbol{f}$ is the vector of external and internal forces, and $\boldsymbol{f}_g$ represent the gyroscopic and centrifugal components of the inertial forces (the full inertial forces are in fact $\boldsymbol{f}_i = M\ddot{\boldsymbol{q}} + \boldsymbol{f}_g$).

This DAE can be linearized about the dynamic equilibrium point, obtaining:

$$\begin{cases} M(\boldsymbol{q})\delta\ddot{\boldsymbol{q}} + R(\boldsymbol{q},\dot{\boldsymbol{q}})\delta\dot{\boldsymbol{q}} + K(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}},\gamma)\delta\boldsymbol{q} + C_q(\boldsymbol{q},t)^T\delta\gamma = \boldsymbol{0} & (3) \\ C_q(\boldsymbol{q},t)\delta\boldsymbol{q} = \boldsymbol{0} & (4) \end{cases}$$

In the formula above, the damping matrix R comes from the linearization of internal/external forces $\boldsymbol{f}$ about $\dot{\boldsymbol{q}}$, plus the linearization of $\boldsymbol{f}_g$, the quadratic part of the inertial forces, hence:

$$\begin{aligned} R(\boldsymbol{q},\dot{\boldsymbol{q}}) &= -\frac{\partial \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t)}{\partial \dot{\boldsymbol{q}}} + \frac{\partial \boldsymbol{f}_g(\dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}} && (5) \\ &= R_f + R_i && (6) \end{aligned}$$

Note that the R_i part is responsible of the so called *gyroscopic damping* , and it is null for $\dot{\boldsymbol{q}} = \boldsymbol{0}$.

The tangent stiffness K contains both the effect of the linearization of internal and external forces, that is the conventional stiffness matrix K_f, plus K_i, the linearization of the inertial forces[1] about $\boldsymbol{q}$, plus K_c, the linearization of the constraint reaction forces $C_q(\boldsymbol{q},t)^T\gamma$: the latter can introduce a contribution to the tangent stiffness because of the geometric effect of changes in $C_q(\boldsymbol{q},t)$ about the linearization point [2][3], that is:

$$\begin{aligned} K(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}},\gamma) &= \frac{\partial(M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{f}_g)}{\partial \boldsymbol{q}} + \frac{\partial(C_q(\boldsymbol{q},t)^T\gamma)}{\partial \boldsymbol{q}} - \frac{\partial \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t)}{\partial \boldsymbol{q}} && (7) \\ &= K_i + K_c + K_f && (8) \end{aligned}$$

Often times, especially in the FEA literature, the K_f matrix is split in two components $K_f = K_{f_m} + K_{f_g}$ where K_{f_m} is the material stiffness and K_{f_g} is the geometric stiffness - the latter is caused, for example, by change of orientation of internal forces in beams, and its effect is null in configurations that have no initial stress at the linearization point. A further splitting can be done by distinguishing internal forces, caused by finite elements, and external forces, caused by applied loads, thus $\boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t) = \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t)_{int} + \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t)_{ext}$, and $K_f = K_{f_m int} + K_{f_g int} + K_{f_m ext} + K_{f_g ext}$. In many cases $K_{f ext}$ matrices are of small value if compared to $K_{f int}$ matrices and can be neglected, but in other cases, for example when considering aerodynamic loads, they might be relevant.

We remark that (3) and (4) require the introduction of constraints via jacobian matrices $C_q(\boldsymbol{q},t)$ and Lagrange multipliers $\delta\gamma$: in fact in the following we will handle this complication by solving *constrained* eigenvalue problems. However one might wonder if there is an alternative approach that avoids $C_q(\boldsymbol{q},t)$ and $\delta\gamma$ at all, so that a conventional (not constrained) eigenvalue solver could be used. Actually this would be possible, for example by running a QR decomposition on the C_q matrix in order to find a $\Xi \in \mathbb{R}^{n\times m}$ matrix such that $\Xi^T C_q(\boldsymbol{q},t)^T = \boldsymbol{0}$. In this way, one could introduce a smaller set of independent variables $\boldsymbol{y} \in \mathbb{R}^{n-m}$ for whom $\dot{\boldsymbol{q}} = \Xi\dot{\boldsymbol{y}}$, hence rewriting the DAE (1) as a simple ODE

$$\Xi^T M(\boldsymbol{q})\Xi\ddot{\boldsymbol{y}} + \Xi^T M(\boldsymbol{q})\dot{\Xi}\dot{\boldsymbol{y}} = \Xi^T \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t) - \Xi^T \boldsymbol{f}_g(\dot{\boldsymbol{q}}) \tag{9}$$

[1] This term is null if the system is studied in a static configuration, as often happens, but might be relevant otherwise, for example when studying eigenmodes of a wind turbine while it is rotating.

[2] For instance, this term accounts for the gravity-induced stiffness of a pendulum, where increments in the rotation of the pendulum generate changes in C_q to account for the rotation in the direction of the reaction force at the pendulum hinge. If the other sources of stiffness are more relevant (ex. springs, elastic internal forces in beams, etc.), or if λ is small at the linearization point, then this term might be neglected.

[3] This implies that a static or dynamic analysis should be performed right before computing eigenvectors, as the value of γ must be known when computing (7).

This can be linearized to give a single expression which is alternative to (3)(4):

$$M_Y(\boldsymbol{q})\delta\ddot{\boldsymbol{y}} + R_Y(\boldsymbol{q},\dot{\boldsymbol{q}})\delta\dot{\boldsymbol{y}} + K_Y(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})\delta\boldsymbol{y} = \mathbf{0} \tag{10}$$

with

$$M_Y(\boldsymbol{q}) = \Xi^T M(\boldsymbol{q})\Xi \tag{11}$$

$$R_Y(\boldsymbol{q}) = -\Xi^T \frac{\partial \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t)}{\partial \dot{\boldsymbol{y}}} + \Xi^T \frac{\partial \boldsymbol{f}_g(\dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{y}}} + \Xi^T M(\boldsymbol{q})\frac{\partial \dot{\Xi}\dot{\boldsymbol{y}}}{\partial \dot{\boldsymbol{y}}} \tag{12}$$

