Direct Solver Aiming at Elimination of Systematic Errors in 3D Stellar Positions

Konstantin Ryabinin[1][0000−0002−8353−7641],
Gerasimos Sarras[1][0009−0005−5316−4062], Wolfgang Löffler[1][0009−0003−1319−5601],
and Michael Biermann[1][0000−0002−5791−9056]

Astronomisches Rechen-Institut, Center for Astronomy of Heidelberg University,
Mönchhofstr. 12–14, 69120 Heidelberg, Germany
konstantin.riabinin@uni-heidelberg.de,
gerasimos.sarras@uni-heidelberg.de, loeffler@ari.uni-heidelberg.de,
biermann@ari.uni-heidelberg.de

Abstract. The determination of three-dimensional positions and velocities of stars based on the observations collected by a space telescope suffers from the uncertainty of random as well as systematic errors. The systematic errors are introduced by imperfections of the telescope's optics and detectors as well as in the pointing accuracy of the satellite. The fine art of astrometry consists of heuristically finding the best possible calibration model that will account for and remove these systematic errors. Since this is a process based on trial and error, appropriate software is needed that is efficient enough to solve the system of astrometric equations and reveal the astrometric parameters of stars for the given calibration model within a reasonable time. In this work, we propose a novel architecture and corresponding prototype of a direct solver optimized for running on supercomputers. The main advantages expected of this direct method over an iterative one are the numerical robustness, accuracy of the method, and the explicit calculation of the variance-covariance matrix for the estimation of the accuracy and correlation of the unknown parameters. This solver is supposed to handle astrometric systems with billions of equations within several hours. To reach the desired performance, state-of-the-art libraries for parallel computing are used along with the hand-crafted subroutines optimized for hybrid parallelism model and advanced vector extensions of modern CPUs. The developed solver is tested using the mock science data related to the Japan Astrometry Satellite Mission for INfrared Exploration (JASMINE).

Keywords: Astrometry · Data Fitting · Least Squares Method · Eigenproblem · Pseudo-Inverse Matrix · High-Performance Computing · Model Errors.

1 Introduction

The Japan Astrometry Satellite Mission for INfrared Exploration (JASMINE) is a proposal by the Institute of Space and Astronautical Science (ISAS) for a near-infrared space telescope mission by the Japan Aerospace Exploration Agency

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63783-4_23

(JAXA). With its three years mission length JASMINE aims at two complementary science goals: firstly, an astrometric survey of the three-dimensional positions and motions of stars around the centre of our Milky Way, and secondly, a survey aiming at discovering Earth-like exo-planets in the habitable zone around cool red dwarf stars, i.e. stars smaller and cooler than our Sun.

The three-dimensional positions of the stars in the Galactic Centre region are being directly determined by geometric methods, i.e. the change of the aspect angle (parallax angle) of the target star as seen in spring and autumn for opposite locations of the Earth's orbit around the Sun. Since there exist no coordinate markers in space, these aspect angles can only be measured as relative angles between the set of target stars. The mathematical problem of the astrometric data reduction is thus the very same as the geodetic reduction of measured field angles into three-dimensional geographical positions of landscape marks on the surface of Earth. The main difference between geodesy and modern space-based astrometry are the micro-arcsecond accuracy requirements on the calibration of the optical and electronic imaging properties and spatial orientation of the measuring instrument. In space these can only be determined by measuring the very same aspect angles between the different stars.

At the core of the "fine art of astrometry" lies thus finding the best formulation of such a calibration model as well as the subsequent determination of its parameters with high accuracy. As this is an iterative process based on trial and error, there is a need for a software that can solve a given astrometric problem in short time with high accuracy and precision.

The fundamental problem of space-based astrometry is, however, that the number of stars being observed and the number of angles being measured is always extremely large compared to the data storage size and computing power available at the time. In the past the astrometric problem at hand was thus simplified mathematically and the exact solution was approximated iteratively.

The main motivations to choose a direct approach over an iterative one, are the following. From an algorithmic perspective, the direct method terminates in a finite number of steps, thus avoiding arbitrary stopping criteria introduced by an iterative scheme [3]. From an astrometric perspective, only the direct methods can provide estimates of the variances of the unknown parameters and of the residuals based on the covariance matrix which is explicitly calculated and stored in memory for further use. Once these estimates are available, then the scientific investigation of the statistical properties of the solution can start with the aim to identify systematic errors, remove hidden model biases/incompleteness and calibrate out further non-random uncertainties of the model to reach the desired accuracy of the astrometric mission.

This paper is a first step towards the cluster-based implementation of the method and algorithm for a direct non-iterative solution to the JASMINE astrometric problem of determining the three-dimensional positions and velocities of stars around the centre of our Milky Way.

2 The Astrometric Problem

The method for solving geodetic and astrometric problems, the so-called adjustment via mediating observations has been developed in the year 1794 by Carl Friedrich Gauß (eventually published in 1823, [9]) and a few years later in 1805 independently by Adrien-Marie Legendre [12]. In this type of problem the *unknowns* $\boldsymbol{p} = (p_1, \dots p_\iota, \dots p_I)^\mathsf{T}$ are not directly accessible to observation, but each of the *observations* $\boldsymbol{o} = (o_1, \dots o_\ell, \dots o_L)^\mathsf{T}$ can, in principle, be expressed as a problem-dependent function of these unknowns

$$\boldsymbol{o} = f(\boldsymbol{p}) \tag{1}$$

$$\text{with} \quad o_\ell = f_\ell(p_1, \dots p_\iota, \dots p_I). \tag{2}$$

In general the function f will be non-linear. If an estimate $\hat{\boldsymbol{p}}$ for the unknowns $\boldsymbol{p}$ exists, the problem can, however, be linearised around $\hat{\boldsymbol{p}}$ using a Taylor expansion and then neglecting all terms of second order and higher.

In the full multi-dimensional case this kind of linearisation yields

$$\begin{pmatrix} o_1 \\ \vdots \\ o_\ell \\ \vdots \\ o_L \end{pmatrix} \simeq \begin{pmatrix} \hat{o}_1 \\ \vdots \\ \hat{o}_\ell \\ \vdots \\ \hat{o}_L \end{pmatrix} + \underbrace{\left. \begin{pmatrix} \frac{\partial}{\partial p_1} f_1 & \cdots & \frac{\partial}{\partial p_\iota} f_1 & \cdots & \frac{\partial}{\partial p_I} f_1 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial p_\ell} f_\ell & \cdots & \frac{\partial}{\partial p_\iota} f_\ell & \cdots & \frac{\partial}{\partial p_I} f_\ell \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial p_L} f_L & \cdots & \frac{\partial}{\partial p_\iota} f_L & \cdots & \frac{\partial}{\partial p_I} f_L \end{pmatrix} \right|_{\hat{\boldsymbol{p}}}}_{\boldsymbol{\mathcal{D}}} \begin{pmatrix} p_1 - \hat{p}_1 \\ \vdots \\ p_\iota - \hat{p}_\iota \\ \vdots \\ p_I - \hat{p}_I \end{pmatrix} \tag{3}$$

