# developer.skao.int Documentation

***Release 0.1.0-beta***

**Marco Bartolini**

**Oct 04, 2021**

# USER GUIDE

The Sensitivity Calculator is an application for calculating the sensitivity of the SKA Mid-Frequency Aperture Array.

The project uses a Docker container to make the results independent of host environment. Starting and stopping the Calculator is done using `make`, but first the code must be downloaded from the SKA Git repository.

The necessary steps are:

1. Install Git if you don't already have it.

   To find if Git is installed on your computer, type in a terminal: `git --version`. The output will either say which version of Git is installed, or that `git` is an unknown command.

   If Git is not there, point your browser to https://git-scm.com/book/en/v2/Getting-Started-Installing-Git and follow the instructions for installation.

2. Install Docker if you don't already have it.

   To find if Docker is installed on your computer, type in a terminal: `docker -v`. The output will either say which version of Docker is installed, or that `docker` is an unknown command.

   If Docker is not there, point your browser to https://docs.docker.com/get-docker and follow the instructions for installation.

3. Clone the sensitivity calculator from the SKA Git repository by moving to the destination directory on your machine, and typing: `git clone https://gitlab.com/ska-telescope/ska-ost-senscalc.git`.

4. Enter the code directory with: `cd ska-ost-senscalc`.

5. Type: `make up` to build and run a Docker container with the Sensitivity Calculator. The process may take several minutes the first time.

6. When the build process has finished, point your browser at `http://127.0.0.1:5000/` to see the Calculator.

7. To shut down the Sensitivity Calculator and remove its Docker container, type: `make down`.

Further documentation, including a `User Guide` can be found in the `docs` folder. To build the html version of the documentation, start from the `ska-ost-senscalc` directory and type `cd docs; make html`. Read the documentation by pointing your browser at `docs/build/html/index.html`.

CHAPTER

# ONE

# USER GUIDE

Users control the Calculator via a web page interface, published at URL `http://127.0.0.1:5000/`.

The observing configuration is described by setting values in the web interface. ‘Universal’ inputs required for all observing modes, such as the target position, are displayed in the upper part of the web page as shown in Fig. 1.