$$\begin{aligned} K_Y(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = & \left(\frac{\partial \Xi^T}{\partial \boldsymbol{y}} M(\boldsymbol{q})\Xi + \Xi^T \frac{\partial M}{\partial \boldsymbol{y}}(\boldsymbol{q})\Xi + \Xi^T M(\boldsymbol{q})\frac{\partial \Xi}{\partial \boldsymbol{y}}\right)\ddot{\boldsymbol{y}} \\ & + \left(\frac{\partial \Xi^T}{\partial \boldsymbol{y}} M(\boldsymbol{q})\dot{\Xi} + \Xi^T \frac{M(\boldsymbol{q})}{\partial \boldsymbol{y}}\dot{\Xi}\right)\dot{\boldsymbol{y}} \\ & - \frac{\partial \Xi^T}{\partial \boldsymbol{y}} \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t) - \Xi^T \frac{\partial \boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t)}{\partial \boldsymbol{y}} + \frac{\partial \Xi^T}{\partial \boldsymbol{y}} \boldsymbol{f}_g(\dot{\boldsymbol{q}}) + \Xi^T \frac{\partial \boldsymbol{f}_g(\dot{\boldsymbol{q}})}{\partial \boldsymbol{y}} \end{aligned} \tag{13}$$

However we note that the expression of M_Y, R_Y and K_Y is substantially more intricate[4] than the expression of M, R and K in (5),(7), for instance (13) require the knowledge of $\dot{\Xi}$ and $\partial \Xi^T / \partial \boldsymbol{y}$. Moreover, the multiplications by Ξ and Ξ^T will destroy the sparsity of the original matrices M, R, K: this is not an issue in problems of small size, but for large problems this would lead to unacceptable memory and performance requirements. For these reasons, we prefer to proceed with the linearization expressed in (3)(4), at the cost of dealing with contraints during the iterative eigenvalue solution process. The following section will explain how to use the M, R, K, C_q matrices to this end.

3. Modal Analysis

We can distinguish two types of modal analysis: in the first case we search for real-valued eigenvalues of the undamped system, in the second case we search for complex-valued eigenvalues of the damped system. The former can be considered a sub case of the latter for $R = 0$, hence a single solver could attack both problems, however it is better to adopt two different solution schemes in order to exploit some optimizations that lead to high computational performance if the damping is of no interest.

3.1. Undamped Case - Real-Valued

We recall some basic concepts in eigenvalue analysis of dynamic systems.

For the simple case of an unconstrained, undamped system with $R = 0$, with solutions $\boldsymbol{q} = \Sigma(\boldsymbol{\Phi}_i e^{i\omega t} + \boldsymbol{\Phi}_i e^{-i\omega t})$

$$M\ddot{\boldsymbol{q}} + K\boldsymbol{q} = \mathbf{0} \tag{14}$$

it is possible to compute the eigenmodes from the following characteristic expression:

$$\left(-\omega_i^2 M + K\right)\boldsymbol{\Phi}_i = \mathbf{0} \tag{15}$$

[4] Under simplifying assumptions such as negligible effects terms $\dot{\Xi}$ and $\partial \Xi^T / \partial \boldsymbol{y}$, the reduced matrices can be approximated as $M_Y(\boldsymbol{q}) \approx \Xi^T M \Xi$, $R_Y(\boldsymbol{q}) \approx \Xi^T R \Xi$, $K_Y(\boldsymbol{q}) \approx \Xi^T K \Xi$. However we experienced that such a simplification is possible only when assuming that the constraints do not change direction in a significant way: in fact an oscillating pendulum would give zero natural frequency in this case.

that leads to a standard eigenvalue problem (SEP) with eigenvalues $\lambda_i = \omega_i^2$ and matrix $C = M^{-1}K$:

$$\left(M^{-1}K - \lambda_i I\right)\mathbf{\Phi}_i = \mathbf{0} \tag{16}$$

$$(C - \lambda_i I)\mathbf{\Phi}_i = \mathbf{0} \tag{17}$$

For symmetric K and M, by the spectral theorem, eigenpairs are real.

However there are some difficulties that prevent the direct use of (16) in engineering problems of practical interest:

- it works only if there are no constraints (no C_q jacobian matrix);
- it requires the inversion of the M matrix: even if M is often diagonal-dominant and easy to invert, this is not true in general and it could destroy the sparsity of the matrices in case of large systems;
- we may be interested in just a small subset of eigenvalues, usually the lower modes, so we need an iterative scheme that is able to do this.

We compute the modes of the *constrained undamped* multibody system with the following *Generalized Eigenvalue Problem* (GEP):

$$-\begin{bmatrix} K & C_q^T \\ C_q & 0 \end{bmatrix}\hat{\mathbf{\Phi}}_i = \lambda_i \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}\hat{\mathbf{\Phi}}_i \tag{18}$$

where we introduced the augmented eigenvector

$$\hat{\mathbf{\Phi}}_i = \{\mathbf{\Phi}_i^T, \boldsymbol{\xi}_i^T\}^T.$$

and where we recover natural frequencies as:

$$\omega_i = \sqrt{-\lambda_i}, \quad f_i = \omega_i / 2\pi \tag{19}$$

We remark that one could change the sign in the left hand side of (18), this would get positive eigenvalues and then one would compute $\omega_i = \sqrt{\lambda_i}$ instead.

The solution of the problem (18) generates $n + m$ eigenvalues, where only n are of interest, and m are spurious modes with $\lambda \approx \pm\infty$ that can be discarded. The same filtering must be done for the corresponding eigenvectors. Moreover, the last m components of the eigenvectors, namely $\boldsymbol{\xi}_i$, can be just discarded or used to get an insight about reaction forces, because they represent the amplitude of reactions in constraints during the periodical motion of the system.

Alternatively, one can solve

$$-\begin{bmatrix} K & 0 \\ 0 & 0 \end{bmatrix}\hat{\mathbf{\Phi}}_i = \lambda_i \begin{bmatrix} M & C_q^T \\ C_q & 0 \end{bmatrix}\hat{\mathbf{\Phi}}_i \tag{20}$$

but this would produce m spurious modes with $\lambda \approx 0$. This is acceptable unless one is interested in structures featuring rigid body modes [5], because the rigid body modes have $\lambda_i \approx 0$ too, and this would prevent an easy detection of such modes because they will mix with the spurious modes. In this formulation (20), the last m components of the eigenvectors, namely $\boldsymbol{\xi}_i$, represent the second integration of reaction forces/moments of the constraints, which can be discarded since no physical meaning exists.

