WCSLIB - its origins and development

ADASS XXXV software prize, invited talk

Mark Calabretta (mark@calabretta.id.au)
Australia Telescope National Facility, CSIRO (retired)
05 November 2025

Abstract

2025 marks thirty years since the first release of WCSLIB, a software library closely linked to FITSWCS, the FITS World Coordinate System standard. From the start, WCSLIB informed the development of FITSWCS and now provides one of several practical implementations. In this talk I will describe WCSLIB's origins, its close connection with FITSWCS, and major milestones in their evolution. I conclude with some general observations on software development as I have experienced it in relation to WCSLIB over the past decades.

1 Origins

WCSLIB is an implementation of the FITS World Coordinate System standard (FITSWCS). Its origins and evolution are deeply entwined with FITSWCS, not just in implementing the standard but also in informing and validating its development. As such, it fulfils the role of a reference implementation. Yet it is and has always been intended primarily as a software library for use in practical astronomical applications.

The FITSWCS standard as we know it today was born from AIPS++, now known as CASA, a software package intended as the successor to AIPS, the venerable Astronomical Image Processing System. The AIPS++ project was undertaken by an international consortium of radio astronomy institutions, including ATNF. Thus, I found myself as part of a team of programmers gathered at NRAO's headquarters in Charlottesville, Virginia, for the first six months of 1992 to begin development on the package. Stemming from a long-held interest in map projections, I volunteered to develop the image coordinate classes and was duly assigned the job.

At the time, the widely-used "AIPS convention", defined by the oft-cited AIPS memos 27 and 46 written by Eric Greisen (1983, 1986), specified how celestial coordinates were to be represented in FITS. It comprised eight map projections, of which four were in common use, with a formalism that did not handle oblique coordinate grids. I envisaged a wider range of projections for AIPS++ with general handling of obliquity, as elucidated in AIPS++ memo 107 (1992). However, I realised that such an extension would be futile if not accomodated by FITS.

By a quirk of fate, Eric had spent two years on sabbatical at ATNF in 1990 and 1991. Based at NRAO, Charlottesville, Eric was the AIPS project manager and, as second author on the paper that introduced FITS — Wells, Greisen, & Harten (1981), was also a co-founder of FITS, the Flexible Image Transport System, adopted formally by the IAU in 1982 for general use in astronomy. As I maintained AIPS, and we both had an interest in birding, I got to know Eric quite well while he was in Sydney. In Charlottesville the following year our birding activity continued nearly every weekend with the Monticello Bird Club so that at the end of six months I had quite a decent US bird list. In fact, it exceeded my Australian list at the time. During this period Eric and I had occasionally discussed FITSWCS in general terms, and a vague notion developed that we should put our heads together and "do it properly". While the ornithological interest may seem irrelevant, in retrospect I think this secondary connection probably helped to unite us over the ten years of hard slog that it ultimately took to bring FITSWCS to fruition.

By November 1992, well after I had returned to Sydney, and following ADASS II in Boston where a special session was held on it, interest in FITSWCS had again come to the fore. With some nudging from Brian Glendenning and Peter Teuben, Eric and I embarked on a project that I naively imagined would take a year or two! At least, I thought, the timeframe would be consistent with my intentions for AIPS++.

I hurriedly completed AIPS++ memo 107 in mid-November 1992 and, collaborating with Eric by copious emails, we soon added FITS to it. This included the CD matrix formalism developed by Bob Hanisch and Don Wells (1988), that being the limit of progress on FITSWCS when we started. We announced the first draft of *Representations of Celestial Coordinates in FITS*, now known as WCS Paper II (2002), for public comment on sci.astro.fits on 4 February 1993.

Already by April 1993, Rick Balsano, a summer student at LANL working under Wayne Landsman, became the first to code our FITSWCS proposal. His programming in IDL uncovered a handful of errors in the projection equations, mostly in the nature of typos, but with a couple of significant fixes. We were surprised at how quickly things were moving. Development proceeded subsequent to Eric presenting a poster paper in June at the American Astronomical Society meeting in Berkeley, with some useful feedback from the community resulting in a further three public drafts. Thereafter, Eric presented a poster and a talk at the FITS BoF at ADASS IV in Baltimore in September 1994. As AIPS++ could not use the IDL implementation, it seemed about time to start work on WCSLIB.

