Computing Project E: Surveying using stars

BGN: 6997V

May 2022

Abstract

The CHARA 6-telescope array uses techniques of optical interferometry to improve angular resolution. This project looks at large data sets of measurements for the array and uses the assumption that the path lengths have been equalised to form a matrix equation. Linear algebra can therefore be used to solve for the positions of the telescopes. A lower bound on the position error was set at $680\mu m$ which was the average drift of fixed delay 'POP' settings between adjacent nights. The chance of there being significant movement increased with as nights became further apart within the same year. Though some positions in 2019 were not significantly different to some points in 2012 suggesting that there is some kind of oscillations back and forth. The baselines of the telescopes were seen to have drifted 3-7cm when compared to the initial calibration of the array in 2005. (total words: 2991)

1 Introduction

The angular resolution of a telescope is the smallest angle between objects that can be visibly resolved (the smaller the better). Angular resolution of a single telescope $\propto D$ where D is the diameter. It is possible to find the required resolution and therefore telescope diameter if it is known how far away an object is and how large the features you care about resolving are. Often, the required diameter is far too large both practically and economically, this is where telescope arrays become useful. When using two telescopes together, the angular resolution is set by the distance between the telescopes and not the sizes of the individual telescopes. This means two telescopes that are distance D apart have the same resolving capability (but not light collecting capability) as a larger continuous telescope of diameter D [CHARA, 2022]. To get the telescope joining benefits, the light in both branches must have travelled the same distance down to the tenths of wavelength to see interference fringes.

The first telescope to take advantage of this effect was the Michelson stellar interferometer in 1920 when it was set up on Mount Wilson, California to image the star Betelgeuse. Today the Centre for High Angular Resolution Astronomy (CHARA) array sits on Mount Wilson. CHARA consists of six 1m telescopes arranged in a Y-shape with baseline distance distances between telescopes ranging

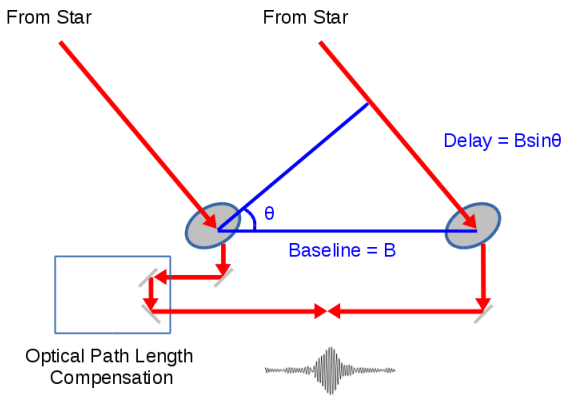

Figure 1: Shows how there is an external optical path difference introduced due to this different telescope positions. This is cancelled out by the internal optical path length compensation which equalises path length and allow for fringes to be found

from 34m to 330m. It is possible to use measurements from the CHARA array to find the positions of the telescopes relative to each other and find whether there has been significant movement of the telescopes between different dates.

At CHARA, the light from each telescope is sent down vacuum tubes to the central Beam Synthesis Facility (BSF), located at the centre of the Y shape, where beams are combined. Before reaching the telescope there is generally a path difference introduced. For example, in figure 1, light travels less distance to collector 1 than it does to collector 2. This is the external delay, it is the inner product of the vector joining the telescopes (baseline) and the unit vector of the telescopes to the star (i.e. $\hat{\mathbf{S}} \cdot \mathbf{B}_{1,2}$ where $\mathbf{B}_{1,2} = \mathbf{r}_1 - \mathbf{r}_2$). The direction of the star is the same for all collectors as the star is so far away.

There are also internal delays: the distances to the BSF the "constant term", the addition of discrete amount of optical distance using fixed mirrors delays (Pipes of Pans or POP) and the use of moving carts that are computer driven to equalise the path length as the stars move through the sky. The cart position is given to micron precision via the use of lasers. The path lengths are assumed to be equalised when the fringe visibility is maximised with respect to the moving carts. The two delay types allow for the construction of 1. The first term is the external path difference and the second term is the internal path difference.

$$\text{optical path difference} = \hat{\boldsymbol{S}} \cdot (\boldsymbol{r}_1 - \boldsymbol{r}_2) + d_1 - d_2 \tag{1}$$

For any measurement in the data, it is assumed that cart has moved to ensure the optical path difference vanishes. Therefore, if the internal path difference is split up into its POP and Cart components (as $d_i = POP_i + CART_i$) equation 1 can be rearranged to get equation 2.

$$\mathrm{CART}_2 - \mathrm{CART}_1 = \hat{\boldsymbol{S}} \cdot (\boldsymbol{r}_1 - \boldsymbol{r}_2) + \mathrm{POP}_1 - \mathrm{POP}_2 \tag{2}$$

Equation 2 can be formed for each measurement, as it is a series of linear equations the matrix equation 3 can be formed. y is a vector of the difference in cart positions for each measurement. X is known as the design matrix and contains the information about the each measurement, such as which telescopes/POP settings that were used and the direction of the star. β is the vector of model parameters, these are the position vectors of each telescopes and values related to the amount of delay a POP setting introduces. The model parameters are the unknowns of the problem.

$$y = X\beta \tag{3}$$

2 Mathematical background

2.1 Motivation for the Moore-Penrose inverse

For large data sets, he matrix equation $y = X\beta$ often has more measurements/equations than model parameters/unknowns, i.e. the design matrix is "skinny". These problems are known as overdetermined.

The model parameter space of the overdetermined system in matrix equation 5 can be seen in figure 2, where there are 3 equations for 2 unknowns. Figure 2 shows that there is not a single point in parameter space that satisfies all three equations, i.e $y - X\beta \neq 0$ for all β. This is equivalent to saying no inverse of X exist (inverses only exist for squares matrices), if one did exist $X^{-1}y = \beta$ and so $y - X(X^{-1}y) = 0$. $y - X\beta$ is the residual vector and is the difference between the data y and the model prediction $X\beta$.