$$\boldsymbol{o} \quad \simeq \quad \hat{\boldsymbol{o}} \quad + \quad \boldsymbol{\mathcal{D}} \quad \Delta\hat{\boldsymbol{p}}, \tag{4}$$

where $\boldsymbol{c} = \hat{\boldsymbol{o}} = F(\hat{\boldsymbol{p}})$ are the estimates for the observations $\boldsymbol{o}$ *calculated* from the estimate parameters $\hat{\boldsymbol{p}}$. $\Delta\hat{\boldsymbol{p}}$ are the updates of the estimate unknowns $\hat{\boldsymbol{p}}$. The matrix $\boldsymbol{\mathcal{D}}$ holding the Jacobian derivatives of the functions f_ℓ with respect to the unknowns $\boldsymbol{p}$ is called the *design matrix* of the problem.

Subtracting $\hat{\boldsymbol{o}} = \boldsymbol{c}$ from Eqn (4) we arrive at the linear *observation equation*

$$\boldsymbol{o} - \boldsymbol{c} = \boldsymbol{\mathcal{D}} \ \Delta\hat{\boldsymbol{p}}. \tag{5}$$

The basic interpretation of this observation equation is that the knowledge and understanding of the problem including an initial estimate $\hat{\boldsymbol{p}}$ for the unknowns $\boldsymbol{p}$ is being used to compute in a first step an estimate prediction $\boldsymbol{c}$ where and when the observations $\boldsymbol{o}$ will be made. The inevitable difference between the estimated prediction $\boldsymbol{c}$ and the actual observation $\boldsymbol{o}$ is then used in a second step to compute the update $\Delta\hat{\boldsymbol{p}}$ to the estimate $\hat{\boldsymbol{p}}$ that would be needed to account for that difference between estimated prediction and actual observation, or

$$\boldsymbol{o} - \boldsymbol{c} - \boldsymbol{\mathcal{D}} \ \Delta\hat{\boldsymbol{p}} \overset{!}{=} 0. \tag{6}$$

3 Methods

3.1 The Astrometric Least-Squares Solution

Given the system of linear observation equations

$$\boldsymbol{l} = \boldsymbol{\mathcal{D}}\boldsymbol{x} \tag{7}$$

with the *unknowns* $\boldsymbol{x} = \Delta\hat{\boldsymbol{p}}$ and differences $\boldsymbol{l} = \boldsymbol{o} - \boldsymbol{c}$ between *observations* and *calculated predictions*, a unique solution exists if the number of unknowns I is equal to the number of observations L and if all observations are linearly independent, i.e. if each observation adds unique information to the system. Then the matrix $\boldsymbol{\mathcal{D}}$ is square, has no zero eigenvalues, has a non-zero determinant and is therefore invertible. The solution is then given by

$$\boldsymbol{x} = \boldsymbol{\mathcal{D}}^{-1}\boldsymbol{l}. \tag{8}$$

In an astrometric problem we have, however, many more observations than we have unknowns. This problem is redundant to a degree $L-I$. Due to the errors inherent to each observation, the observation equations do, however, not only provide redundant information to the problem but also inconsistencies. There exists no longer an exact solution to the problem.

But the observations can be combined in such a way that an approximate solution can be computed which fits the problem in question to some degree. Following the approach published by Legendre [12] the best approximate solution is given by

$$\hat{\boldsymbol{x}} = (\boldsymbol{\mathcal{D}}^{\mathsf{T}}\boldsymbol{\mathcal{D}})^{-1}\boldsymbol{\mathcal{D}}^{\mathsf{T}}\boldsymbol{l} = \boldsymbol{\mathcal{N}}^{-1}\boldsymbol{\mathcal{D}}^{\mathsf{T}}\boldsymbol{l}, \tag{9}$$

which minimises the square sum of the residuals, i.e. the quadratic sum of the differences between the individual observations and the computed solution.

If the number of independent, i.e. absolute observations is greater or equal to the number of unknowns, then the matrix $\boldsymbol{\mathcal{D}}$ has full rank and the normal matrix $\boldsymbol{\mathcal{N}} = \boldsymbol{\mathcal{D}}^{\mathsf{T}}\boldsymbol{\mathcal{D}}$ is regular and positive definite.

If the observations are, however, not independent but, for example, relative to each other, as is the case in astrometry, then the matrix $\boldsymbol{\mathcal{D}}$ is rank deficient. This means that the normal matrix $\boldsymbol{\mathcal{N}}$ is singular and no unique inverse exists. The solution to the problem is no longer unique (with respect to rotation and spin). This can be fixed by adding either constraints to the equations such that the constrained normal matrix becomes regular, or by computing a pseudo-inverse matrix to get one arbitrary astrometric solution and then de-rotating and de-spinning that particular solution in a post-processing step to get the one physical solution.

3.2 The Hipparcos, Gaia and JASMINE Astrometric Solutions

As mentioned already in the introduction, the fundamental problem of space-based astrometry is the sheer amount of unknowns (ten billions in the case of the ESA Gaia astrometry mission) to solve for using an even larger amount of angle measurements (more than two trillion measurements in the case of Gaia), which hitherto prevented any attempt at solving the astrometric problem directly without cutting it into smaller pieces first and then iterating for the full solution.

In the case of the ESA Hipparcos astrometry mission [6] the global astrometric problem with $3 \cdot 10^6$ unknowns was first split into many independent and much smaller astrometric solutions along rings on the celestial sphere made up by the observations during 4.5 revolutions of the satellite around its slowly precessing spin axis. These astrometric solutions computed for each of these rings were then combined into one spherical solution in a second step by simply rotating the rings for a best fit. From this spherical solution, updated spatial orientations of the individual rings were derived and these two steps then iterated for convergence. In addition, the astrometric problem for each individual ring was reduced in size by forward eliminating the attitude unknowns (describing the spatial orientation of the instrument) from the normal equations before the remaining normal matrix was inverted.

In the case of the ESA Gaia astrometry mission [8], the Astrometric Global Iterative Solution (AGIS) [13] is employing a block-iterative scheme, in which the normal matrix for the $8.9 \cdot 10^9$ unknowns is blocked along the astrometric unknowns, the attitude unknowns describing the orientation of the instrument and the calibration unknowns of the instruments, and in which the off-diagonal blocks are all set to zero. This results in a normal matrix in which the three non-zero diagonal blocks are block- or band-diagonal themselves and thus trivially or at least easily inverted. Each of these three main diagonal blocks is inverted and solved for separately by assuming a prior solution for the other two blocks. This solution of one block is serving as prior solution for the other two. The solving of these three blocks is iterated until convergence is achieved.