2 Development

Early in November 1994 I started coding our FITSWCS proposal independently in Fortran 77 and K&R C. I was targetting the C library at AIPS++, which was written in C++, but with an eye to anything else that could bind to C. This was against some grumbles from within AIPS++ that it should be written directly in C++, which, I countered, would have restricted its applicability. The Fortran library was aimed at AIPS, Miriad, IRAF, GIPSY, and any others of the major astronomical data reduction packages of the time.

Being strongly influenced by its coding style, I chose the name, WCSLIB, as a nod to Pat Wallace's SLALIB, which I had studied closely in writing the ephemeris routines for the Australia Telescope Compact Array, which, likewise, I had named ATELIB.

WCSLIB 1.0 was released on 31 January 1995 under a GNU Library General Public License, in accordance with AIPS++ policy. In hindsight, the choice of LGPL, versus GPL, certainly helped to foster the adoption of WCSLIB. Later, under the influence of an article written by Richard Stallman, *Why you shouldn't use the Lesser GPL for your next library*, I switched to GPL in WCSLIB 4.0. However, this created licensing problems. I am reminded particularly of a project at Brigham and Women's Hospital in Boston that was adapting medical imaging software for use in astronomy. Confronted with legal issues of which I had limited interest, I was relieved to switch back to LGPL in release 4.3.

The first release of WCSLIB concerned itself solely with coding the projection equations and related spherical coordinate transformations, essentially as had been done previously in libraries that implemented the AIPS convention. In the process I uncovered several more significant corrections to the FITSWCS draft. There was little in WCSLIB 1.0 relating to FITS other than instructions on how to set the Euler angles from FITS header keywords. In fact, my intention was to make the implementation of the projection equations as general as possible so that, even today, they only acknowledge FITS's existence in setting the default value of the radius of the generation sphere to $180°/\pi$, and in the three-letter projection codes. This policy carried on later in the implementation of spectral, logarithmic, and tabular coordinates, distortions, etc.

My coding of the projection equations was informed by a program I had written for my own amusement during my Ph.D. years. It plotted celestial charts with various coordinate systems and map projections, inspired by a text book I had tracked down in the geography library at Sydney University, *Map Projections*, by Kellaway (1946). In particular, I opted to use angles in degrees specifically because, for some projections, it is important to be able to represent the latitude at the pole exactly: 90° rather than $\pi/2$ radians. It also accords with the FITS prescription of measuring angles in degrees.

Reviewing it now after thirty years, WCSLIB 1.0 does seem rather rudimentary. Nevertheless, it garnered much impactful feedback. Most notably, Rick Ebert (IPAC/JPL) and Lindsay Davis (NOAO) independently reported an apparent bug that actually arose from a serious shortcoming in the FITSWCS formalism. Here Eric and I were confronted with the dreaded monster emerging from under the bed! We were in a quandry for a week or two as to what to do about it. Ultimately we decided to go the whole hog and generalise the formalism to allow the reference point to be anywhere on the sphere, in particular for the sake of the conic projections. It led to the introduction of the `LATPOLE` keyword as elucidated in Sect. 2.4 of WCS Paper II, which I consider to be the most abstruse part of FITSWCS.

WCSLIB 2.1, the next release in November 1995, through to WCSLIB 2.9, in April 2002, progressively implemented the rest of the celestial coordinates part of the FITSWCS proposal, including developments within that period, which it validated. WCSLIB 2.1 began to acknowledge the existence of FITS by parsing the CTYPEia keywords. However, it was still left to the applications programmer to obtain the WCS keyvalues from the FITS header and plug them into disparate parts of WCSLIB.

By WCSLIB 2.4, Ray Plante (NCSA) had translated the C library into Java, which was included in that release only, and Brian Glendenning (NRAO) had written the C++ class wrappers for AIPS++.

Also in this period, WCSLIB 2.5 was joined by a curvilinear coordinate plotting routine, PGSBOX, ultimately intended to become part of PGPLOT. While general in nature, it came with an interface to WCSLIB for plotting celestial coordinate grids. It was written expressly for AIPS++ and proved to be a devilishly tricky piece of coding. However, it never did made it into PGPLOT.

Our FITSWCS proposal was split into two papers in 1999, with Eric taking Paper I while I took Paper II. They were accepted together for publication on 9 September 2002 and ratified as a FITS standard that December. This provided an impetus to reconsider the design of WCSLIB at a fundamental level. I had coded WCSLIB 1.0 first in Fortran 77 and then translated it to an independent C implementation as I figured it would be harder to do the other way around. However, Fortran severely restricted what could be done in C, which I had always considered to be the primary implementation. By WCSLIB 2.9 this had become such an impediment, not to mention irritation, that I decided to abandon the independent Fortran implementation and replace it with a set of wrappers on the C library. This would give me the freedom to design the various WCSLIB data structs as I saw fit. Considering what they have to do, the Fortran wrappers, which are written in C, continue to amaze me at how well they work.