The "best-fit" solution is found via minimising the 2-norm of the residuals, $\|y - Xb\|_2$, with respect to β. The minimum can be found by assuming a minimum norm value at $\hat{b}$ and taking small perturbations around this point (see appendix A). This gives the solution $X^T X\hat{b} = A^T y$ (this is known as the normal matrix equation) which can rearranged to give the expression an $\hat{b}$, as seen in equation 4. A^+ is known as the Moore-Penrose inverse or pseudoinverse and is a way of generalising the matrix inverse for non-square matrices. It can be used to find $\hat{b}$.

By being the solution to minimising the 2-norm, the pseudoinverse is the solution that minimises the sum of squares of the residuals and is know as ordinary least-squares (OLS). OLS is preferred over other loss functions as it maximises likelihood, though it is very sensitive to outliers so cleaning data is especially important. In the example of the 3×2 overdetermined matrix, the pseudoinverse gives the point, $\hat{b}$, which is the OLS from all the solution points and is shown as red dot on the graph. In this project, the Moore-Penrose pseudoinverse was used to find the model parameters.

$$\hat{\beta} = (A^T A)^{-1} A^T y = A^+ y \tag{4}$$

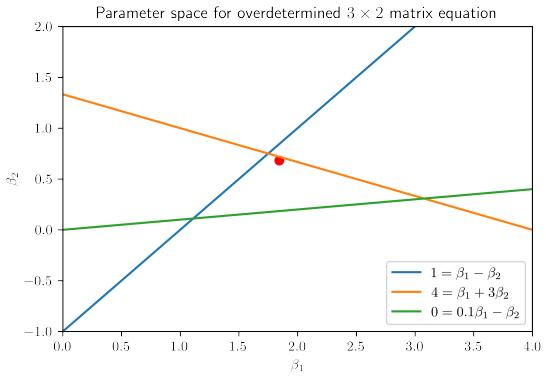

Figure 2: Plotting equations from a 3×2 matrix showing that there is no exact solution to the equation i.e. the matrix cannot be inverted

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ 1 & 3 \\ 0.1 & -1 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} \tag{5}$$

2.2 Singular Value Decomposition

$$X = U\Sigma V^T \tag{6}$$

The Singular Value Decomposition (SVD) is the generalisation of the eigen-decomposition, for non-square matrices. Equation 6 shows the expression for the decomposition for X, a $N \times M$ real matrix ($N > M$). U and V are $N \times N$ and $M \times M$ orthogonal matrices respectively, meaning they have orthonormal columns (so V^T has orthonormal rows). Σ is populated diagonally by the singular values w_i, this allows the SVD to be presented as a linear combination of matrices weighted by the singular values (equation 7).

$$X = \sum_i w_i u_i v_i^T \tag{7}$$

Finding the pseudoinverse via the normal matrix equation requires the inversion of $A^T A$ which is not very efficient. Instead, the pseudoinverse can be found via elements of the SVD as seen in equation 8 (the full derivation is in the appendix A). This is the method that the computer algorithms use.

$$A^+ = V\Sigma^{-1}U^T = V[\text{diag}(1/w_i)]U^T \tag{8}$$

2.3 SVD errors

$$\chi^2 = \sum_i \left(\frac{y_i - X_{ij}\beta_j}{\sigma_d} \right)^2 = \sum_i \left(\frac{i\text{th residual}}{\sigma_d} \right)^2 \tag{9}$$

Errors can be calculated for the model parameters with respect to some given data using the SVD. The χ^2 value of a model and some data is given by equation 9. For a linear best-fit model this follows the chi-squared distribution. For a data set of N rows and M columns there are $N - M$ degrees of freedom, this is roughly equal to the chi-squared distribution for large amount of data such that $\chi^2 \sim N - M$ [Buscher, 2021]. σ_d is the root mean square errors for the y values, it is reasonable to assume that this is the same for all measurements. Therefore σ_d can be found by rearrangement. Equation 10 then gives the error on each model parameter [Press et al., 2007]. When there is a zero singular value $1/w_i$ is set to 0.

$$\sigma^2(\beta_j) = \sum_i \left(\frac{Vji}{w_i} \right)^2 \sigma_d^2 \tag{10}$$

Each zero singular value corresponds to a degeneracy, in this way the SVD very quickly and robustly tells you whether the problem is complete. The row of the V^T matrix corresponding to the zero singular value occupies the null space of the matrix (i.e. $Xv_i = 0$). This can be seen using 7, as $Xv_j = \sum_i w_i u_i v_i^T v_j = \sum_i w_i u_i \delta_{ij} = 0$ if $w_j = 0$ as v_j are orthonormal. This means the data is not sensitive to that combination of model parameters. Equation 11 shows that if you add on a scalar multiple of one of the degenerate v_i vectors, that the output of the model is the same. This means there are many parameter values to which the data will give the same output i.e. the parameters are not well defined. The number of degeneracies is equal to the number of ranks deficient the design matrix is. The rank of a skinny matrix is the dimension of output space, for full rank the output space is the same dimension as the number of model parameters meaning the model parameters can be uniquely determined. If rank deficient, the dimensions are absorbed into the null space meaning that there are no longer enough dimensions in output space to uniquely determine the model parameters.

$$X(\beta + \beta') = X\beta + X(\gamma v_i) = X\beta + 0 \tag{11}$$

3 Implementation Performance

3.1 Python for data science

Python is much more popular than C++ for data science as it is possible to achieve C-like speeds by taking advantage of vectorisation and functions that are built on compiled language such as C++ and FORTRAN.

