Author: J.E. Noordam, M. Mevius	Date of issue: Version 1.1: 8 July 2005 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-ASTRON-DOC-00000
	Status: Draft Revision nr.: 1.0	File: *lofar/*

Functional description of MeqTree parameter handling

J.E. Noordam, M. Mevius

(jnoordam@astron.nl)

Dwingeloo

Version 1.1: 8 July 2005

Verified:			
Name	Signature	Date	Rev.nr.
	o.p.v.	15th November 2005	0.9.5

Accepted:		
Work Package Manager	System Engineering Manager	Program Manager
J.E. Noordam .... *date:*	C.M. de Vos .... *date:*	J. Reitsma .... *date:*

Distribution list:

Group:	For Information:
ASTRON	dOList
O.M. Smirnov (OMS)	C.M. de Vos
R. J. Nijboer (RJN)	K. van der Schaaf (KvdS)
A. G. Willis (AGW)	H. Holties
R. Assendorp (RXA)	A.G. de Bruyn
T. A. Oosterloo (TAO)	M. A. van Haarlem
G. van Diepen (GVD)	J. D. Bregman (JDB)
W. N. Brouw (WNB)	
M. Brentjens (MAB)	
M. Mevius (MXM)	

Document revision:

Revision	Date	Section	Page(s)	Modification
0.5	2005-July-01	-	-	Creation
1.0	Version 1.1: 8 July 2005	??	??-??	Updated class description

Abstract

The

1 Introduction

MeqTrees are a way to implement an arbitrary Measurement Equation, and to solve for (arbitrary subsets of) its parameters. Figs 5 and 6 are a quick reminder of MeqTree basics.

This document gives a functional description of the various options, modes and tools that are needed to maximize the power of MeqTree parameter solving, and to make them easy to use. In that sense it is an input requirements document. Eventually, there will be one or more companion documents that describe the actual implementation in greater detail. However, since this document will be updated with evolving insights, it can be used as kind of high-level users manual.

The over-arching principle of everything we offer is that it should have a low threshold. The user should be required to specify as little as possible. Everything should have sensible (context-sensitive) default values. In addition, the various nodes should be able take sensible and optimising decisions on their own. However, the user should be able to customise the processing, usually by modifying the TDL script (with the editor in the browser) and regenerating the UF.

2 MeqTree use case

It will be assumed that each data reduction is part of a project, which has its own set of MeqParm tables, in a separate project sub-directory.

- Select a uv-data set (MS) to be processed.
- Optionally, create a Local Sky Model (LSM). Note that the LSM has a MeqParm table of its own.
- Select a TDL script for a User Forest (UF) to operate on the MS.
- Generate the UF from the TDL script. If a LSM is available, its source/patch prediction subtrees will be automatically integrated with the UF trees.
- Transfer any calibrator MeqParm values to the UF MeqParm tables as initial values. It should be sufficient to supply the names of the relevant MeqParm table(s), after which the available funklets for the relevant MeqParms (names!) are transferred. Note multiple calibrations before, during and after the observation are automatically interpolated by the standard MeqParm interpolation scheme (see section ..).
- Execute the UF with the stream control panel. This offers the possibility of changing things like channel selection, snippet size etc, but it should have sensible defaults already. The latter are a combination between MS information, and inbuilt instructions for this particular UF (which after all represents a particular data operation).