In contrast to this, the Gaia One Day Astrometric Solution (ODAS) [14] comprised the much smaller but still quite challenging astrometric problem with $5 \cdot 10^5$ unknowns spanning only one ring containing the observations made during about 4.5 revolutions of the satellite around its own slowly precessing spin axis. Here the size of the problem was reduced by only forward-eliminating the astrometric unknowns. The remaining reduced normal matrix was then directly inverted using a singular value decomposition.

The aim of the current work is to find out whether such a direct astrometric solution based on forward elimination and then direct inversion using singular value decomposition can be applied to astrometric problems much larger than the ODAS problem, i.e. to the full JASMINE astrometric problem solving for up to $2.5 \cdot 10^7$ unknowns covering the full 18 months of astrometric mission time. Thus was born the idea of the Astronomisches Rechen-Institut (ARI) JASMINE Astrometric Solution (AJAS).

3.3 The ARI JASMINE Astrometric Solution

The observations in AJAS are two-dimensional. The primary data of these observations are the 2D coordinates (κ, μ) of the corresponding centroid of the light spot (trace of a star detected by a telescope) expressed in the coordinate system of the telescope detector, and the guessed identifier of a source. The pixel coordinates (κ, μ) are determined via segmentation of the exposure into spots and calculation of their centroids. Source identifiers are determined via a so-called cross-match procedure, in which the catalog of known stars is used as *a priori* data. Both procedures are parts of raw data processing and lay beyond the scope of this work.

A system of astrometric equations is built based on some reference frame. In AJAS, we have chosen the telescope's field of view reference frame (FoVRS), because it simplifies the subsequent calculations. Being two-dimensional, each observation contributes twice to the system of astrometric equations spawning two equations for perpendicular directions (η, ζ) of the FoVRS. The unknowns in each equation are the set of calibration (nuisance) and source parameters. The coefficients are the derivatives of field angles η and ζ with respect to the corresponding parameters.

The design matrix of the AJAS astrometric problem is $\boldsymbol{\mathcal{D}} = (\boldsymbol{\mathcal{C}}_0 \, \boldsymbol{\mathcal{C}}_1 \, \boldsymbol{\mathcal{S}})$, where $\boldsymbol{\mathcal{C}}_0$ is a lower-order calibration part encoding the attitude of the satellite telescope, $\boldsymbol{\mathcal{C}}_1$ is a higher-order calibration part taking care of the above-mentioned imperfections of the telescope optics, and $\boldsymbol{\mathcal{S}}$ is a part related to source parameters. In the end, only the unknowns related to $\boldsymbol{\mathcal{S}}$ have a scientific meaning, still, all the unknowns should be determined to ensure the error calculus.

The vector side $\boldsymbol{l}$ of the system is built based on the difference between observed and calculated η and ζ coordinates of the observations:

$$\boldsymbol{l} = \begin{pmatrix} \eta_\ell^{\text{obs}} - \eta_\ell^{\text{calc}} \\ \zeta_\ell^{\text{obs}} - \zeta_\ell^{\text{calc}} \end{pmatrix}_{\ell=\overline{1,L}} = \begin{pmatrix} \eta^{\text{obs}}(\kappa_\ell, \mu_\ell) - \eta^{\text{calc}}(t_\ell, \lambda_\ell) \\ \zeta^{\text{obs}}(\kappa_\ell, \mu_\ell) - \zeta^{\text{calc}}(t_\ell, \lambda_\ell) \end{pmatrix}_{\ell=\overline{1,L}},$$

where L is the total number of observations, t_ℓ is the timestamp of observation ℓ, and λ_ℓ is the source assigned to the observation ℓ.

For the very simple case with a single detector, 4 exposures, 40 observations, 10 sources, 2 source parameters per source, 6 nuisance parameters per observation in the lower-order calibration part, and 6 nuisance parameters per observation in the higher-order calibration part, the design matrix $\boldsymbol{\mathcal{D}}$ looks like in Fig. 1a. For the real data, $\boldsymbol{\mathcal{D}}$ will be way bigger, but the overall structure will be retained.

For a 3-year JASMINE mission, 4 detectors and $\sim 10^6$ exposures are planned. Around $1.15 \cdot 10^5$ sources are expected to be observed with a total number of observations to be around $8.25 \cdot 10^9$ observations, and 5 astrometric parameters per source should be determined (2 celestial coordinates, 2 proper motion components, and parallax). For this amount of input data, the number of rows in $\boldsymbol{\mathcal{D}}$ will be $1.65 \cdot 10^{10}$ (twice as many, as observations, because of two coordinates η and ζ). The number of columns in $\boldsymbol{\mathcal{C}}_0$ will be $2.4 \cdot 10^7$ (count of detectors $\times$ count of exposures $\times$ 2 coordinates $\times$ count of lower-order calibration parameters). The exact number of columns in $\boldsymbol{\mathcal{C}}_1$ depends on the particular higher-order calibration model but is expected to be no more than 100. The number of columns in $\boldsymbol{\mathcal{S}}$ will be $5.75 \cdot 10^5$ (count of sources $\times$ count of source parameters).

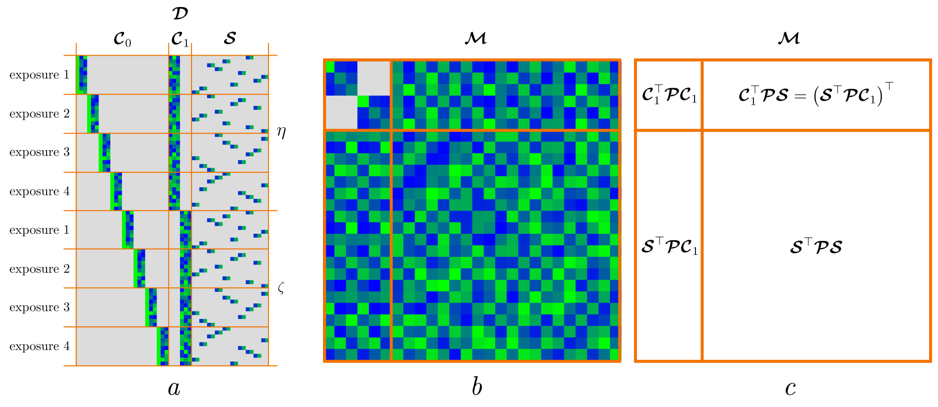
$\mathcal{D}$
$\mathcal{C}_0$
$\mathcal{C}_1$
$\mathcal{S}$
exposure 1
exposure 2
exposure 3
exposure 4
exposure 1
exposure 2
exposure 3
exposure 4
η
ζ
$\mathcal{M}$
$\mathcal{M}$
$\mathcal{C}_1^\top \mathcal{P} \mathcal{C}_1$
$\mathcal{C}_1^\top \mathcal{P} \mathcal{S} = (\mathcal{S}^\top \mathcal{P} \mathcal{C}_1)^\top$
$\mathcal{S}^\top \mathcal{P} \mathcal{C}_1$
$\mathcal{S}^\top \mathcal{P} \mathcal{S}$
a
b
c