For example, the most computationally intensive functions in this task were the SVD and the pseudoinverse. Both of these functions in `numpy` are built on

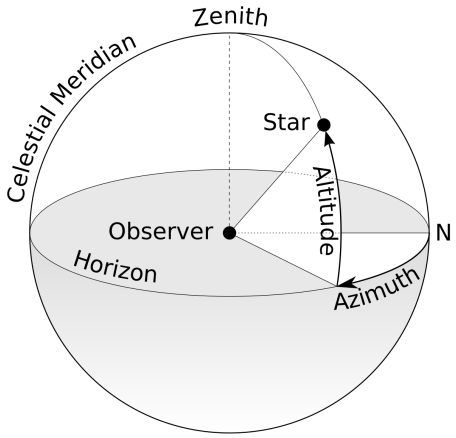

Figure 3: Horizontal coordinates: azimuth starts at North and goes through 2π and elevation which starts at the horizon and goes up to $\pi/2$. This uniquely determines the upper half of a sphere i.e. the bits of the sky that it is possible to see

top of the a linear algebra package in FORTRAN. Performance of `np.linalg.svd()` on $2034x35$ design matrix of 7th April dataset was tested using `\%timeit`. The time taken per decomposition was 224 ± 7ms therefore performance was not an issue for the project.

The main functions were written in scripts.py and were imported to Jupyter notebooks for different tasks e.g. data pre-processing, visualisation etc. Jupyter notebooks (file extension .ipynb) are a great tool for data science projects as they allow for the running of isolated 'cells' of code without rerunning all functions.

3.2 Data preprocessing

There were five datasets, four from individual days in 2019 and one which spanned the whole of 2012, albeit with far fewer measurements per day. Comma separated variable (csv) files were loaded in python as DataFrame objects using `pandas`, table 3.2 shows the columns in the data.

utc	star	azimuth	elevation	tel_1	tel_2	pop_1	pop_2	cart_1	cart_2

Table 1: The columns for the data provided. The values of tel and pop were categorical, the values of azimuth, elevation and cart were floats

A few changes needed to be made to the data before analysis. The star position data for all measurements was given in horizontal coordinates (see figure 3) in which the azimuth starts at North, increasing clockwise, taking values from 0 to 2π. The elevation is the angle of the star to the horizon taking values from 0 to $\pi/2$. The horizontal and elevation values were converted to

radians as this is the default for `numpy` trigonometry. For the 2019 data the elevation and azimuth columns were the wrong way around, this was known as some elevation values were $> \pi/2$. For the 2012 dataset a `dayofyear` column was also added for filtering.

3.3 Creating the design matrix

3.3.1 Design matrix shape

To represent categorical data in a matrix a model parameter is needed for each category. With six telescopes whose positions can only be known relative to each other there are 15 model parameters for telescope positions (x, y, z) as one telescope, which can be arbitrarily picked (E1), is placed at the origin. There can be a minimum of six POP parameters when each telescope only used one POP settings and up to 180 when all six telescopes use all 30 available POP settings. Note 'PXBY' on one telescope is not the same when used on another telescope. Once the number of unique POP settings is found an empty design matrix can be initialised with shape (data points, $15 +$ unique POPs).

3.3.2 One-hot encoding

$$y_i = X_{ij}\beta_j = \hat{\mathbf{S}}_i \cdot (\boldsymbol{r}_{i,1} - \boldsymbol{r}_{i,2}) + \mathrm{POP}_{i,1} - \mathrm{POP}_{i,2} \tag{12}$$

When the inner product of the ith row of the design matrix is taken with the model parameter vector β the right hand side of equation 2 must be recovered. For this reason the only non-zero elements in the matrix are for the columns corresponding to the telescopes and POP settings present. For POP settings this means the column for the position 1 POP has +1 and position 2 has -1. The telescope requires the star vector to dot product the position vectors. Therefore, for three components are each telescope present should have $+(S_x, S_y, S_z)$ for 1 position and $-(S_x, S_y, S_z)$ for 2 position. The star vector can be found by converting the horizontal coordinates to Cartesian coordinates using equation 13.

Populating the telescope and POP settings makes use of vectorisation by populating whole columns at a time instead of individual elements. Finally, the difference in the cart values (y) can be calculated easily, again using the vectorisation in `pandas` columns.

$$\hat{\boldsymbol{S}}(\theta, \phi) = (\cos\theta\cos\phi, \cos\theta\sin\phi, \sin\theta) \tag{13}$$