The matrices that appear in the two forms of the GEP have different properties, and this is relevant when we will choose the optimal solution scheme. In the GEP (18), the A matrix is non-singular only if there are no rigid body modes, as it is z-times rank deficient

[5] Rigid body modes are sometimes called free-free modes in literature. In 3D space, a structure that is not constrained to the ground, like a flying boomerang, will produce 6 rigid body modes with $\omega_i = 0$. In 2D space, one expects 3 rigid body modes.

in presence of z rigid body modes, moreover the B matrix is always singular and not invertible. Hence both matrices are not invertible in the most general case. On the other hand, in the GEP (20), the A matrix is singular, but B matrix is always non-singular and invertible, regardless of the presence of rigid body modes, because M is positive definite and C_q is assumed to be full rank. This would make GEP (20) a better choice respect to GEP (18) because one could always transform it to a SEP via $C = B^{-1}A$. However, as we will see later, solving the SEP in this form is not what we need in case of large systems, where we want a limited number of eigenpairs starting from the smallest ones: if so, a shift and invert approach is needed, where the non-singularity of B is irrelevant, and we would rather need the inversion of A: in this case, neither GEP (18) nor GEP (20) would fit this requirement. However, the shift and invert approach requires a regularized form of the inverse matrix, by means of a σ shift parameter as in $C = (A - \sigma B)^{-1}B$, so both approaches could work in this setting, except for $\sigma = 0$.

Finally, we note that, when the K matrix is symmetric, both A and B are symmetric, therefore optimized linear solvers for the inner loop of the Krylov-Schur solver could be used in sake of higher speed (that is, the $(A - \sigma B)^{-1}$ problem can be approached via LDLt decompositions rather than LU decompositions in case of direct solvers, or via MINRES rather than GMRES in case of Krylov solvers).

3.2. *Damped Case - Complex-Valued*

The conventional modal analysis of the damped system

$$M\ddot{\boldsymbol{q}} + R\dot{\boldsymbol{q}} + K\boldsymbol{q} = \mathbf{0} \tag{21}$$

with solutions $\boldsymbol{q} = \boldsymbol{\Phi}e^{\lambda t}$ is formulated as a quadratic eigenvalue problem (QEP), either with left or right eigenvectors:

$$(\lambda^2 M + \lambda R + K)\boldsymbol{\Phi} = \mathbf{0} \tag{22}$$

$$\boldsymbol{\Psi}^*(\lambda^2 M + \lambda R + K) = \mathbf{0} \tag{23}$$

We recall some useful properties. Since coefficients of (22) are real, any complex roots must appear as complex conjugate pairs. The QEP generates $2n$ eigenvalues, that are finite if M is non singular; if M, R, K are real, or Hermitian, then eigenvalues can be a mix of real values or complex conjugate pairs $(\lambda, \overline{\lambda})$; if M is Hermitian positive definite and R, K are Hermitian positive semidefinite, then $\mathrm{Re}(\lambda) \leq 0$.

- complex conjugate pairs $(\lambda, \overline{\lambda})$ correspond to underdamped modes, oscillatory and decaying for $\mathrm{Re}(\lambda) < 0$;
- purely imaginary conjugate pairs $(\lambda, \overline{\lambda}), \mathrm{Re}(\lambda) = 0$ correspond to undamped modes, purely harmonic and not decaying;
- real modes with $\mathrm{Re}(\lambda) \leq 0$ and no imaginary part correspond to overdamped modes, not oscillatory, exponential decaying;
- in all cases, $\mathrm{Re}(\lambda) > 0$ indicates an unstable system;
- for the class of damped systems, also eigenvectors $\boldsymbol{\Phi}_i$ are complex, with elements:

$$\boldsymbol{\Phi}_{i,j} = a_{i,j} + i\, b_{i,j} = \delta_{i,j}e^{i\beta_{i,j}}$$

  where both the amplitude and the phase of the entire eigenvector can be arbitrary (but the relative amplitude $\delta_{i,j}/\delta_{i,k}$ of each component is unaltered by whatever normalization, and the relative phase of each component is constant $\beta_{i,j} - \beta_{i,k} = \mathrm{const}_{jk}$);
- the two eigenvectors of a complex conjugate pair are also conjugate.

Oscillatory modes, corresponding to a complex conjugate pair $(\lambda, \overline{\lambda})$, $\mathrm{Re}(\lambda) < 0$, can be written in a more engineering-oriented way as done in 1-dof systems, $Ae^{(-\zeta\omega + i\omega_d)t} +$

$Be^{(-\zeta\omega - i\omega_d)t}$, where one has the following expressions for natural (undamped) frequencies ω_i, damped frequencies $\omega_{d,i}$ and damping factors ζ_i:

$$\omega_i = \|\lambda_i\|, \quad f_i = \omega_i/2\pi \tag{24}$$
$$\omega_{d,i} = \mathrm{Im}(\lambda_i), \quad f_{d,i} = \omega_{d,i}/2\pi \tag{25}$$
$$\zeta_i = -\mathrm{Re}(\lambda_i)/\omega_i \tag{26}$$
$$\omega_{d,i} = \omega_i\sqrt{1-\zeta^2} \tag{27}$$

Although there exist algorithms that can solve (22) directly, often the QEP is transformed to a SEP or GEP so that a conventional solver like Arnoldi or Krylov-Schur can be used. This can be done expressing the problem in state space: we introduce an augmented eigenvector that contains both the eigenvector $\mathbf{\Phi}_i \in \mathbb{R}^n$, and the eigenvector $\lambda_i\mathbf{\Phi}_i \in \mathbb{R}^n$:

$$\underline{\mathbf{\Phi}}_i^T = \{\mathbf{\Phi}_i^T, \lambda_i\mathbf{\Phi}_i^T\}$$

This can be used to transform the QEP (22) into the following GEP with double the original size:

$$\begin{bmatrix} 0 & I \\ -K & -R \end{bmatrix}\underline{\mathbf{\Phi}}_i = \lambda_i \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}\underline{\mathbf{\Phi}}_i \tag{28}$$