At the same time I had been trying to persuade Richard Gooch (ATNF) to use WCSLIB to add FITSWCS support

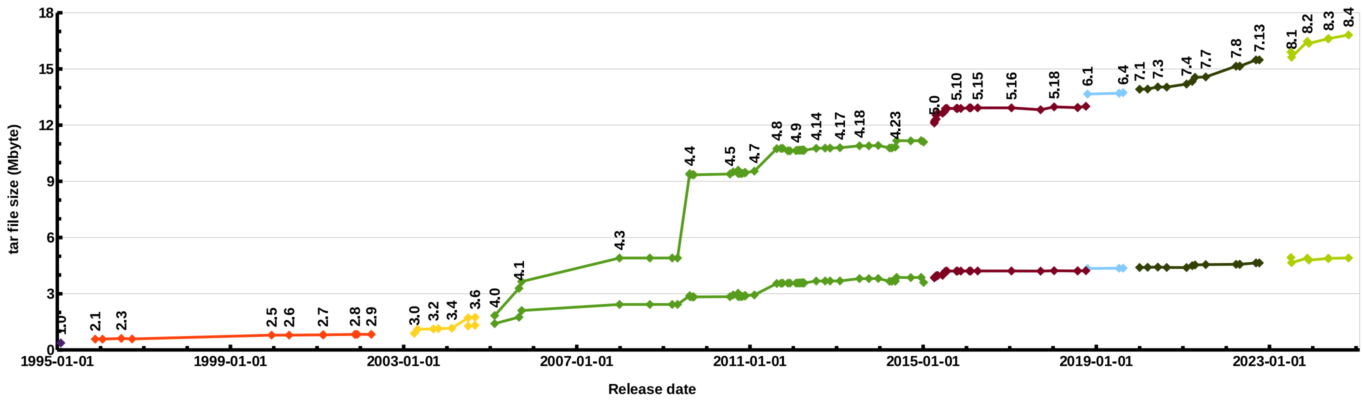

Figure 1: Evolution of the WCSLIB distribution tar file size (uncompressed). Release 2.4 excludes a translation of the C library into java by Ray Plante (NCSA). The large jump at WCSLIB 4.4 was due to the inclusion of the *Doxygen*-generated user manual in HTML and PDF formats, whereas 4.1 and 4.3 were boosted by preprocessed *Flex* code. The lower trace omits these extras, thereby offering a better estimate of the true size of the code base. Other jumps are identified in the text. Omitting the *Doxygen* and *Flex* output, WCSLIB 8.4 is ×13 larger than 1.0. It came as quite a surprise to me when I prepared this graph to see how WCSLIB is still accreting code at a steady rate.

to his *Karma/kvis* package, *kvis* being a visualisation tool developed in-house by ATNF. Richard, being a committed speed daemon, said he would consider it but only if the API was vectorised. The idea appealed to me and I decided to add it to the mix of a major revamp. However, WCSLIB never did find its way into *Karma*.

Thirdly, I had decided to give WCSLIB a FITS header parser as the next logical progression and consequently the `wcsprm` struct would have to hold the keyvalues that it extracted. That also meant that WCSLIB would have to do its own memory management.

Finally, by this time WCS Paper III (2006) was in an advanced state, including its specification of tabular coordinates, and I had written several drafts of WCS Paper IV on distortions (2004). Consequently, I had a pretty good idea of what would be required of WCSLIB to implement them. The changes would necessarily affect the API, which I realised would be painful for WCSLIB's users. However, I figured it would be best to get it over and done with in one hit. Perl, Python, Qt, and many others have gone through similar upheavals. I pre-released the header files and test programs in December 2002 and, following positive feedback from Jessica Mink (CfA), issued WCSLIB 3.0 as a beta release on 1 April 2003, which seemed an appropriate date! WCSLIB 3.1 followed at the end of that month.