Fig. 1: Structure of the matrix $\boldsymbol{\mathcal{D}}$ (a), structure of the matrix $\boldsymbol{\mathcal{M}}$ (b), and main blocks of the matrix $\boldsymbol{\mathcal{M}}$ (c), gray indicates zero values, green-to-blue indicates non-zero values

Altogether, this gives a size of $\boldsymbol{\mathcal{D}}$ to be in the order of $10^{10} \times 10^7$. Despite sparsity, this matrix is far too big to be treated directly. Instead, another system is to be solved, which is based on the reduced normal matrix $\boldsymbol{\mathcal{M}}$ derived from the normal matrix $\boldsymbol{\mathcal{N}}$:

$$\boldsymbol{\mathcal{N}} \boldsymbol{x} = \boldsymbol{\mathcal{D}}^\mathsf{T} \boldsymbol{l}, \quad \boldsymbol{\mathcal{N}} = \boldsymbol{\mathcal{D}}^\mathsf{T} \boldsymbol{\mathcal{D}} = \begin{pmatrix} \boldsymbol{\mathcal{C}}_0^\mathsf{T} \boldsymbol{\mathcal{C}}_0 & \boldsymbol{\mathcal{C}}_0^\mathsf{T} \boldsymbol{\mathcal{O}} \\ \boldsymbol{\mathcal{O}}^\mathsf{T} \boldsymbol{\mathcal{C}}_0 & \boldsymbol{\mathcal{O}}^\mathsf{T} \boldsymbol{\mathcal{O}} \end{pmatrix}, \quad \boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_{\boldsymbol{\mathcal{C}}_0} \\ \boldsymbol{x}_{\boldsymbol{\mathcal{O}}} \end{pmatrix}, \quad \boldsymbol{\mathcal{O}} = (\boldsymbol{\mathcal{C}}_1 \ \boldsymbol{\mathcal{S}}).$$

The matrix $\boldsymbol{\mathcal{M}}$ is built using a forward elimination of the $\boldsymbol{\mathcal{C}}_0^\mathsf{T} \boldsymbol{\mathcal{C}}_0$ component of $\boldsymbol{\mathcal{N}}$:

$$\begin{pmatrix} \boldsymbol{\mathcal{I}} & \boldsymbol{\mathcal{V}} \boldsymbol{\mathcal{O}} \\ \boldsymbol{0} & \boldsymbol{\mathcal{M}} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_{\boldsymbol{\mathcal{C}}_0} \\ \boldsymbol{x}_{\boldsymbol{\mathcal{O}}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mathcal{V}} \boldsymbol{l} \\ \boldsymbol{b}_{\boldsymbol{\mathcal{O}}} \end{pmatrix},$$

where

$$\boldsymbol{\mathcal{V}} = \left(\boldsymbol{\mathcal{C}}_0^\mathsf{T} \boldsymbol{\mathcal{C}}_0 \right)^{-1} \boldsymbol{\mathcal{C}}_0^\mathsf{T}, \quad \boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{O}}^\mathsf{T} \boldsymbol{\mathcal{P}} \boldsymbol{\mathcal{O}}, \quad \boldsymbol{b}_{\boldsymbol{\mathcal{O}}} = \boldsymbol{\mathcal{O}}^\mathsf{T} \boldsymbol{\mathcal{P}} \boldsymbol{l}, \quad \boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{I}} - \boldsymbol{\mathcal{C}}_0 \boldsymbol{\mathcal{V}}, \tag{10}$$

and $\boldsymbol{\mathcal{I}}$ is the identity matrix. Then, the system to be solved is:

$$\boldsymbol{\mathcal{M}} \boldsymbol{x}_{\boldsymbol{\mathcal{O}}} = \boldsymbol{b}_{\boldsymbol{\mathcal{O}}}. \tag{11}$$

The structure of the matrix $\boldsymbol{\mathcal{M}}$ is shown in Fig. 1b and Fig. 1c. It is symmetric, singular, positive semi-definite. Its size is driven by the sum of the column numbers of $\boldsymbol{\mathcal{C}}_1$ and $\boldsymbol{\mathcal{S}}$ and is expected to be within $10^6 \times 10^6$ for the JASMINE mission.

From (11), the unknowns related to $\boldsymbol{\mathcal{C}}_1$ and $\boldsymbol{\mathcal{S}}$ can be calculated as:

$$\boldsymbol{x}_{\boldsymbol{\mathcal{O}}} = \boldsymbol{\mathcal{M}}^+ \boldsymbol{b}_{\boldsymbol{\mathcal{O}}} = \boldsymbol{\mathcal{Z}} \boldsymbol{\mathcal{E}}^{-1} \boldsymbol{\mathcal{Z}}^\mathsf{T} \boldsymbol{b}_{\boldsymbol{\mathcal{O}}}, \tag{12}$$

where $\boldsymbol{\mathcal{M}}^+$ is pseudo-inverse, $\boldsymbol{\mathcal{Z}}$ is a matrix of eigenvectors, and $\boldsymbol{\mathcal{E}}$ is a diagonal matrix of eigenvalues of $\boldsymbol{\mathcal{M}}$. The $\boldsymbol{\mathcal{M}}^+$ is the covariance matrix which stores the standard error estimates of the higher calibration parameters a s well as of the source parameters and is needed for the statistical analysis of the residuals.

The rest of the unknowns (related to $\boldsymbol{\mathcal{C}}_0$) can be determined via a backsubstitution:

$$\boldsymbol{x}_{\boldsymbol{\mathcal{C}}_0} = \left(\boldsymbol{\mathcal{C}}_0^{\mathsf{T}}\boldsymbol{\mathcal{C}}_0\right)^{-1} \boldsymbol{\mathcal{C}}_0^{\mathsf{T}} \left(\boldsymbol{l} - \boldsymbol{\mathcal{O}}\boldsymbol{x}_{\boldsymbol{\mathcal{O}}}\right). \tag{13}$$

Finally, for the error calculus, the residuals are calculated as

$$\boldsymbol{r} = \boldsymbol{l} - \boldsymbol{\mathcal{O}}\boldsymbol{x}_{\boldsymbol{\mathcal{O}}} - \boldsymbol{\mathcal{C}}_0\boldsymbol{x}_{\boldsymbol{\mathcal{C}}_0}, \tag{14}$$

where they should follow a normal distribution based on the assumption that the observational errors are uncorrelated to each other and follow also a normal distribution.

The scientific goal of the JASMINE mission is the determination of the 3D angular positions of the stars on the celestial sphere with a target accuracy of down to 10 µas. The accuracy of the instrument calibration must therefore be better than this. And that means that we have to calibrate the geometric locations of the photosensitive elements in the focal plane to an accuracy of at least 20 nm or 100 silicon atom diameters.