Additionally, one can consider the constraints by introducing Lagrange multipliers $\boldsymbol{\xi}_i \in \mathbb{R}^m$ that correspond to the m constraints enforced as $C_q\mathbf{\Phi}_i = 0$, thus obtaining a constrained QEP:

$$\begin{cases} \lambda_i^2 M\mathbf{\Phi}_i + \lambda_i R\mathbf{\Phi}_i + K\mathbf{\Phi}_i + C_q^T\boldsymbol{\xi}_i = \mathbf{0} & (29) \\ -C_q\mathbf{\Phi}_i = \mathbf{0} & (30) \end{cases}$$

Finally, introducing the augmented eigenvector $\hat{\underline{\mathbf{\Phi}}}_i \in \mathbb{R}^{2n+m}$ as

$$\underline{\mathbf{\Phi}}_i^T = \{\mathbf{\Phi}_i^T, \lambda_i\mathbf{\Phi}_i^T, \boldsymbol{\xi}_i^T\}$$

and by making use of simple linear algebra, we can write the constrained QEP as a constrained GEP:

$$\begin{bmatrix} 0 & I & 0 \\ -K & -R & -C_q^T \\ -C_q & 0 & 0 \end{bmatrix}\hat{\underline{\mathbf{\Phi}}}_i = \lambda_i \begin{bmatrix} I & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & 0 \end{bmatrix}\hat{\underline{\mathbf{\Phi}}}_i \tag{31}$$

An alternative formulation is based on the solution of the following GEP, where the spurious modes related to the constraint equations are zero instead of infinite:

$$\begin{bmatrix} 0 & I & 0 \\ -K & -R & 0 \\ 0 & 0 & 0 \end{bmatrix}\hat{\underline{\mathbf{\Phi}}}_i = \lambda_i \begin{bmatrix} I & 0 & 0 \\ 0 & M & C_q^T \\ C_q & 0 & 0 \end{bmatrix}\hat{\underline{\mathbf{\Phi}}}_i \tag{32}$$

that corresponds to

$$\begin{cases} \lambda_i^2 M\mathbf{\Phi}_i + \lambda_i R\mathbf{\Phi}_i + K\mathbf{\Phi}_i + C_q^T\lambda_i\boldsymbol{\xi}_i = \mathbf{0} & (33) \\ C_q\lambda_i\mathbf{\Phi}_i = \mathbf{0} & (34) \end{cases}$$

We experienced that the most efficient way to compute eigenpairs of the constrained damped system is the first approach, i.e. the GEP (31).

4. Computing Eigenpairs with Sparse Matrices

When the number of unknowns n grows, it is not possible to compute all the n eigenvalues and eigenvectors, both for reasons of computational time and for the extreme requirement of memory needed for storing all the eigenvectors. In fact, many analyses that require the computation of eigenmodes in practice require a small set of them.

There are iterative methods that preserve the sparsity of matrices and that can compute a limited set of k eigenvectors: most notably these are the IRAM (Implicitly Restarted Arnoldi method), Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) and lastly the Krylov-Schur method.

The problem is that they compute the largest k, not the smallest ones, that is exactly the opposite of our interest. This issue can be solved adopting a Moebius transform of the eigenvalue problem. We proceed as follows.

For the undamped constrained case, first we formulate the generalized eigenvalue problem (GEP):

$$A\hat{\boldsymbol{\Phi}}_i = \lambda_i B\hat{\boldsymbol{\Phi}}_i \tag{35}$$

$$A = \begin{bmatrix} -K & -C_q^T \\ -C_q & 0 \end{bmatrix} \tag{36}$$

$$B = \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \tag{37}$$

then we adopt a Moebius transform of the eigenvalue problem, namely the *shift and invert* strategy that computes eigenvalues μ_i in the following problem:

$$(C - \mu_i I)\hat{\boldsymbol{\Phi}}_i = 0 \tag{38}$$

$$C = (A - \sigma B)^{-1} B \tag{39}$$

$$\mu = \frac{1}{\lambda - \sigma} \qquad \lambda = \frac{1}{\mu} + \sigma \tag{40}$$

After the eigenvalue problem (38) is solved for k pairs of $(\mu_i, \hat{\boldsymbol{\Phi}}_i)$, one recovers the original λ_i hence the original ω_i using (40).

For the damped constrained case, we formulate a GEP of the type

$$A\underline{\hat{\boldsymbol{\Phi}}}_i = \lambda_i B\underline{\hat{\boldsymbol{\Phi}}}_i \tag{41}$$

$$A = \begin{bmatrix} 0 & I & 0 \\ -K & -R & -C_q^T \\ -C_q & 0 & 0 \end{bmatrix} \tag{42}$$

$$B = \begin{bmatrix} I & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{43}$$

then, similarly to the undamped case, we apply the shift-invert Moebius transformation to solve $(C - \mu_i I)\underline{\hat{\boldsymbol{\Phi}}}_i = 0$ with $C = (A - \sigma B)^{-1} B$, obtaining pairs $(\mu_i, \underline{\hat{\boldsymbol{\Phi}}}_i)$, and finally recovering $\lambda = \frac{1}{\mu} + \sigma$.

Right eigenvectors $\boldsymbol{\Phi}_i$ are not affected by the Moebius transform. Just in case one is interested in the left eigenvectors as in $\boldsymbol{\Psi}_i^*(C - \lambda_i I) = 0$, then those are recovered solving $z_i^*(C - \mu_i I) = 0$ and using the transform $z = (A - \sigma B)^* \boldsymbol{\Psi}_i$.

User-defined values of σ can be used to extract eigenvalues in specific frequency ranges. In fact the iterative solver will return the k eigenvalues that are closer, in absolute value, to σ.

If the shift parameter σ is zero or close to it, as often happens, one can see that the largest k eigenvalues μ computed by the Krylov-Schur solver will become the smallest k eigenvalues λ, for the modes closer to zero frequency.

As a special case, for $\sigma = 0$, one has $C = A^{-1}B$ and $\lambda = \frac{1}{\mu}$, that for a unconstrained problem (no C_q jacobian) it corresponds to solving the inverse eigenvalue problem $\left(K^{-1}M - \mu_i I\right)\boldsymbol{\Phi}_i = \mathbf{0}$.

Table 1. Different options for the eigenpair computation.

