47. The iterative solution: implementation

FORMULATING THE MATHEMATICAL description of the astrometric solution was one part of the challenge for the Gaia astrometric data processing. But its actual computer implementation was quite another.

As I have described here separately, the problem is formidable: both in terms of the amount of data to be treated (the total number of unknowns is around one billion, and the solution treats some 100 billion observations extracted from some 100 000 Gbytes of raw satellite data), and in terms of the way in which the iterative solution has to be executed, with its four 'blocks' (of source, attitude, calibration, and global parameters) being evaluated in a cyclic sequence until convergence.

The implementation and the data management required to make the Astrometric Global Iterative Solution (AGIS) function has been absolutely crucial to the ultimate goal, and success, of Gaia.

A KEY FIGURE in this task was William O'Mullane. While Gaia was still in its study phase, in the 1990s, O'Mullane conceived and set up the framework for running the set of iterative equations in a distributed manner. The prototype was based on the Hipparcos satellite data, which employed a similar sky-scanning.

This approach was novel in using the relatively new Java language, and in exploiting the message passing and networking capabilities to manage multiple distributed computing nodes.

Initially using an object oriented database, this was later largely replaced by more traditional solutions, although AGIS ultimately used the high-performance matrix database InterSystems Caché (which itself now uses the Gaia data as a key example on its home page!).

In 2005, O'Mullane returned to ESA with the task of developing an implementation of the global astrometric solution at scale. This led to the building up of a team of a dozen computer scientists at ESA's European Space Astronomy Centre (ESAC), outside Madrid.

Using 'agile' programming (and, in particular, 'extreme programming', XP), the group developed a system capable of producing the core astrometric solution underpinning the successive catalogue releases.

CENTRAL TO THE technical implementation was the concept of a 'data train', which performs a 'sweep' through the observational data, drawing the various algorithms behind it. The train 'picks up' an object, and passes it to all algorithms in a data-driven manner. Algorithms are called – and objects are then passed to them – allowing the system to access data efficiently, while insulating the algorithmic code from storage aspects.

As an example, while early implementations of AGIS required four passes through the billions of observations to execute the four processing blocks, a later implementation executed a single 'outer iteration' with just a single pass through the entire data set.

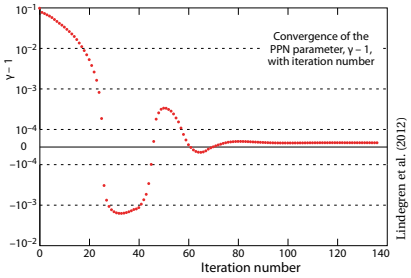
A critical constraint was to minimise disk access at all stages, also within each iteration. For example, holding the satellite attitude vectors for the entire mission in memory on each processing node was crucial, since practically every calculation used the attitude data. Likewise, calibration results from each previous iteration could also be stored in memory on each node.

Gaia processor, ESAC (Joeé Hernandez)

LET ME FIRST LOOK at a major demonstration solution run at ESAC on simulated data, and before the satellite launch, as described by Lindegren et al. (2012).

This used an IBM cluster with 14 nodes, each node having two processors with four cores each, corresponding to 112 CPUs in total. This configuration of 14 nodes was estimated to have a total performance of 0.65 teraflop/sec (0.65×10^{12} floating point operations per second). One iteration took about 1 hr (with typically 90% CPU occupancy). The total run time for the 135 iterations executed was nearly 6 days, corresponding to a total of about 3×10^{17} floating point operations.

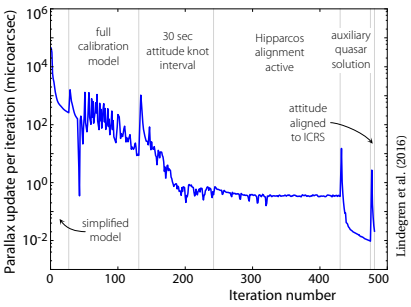
Scaled up to the projected 10^8 primary sources of a real AGIS run, this would amount to 1.5×10^{19} flop. Using a more conservative estimate of 5×10^{19} flop to account for additional features not included in that particular demonstration run, they predicted a requirement of some 60 days on a dedicated 10 teraflop/sec machine.



The figure above, from the same simulation exercise, shows how one of the 'global' terms, specifically the estimate of the parameter γ (in the PPN formulation of general relativity), eventually converges, from some initial 'assumed' value of 1.1, to a stable value close to 1.0, from around iteration 80 onwards.

THE FOLLOWING IS another example illustrating the convergence of AGIS, taken from the processing involved in the preparation of Gaia Data Release 1 (DR1, Lindegren et al. 2016). It shows how the parallax update (in microarcseconds) evolved with the iteration number. Different regimes of the underlying calibration model were adopted as the iterations proceeded.

Starting with parallax updates of around 10 milliarcsec for the initial iterations, they converged to values of around 1 microarcsec or less by around iteration 200.



THE ASTROMETRIC SOLUTION for Gaia Early Data Release 3 (EDR3) is described in detail by Lindegren et al. (2021). But I will focus here on some of the numbers related to the hardware, and to the execution times.

The astrometric solution for Gaia EDR3 was run using 32 nodes, each comprising 64 GB of RAM and 24 cores (which are Intel–Xeon CPU E5–2670 v3 running at 2.30 GHz). Together these provided a total theoretical performance of 30 teraflop/sec.

In terms of disk storage, 55 Tbytes was available for the solution for primary stars (around 40 Tbytes was used), and a further 20 Tbytes for the secondary stars.

In practice, data from the satellite is sorted before being ingested by AGIS. This AGIS-preprocessor reads and writes the data three times in order to have the main astrometric data sorted for every source. Then the data is grouped, typically in chunks of 1000 sources in the primary store and 10 000 sources (depending on the star density) in the secondary store. The AGIS-preprocessor takes about one week to run.

A total of 181 iterations were executed in generating EDR3 (as detailed in their Table 3). Each outer iteration of the 14 million primary sources (performing the source, attitude, calibration and global blocks) took 90 minutes. Although this has been further optimised (taking around 60 minutes) for the next major reduction, DR3, more primary stars will be used.

What has taken much time and resources is refining the calibration model to further improve the solution, correcting the biases observed in the preliminary runs. In practice, some 200 preliminary tests were scheduled before starting the final operational run.

THE PRIMARY STAR SOLUTION for EDR3 had to process about 6.5 billion CCD observations for the 14.3 million primary sources. The solution determined 71.5 million source parameters, together with 10.7 million attitude, 1.1 million calibration, and 2.0 million global parameters. The 'redundancy factor', being the mean number of observations per unknown, was about 76.

The secondary star solution processed nearly 78 billion field-of-view transits, generating converged solutions for 2.495 billion sources (of which 585 million were 5-parameter, 883 million 6-parameter, and 1.027 billion were 2-parameter solutions). Subsequently some of the 5- and 6-parameter solutions, and most of the two-parameter solutions, were removed because they failed to meet the rigorous acceptance criteria.

LOOKING BACK, this crucial step in the success of Gaia could easily have gone spectacularly wrong.

In the technology preparation phase in 1999, I identified this data organisation/analysis problem as one of the highest priority items for immediate further study, entering the Science Technology Document (Peacock & Scoon 1999) as Item G24. But a review panel rejected the funding request, with the one-line comment: *'Not a priority, just wait for commercial products to come'*.

It was a long process for me to reverse this uninformed dismissal of a most complex issue. After that, a 2-year industrial study, and a parallel academic study extending over several years, failed to progress this challenging implementation. Only bringing O'Mullane to ESA to work on the problem under my direct authority eventually led its successful implementation.