This kind of calibration accuracy cannot be achieved in the laboratory on-ground before launch. Launching the instrument into space will change its geometry in an uncertain way at a far greater scale. The instrument calibration must therefore be based on the very same observational data that determine the stellar positions. This can be achieved by setting up a calibration model and treating its parameters as unknowns of the overall problem. Any error or uncertainty in the formulation of the calibration model will show up as systematic effect in the distribution of the residuals defined in Eqn 14 and their standard errors. These systematics need to be accounted for in the formulation of an improved calibration model and a new solution based on this improved model should be computed to reduce this model uncertainty.

Solving the overall astrometric problem requires therefore a speedy and accurate method with which to compute the solution for a whole series of increasingly more accurate calibration models. The actual and detailed formulation of these calibration models is, however, based on mining the residual data and their standard errors for correlations and systematics.

3.4 Approaches to Build a Direct Astrometric Solver

The direct astrometric solver is defined by Eqns (11–14). Each formula allows for parallel calculations, therefore, the direct solver can reach correspondingly high performance. The most computationally intensive part is the calculation of the pseudo-inverse matrix $\boldsymbol{\mathcal{M}}^{+}$ (Eqn (12)), and specifically, the calculation of eigenvalues $\boldsymbol{e}$ and eigenvectors $\boldsymbol{\mathcal{Z}}$ of the matrix $\boldsymbol{\mathcal{M}}$. This is because $\boldsymbol{\mathcal{M}}$ is nearly a dense matrix, while the rest of operands is rather sparse. So, the general architecture of parallelism for the direct solver should be chosen in a way that ensures the best performance of the eigenproblem solution. The rest of the operations should be implemented in the same architecture to minimize the overhead of moving the intermediate data. In this regard, our first experiments were aimed at finding the most optimal hardware and software configuration to solve an eigenproblem of a real symmetric matrix of size $10^6 \times 10^6$ (as this is the upper estimation for the matrix $\boldsymbol{\mathcal{M}}$ in the JASMINE mission).

A state-of-the-art library to solve the eigenproblem for dense matrices is EigenExa [16]. It is written in modern Fortran and implements a hybrid parallelism model based on MPI and OpenMP standards. Compared to ScaLAPACK library [2], which is a *de facto* standard for distributed parallel computations involving linear algebra subroutines, EigenExa has the following distinctive features [16]:

1. The traditional way to solve the eigenproblem in ScaLAPACK is using the Householder transformation [18] of the input symmetric real matrix $\boldsymbol{\mathcal{M}}$ to tridiagonal form $\boldsymbol{\mathcal{T}}$, then performing the Divide-and-Conquer (DC) algorithm [4] or the more modern and efficient algorithm of Multiple Relatively Robust Representations (MRRR) [5] to find the eigenvalues and eigenvectors of $\boldsymbol{\mathcal{T}}$, and then transforming the eigenvectors back to the initial matrix $\boldsymbol{\mathcal{M}}$ (while eigenvalues of $\boldsymbol{\mathcal{T}}$ match eigenvalues of $\boldsymbol{\mathcal{M}}$ since Householder transformation preserves the spectrum). Instead, EigenExa transforms the matrix $\boldsymbol{\mathcal{M}}$ to a pentadiagonal form $\boldsymbol{\mathcal{P}}$, then performs a modified version of the DC algorithm to reveal eigenvalues and eigenvectors and transforms the eigenvectors back to matrix $\boldsymbol{\mathcal{M}}$. This approach reduces the number of numerical operations involved in the calculation [7].
2. EigenExa uses fine-tuned symmetric multiprocessing parallelism inside the distributed processes allowing for a better ratio of computation operations to service operations.
3. EigenExa implements a set of data arrangement optimizations, which improve the CPU cache usage (including reducing the cache thrashing).

These features significantly increase the overall processing speed. According to our experiments on small-scale matrices (order of $10^4 \times 10^4$) and small process grid (4 nodes with 2 cores each), the average speedup of EigenExa compared to ScaLAPACK (with MRRR) is about a factor of 10. For big-scale matrices, as reported by Sakurai et al. [16] and Imamura et al. [11], EigenExa is capable of solving the eigenproblem for $10^6 \times 10^6$ matrix in less than 1 hour using 82944 nodes with 8 cores each of the K supercomputer or 4096 nodes with 48 cores each of the Fugaku supercomputer, whereby the scaling is close to linear. This brings us to the conclusion, that the cluster-based architecture with a hybrid model of parallelism is the way to implement a direct astrometric solver to meet our performance requirements.

4 Proposed Architecture of the Direct Astrometric Solver

We adopt the model of hybrid parallel computations within AJAS.

AJAS is a software pipeline written in C++ and optimized for CPU clusters. Its schema is shown in Fig. 2.

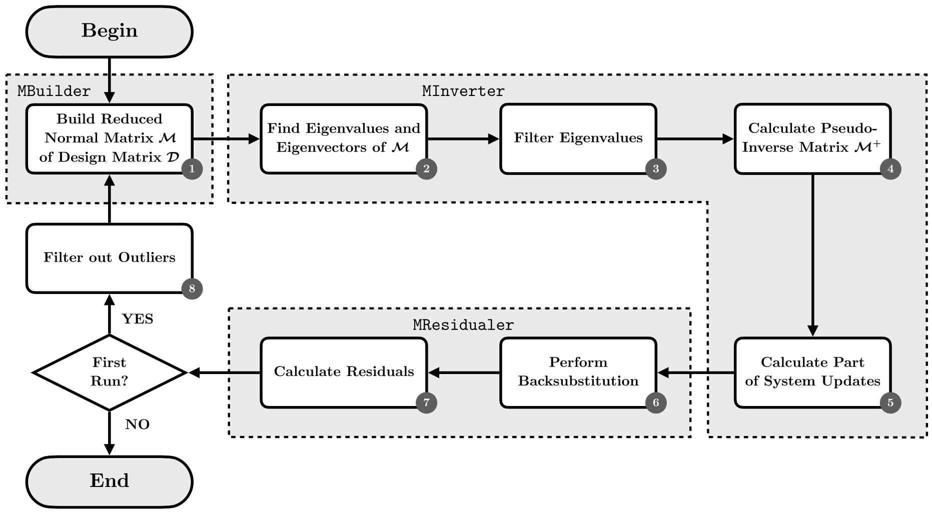
Begin
MBuilder
Build Reduced Normal Matrix $\mathcal{M}$ of Design Matrix $\mathcal{D}$
1
MInverter
Find Eigenvalues and Eigenvectors of $\mathcal{M}$
2
Filter Eigenvalues
3
Calculate Pseudo-Inverse Matrix $\mathcal{M}^+$
4
Filter out Outliers
8
YES
First Run?
NO
MResidualer
Calculate Residuals
7
Perform Backsubstitution
6
Calculate Part of System Updates
5
End

Fig. 2: Direct astrometric solver pipeline

AJAS consists of 3 modules (`MBuilder`, `MInverter`, and `MResidualer`) executed sequentially, while each one performs calculations using a hybrid parallelism. The reason for grouping the pipeline steps into these modules is the semantics of operations. Currently, the outlier filtering step (8) is not yet implemented, but very probably it will be attached to `MResidualer` in the future.