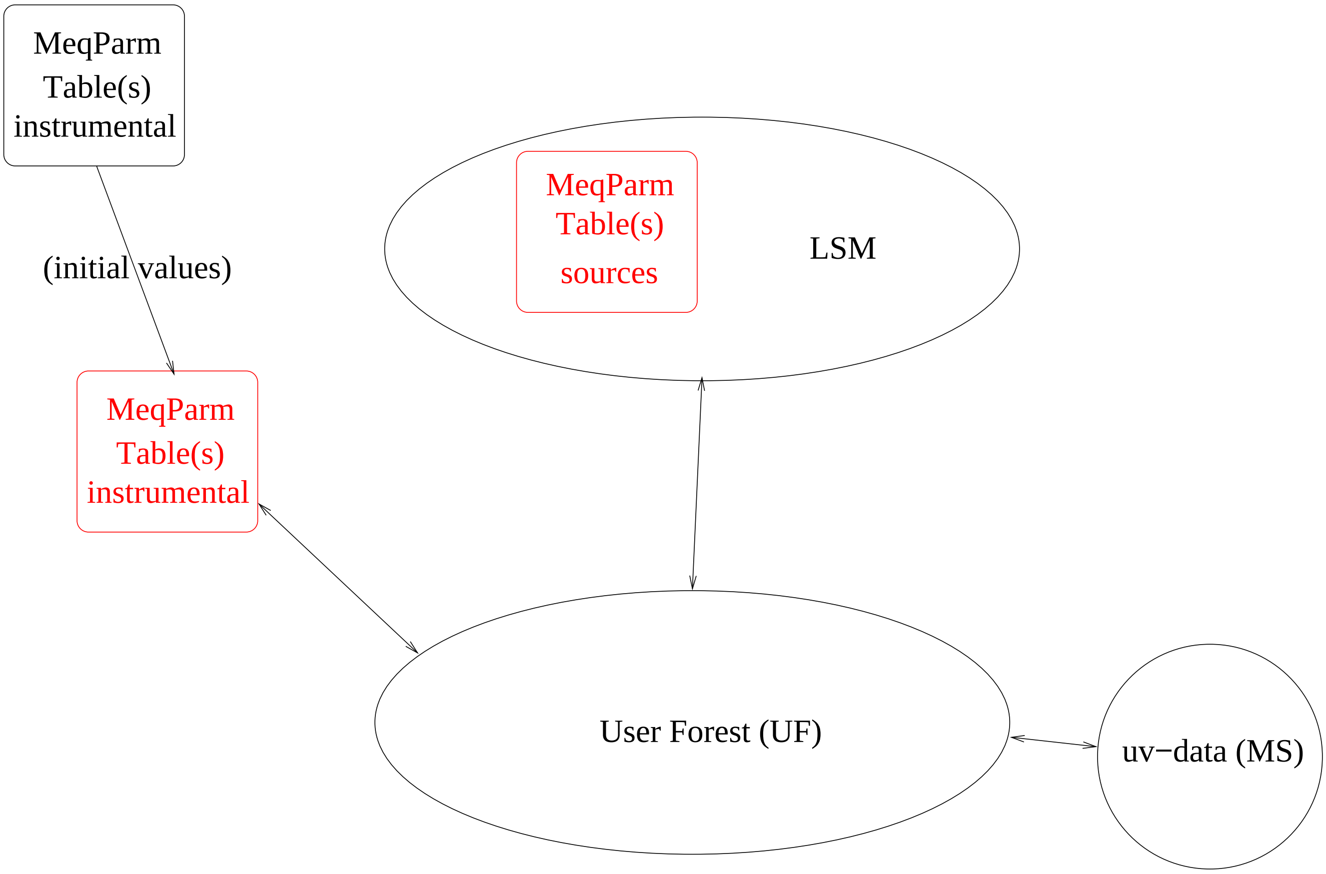


Figure 1: *Blob diagram of a User Forest (UF) and its environment. Note that the Local Sky Model (LSM) has its own MeqParm table, which should be accessible by the user in the same way as the instrumental MeqParm table.*