Parsing WCS keyrecords from FITS headers was the next notable advance, the difficulty here being that WCSLIB doesn't and won't ever read FITS files, so needs assistance from some other piece of software. As it happened, the IAU General Assembly was held in Sydney in July that year and I found myself discussing it with Bill Pence. He agreed to provide a function in CFITSIO to scoop up a whole header in a character array, which an applications programmer could then pass to WCSLIB to sift out the WCS goodies. I coded the header parser, *wcspih()*, entirely in *Flex*, an entertaining exercise for a novice, and quite sufficient for FITS's simple *"keyword = keyvalue"* syntax. It appeared first in WCSLIB 3.5, released in June 2004.

In retrospect, this was a watershed in that it allowed an applications programmer to compute world coordinates for a FITS image without needing to know anything about what was in the header or, indeed, anything much about FITSWCS. Further parsers followed in due course: for parsing and converting units, and a completely generic header parser, *fitshdr()*, not related to FITSWCS, which I was asked to write for the benefit of AIPS++. A parser for binary table headers, including pixel lists, *wcsbth()*, a fair bit more complicated than image headers, eventually appeared in WCSLIB 4.3 of December 2007.

The first few releases of WCSLIB 4.x, dating from 2005, saw a spurt of progress, in particular in implementing tabular coordinates. Milestones were:

- 4.0 (2005/02/07): HEALPix projection, `HPX`, implemented, validating WCS Paper V (2007), *"Mapping on the HEALPix Grid"*.
- 4.1 (2005/08/31): Implementation of tabular coordinates (`-TAB`).

  Conversion from K&R C to ANSI C89 completed.
- 4.2 (2005/09/23): Installation process brought under GNU *autoconf*.
- 4.3 (2007/12/27): *wcsbth()*, binary table image array and pixel list header parser.

  wcshdo() writes out a `wcsprm` struct as a FITS header.

  Utility programs *wcsgrid*, *HPXcvt*, and *fitshdr*.
- 4.4 (2009/08/10): Generation of WCSLIB user manual from header prologues using *Doxygen*.

  Utility program *wcsware*.
- 4.8 (2011/08/15): Error diagnostics mechanism, contributed by Michael Droettboom (STScI).
- 4.18 (2013/07/13): Butterfly projection, `XPH`, implemented, validating WCS Paper VI (2013).

Regarding tabular coordinates, again WCSLIB relies on external software to read the necessary information from a FITS binary table extension. However, this was rather more complicated than simply reading a header into a character array and necessitated providing a function, *fits_read_wcstab()*, for inclusion in CFITSIO or to act as a programming template for other FITS readers. Another challenge here was to keep WCSLIB and CFITSIO independent of one another, which in practice meant not including each other's header files. I described the work in a poster paper at ADASS XV in San Lorenzo de El Escorial (2005).

WCSLIB was, and still is documented primarily by plaintext prologue comments in its header files. These are easy to maintain as a natural part of code development — good, up-to-date documentation being a crucial part of any software package. In adapting document generation to *Doxygen*, which undoubtedly improved things for applications programmers, I didn't want to clutter the code with document markup. However, now with experience in writing the FITS header parsers, I realized that it could be done essentially without any markup by strictly formatting these prologue comments. A *Flex* parser would then be able to extract them and insert the appropriate *Doxygen* markup as part of the document compilation process. The parser, *doxextr*, is augmented by sed scripts that control the font used for particular keywords or phrases. Looking now at the output from *doxextr*, blighted as it is by document markup, there is no way that I would want to work directly with it.

After retiring in 2012, while continuing to maintain WCSLIB, I mostly spent the first 18 months completing the CHIPASS 1.4 GHz radio continuum survey, a complex data reduction project I had started years before. Once published, and also with WCS Paper VI out of the way, I was able to devote time to adding missing capabilities to WCSLIB, chief among which was the implementation of distortions, as per WCS Paper IV, which remains unpublished. This went pretty much according to plan, noting, in particular, the ease with which record-valued keywords could be parsed. However, supporting the legacy distortion functions — DSS, TPV, SIP, TNX, and ZPX, which between them cover the gamut of possible ways of defining a distortion function! — required creating a superset of all of them. Thus was born the Template Polynomial Distortion function or TPD. Distortion handling first appeared in WCSLIB 5.0 of 5 April 2015, a beta release, with refinements and developments following in eleven releases over the next six months with help of much feedback. The formalism proposed in WCS Paper IV was thus validated as handling everything expected of it.