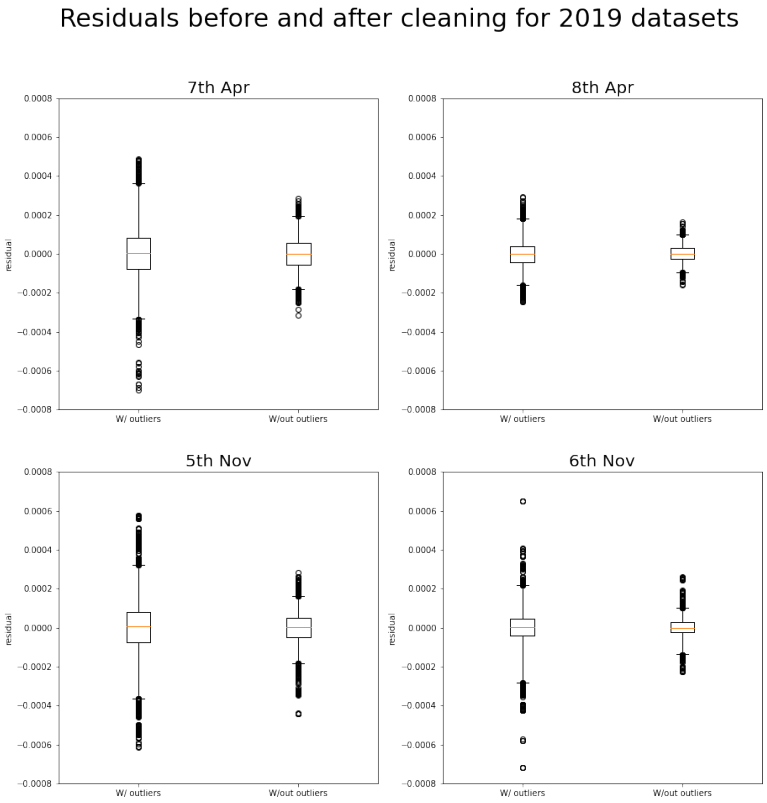

Figure 4: removal of $\pm 1.5\sigma$ residual outliers for 2019 dataset

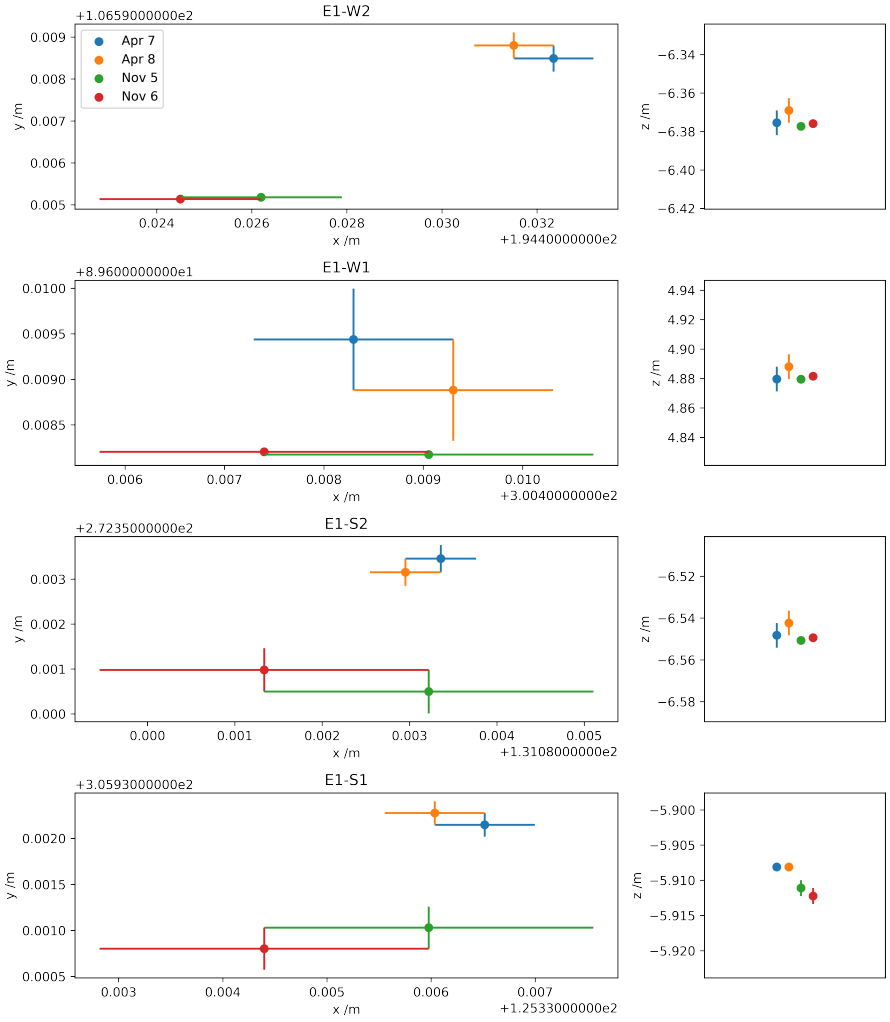

Figure 5: the positions of the telescopes in April vs. November taking the uncertainty to be the difference in value on two consecutive nights