	GEP	Notes
Undamped	$A = [K] \quad B = [M]$	real eigenpairs, $\omega_i = \sqrt{\lambda_i}$ A singular if rigid body modes
	$A = [-K] \quad B = [M]$	real eigenpairs, $\omega_i = \sqrt{-\lambda_i}$ A singular if rigid body modes
Undamped Constrained	$A = \begin{bmatrix} -K & -C_q^T \\ -C_q & 0 \end{bmatrix} \quad B = \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}$	real eigenpairs, $\omega_i = \sqrt{-\lambda_i}$ $\|\lambda_i\| = \infty$ for each constraint A singular if rigid body modes B singular
	$A = \begin{bmatrix} -K & 0 \\ 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} M & C_q^T \\ C_q & 0 \end{bmatrix}$	real eigenpairs, $\omega_i = \sqrt{-\lambda_i}$ $\lambda_i = 0$ for each constraint A singular B nonsingular
Damped	$A = \begin{bmatrix} 0 & I \\ -K & -R \end{bmatrix} \quad B = \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}$	complex eigenpairs, $\omega_i = \|\lambda_i\|$ A singular if rigid body modes B singular
Damped Constrained	$A = \begin{bmatrix} 0 & I & 0 \\ -K & -R & -C_q^T \\ -C_q & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} I & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & 0 \end{bmatrix}$	complex eigenpairs, $\omega_i = \|\lambda_i\|$ $\|\lambda_i\| = \infty$ for each constraint A singular if rigid body modes B singular
	$A = \begin{bmatrix} 0 & I & 0 \\ -K & -R & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} I & 0 & 0 \\ 0 & M & C_q^T \\ C_q & 0 & 0 \end{bmatrix}$	complex eigenpairs, $\omega_i = \|\lambda_i\|$ $\lambda_i = 0$ for each constraint A singular B nonsingular

In general one can adjust the σ shift value so that it provides the best numerical performance, in detail it provides a regularization of A and helps solving the linear problem in (39) also in the case where A or B are singular or close to singularity. This is what happens in many cases when doing modal analysis of engineering structures, especially if the structure has rigid body modes. In fact, our default method is to extract all lower modes including rigid body modes, and at this end we experienced that a value of $\sigma = 1 \times 10^{-3}$ works well retrieving also the six $\lambda \approx 0$ modes and cures the ill-posedness problems.

Krylov-Schur and Arnoldi methods draw on a single computational primitive, that is the product of a sparse matrix C by a vector $\boldsymbol{v}$ for the solution of the problem $(C - \mu_i I)\Phi_i = 0$. However in our case $C = (A - \sigma B)^{-1}B$, hence pre-computing such C matrix is out of question because the exact inversion of $(A - \sigma B)$ would require too much CPU time and would destroy the sparsity. Since only the product primitive $C\boldsymbol{v}$ is required for the iterative solver, an acceptable tradeoff is to return the result of the product $\boldsymbol{r} = C\boldsymbol{v}$ by doing these steps:

$$\boldsymbol{z} = B\boldsymbol{v} \tag{44a}$$
$$\boldsymbol{r} = (A - \sigma B)^{-1}\boldsymbol{z} \tag{44b}$$

Here we note that Eq.(44b) in the second step requires a linear system solution. This can be a computational bottleneck, but a substantial speedup can be achieved observing that the coefficient matrix $(A - \sigma B)$ is constant, therefore one can factorize it once at the beginning of the Krylov-Schur iterations, and perform only the back substitutions in (44b).

An alternative, that preserve sparsity of the matrices and can fit better in scenarios with millions of unknowns, is that (44b) is solved iteratively via truncated MINRES or GMRES iterative methods. If the number of unknowns is in the range of tens of thousands, however, we experienced that the factorization via a direct method performs faster.

5. Implementation of the Krylov-Schur Solver

The Krylov-Schur method has been introduced in 2001 [14], leading to improved performance respect to other Krylov subspace methods, such as Arnoldi and Lanczos, which have been used for decades in the field of eigenvalue computation. The notorious Implicitly-Restarted Arnoldi Method, implemented for example in the ARPACK library, or the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG), implemented for example in the BLOPEX library, both fail to converge for those problems whose matrix is of type (18), (20), (31) or (32) and wide mass ratios or strongly ill-conditioned blocks are present.

On the contrary, the robustness of the Krylov-Schur method guarantees satisfying results also for the most critical conditions, thus becoming the elected choice for the following tests. Our implementation follows the guidelines in [19] as reported in the Algorithm 1. It has been extended to the case of complex and sparse matrices, and has been included in the open-source multibody library CHRONO [20].

On a parent level of the Krylov-Schur solver, specific routines constructs an eigenvalue problem in accordance with (31) or with (18) for the undamped case. This will push the spurious constraint modes to infinity, being of less disturbance for the usual low-frequency area of interest for engineering applications.

The code offers the possibility to specify different problem formulation, either direct or in shift-invert, by providing different OP_CV(v) operators in Algorithm 2. For instance, in Algorithm 3 we show the implementation for the shift-invert case, implementing (44a) and (44b).

Solutions of linear systems required by the method can be theoretically provided by any linear solver enabled for complex values; practically, given the relatively high accuracy required by the solution and the ill-conditioning of some matrices, direct solvers are almost mandatory for this application, relegating iterative solvers only for systems with higher degrees of freedom. While for smaller and simpler problems the choice of the solver is not critical (allowing the use of e.g. SparseLU and SparseQR functions from the popular C++ linear algebra library EIGEN [21]), for most of the real cases more advanced solvers are required, like Pardiso MKL or MUMPS [22]. Given the importance of this choice, our Krylov-Schur implementation has been made solver agnostic: the user can indeed provide one of its own choice.

For the undamped case, as (18), the value of σ in the shift-invert procedure is assigned as a small positive real value $\sigma = \epsilon$ (by default we used $\sigma = 10^{-3}$ in our tests) in order to return the lowest eigenmodes, including those with zero eigenvalues in case there are rigid body modes. A small negative real value would work as well, but the numerical conditioning of the problem would be worse. If one needs specific eigenmodes clustered about some specific frequency f_c, we set it as $\sigma = -f_c^2$. For the damped case, we use a complex shift $\sigma = \sigma_R + i\sigma_I$, with a small real value $\sigma_R = \epsilon$ and no imaginary part if we are interested in the lowest eigenvalues, for instance $\sigma = 10^{-3} + i0$, or with a finite imaginary part if we need eigenmodes clustered about a f_c frequency: $\sigma = \epsilon_R + if_c$.

The Krylov-Schur decomposition is then solved by using EIGEN linear algebra library eigensolvers [21].