WCS Paper VII (2015) on time-related keywords was implemented in WCSLIB 6.1, released on 19 October 2018. It required the header parsers to recognise the new keywords, and the `wcsprm` struct to be expanded to accomodate them. Subsequently, I was asked twice separately to accomodate various non-standard keywords used in solar astronomy, leading to WCSLIB 7.x and 8.x

3 In retrospect

In retrospect, one thing that impresses me regarding development of WCSLIB is the critical importance of user feedback. This has ranged from bug reports, sometimes with patch; requests for new functionality, often leading to new service routines; or even contributions of major sections of code, such as the error diagnostics reporting. Warnings issued from compilers or runtime analysers such as *valgrind* have been a major contributor, often reported by various Linux package maintainers, who run the latest version of the compiler with strict checking enabled on platforms that I cannot test directly. While often just nuisance warnings, occasionally they have uncovered real problems and must always be investigated. Feedback relating to thread-safety has been another major contributor.

Another important aspect of user feedback is that is has often provided me with the only indication that anyone actually uses WCSLIB! For example, when WCSLIB first implemented distortions I immediately received copious feedback, which was invaluable in honing the code, but also revealed that it was useful to many people. On the other hand, I've had little feedback over the years relating to tabular coordinates (`-TAB`) and am left wondering whether this complex piece of code gets much use. Likewise, I was not keen to add *wcshdo()* to the library as it seemed superfluous, after all if a `wcsprm` struct was constructed from a header, why would you need to recreate it? However, the amount of feedback I have received tells me that it is widely used.

Feedback has also alerted me to WCSLIB wrappers in other languages, such as Python, Java, Javascript, Julia, and R, and its incorporation into other packages, such as Astropy and JuliaAstro. Likewise, its presence as a package in various Linux distros, ranging from Homebrew to Ubuntu.

In light of the pivotal role of user feedback, I felt it important to acknowledge contributors and have done so via the THANKS file included with WCSLIB, which has now accumulated 115 names. Some have made many contributions, others one or two. Contributors are also acknowledged for specific contributions via the CHANGES file, where some names will be found to occur quite frequently!

Of course, ideally code that is released should not contain bugs, but we all know that is impossible. From the first, WCSLIB has been distributed with a suite of test programs, which are also meant to serve as example code. They have proved essential in picking up silly, and not so silly coding errors before they get out the door. However, more subtle errors may take years or even decades to reveal themselves. For example, accounting for the many special cases in the spherical coordinate transformation equations of WCS Paper II proved to be exasperating and, nine years after first coding them, finally led me to write the mathematical notes supplement to the paper. As another example, recently it was pointed out that a coordinate vector containing a mix of NaN and non-NaN elements should survive unchanged if multiplied by the unit matrix. However, noting that 0 * NaN = NaN, this wasn't happening and a weird looking test was needed to account for the possibility. This was in a basic piece of code dating back to WCSLIB 2.1! While the rate of bug reports may be slowing, it seems they will never stop!

One of the biggest frustrations when writing a software library is finding that you need to change the API. There are remedies for this in C++ but in C and Fortran the only recourse is to write a new function with a different name and deprecate the old one. To do otherwise risks losing your user base. I bit the bullet at WCSLIB 3.0 but would never countenance doing so again. If I was to start again from scratch, I would keep much the same structure but alter the API in some respects, particularly relating to the error reporting mechanism and in extracting most of the auxiliary data from the `wcsprm` struct.

Another thing that strikes me about WCSLIB is having to support non-standard usage. The history of FITSWCS, particularly its long gestation time, allowed what is now errant usage to get a foothold and there is an expectation that WCSLIB will account it. My approach has been to quarantine non-standard usage as much as possible by providing WCSLIB with a suite of routines that repair non-standard constructs at as early a stage as possible so that the rest of the library doesn't need to deal with them.

Technological change over the past 30 years has been relentless and has affected WCSLIB in many ways. Compilers have become cleverer and cleverer, mostly for the better, but sometimes I think too clever for their own good! While I embraced selected features of C99, Link Time Optimisation (LTO) essentially forced Fortran 2003 onto the Fortran wrappers at WCSLIB 8.3 in specific cases, as required by some Linux distros. As compilers have become better at finding potential problems with code they have also become more prone to producing false alarms. Annoyingly, these "cry wolf" warnings have to be fixed lest they obscure signficant warnings; the goal has always been that WCSLIB should compile cleanly, that is, with strict warnings enabled but without producing any.