The communication between processes adheres to the Message Passing Interface (MPI) standard. Each process is supposed to run on an individual cluster node, whereby $\mathit{\Pi}$ available nodes are logically organized as a square-shaped $\left\lfloor\sqrt{\mathit{\Pi}}\right\rfloor \times \left\lfloor\sqrt{\mathit{\Pi}}\right\rfloor$ grid. Each process utilizes multi-threading to organize a fine-grain parallelism. The workload is balanced dynamically based on the actual dimensions of the problem.

4.1 Matrix Building Module

The `MBuilder` module (comprising step (1) in Fig. 2) is responsible for building the reduced normal matrix $\boldsymbol{\mathcal{M}}$ and vector $\boldsymbol{b}_{\boldsymbol{\mathcal{O}}}$ of the astrometric equations system. It computes the following formulas derived from (10):

$$\boldsymbol{\mathcal{M}} = \sum_{n=1}^{N}\sum_{\tau=1}^{\Upsilon} \boldsymbol{\mathcal{O}}_{n\tau}^{\mathsf{T}} \boldsymbol{\mathcal{P}}_{n\tau} \boldsymbol{\mathcal{O}}_{n\tau}, \qquad (15) \qquad \boldsymbol{b}_{\boldsymbol{\mathcal{O}}} = \sum_{n=1}^{N}\sum_{\tau=1}^{\Upsilon} \boldsymbol{\mathcal{O}}_{n\tau}^{\mathsf{T}} \boldsymbol{\mathcal{P}}_{n\tau} \boldsymbol{l}_{n\tau}, \qquad (16)$$

where $\boldsymbol{\mathcal{O}}_{n\tau} = (\boldsymbol{\mathcal{C}}_{1,n\tau}\ \boldsymbol{\mathcal{S}}_{n\tau})$, $\boldsymbol{\mathcal{P}}_{n\tau} = \boldsymbol{\mathcal{I}} - \boldsymbol{\mathcal{C}}_{0,n\tau}\left(\boldsymbol{\mathcal{C}}_{0,n\tau}^{\mathsf{T}}\boldsymbol{\mathcal{C}}_{0,n\tau}\right)^{-1}\boldsymbol{\mathcal{C}}_{0,n\tau}^{\mathsf{T}}$, N is the number of telescope detectors, Υ is the number of exposures made during the mission.

Each process of `MBuilder` composes a block of the matrix $\boldsymbol{\mathcal{M}}$ and a block of the vector $\boldsymbol{b}_{\boldsymbol{\mathcal{O}}}$ by computing $N\Upsilon/\Pi$ "layers" (double-sum internals from (15) and (16)) in a multi-threaded way. The elements of these layers, which belong to the matrix and vector blocks of a particular process, are summed up locally; the rest is transmitted for summation to other processes. The result blocks of $\boldsymbol{\mathcal{M}}$ are then processed by the `MInverter` module. Upon calculation, the intermediate data are stored in binary files for subsequent use by the `MResidualer` module.

To save memory and numerical operations, the layers are compressed exploiting the fact that their components are rather sparse. The zero elements are predicted analytically and excluded from both storage and calculations. To tackle this, a customized code for matrix operations (matrix-matrix and matrix-vector multiplication, matrix addition, etc.) has been implemented. The stored and summed/transmitted layer components with their minimal compression ratios and memory footprint fractions are listed in Table 1. The actual values depend on the calibration model, which will be adjusted during the real mission. On one hand, the more complicated the calibration model is the more unknowns are introduced to the system of astrometric equations but on the other, they will also contain more zero entries. So, the compression ratio will grow.

Table 1: Approximate values for data components compression and memory footprints calculated for the expected JASMINE data

	Element	Compression ratio	Fraction of total memory footprint
Stored on disk	$\boldsymbol{\mathcal{C}}_{0,n\tau}$	2	$1.16 \cdot 10^{-4}$
	$\boldsymbol{\mathcal{C}}_{1,n\tau}$	8	$6.95 \cdot 10^{-4}$
	$\boldsymbol{\mathcal{S}}_{n\tau}$	$1.15 \cdot 10^{5}$	$1.93 \cdot 10^{-4}$
	$\boldsymbol{l}_{n\tau}$	1	$3.86 \cdot 10^{-5}$
	$\left(\boldsymbol{\mathcal{C}}_0^{\mathsf{T}}\boldsymbol{\mathcal{C}}_0\right)^{-1}$	2	$6.73 \cdot 10^{-7}$
Stored in RAM; transmitted and summed up	$\boldsymbol{\mathcal{C}}_{1,n\tau}^{\mathsf{T}}\boldsymbol{\mathcal{P}}_{n\tau}\boldsymbol{\mathcal{C}}_{1,n\tau}$	61	$1.28 \cdot 10^{-5}$
	$\boldsymbol{\mathcal{S}}_{n\tau}^{\mathsf{T}}\boldsymbol{\mathcal{P}}_{n\tau}\boldsymbol{\mathcal{C}}_{1,n\tau}$	892	$3.47 \cdot 10^{-3}$
	$\boldsymbol{\mathcal{S}}_{n\tau}^{\mathsf{T}}\boldsymbol{\mathcal{P}}_{n\tau}\boldsymbol{\mathcal{S}}_{n\tau}$	$1.24 \cdot 10^{4}$	$9.95 \cdot 10^{-1}$
	$\boldsymbol{\mathcal{C}}_{1,n\tau}^{\mathsf{T}}\boldsymbol{\mathcal{P}}_{n\tau}\boldsymbol{l}_{n\tau}$	4	$1.35 \cdot 10^{-6}$
	$\boldsymbol{\mathcal{S}}_{n\tau}^{\mathsf{T}}\boldsymbol{\mathcal{P}}_{n\tau}\boldsymbol{l}_{n\tau}$	111	$1.93 \cdot 10^{-4}$

The communication and data access schemes of `MBuilder` are a potential bottleneck that should be investigated and optimized further.

4.2 Matrix Inversion Module

The `MInverter` module inverts the previously created matrix $\boldsymbol{\mathcal{M}}$ and calculates the part of the astrometric system's solution implementing Eqn (12). As mentioned above, $\boldsymbol{\mathcal{M}}$ is singular, so it is inverted by a singular value decomposition [1], whereby the diagonalization is performed using the eigenvectors and reciprocal eigenvalues.