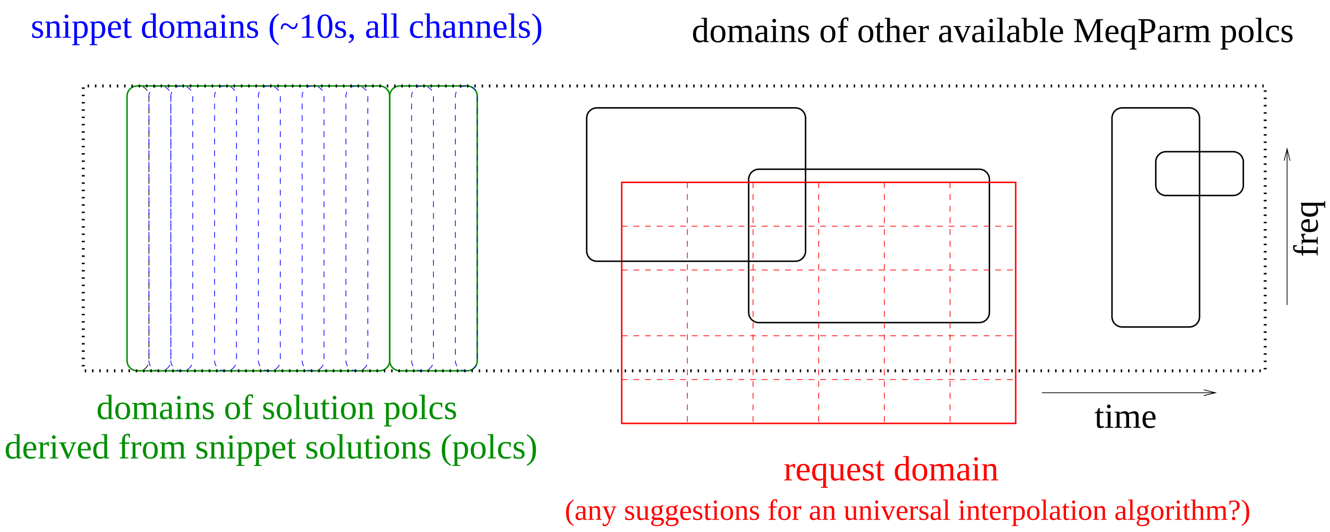


Figure 2: *Actual values for a MeqParm take the form of zero or more funklets, each with its validity domain. Illustrated is the default case, where domains are rectangular areas in the 2D (freq,time) plane. The domains of the available funklets may overlap.*
Whenever a MeqParm node gets a request for values for a given domain, it uses its available funklets to satisfy the request by means of interpolation or extrapolation. It always produces values. If no funklets are available for it (yet), it uses the default funklet that has been specified when creating this particular MeqParm.

- The UF will have in-built views, which can be activated.
- There should be some kind of progress meter.
- Inspect/display the contents of the various 'memory' nodes that have accumulated information about the reduction (e.g. solution quality, use of resources, flag summary, etc)
- Inspect/display the new MeqParm values.
- Clean them up by reorganising (smoothing, combining, splitting) the available funklets.

2.1 Multiple uv-data sets

2.2 Calibrator observations

Do a short calibrator observation. The resulting MeqParm values are stored in a MeqParm table, which may already exist. We might think of an observatory table of calibrator data, which is added to each time.