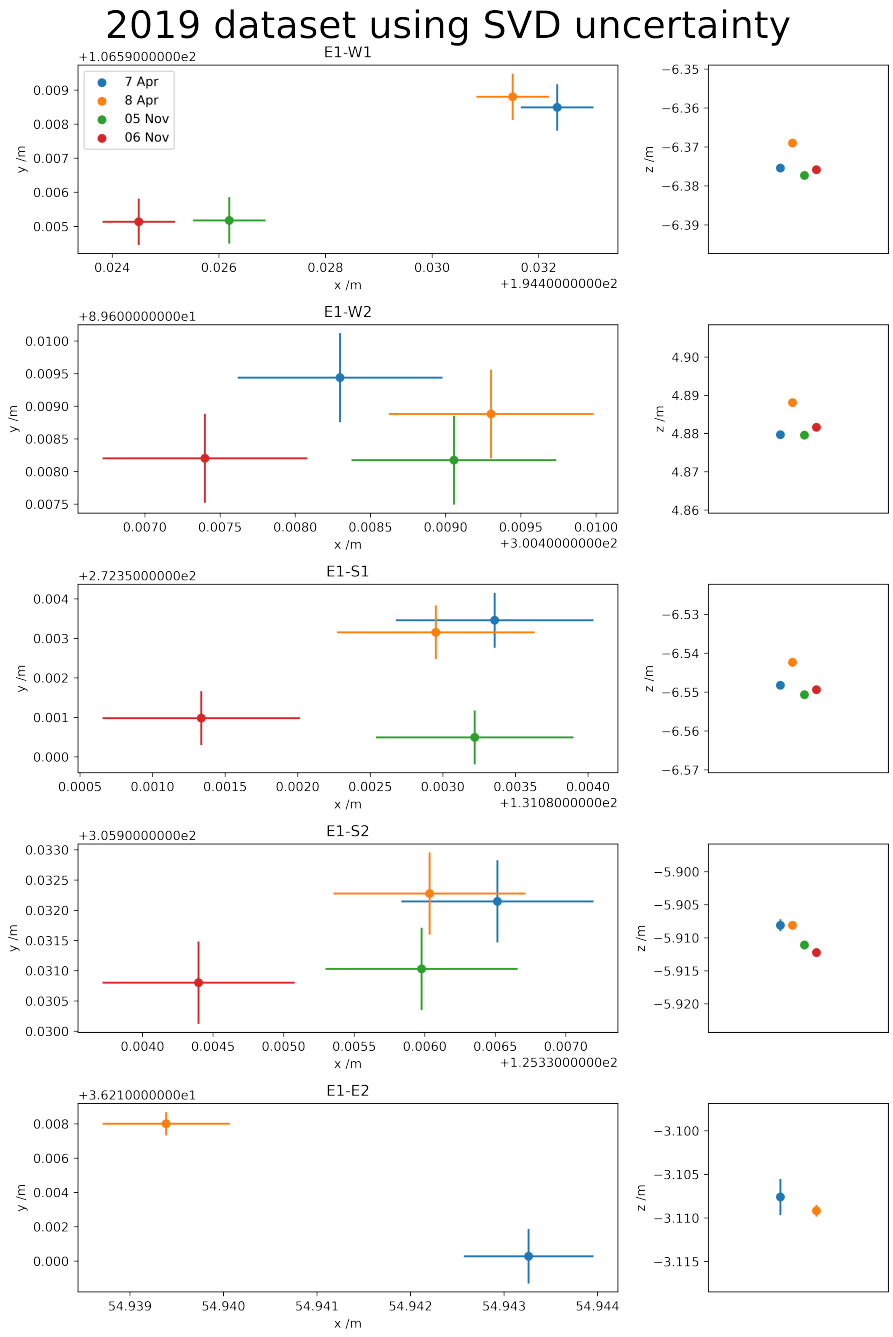

Figure 6: the positions of the telescopes in April vs. November taking the uncertainty to be the SVD error formula or 680μm due to the POP drift, whichever is largest

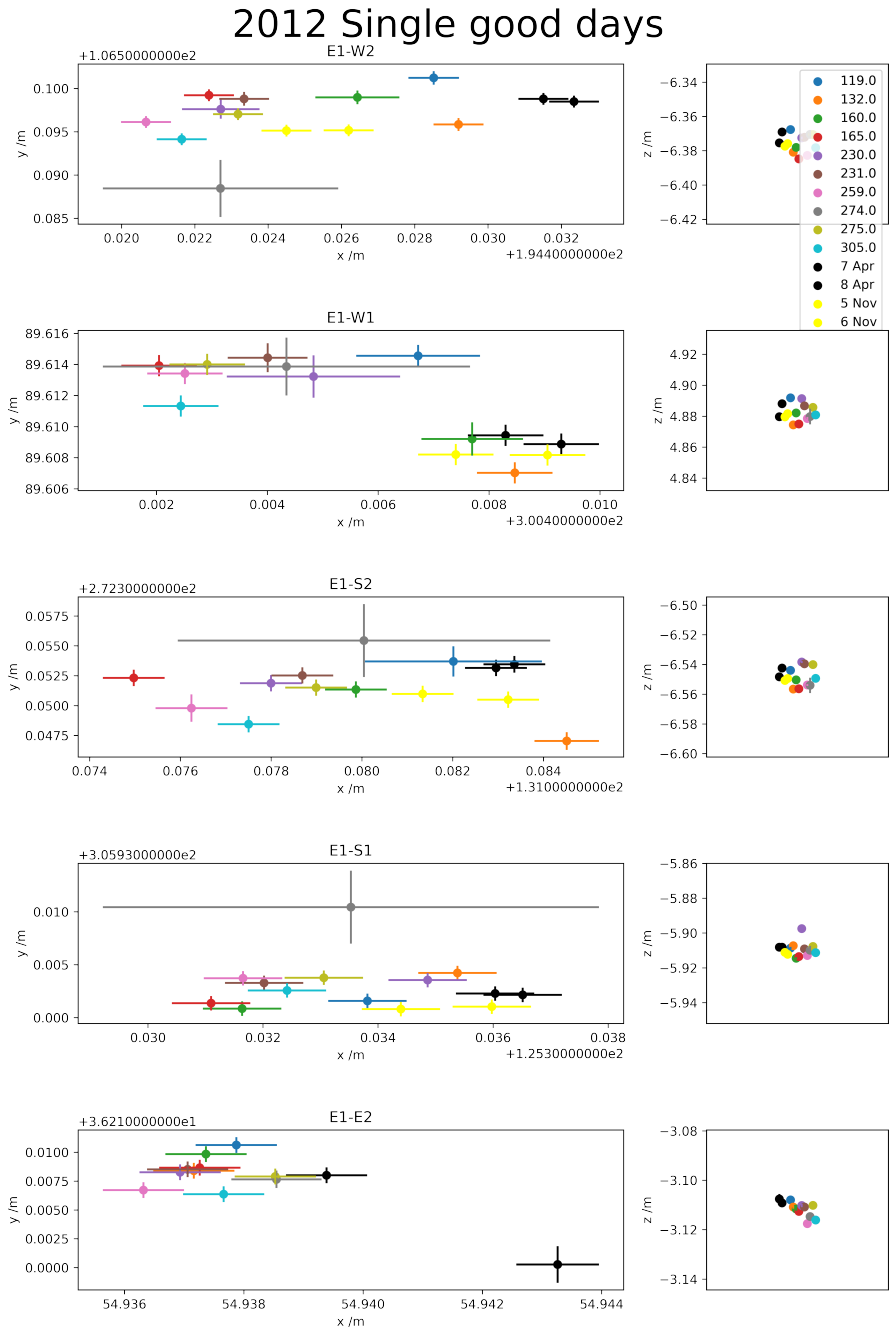

Figure 7: The baseline estimates for the days in 2012 with only a single degeneracy