An important contribution to the stability of the method is given by a trivial and inexpensive pre-conditioning of the Jacobian matrix C_q. While stiffness and damping matrices have usually terms on the order at least of 10^6, the Jacobian matrix is usually in the order of 10^0. This change affects only the Lagrange multipliers γ and the relative eigenvector counterpart $\boldsymbol{\xi}_i$ that should be re-scaled back of the same factor (if they are of any interest to the user). This simple change in the matrices allows, in some corner case, a significant reduction of the residuals even just after the first iteration of the method.

6. Results

The Krylov-Schur method has been tested on various scenarios, including real-case problems, in order to assess the accuracy and scalability of the method. Relevant test conditions include flexible elements, rigid bodies, generic constraints, free-free modes in various combination.

Test are using the newly-implemented quadratic Krylov-Schur eigen solver, Pardiso MKL direct linear solver and results are compared to the *eigs* solver of MATLAB (that turns out to be an implementation of Krylov-Schur solver as well). The hardware includes an Intel i7 6700HQ with 16 GB RAM.

Algorithm 1 Krylov-Schur

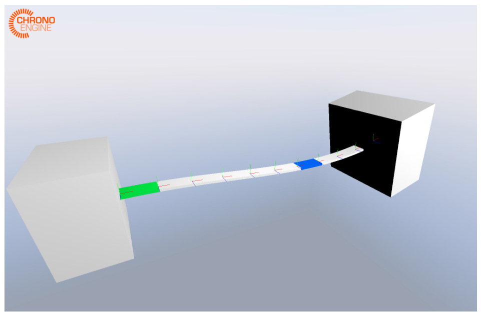
```
1: procedure KRYLOV-SCHUR(OP_CV(),$k$, $m$)
2:     $Q(:,1) := v1/norm(v1)$
3:     $p := 1$
4:     $isC := 0$
5:     $[Q, H] :=$ KRYLOVEXPANSION(OP_CV(),$Q, H, 0, k$)
6:     while $i < i_{\max}$ & $p < k$ do
7:         $i++$
8:         $isC := 0$
9:         $[Q, H] :=$ KRYLOVEXPANSION(OP_CV(),$Q, H, k + isC, m$)
10:        $[U, T, isC] :=$ SORTSCHUR($H(p : m, p : m), k - p + 1$)
11:        $H(p : m, p : m) := T$
12:        $H(1 : p - 1, p : m) := H(1 : p - 1, p : m)\, U$
13:        $Q(:, p : m) := Q(:, p : m)\, U$
14:        $H(m + 1, p : m) := H(m + 1, m)\, U(end, :)$
15:        $Q := [Q(:, 1 : k), Q(:, m + 1)]$
16:        $H := [H(1 : k, 1 : k); H(m + 1, 1 : k)]$
17:        CHECKCONVERGENCE($H, k + isC, p, tol$)
18:    end while
19:    $[\mu, \mathbf{\Phi}_H] :=$ EIG($H(1 : k + isC, 1 : k + isC)$)
20:    $\mathbf{\Phi} = Q(:, k + isC)\mathbf{\Phi}_H$
21:    return $\mu, \mathbf{\Phi}$
22: end procedure
```

Algorithm 2 Krylov Expansion

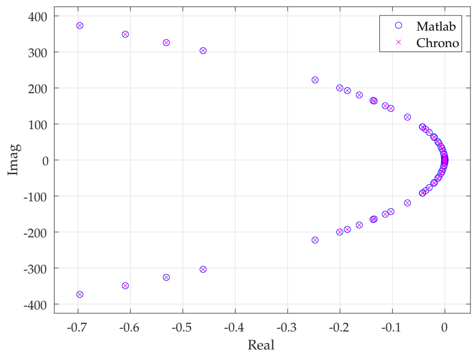
```
1: procedure KRYLOVEXPANSION(OP_CV(),$Q$, $H$, $k_s$, $k_e$)
2:     for $k = k_s + 1 : k_e$ do
3:         $v :=$OP_CV($Q(:, k)$)
4:         $isC := 0$
5:         $w := Q(:, 1 : k)'\, v$
6:         $v -= Q(:, 1 : k)\, w$
7:         $w2 := Q(:, 1 : k)'\, v$
8:         $v -= Q(:, 1 : k)\, w2$
9:         $w += w2$
10:        $nv := norm(v)$
11:        $Q(:, k + 1) := v/nv$
12:        $H(1 : k + 1, k) := [w; nv]$
13:    end for
14: end procedure
```

Algorithm 3 Op_Cv operator

```
1: procedure OP_CV($v$)
2:     $\mathbf{z} = Bv$
3:     $\mathbf{r} = (A - \sigma B)^{-1}\mathbf{z}$
4:     return $\mathbf{r}$
5: end procedure
```



(a) *Cantilever Models.*



(b) *Cantilever Eigenvalues.*

Figure 1. Cantilever Test.

Table 2. Test beams properties.

Property	Value
Young Modulus	100 MPa
Density	$1000\,\mathrm{kg\,m^{-3}}$
Section	$0.3\,\mathrm{m} \times 0.05\,\mathrm{m}$
Poisson Ratio	0.31
Rayleigh Damping	$\alpha = 1 \times 10^{-3}, \beta = 1 \times 10^{-5}$

6.1. *Hybrid Flexible and Rigid Bodies with Constraints* 276

Constraining the system results in additional zero-valued block in the system matrices, thus potentially compromising the stability for the inner linear solver. In the following test case an Euler beam with properties set according to Tab. 2 is fixed at the base, while its tip is constrained to a rigid body of heavier mass (4000 kg) (Fig. 1). The method has been tested with end masses up to 10×10^8 in order to prove its robustness. An additional test case with a crank-rod-piston assembly shows the use of tetrahedral mesh (Fig. 2).