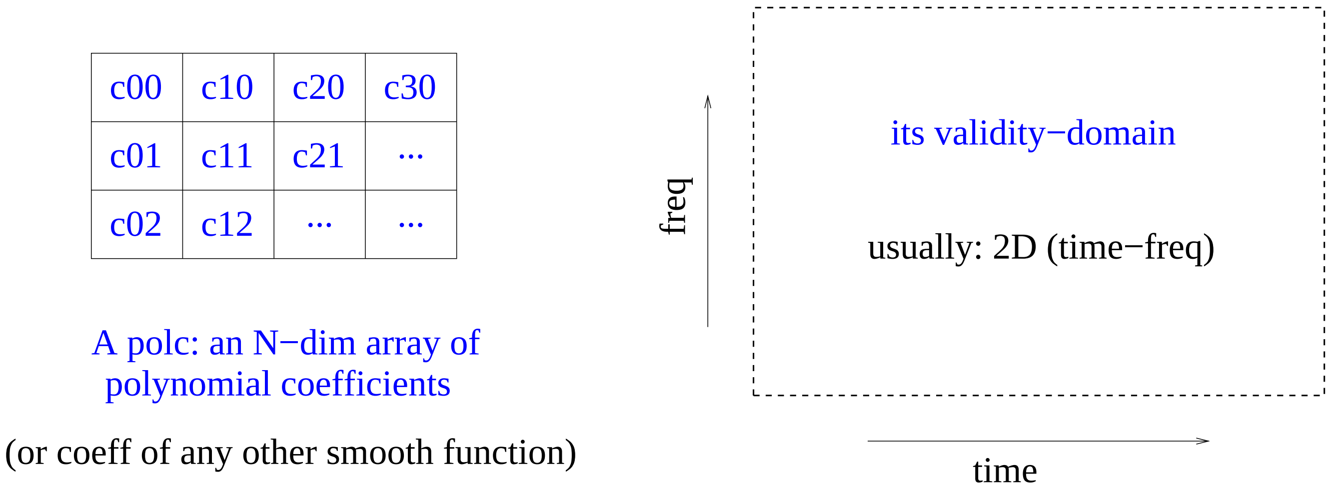


Figure 3: *A funklet is a N-dimensional array of base-function coefficients. Since a MeqSolver solves for these coefficients, they are the actual M.E. parameters. The fact that all MeqParms are functions of (usually) freq and time is a very powerful feature of MeqTrees.*
The default funklet is the ***polc****, which has polynomial base-functions along all the defined axes (default 2: freq and time). A second type of funklet, the PolcLog, was created specifically for the spectral index. It assumes a polynomial with terms* $(log(f/f_0))^n$ *along all the axes, and requires special definition parameters. It works fine, i.e. it solves for its coefficients.*
Ultimately, we want a entire family of funklet types, with arbitrary base-functions for the different axes, and definition scheme that is as uniform as possible. Of course it should be possible for the user to define his own base-functions, perhaps by defining a (Python) math expression for each axis. NB: It is not clear to me whether automatic conversion to reduced domain values will work in all cases....

3 MeqParms and MeqParm tables

Fig

4 Debugging and fiddling

MeqTrees have a debugging mode, which is enabled and controlled from the browser. It is possible to pause execution, set breakpoints, execute step-by-step (with various step-sizes), and inspect each node individually.

A very powerful feature is the MeqParm Fiddler, which can be invoked at any point in the tree. From there, it gives access to all the MeqParms that are downstream from it. It is possible to inspect, but also to modify the values and state (e.g. whether they are solvable) of MeqParms, either individually and in groups. It is then possible to re-execute the tree from the fiddler position, using the new MeqParm states.

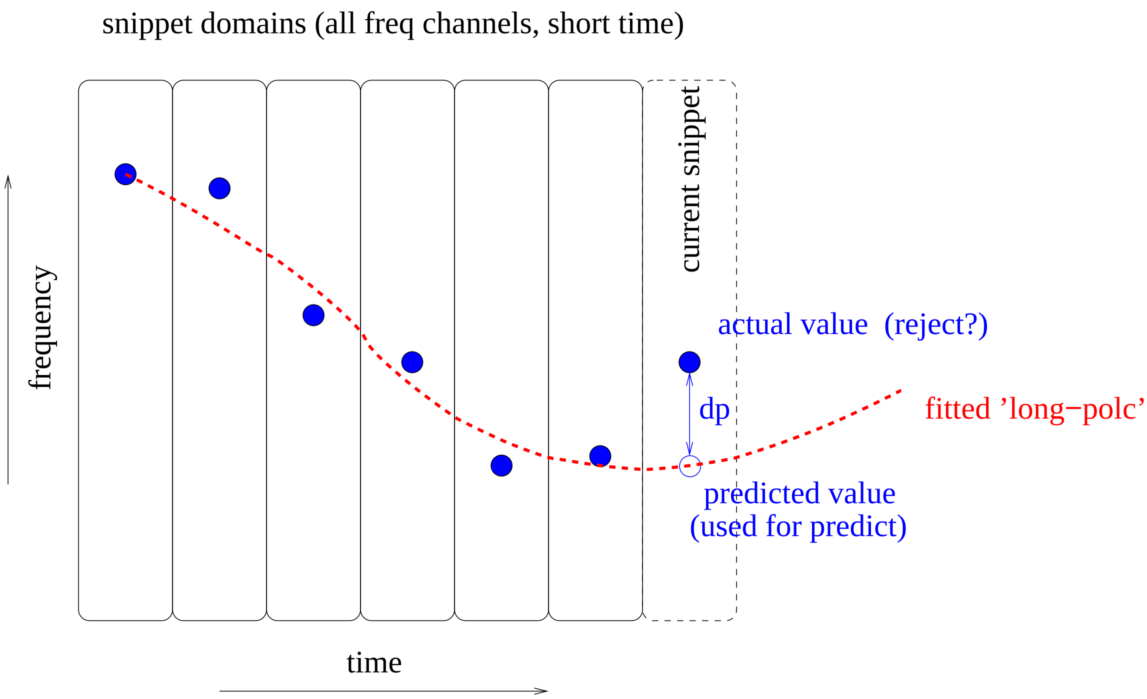
5 MeqParm solving modes

When solving for (subsets of) MeqParms, the user may choose from various modes:

- Independent snippets. This is the way it is done in all existing packages. Snippet size is of the order of one minute. Often separate solutions per frequency channel...
  - Fitting larger funklet(s) afterwards. Done by IRAM/AIPS++. Allows removing of bad snippets, and automatic distribution over multiple funklets with adjacent or overlapping domains.
  - Fitting larger funklet(s) continuously. Allows Kalmann prediction. Important for ionosphere tracking. Small (6x6) solution matrices may be accumulated continuously, and inverted when necessary. Equivalent (S/N) to single large solution?
- One huge domain
- Tiled solutions (different solving intervals for different MeqParms)

6 Solution stability

Linear dependence. Matrix rank.



An elegant scheme for MeqParm solution behaviour:

− The blue dots (●) are the results of snippet solutions (c0f only)

− Each provides an equation that is added to the 'long−polc' solution matrix

− The latter is inverted each time, to give better long−polc coefficients

− The validity domain of the current long−polc is increased with each snippet

− The updated long−polc is used to predict the value for the current (next) snippet

− The differences (dp) between predicted and actual snippet values are kept

− These can be used to accept/reject a snippet solution in a policy−free manner

− They can also be used to decide to start a new long−polc

Question: Is this scheme equivalent to a single long−snippet solution for a long−polc?

Figure 4: Snippet-solutions.

Use of covariance matrix (and other solver metrics)

Automatic conversion to reduced domains (-1,1)

Automatic ignoring of super-dimensional triangle

S/N of polynomial coeff (only the average is intuive). Bas van der Tol.

Levenberg Marquardt strategies.

7 An observatory MeqParm naming convention

A MeqParm node will always return values for a requested domain, using its available funklets in the (its?) MeqParm table. The trick is to identify which MeqParm is which, especially when using an existing MeqParm table: transferring calibration values etc.

What is needed is an observatory MeqParm naming convention, to be used in all TDL scripts. Basically, this is a kind of generalised Measurement Equation, in which all instrumental effects have standard names. This can be helped by issuing TDL scripts for standard Jones matrices for this particular instrument, which can be inserted into user-defined trees.

Problem: alternative Jones matrices with different parametrization. How are these related to one another (should we want to?).

8 MeqParm Table organisation

Naming, etc

Transfer of external values

Before offering canned trees for operational use, we must define the simplest possible book-keeping scheme

Step-wise development, driven by experience. The minimum level that we offer must deal with calibrator observations, and using their MeqParm results as initial values for 'science' observations.

-) A 'project' resides in a sub-directory. It comprises:

-) MS (multiple?)

-) MeqParm table (for instrumental MeqParms)

-) LSM

-) Project file (can be inspected from the browser)

- Contains processing history and other information

- Automatically updated

-) Images etc

-) A calibrator observation is a project

-) After starting the browser, starting the kernel and loading a canned forest, the user uses the Stream Control to select an MS.

-) A NEW instrumental MeqParm table is created, with a name built up from the following elements:

-) MS name

-) MS time-domain (from MS)

-) forest_function (from forest state record)

-) If one or more input MeqParm tables are specified, they are merged into the new MeqParm table, to serve as initial values.

-) NB: The source parameters reside in a separate MeqParm table inside the LSM. It is not possible to use it before we have the Tree Definiton Language (TDL).

9 MeqParm visualization

Groups

10 MeqParm manipulation tools

Combination, splitting, smoothing over edge discontinuities

11 Conclusions

Tsja...