You may wonder what motivated my interest in WCSLIB. Initially, as I related, FITSWCS and WCSLIB were a part of AIPS++ that I had responsibility for. When the Virtual Observatory concept was taking off in the 1990s, it was understood that such standards and the software to implement them would be of key importance and ATNF was keen to be involved. When I left AIPS++ in 1998, I took over the software used for reducing data from the 21 cm Parkes Multibeam system, of which I was the sole maintainer until retirement. Circa 2008, I became interested in mining the HIPASS spectral line data set to construct a continuum map of the southern sky. As part of the process I added full WCS capability to the map-producing gridder named *gridzilla*. The ability to grid onto a few well-chosen map projections proved to be a key part of the data analysis, ultimately resulting in CHIPASS. Later, I was given responsibility for maintaining ATNF's fork of Miriad, which, I decided, would benefit from full WCS capability. Inevitably, this motivated a refinement of WCSLIB's Fortran wrappers. Truely, there is no substitute for using your own code to discover, and repair, its shortcomings.

I've maintained WCSLIB since retiring in 2012 on a volunteer basis, essentially as a hobby and a way of keeping in touch with current activity in astronomy. Over the years, through FITSWCS and WCSLIB, I've corresponded with a wide gamut of astronomers: from grad students to old-timers, including a Nobel Prize winner. I've occasionally thought about divesting WCSLIB and looked briefly at putting it on Github. However, over time the code base has accumulated a wealth of RCS check-in comments, which I am meticulous about and often consult, and I couldn't immediately see a way to retain them. Also, I develop on RCS branches and have devised an automated release procedure, such that the RCS revision matches the WCSLIB release number. I couldn't see a way to maintain that or anything close to it. Consequently, I put it aside, at least for now.

My involvement with FITSWCS and WCSLIB has been challenging at times, but always stimulating and certainly rewarding in many ways. It's consumed a decent chunk of my professional career, and quite a bit of my retirement as well! I'd like to thank the ADASS community and Program Organising Committee once again for recognising it as the recipient of this year's software prize.

References

[1] Calabretta, M. R. 1992, Spherical Map Projections, AIPS++ Memo No. 107, The AIPS++ Consortium. `http://aips2.nrao.edu/docs/memos/107.ps`

[2] Calabretta, M. R., & Greisen, E. W. 2002, A&A, 395, 1077 (WCS Paper II)

[3] Calabretta, M. R. et al. (WCS Paper IV, draft dated 2004/04/22) `http://www.atnf.csiro.au/people/Mark.Calabretta`

[4] Calabretta, M. R. 2006, Astronomical Data Analysis Software and Systems XV, ASP Conference Series, 351, 591

[5] Calabretta, M. R., & Roukema, B. F. 2007, MNRAS, 381, 865 (WCS Paper V)

[6] Calabretta, M. R., & Lowe, S. R. 2013, PASA, 30, e050 (WCS Paper VI)

[7] Greisen, E. W. 1983, Non-linear Coordinate Systems in AIPS, AIPS Memo No. 27, National Radio Astronomy Observatory, Charlottesville, Virginia. Revised January 1993. `http://www.cv.nrao.edu/fits/wcs/aips27.ps.Z`

[8] Greisen, E. W. 1986, Additional Non-linear Coordinates, AIPS Memo No. 46, National Radio Astronomy Observatory, Charlottesville, Virginia, May 20. `http://www.cv.nrao.edu/fits/wcs/aips46.ps.Z`

[9] Greisen, E. W. & Calabretta, M. R. 1995, Astronomical Data Analysis Software and Systems IV, ASP Conference Series, 77, 233

[10] Greisen, E. W., & Calabretta, M. R. 2002, A&A, 395, 1061 (WCS Paper I)

[11] Greisen, E. W., Calabretta, M. R., Valdes, F. G., & Allen, S. L. 2006, A&A, 446, 747 (WCS Paper III)

[12] Hanisch, R. J., & Wells, D. C. 1988, World Coordinate Systems Representations Within the FITS Format, draft based on notes from a meeting sponsored by the National Aeronautics and Space Administration, Code EZ, held in Charlottesville, Virginia, January 1988.
`http://www.cv.nrao.edu/fits/wcs/wcs88.ps.Z`

[13] Kellaway, G. P. 1946, Map Projections (Methuen & Co. Ltd., London, England)

[14] Rots, A. H., Bunclark, P. S., Calabretta, M. R., Allen, S. L., Manchester, R. N., & Thompson, W. T. 2015, A&A, 574, A36 (WCS Paper VII)

[15] Wells, D. C., Greisen, E. W., & Harten, R. H. 1981, A&AS, 44, 363