6.2. *Free-free Modes* 283

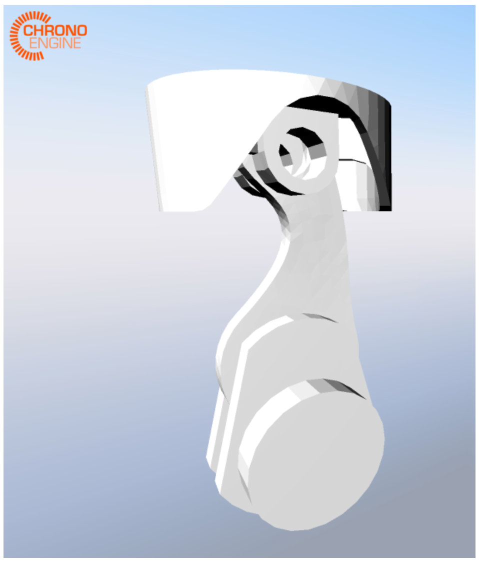
The presence of unconstrained bodies results in degenerated modes whose eigenvalues are pushed towards infinity. The method guarantees proper stability also for this degenerate case (Fig. 3). It might be noticed how each degree of freedom contributes to the overall residual: the first half represents the positional degrees of freedom, while the second represents the velocities (usually of less interest). Beam properties are the same as Tab.2.

6.3. *Wind turbine* 289

This medium-scale real test case involves a modern large-size wind turbine, courtesy of a commercial original equipment manufacturer in wind industry. The test includes constrained flexible as well as free rigid bodies. Given the wide ratio between smaller and bigger eigenvalues (the A matrix results in a reversed conditioning number of 10^{-19}), the preconditioning of the Jacobian matrix of the constraints has proved to be essential for the robustness and accuracy of the results. The problem is non-symmetric. Given the sensitivity of the results they have been hidden and only the residuals are shown in Fig. 4. Problem size is on the order of the thousands.

6.4. *Scalability* 298

The scalability of the method has been tested against a grid of Euler beams, whose size and number of cells is parametrized in order to provide different scales to the same problem. Each beam is fixed at every intersection with the grid. For each test, the lower 100 modes have been computed. The number of elements are three and two, respectively along


(a) *Second Mode at 52Hz .*

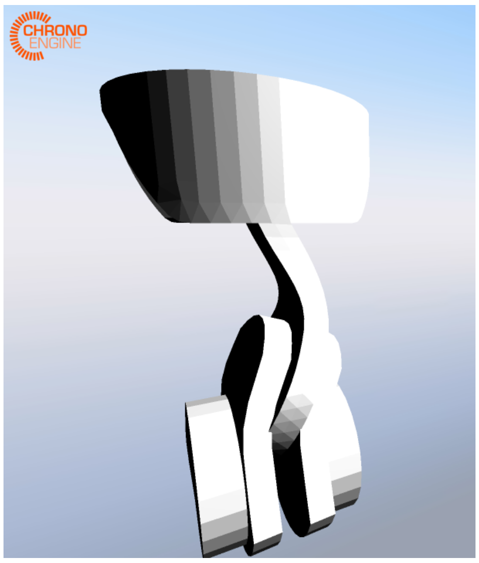
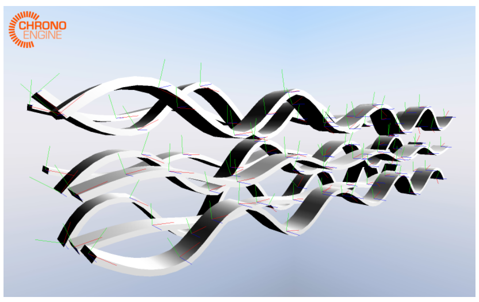
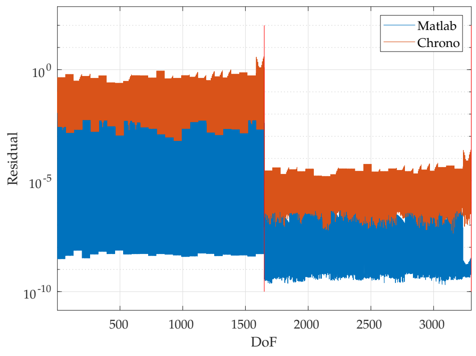

(b) *Sixth Mode at 200 Hz.*

Figure 2. Benchmark for multi-body system with constraints: flexible crank, rod and piston bodies with bearings.


(a) *Free-free Models.*



(b) *Free-free eigensolver residuals.*

Figure 3. Benchmark with multiple rigid body modes.

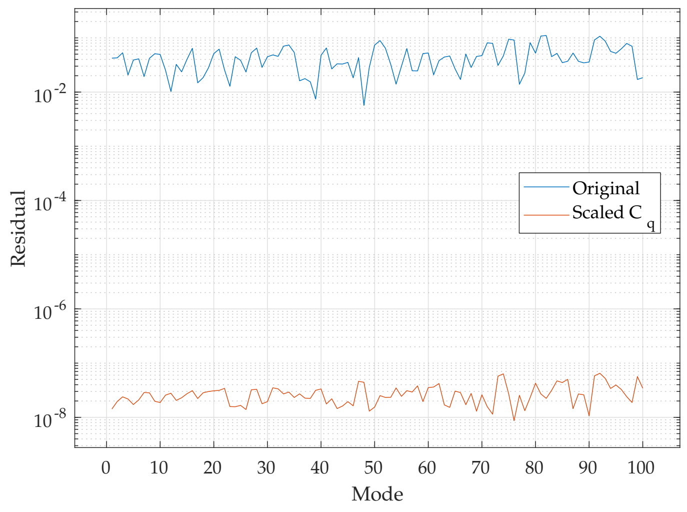
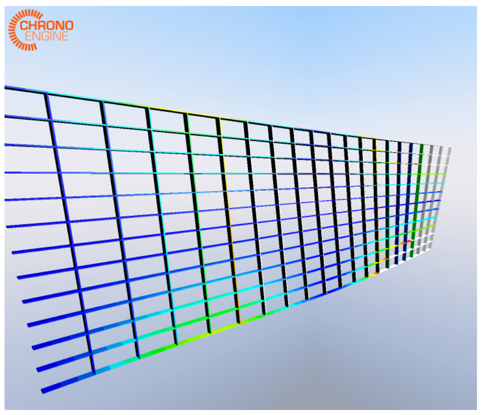
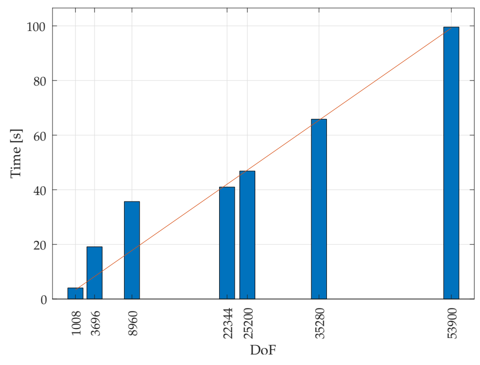


Figure 4. Wind Turbine Test Residuals.