The eigenproblem is efficiently solved by the EigenExa library (Fig. 2, step 2). It requires $\boldsymbol{\mathcal{M}}$ to be distributed over the square-shaped $\left\lfloor\sqrt{\Pi}\right\rfloor \times \left\lfloor\sqrt{\Pi}\right\rfloor$ process grid using the block-cyclic distribution pattern [15], whereby despite the matrix symmetry, all the elements of $\boldsymbol{\mathcal{M}}$ should be presented for the sake of optimal processing. Moreover, to minimize the probability of cache thrashing, blocks are padded in memory with some empty rows and columns (the exact number of which is calculated by EigenExa).

When EigenExa finishes, each cluster node has a corresponding block of the eigenvector's set $\boldsymbol{\mathcal{Z}}$ stored in RAM using the same block-cyclic layout and paddings as the blocks of the input matrix $\boldsymbol{\mathcal{M}}$. Along with it, each node has a copy of the full set of eigenvalues $\boldsymbol{e}$.

Following the pipeline shown in Fig. 2, the next step is the eigenvalues filtering (step 3). It is very lightweight and performed by each node individually. The filtering zeroes out the expected zero eigenvalues that are numerically close to but not exactly equal to zero.

Next, the pseudo-inverse is computed (step 4). For the sake of efficiency, first, each node scales each i-th column of the $\boldsymbol{\mathcal{Z}}$ block with $\sqrt{1/\boldsymbol{e}_i}$, if $\boldsymbol{e}_i > 0$, and 0, if $\boldsymbol{e}_i = 0$. As long as $\boldsymbol{\mathcal{M}}$ is positive semi-definite, all the eigenvalues are non-negative. This allows us to reformulate Eqn (12) as $\boldsymbol{\mathcal{M}}^{+} = \boldsymbol{\mathcal{Z}}_{\boldsymbol{e}}\boldsymbol{\mathcal{Z}}_{\boldsymbol{e}}^{\mathsf{T}}$, where $\boldsymbol{\mathcal{Z}}_{\boldsymbol{e}}$ is scaled $\boldsymbol{\mathcal{Z}}$.

This, in turn, allows the use of the ScaLAPACK `PDSYRK` function that efficiently multiplies a real matrix by its transpose. For this operation, the block-cyclic layout of $\boldsymbol{\mathcal{Z}}$ is reused, so no data is moved or copied in memory. The blocks of multiplication results overwrite the blocks of matrix $\boldsymbol{\mathcal{M}}$, so no additional memory is allocated.

Step 5 is the multiplication of the $\boldsymbol{\mathcal{M}}^{+}$ matrix by the vector $\boldsymbol{b}_{\boldsymbol{\mathcal{O}}}$, which has been distributed over the nodes in a block-cyclic way by the `MBuilder` module. The ScaLAPACK `PDSYMV` function is used to perform the matrix-vector multiplication. The result vector $\boldsymbol{x}_{\boldsymbol{\mathcal{O}}}$ is gathered on the first node and stored in a file.

4.3 Residuals Calculation Module

The `MResidualer` module adopts Eqns (13) and (14) (corresponding to the steps (6) and (7) in Fig. 2) split into blocks by detectors and exposures:

$$\boldsymbol{x}_{\boldsymbol{\mathcal{C}}_0,n\tau} = \left(\boldsymbol{\mathcal{C}}_{0,n\tau}^{\mathsf{T}}\boldsymbol{\mathcal{C}}_{0,n\tau}\right)^{-1}\boldsymbol{\mathcal{C}}_{0,n\tau}^{\mathsf{T}}\left(\boldsymbol{l}_{n\tau} - \boldsymbol{\mathcal{O}}_{n\tau}\boldsymbol{x}_{\boldsymbol{\mathcal{O}},n\tau}\right), \tag{17}$$

$$\boldsymbol{r}_{n\tau} = \boldsymbol{l}_{n\tau} - \boldsymbol{\mathcal{O}}_{n\tau}\boldsymbol{x}_{\boldsymbol{\mathcal{O}},n\tau} - \boldsymbol{\mathcal{C}}_{0,n\tau}\boldsymbol{x}_{\boldsymbol{\mathcal{C}}_0,n\tau}. \tag{18}$$

It reuses the components stored by `MBuilder`. The calculations are embarrassingly parallel (having no dependencies and no inter-process communication) and distributed over Π available cluster nodes so that each one has to process $N\Upsilon/\Pi$ blocks.

5 Discussion

To preliminarily validate the AJAS software, we created a set of unit tests for all its modules. For this, we recreated the AJAS pipeline in pure sequential code without any optimizations (using just trivial matrix-matrix and matrix-vector operations) in Python using the NumPy library [10]. This pipeline "twin" can handle very small-scale problems only but allows us to ensure that the AJAS pipeline's numerical part works correctly on all the steps.

To test the accuracy of AJAS, we compared the solution of different small-scale systems found by AJAS, by singular value decomposition [1] of reduced normal matrix $\boldsymbol{\mathcal{M}}$ implemented in Python (using NumPy) and by QR decomposition [17] of the design matrix $\boldsymbol{\mathcal{D}}$ implemented in Julia. The comparison showed the equality of those solutions up to the machine epsilon.

To preliminary test the efficiency of the AJAS implementation, we run it on 2×2 process grid using 4 nodes of the Baden-Württemberg cluster bwUniCluster 2.0, which is available for academic use for the universities of Baden-Württemberg (Germany). Each node has 40 Intel®Xeon Gold 6230 2.1 GHz CPU cores and 96 Gb RAM. To allow for basic profiling, we also developed a simple generator for mock science data in Julia language that produces artificial observations. The needed amount of calculations in AJAS is mainly driven by the size of the $\boldsymbol{\mathcal{M}}$ matrix, so we generated the data in the amount needed to build matrices of size $m \times m$, where $m = 10^4;\ 2 \cdot 10^4;\ 3 \cdot 10^4;\ 4 \cdot 10^4$.

The approximate time dependencies of the AJAS pipeline steps revealed from the profiling are the following ($m \approx A\Lambda$, A is the number of astrometric parameters, Λ is the number of sources observed, Υ is the number of exposures made, L is the total number of observations made, N is the number of detectors in the telescope):

1. Building of the matrix $\boldsymbol{\mathcal{M}}$: $t_1 = t_1(N\Upsilon, L, m^2) \approx 1.7 \cdot t_2$.
2. Finding eigenvalues and eigenvectors of $\boldsymbol{\mathcal{M}}$: $t_2 = t_2(m^2)$.
3. Filtering eigenvalues: $t_3 = t_3(m) \approx 0$.
4. Calculating pseudo-inverse matrix $\boldsymbol{\mathcal{M}}^+$: $t_4 = t_4(m^2) \approx 0.4 \cdot t_2$.
5. Calculating part of system updates: $t_5 = t_5(m) \approx 10^{-3} \cdot t_2$.
6. Performing backsubstitution: $t_6 = t_6(N\Upsilon, L) \ll t_2$.
7. Calculating residuals: $t_7 = t_7(N\Upsilon, L) \ll t_2$.