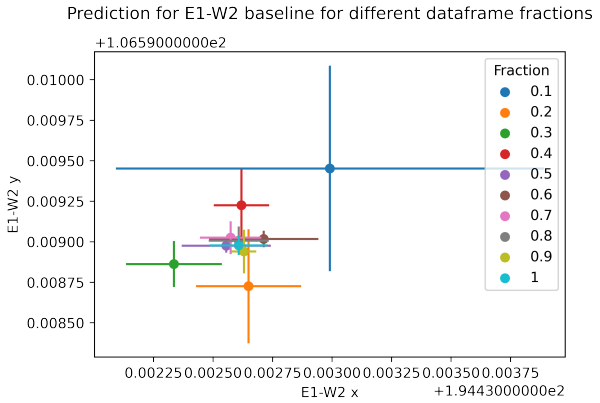

Figure 8: Estimate and error for (x, y) of E1-W2 baseline for fractions of April 7th dataset with only one (POP) degeneracy. This shows that although the position estimate is worse the SVD error increases to account for this

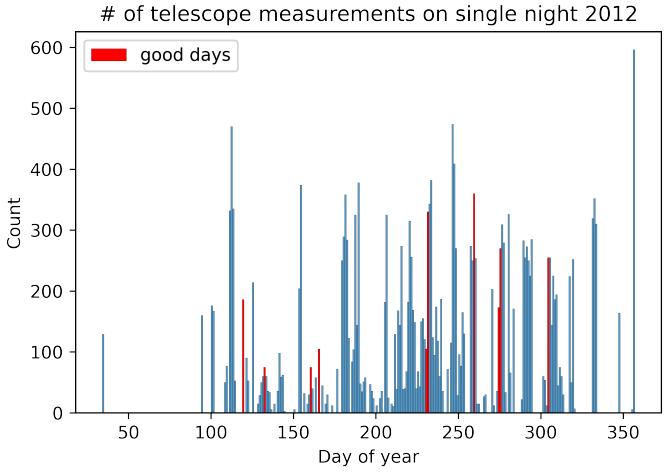

Figure 9: The distribution of measurements for 2012 is weighted towards the end of the year. The red lines denote individual days whose design matrix only has one degeneracy (POP degeneracy) i.e. the days capable of making a good estimate

Telescopes	East (m)	North (m)	Height (m)	Baseline (m)
S2-S1	-5.747	33.579	0.640	34.073
E2-E1	-54.943	-36.210	3.108	65.876
W2-W1	105.976	-16.989	11.255	107.918
W2-E2	-139.489	-70.388	3.268	156.277
W2-S2	-63.349	165.755	-0.173	177.448
W2-S1	-69.096	199.334	0.467	210.970
W2-E1	-194.432	-106.598	6.375	221.828
E2-S2	76.140	236.143	-3.441	248.139
W1-S2	-169.325	182.744	-11.428	249.393
W1-E2	-245.465	-53.399	-7.987	251.333
W1-S1	-175.072	216.323	-10.788	278.500
E2-S1	70.393	269.722	-2.801	278.77
E1-S2	131.083	272.353	-6.548	302.328
W1-E1	-300.408	-89.609	-4.88	313.526
E1-S1	125.337	305.932	-5.908	330.664

Table 2: Baselines values for April 7th 2019. These compare well to the 2005 calibration showing the model was working [Ten Brummelaar et al., 2005]

4 Results & Discussion

After having formed the design matrix and the y vector, the least-squares model parameters can be found via the pseudoinverse (equation 4) - this is easy and quick in python using `numpy.linalg.pinv`. The residuals ($r = y - X\hat{\beta}$) of the model were then plotted and the outliers $\pm 1.5\sigma$ were removed (where σ is the standard deviation of the residuals). This threshold was chosen as empirically it was the amount such that the residuals from the cleaned data were bounded by the 2.5th and 97.5th percentile residuals in the uncleaned data. This can be seen for the 2019 datasets in figure 4.

The calculated baselines for April 7th data set are in seen table 4. It was known that the model was working as the results were comparable to the calibration results from 2005, the largest differences compared to 2005 were for the E1 baselines [Ten Brummelaar et al., 2005].

As discussed, the degeneracy is related to how many ranks deficient a matrix is. The matrices for this data is always deficient at least one rank as all the POP settings can always be offset by some constant amount. The vector corresponding to this degeneracy consists of 0s for the baseline parameters and 1s for the POP parameters, this is in the null space of X. One POP setting is always +1 and the other is always -1, If all POP columns are multiplied by 1 the output vector is $+1 - 1 = 0$. This means means that it is not possible to find absolute values for POP settings and that only difference between POP settings can be found. This is not true for the telescope positions are there is the star vector which is always changing.

4.1 2019

This process of finding the baselines was repeated for the all four data sets in 2019. There were two main methods of finding the uncertainty. The simplest was to take the difference between two nights as the uncertainty this is shown in figure 5. The second was to use the SVD error which is calculated on the assumption that the model parameters are constant throughout a single night. Though it is known that the mechanical drifts in the POP and "constant term" are on the order of 100s microns on a given night. This naive assumption led to SVD uncertainties 10s of times smaller than the difference between adjacent night.

The last measurement on April 7th was taken at 1300UTC, this was 10.5 hours after the first measurement of that day and 14 hours before the first measurement for the following day, this was similar for all 2019 data sets. As these two lengths of time are similar, the amount the POP setting drift between two adjacent nights should be similar to the amount of POP drift within a given day. As discussed, POP values are not absolute due to the degeneracy therefore the drift must be measured through the drift of the difference between two POP settings. This was done for four POP setting pairs S2 Apr P2B4-P3B4, W1 Apr P1B3-P2B3, E1 Nov P1B1-P5B1 and E1 April P1B1-P3B1. The average of these was $\sim 680\mu m$. As the telescope baselines are related to the POP settings by equation 2, the lower bound of the baseline setting uncertainty should be at least the lower bound of the POP setting uncertainty, given errors accumulate in quadrature. Therefore, this was set as the lower bound of the telescope uncertainty via `np.maximum(error, 0.00068)`. The SVD errors were then plotted as seen in figure 6.