(a) *Beam-Grid Model*.



(b) *Beam-Grid Time Cost*.

Figure 5. Scalability Test.

each cell longitudinal and vertical direction. The ratio between longitudinal and vertical number of cells is kept constant across the different tests. The results basically show a linear relation ($R^2 = 0.9996$) between the number of degrees of freedom of the original problem and the time cost of the Krylov-Schur solver Fig. 5, with a little additional overhead for smaller-scale tests. Results do not include the time expense for the assembly of the matrices. Again, the beam properties are set according to Tab.2.

7. Conclusions

The proposed implementation of the Krylov-Schur solver successfully proves to effectively handle a wide variety of test cases, including free-free modes, constraints, rigid and flexible systems, resulting in either real or complex, symmetric or asymmetric matrices of the associated eigenvalue problem. The ample availability of the software guarantees a vast dissemination of the method, offering the best platform for further improvements.

Author Contributions: Conceptualization, *all*; methodology, *all*; software, *all*; validation, *all*; supervision, A.T.; funding acquisition, A.T. and P.C. All authors have read and agreed to the published version of the manuscript.

Data Availability Statement: The C++ version of the Krylov-Schur solver has been made available open-source in the https://github.com/projectchrono/chrono repository, under BSD-3 license.

Acknowledgments: Part of this work has been supported by the Italian PON R&I 2014-2020 initiative (DM 1061/2021, EU cluster n.5: Climate, Energy and Mobility).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SEP	Standard Eigenvalue Problem
GEP	Generalized Eigenvalue Problem
QEP	Quadratic Eigenvalue Problem

References

1. Craig, R.R.; Bampton, M.C.C. Coupling of substructures for dynamic analyses. *AIAA Journal* **1968**, *6*, 1313–1319. https://doi.org/10.2514/3.4741.
2. Rubin, S. Improved Component-Mode Representation for Structural Dynamic Analysis. *AIAA Journal* **1975**, *13*, 995–1006. https://doi.org/10.2514/3.60497.
3. Wu, L.; Tiso, P. Nonlinear model order reduction for flexible multibody dynamics: a modal derivatives approach. *Multibody System Dynamics* **2016**, *36*, 405–425.
4. Balachandran, B.; Magrab, E.B. *Vibrations*; Cambridge University Press, 2018.
5. Hodges, D.H.; Pierce, G.A. *Introduction to Structural Dynamics and Aeroelasticity*; Cambridge University Press, 2011.
6. Ogunwa, T.; Abdullah, E.; Chahl, J. Modeling and Control of an Articulated Multibody Aircraft. *Applied Sciences* **2022**, *12*, 1162. https://doi.org/10.3390/app12031162.
7. Lewis, F.L.; Vrabie, D.L.; Syrmos, V.L. *Optimal Control*, 3rd edition ed.; John Wiley & Sons Inc: Hoboken, 2012.
8. Kamman, J.W.; Huston, R.L. Dynamics of Constrained Multibody Systems. *Journal of Applied Mechanics* **1984**, *51*, 899–903, [https://asmedigitalcollection.asme.org/appliedmechanics/article-pdf/51/4/899/5457564/899_1.pdf]. https://doi.org/10.1115/1.3167743.
9. Zhang, Y.; Wen, B.; Chen, S. Eigenvalue problem of constrained flexible multibody systems. *Mechanics Research Communications* **1997**, *24*, 11–16. https://doi.org/10.1016/S0093-6413(96)00072-9.
10. González, F.; Masarati, P.; Cuadrado, J.; Naya, M.A. Assessment of Linearization Approaches for Multibody Dynamics Formulations. *Journal of Computational and Nonlinear Dynamics* **2017**, *12*. https://doi.org/10.1115/1.4035410.
11. Pappalardo, C.M.; Lettieri, A.; Guida, D. Stability analysis of rigid multibody mechanical systems with holonomic and nonholonomic constraints. *Archive of Applied Mechanics* **2020**, *90*, 1961–2005. https://doi.org/10.1007/s00419-020-01706-2.
12. Lehoucq, R.B.; Sorensen, D.C. Deflation Techniques for an Implicitly Restarted Arnoldi Iteration. *SIAM Journal on Matrix Analysis and Applications* **1996**, *17*, 789–821. https://doi.org/10.1137/S0895479895281484.
13. Lehoucq, R.B.; Sorensen, D.C.; Yang, C. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*; SIAM, 1998.
14. Stewart, G.W. A Krylov-Schur algorithm for large eigenproblems. *SIAM Journal on Matrix Analysis and Applications* **2002**, *23*, 601–614. https://doi.org/10.1137/S0895479800371529.
15. Tzounas, G.; Dassios, I.; Liu, M.; Milano, F. Comparison of Numerical Methods and Open-Source Libraries for Eigenvalue Analysis of Large-Scale Power Systems. *Applied Sciences* **2020**, *10*, 7592. https://doi.org/10.3390/app10217592.
16. Hernandez, V.; Roman, J.E.; Vidal, V. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software* **2005**, *31*, 351–362. https://doi.org/10.1145/1089014.1089019.
17. Heroux, M.A.; Bartlett, R.A.; Howle, V.E.; Hoekstra, R.J.; Hu, J.J.; Kolda, T.G.; Lehoucq, R.B.; Long, K.R.; Pawlowski, R.P.; Phipps, E.T.; et al. An overview of the Trilinos project. *ACM Transactions on Mathematical Software* **2005**, *31*, 397–423. https://doi.org/10.1145/1089014.1089021.
18. Qiu, Y. SPECTRA. https://github.com/yixuan/spectra, 2022.
19. Ding, X. Krylov Schur Algorithm in MATLAB.
20. Tasora, A.; Serban, R.; Mazhar, H.; Pazouki, A.; Melanz, D.; Fleischmann, J.; Taylor, M.; Sugiyama, H.; Negrut, D. Chrono: An open source multi-physics dynamics engine; Springer, 2016; pp. 19–49.
21. Guennebaud, G.; Jacob, B.; et al. Eigen v3. http://eigen.tuxfamily.org, 2010.
22. Cutroneo, F. Development of a Numerical Method for the Modal Analysis of Constrained Multibody Systems. Thesis, University of Parma, 2021.