As seen in the list above, the timing of steps (1)–(5) depends on the size of $\boldsymbol{\mathcal{M}}$ (while the timing of (1) depends mainly on the number of detectors, exposures, and observations). In terms of the amount of computations, step (2) is the heaviest one. However step (1) takes the longest execution time because of the intensive disk and RAM access, as well as inter-process communication. Step (3) is negligibly fast because it implies just a single scan of eigenvalues zeroing out the values smaller than a given threshold. The timing of steps (6) and (7) depends on the number of detectors, exposures, and observations, but is much smaller than the timing of step (2).

To run a bigger problem that will be close to the real JASMINE size, a larger process grid on the cluster is required. 3 years of JASMINE operation will result in the system of approx. $1.65 \cdot 10^{10}$ astrometric equations with $2.5 \cdot 10^7$ unknowns and the matrix $\boldsymbol{\mathcal{M}}$ will be around $6 \cdot 10^5 \times 6 \cdot 10^5$, which requires 10.5 Tb of RAM and 7 Tb of disk storage to run the pipeline.

6 Conclusion

The paper delineates the novel approach to the development of the direct astrometric solver aimed at revealing stellar properties from the observations made by a space telescope. AJAS, the software implementation of the solver, is optimized for execution on CPU clusters. The computational core is based on the EigenExa, ScaLAPACK, Intel®oneMKL, and Intel®MPI libraries, which ensure high performance in computations and communication between the distributed cluster nodes. Preliminary testing on realistic mock science data and a small-scale process grid on the cluster demonstrate the viability of the proposed approach and architecture. The high performance and numerical accuracy of the solver are crucial to allow for multiple runs in a reasonable time, which is needed to fine-tune the calibration model parameters in order to mitigate the model's uncertainty. Thereby, the correctness of the astrometric solution will be improved by a proper accounting for the stellar properties, orientation of the space telescope, and imperfections of its optics.

The foremost step for future work is the detailed testing of the developed software on the larger process grids on the clusters to reveal and eliminate bottlenecks in computation and communication routines as well as to confirm the predictions of the time needed to handle the astrometric problems of realistic size.

7 Acknowledgments

This work was financially supported by the German Aerospace Agency (Deutsches Zentrum für Luft- und Raumfahrt e.V., DLR) through grant 50OD2201. The authors acknowledge support by the state of Baden-Württemberg through bwHPC. The authors also thank the National Astronomical Observatory of Japan (NAOJ) fruitful discussions for the mission specifics.

References

1. Ben-Israel, A., Greville, T.N.E.: Generalized Inverses. Theory and Applications. Springer, New York (2003). https://doi.org/10.1007/b97366
2. Blackford, L.S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R.C.: ScaLAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA (1997)
3. Bombrun, A., Lindegren, L., Holl, B., Jordan, S.: Complexity of the gaia astrometric least-squares problem and the (non-)feasibility of a direct solution method. Astronomy & Astrophysics **516**, A77 (2010). https://doi.org/10.1051/0004-6361/200913503

4. Cuppen, J.J.M.: A divide and conquer method for the symmetric tridiagonal eigenproblem. Numerische Mathematik **36**(2), 177–195 (1980). `https://doi.org/10.1007/BF01396757`
5. Dhillon, I.S., Parlett, B.N., Vömel, C.: The Design and Implementation of the MRRR Algorithm. ACM Transactions on Mathematical Software **32**(4), 533–560 (2006). `https://doi.org/10.1145/1186785.1186788`
6. ESA: The Hipparcos and Tycho Catalogues. ESA SP-1200 (1997), `https://www.cosmos.esa.int/web/hipparcos/catalogues`
7. Fukaya, T., Imamura, T.: Performance Evaluation of the Eigen Exa Eigensolver on Oakleaf-FX: Tridiagonalization Versus Pentadiagonalization. In: 2015 IEEE International Parallel and Distributed Processing Symposium Workshop. pp. 960–969 (2015). `https://doi.org/10.1109/IPDPSW.2015.128`
8. Gaia Collaboration, Prusti, T., de Bruijne, J.H.J., Brown, A.G.A., Vallenari, A., Babusiaux, C., Bailer-Jones, C.A.L., Bastian, U., Biermann, M., Evans, D.W., et al.: The Gaia mission. Astronomy & Astrophysics **595**, A1 (Nov 2016). `https://doi.org/10.1051/0004-6361/201629272`
9. Gauß, C.F.: Theoria Combinationis Observationum Erroribus Minimis Obnoxiae. In: Commentationes Societatis Regiae Scientiarum Gottingensis recentiores – Classis Physicae, vol. 5. Henrich Dieterich, Göttingen (1823), `https://archive.org/details/theoriacombinat00gausgoog`
10. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (Sep 2020). `https://doi.org/10.1038/s41586-020-2649-2`, `https://doi.org/10.1038/s41586-020-2649-2`
11. Imamura, T., Terao, T., Ina, T., Hirota, Y., Ozaki, K., Uchino, Y.: Performance benchmark of the latest EigenExa on Fugaku (2022), `https://sighpc.ipsj.or.jp/HPCAsia2022/poster/108_poster.pdf`
12. Legendre, A.M.: Nouvelles Méthodes Pour La Détermination des Orbites des Comètes. Firmin Didot, Paris (1805), `https://archive.org/details/61Legendre`
13. Lindegren, L., Lammers, U., Hobbs, D., O'Mullane, W., Bastian, U., Hernàndez, J.: The astrometric core solution for the Gaia mission. Overview of models, algorithms, and software implementation. Astronomy & Astrophysics **538**, A78 (Feb 2012). `https://doi.org/10.1051/0004-6361/201117905`, `http://cdsads.u-strasbg.fr/abs/2012A%26A...538A..78L`
14. Löffler, W., Bastian, U., Biermann, M., Jordan, S., Brüsemeister, T., Stampa, U., Bernstein, H.H.: The one-day astrometric solution for the gaia mission. Astronomy & Astrophysics (in preparation)
15. Ostrouchov, S.: Block Cyclic Data Distribution (1995), `https://www.netlib.org/utk/papers/factor/node3.html`
16. Sakurai, T., Futamura, Y., Imakura, A., Imamura, T.: Scalable Eigen-Analysis Engine for Large-Scale Eigenvalue Problems, pp. 37–57. Springer Singapore, Singapore (2019). `https://doi.org/10.1007/978-981-13-1924-2_3`
17. Trefethen, L.N., Bau, D.: Numerical Linear Algebra. Society for Industrial and Applied Mathematics, Philadelphia (1997)
18. Wilkinson, J.H.: Householder's method for symmetric matrices. Numerische Mathematik **4**(1), 354–361 (1962). `https://doi.org/10.1007/BF01386332`