Contents

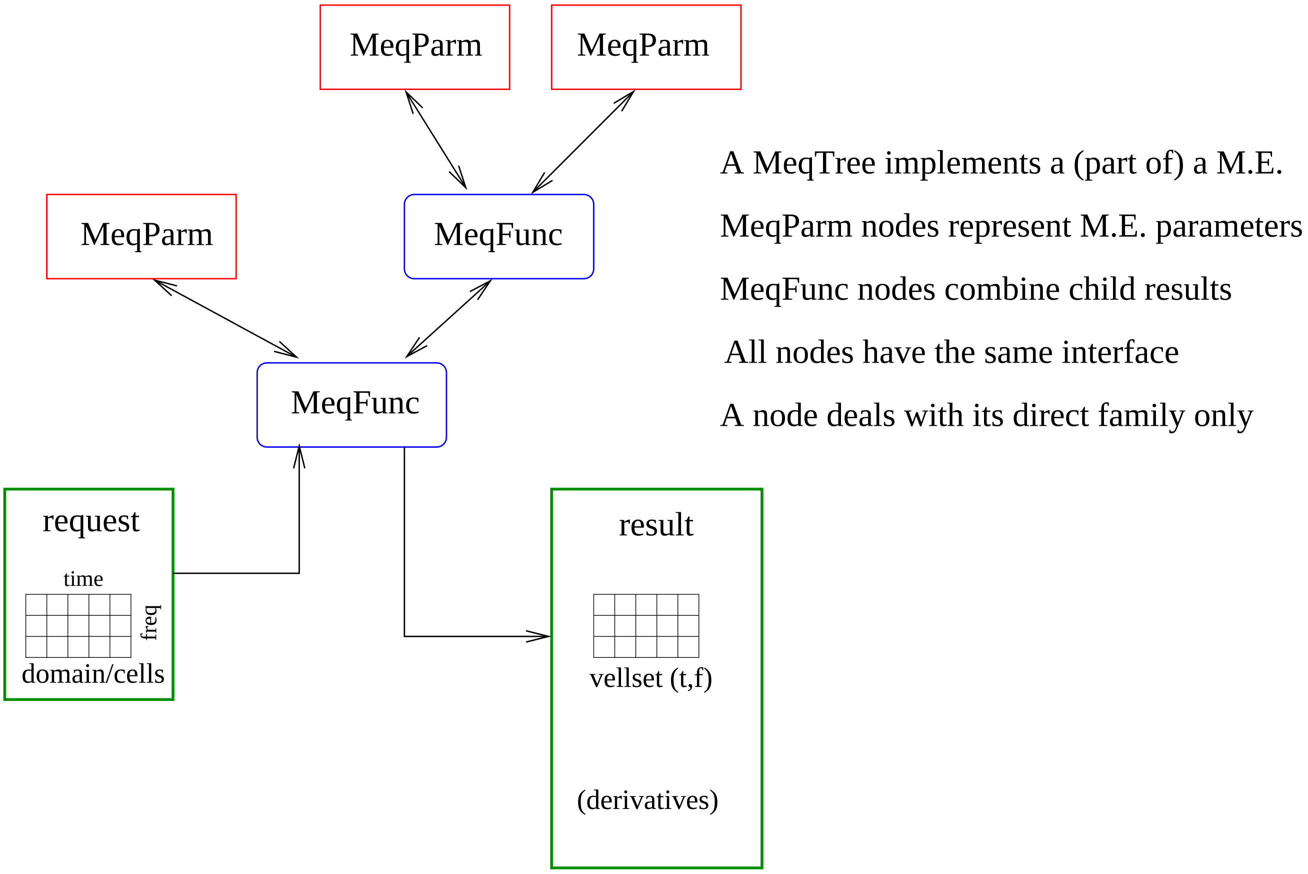


Figure 5: *A basic MeqTree with some nodes and MeqParms. This is a quick reminder of the essential MeqTree concepts. A MeqForest implements a parametrized Measurement Equation (M.E.). MeqSolver nodes solve for arbitrary subsets of these parameters, which are represented by MeqParms nodes. Each MeqParm is associated with a MeqParm table, which contains its available values in the form of zero or more funklets. Funklets are n-dimensional arrays of coefficients for base-functions like polynomials. Each funklet has its own validity domain, usually a rectangular area in 2D freq-time space. Each funklet has a separate row in a MeqParm table.*