For the 2019 data sets, this lower bound was larger than the error from the SVD formula for most parameters. These new errors showed that for April every baseline apart from E1-S1 had significant movement, though this was in the z direction for the first three baselines and in the (x, y) direction for the last baseline. The November baselines didn't change significantly the z direction (the error bars are hidden by the scatter size) but all four baselines changed in the x direction. There was also lots of significant movement between the two times of the year. (note: there were no E2 measurements for Nov 2019).

4.2 2012

The situation in 2012 was slightly different having only 100s instead of 1000s of measurements per day (see figure 9). This led to a few questions as to how this may impacts results. Using random fractions of the April 7th 2019 data set (with only one degeneracy from the POP), the position and error estimate were calculated for E1-W2, as seen in figure 8. For smaller fractions the estimate is worse, but crucially the SVD error grew to account for this inaccuracy. This meant that estimates with small amounts of data are still meaningful. `find_good_days()` function looked for days in 2012 which only has a single degeneracy in the design matrix. These 'good days' are the red

lines on figure 9, are not the most sampled days, for example day 132 only has 75 measurements. Figure 7 shows the results of the estimates for the 'good days' in 2012. The results show that there has been significant movement both within 2012 between 2012 and 2019 for many of the telescopes. There were significant movements for many of the telescopes baselines but the $E1 - W1$ baseline was most pronounced where throughout ~ 180 in 2019 the x and y components both decreased by $\sim 6mm$. The $E1 - E2$ baseline was least pronounced in which all the values stayed within $\sim 5mm$ in the y direction and $\sim 3mm$ in the x direction. The z component tended to increase throughout the year a few mm for all baselines. Day 274 has unusually large errors.

5 Conclusion

Overall, the formulation of the matrix equation for a series of measurements allows for many linear algebra techniques to be used to model the data. This project has shown that whilst these techniques are incredibly useful it is important to keep track of the assumptions made (e.g. that model parameters are constant). This assumption was accounted for by putting in a lower bound error for the baseline position. It was shown that movement between nights is more common the further away the two nights are from each other. In the 2012 data set, it has been shown that some telescopes move more than other and that a general patterns are followed for baselines where there is significant movement.

Figure 7 showed that often time there baselines from 2019 will be within error bars of values from 2012 hinting at seasonal/oscillatory movement instead of movement in a single direction. Though it is hard to investigate the data seasonally as there is much more data towards the end of the year. No obvious signs were seen for the Brawley earthquake in August 2012 [Hauksson et al., 2013]. A systematic approach to analysing the degeneracies would help to understand the combinations of parameters that lie in the null space. This would allow for better predictions of which degeneracies are occurring.

6 Appendix: Pseudoinverse as minimisation of the 2-norm

$y = X\beta$ only has solutions for square matrices. Though we can minimise wrt 2-norm

$$min||A\hat{b} - y||_2$$

$$r(\hat{b}) = ||A\hat{b} - y||_2^2 = (A\hat{b} - y)^T(A\hat{b} - y)$$

$$= \hat{b}^T A^T A\hat{b} - \hat{b}^T A^T y - y^T A\hat{b} + y^T y$$

$$\frac{dr(\hat{b} + \epsilon v)}{d\epsilon} = \frac{d((\hat{b} + \epsilon v)^T A^T A(\hat{b} + \epsilon v) - (\hat{b} + \epsilon v)^T A^T y - y^T A(\hat{b} + \epsilon v) + y^T y)}{d\epsilon}$$

$$= \hat{b}A^T Av + v^T A^T A\hat{b} - v^T A^T y - y^T Av$$

$$= v^T(\hat{b}A^T Av - A^T y) + (v^T(A^T A\hat{b} - A^T y))^T$$

Therefore we see that the solution to the least squares problem is

$$A^T A\hat{b} = A^T y$$

So the pseudoinverse (the matrix that best does the job of an inverse) is:

$$\hat{b} = (A^T A)^{-1} A^T y = A^+ y$$

$$A^+ = (A^T A)^{-1} A^T$$

Now using the expression for the SVD

$$A^+ = (A^T A)^{-1} A^T$$

$$= (V\Sigma U^T U\Sigma V^T)^{-1} V\Sigma U^T$$

$$= (V\Sigma^2 V^T)^{-1} V\Sigma U^T$$

$$(V^T)^{-1}\Sigma^{-2} V^{-1} V\Sigma U^T$$

$$= V\Sigma^{-2}\Sigma U^T$$

$$A^+ = V\Sigma^{-1} U^T$$

7 Main numerical code

```
import numpy as np
import pandas as pd

def create_design_mat(df):
    """

    Creates design matrix from preprocessed dataframe

    Parameters
    ----------
    df : pandas.DataFrame
        dataframe with measurements


    Returns
    -------
    design_mat: numpy.ndarray
        design matrix for the dataframe
    pinv: numpy.ndarray
        Moore-Penrose pseudoinverse found using ‘np.linalg.pinv‘
    y: numpy.ndarray
        Output vector i.e. lists of difference in carts
    beta: numpy.ndarray
        model parameters vector found using ‘pinv @ y‘
        baseline components zero’d on E1 telescope
        15 baselines W2, W1, S2, S1, E2 followed by POP settings

    """
    # ea. row must contain (x, y, z) for ea. telescope (except E1, the zero point)
    # and all 30 POP settings for ea. telescope
    telescopes = ["E1", "W2", "W1", "S2", "S1", "E2"]

    # find the star vectors
    theta = df.elevation
    phi = df.azimuth
    S = np.array(
        [np.cos(theta) * np.sin(phi), np.cos(theta) * np.cos(phi), np.sin(theta)]
    ).T
    k = 0

    tot_unique_pops = []

    width = 15

    for telescope in telescopes:

        pop_tel_1 = df[df.tel_1 == telescope].pop_1
        pop_tel_2 = df[df.tel_2 == telescope].pop_2
        tel_1_unique_pops = np.unique(pop_tel_2)
        tel_2_unique_pops = np.unique(pop_tel_1)
        tel_unique_pops = np.union1d(tel_1_unique_pops, tel_2_unique_pops)
        tot_unique_pops.append(tel_unique_pops)
```

```
        width += len(tel_unique_pops)

    if tot_width:
        width = 195

    design_mat = np.zeros((len(df), width))

    for i, telescope in enumerate(telescopes):

        # for ea. new telescope must jump 3 places for (x, y, z)

        if telescope != "E1":  # keep 'E1' as the zero point

            design_mat[:, 3 * i - 3 : 3 + 3 * i - 3] += S * np.
    where(
                df["tel_1"] == telescope, 1, 0
            ).reshape(-1, 1)
            design_mat[:, 3 * i - 3 : 3 + 3 * i - 3] -= S * np.
    where(
                df["tel_2"] == telescope, 1, 0
            ).reshape(-1, 1)

        for pop in tot_unique_pops[i]:

            # not working because E1 isn't in the list of
    telescopes
            design_mat[:, 15 + k] += np.where(
                (df["pop_1"] == pop) & (df["tel_1"] == telescope),
    1, 0
            )  # add when it is tel_1
            design_mat[:, 15 + k] -= np.where(
                (df["pop_2"] == pop) & (df["tel_2"] == telescope),
    1, 0
            )  # subtract when it is the tel_2

            k += 1

    y = df["cart_2"].values - df["cart_1"].values

    pinv = np.linalg.pinv(design_mat)

    beta = pinv @ y

    return design_mat, pinv, y, beta

def svd_uncertainty(design_mat, y, beta):
    """

    Calculate uncertainty derived from singular value decomposition
     removing zero singular value in the process

    Parameters
    ----------
    design_mat: numpy.ndarray
        design matrix for the dataframe
    pinv: numpy.ndarray
        Moore-Penrose pseudoinverse found using `np.linalg.pinv`
```

```
    beta: numpy.ndarray
        model parameters vector found using ‘pinv @ y‘
        baseline components zero’d on E1 telescope
        15 baselines W2, W1, S2, S1, E2 followed by POP settings

    Returns
    -------
    sigma: numpy.ndarray
        array of standard deviations for the model parameters

    """

    residuals = y - (design_mat @ beta)

    sigma_d_squared = np.sum((residuals)**2) / (len(y) - len(beta))

    _, w, Vt = np.linalg.svd(design_mat)

    # find  the machine precision of the data type being used
    eps = np.finfo(type(w[0])).eps
    N = len(y)

    # this gives the cutoff after which point the singular values
    can be considered zero
    small = w[0] * N * eps

    w = w[w > small]

    # remove eigenvetors orresponding to the small singular values
    Vt = Vt[:, :len(w)]

    sigma = np.sqrt(np.sum((Vt/w)**2, axis=1) * sigma_d_squared)

    # ensure that error is atleast as large as the POP drift
    return np.maximum(sigma, 0.0006807973948568247)

def pre_process(file_name):
    """

    processes by adding in extra datetime columns, swapping the
    elevation and azimuth columns where necessary and converting
    the elevation and azimuth to radians. After this the file is
    saved back to where it came from.

    Parameters
    ----------
    file_name: str
        where the file is located relative to current directory

    Returns
    ---------
    None
    """

    # load the csv into a DataFrame
    df = pd.read_csv(f’data/{file_name}.csv’)

```

```
    # create seperate columns for year, month, dat and time
    dates = pd.to_datetime(df.utc)
    df['year'] = [date.year for date in dates]
    df['month'] = [date.month for date in dates]
    df['day'] = [date.day for date in dates]
    df['month'] = [date.month for date in dates]
    df['dayofyear'] = [date.day_of_year for date in dates]


    # need to swap the elevation and azimuth for the 2019 date sets
    if file_name[:4] == '2019':
        df = df.rename(columns={"elevation": "azimuth", "azimuth":
    "elevation"})

    # change to radians
    df["azimuth"] = df["azimuth"] * 2 * np.pi / 360
    df["elevation"] = df["elevation"] * 2 * np.pi / 360

    df.to_csv(f'data/{file_name}.csv', index=False)

def remove_outliers(df, n_sigma=2):
    """

    Removes outliers residual of the model +- n_sigma standard
    deviations from zero

    Parameters
    ----------
    df: pandas.DataFrame
        dataframe to be cleaned
    n_sigma:
        number of std devs where the cutoff stars (default=2)

    Returns
    -------
    df_new: pandas.DataFrame
        The dataframe with the outliers removed

    """

    design_mat, pinv, y, beta = create_design_mat(df)

    # find residual vector
    residual = y - design_mat @ beta

    # set threshold via standard deviation
    threshold = n_sigma * np.std(residual)

    # set index value of outliers to 1 and remove these
    df_new = df[np.where(abs(residual) > threshold, 1, 0) != 1]

    return df_new
```

Listing 1: main numerical functions

References

[Buscher, 2021] Buscher, D. (2021). *Part II Computational Physics Lectures*.

[CHARA, 2022] CHARA (2022). Basics of interferometry. *https://web.archive.org/web/20220201214654/https://chara.gsu.edu/public/basics-of-interferometry*.

[Hauksson et al., 2013] Hauksson, E., Stock, J., Bilham, R., Boese, M., Chen, X., Fielding, E. J., Galetzka, J., Hudnut, K. W., Hutton, K., Jones, L. M., et al. (2013). Report on the august 2012 brawley earthquake swarm in imperial valley, southern california. *Seismological Research Letters*, 84(2):177–189.

[Press et al., 2007] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

[Ten Brummelaar et al., 2005] Ten Brummelaar, T., McAlister, H., Ridgway, S., Bagnuolo Jr, W., Turner, N., Sturmann, L., Sturmann, J., Berger, D., Ogden, C., Cadman, R., et al. (2005). First results from the chara array. ii. a description of the instrument. *The Astrophysical Journal*, 628(1):453.