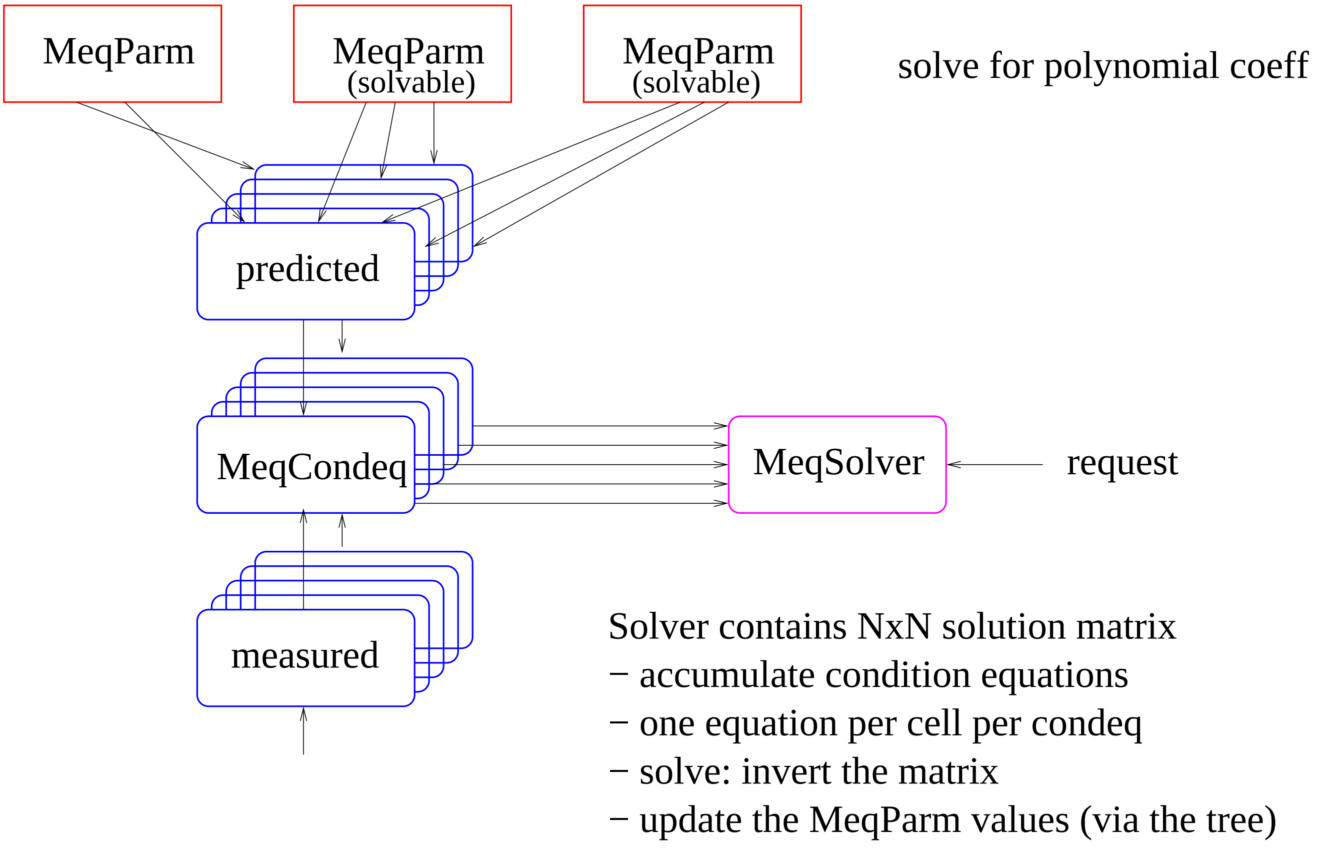


Figure 6: *Solving tree. The children of a MeqSolver node are one or more MeqCondeq nodes. These construct condition equations for the solver by comparing the results of their two children:*

$$\Delta V = V_{pred} - V_{meas} = \Sigma \frac{\partial \Delta V}{\partial p_k} \Delta p_k$$

The solver solves for incremental values Δp_k of the funklet coefficients of those MeqParms that are set solvable for this solver. A different equation is generated for each domain cell (unless it is flagged, of course). The partial derivatives are numeric, i.e. they are calculated by dividing perturbed values by the (small) perturbation per p_k. It is possible to specify a double-sided version, which is more accurate, but expensive. In the future, an analytic derivative may be implemented.
In the future, a MeqCondeq may have a third child, which specifies equation weights (per cell).