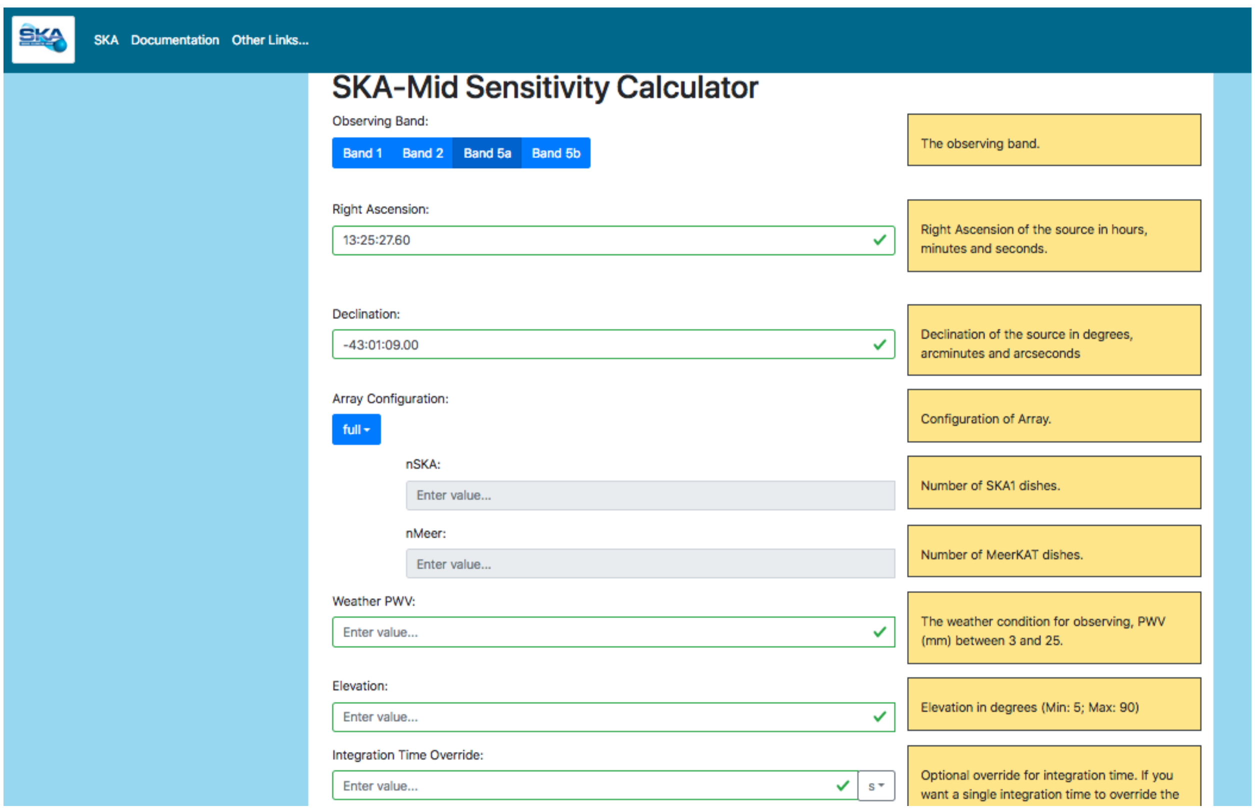



**Figure 1** . Screenshot showing the ‘universal’ part of the Calculator page.

To make the interface more intuitive, configuration details that depend on observing mode become visible on tabs extending from the base of the page when the modes `Line` or `Continuum` are selected, as shown in Fig. 2. `Pulsars` mode is not implemented yet.

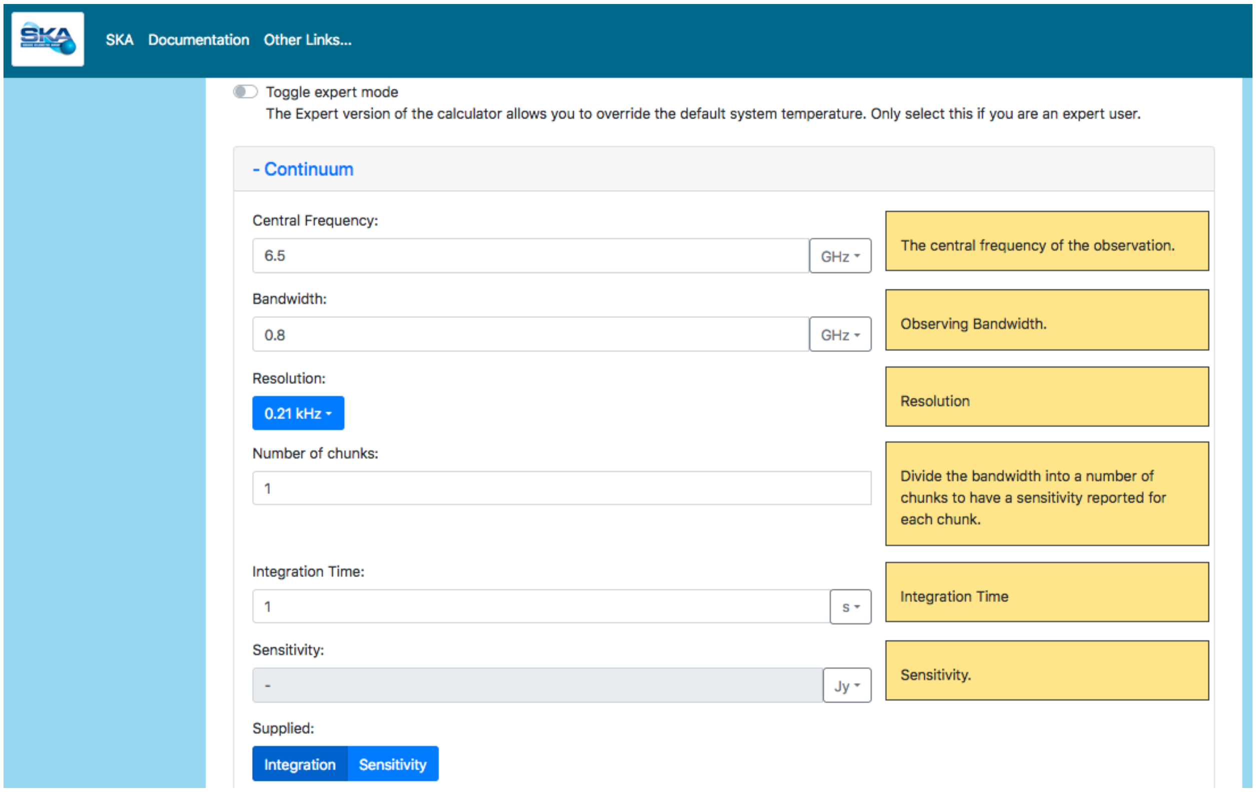
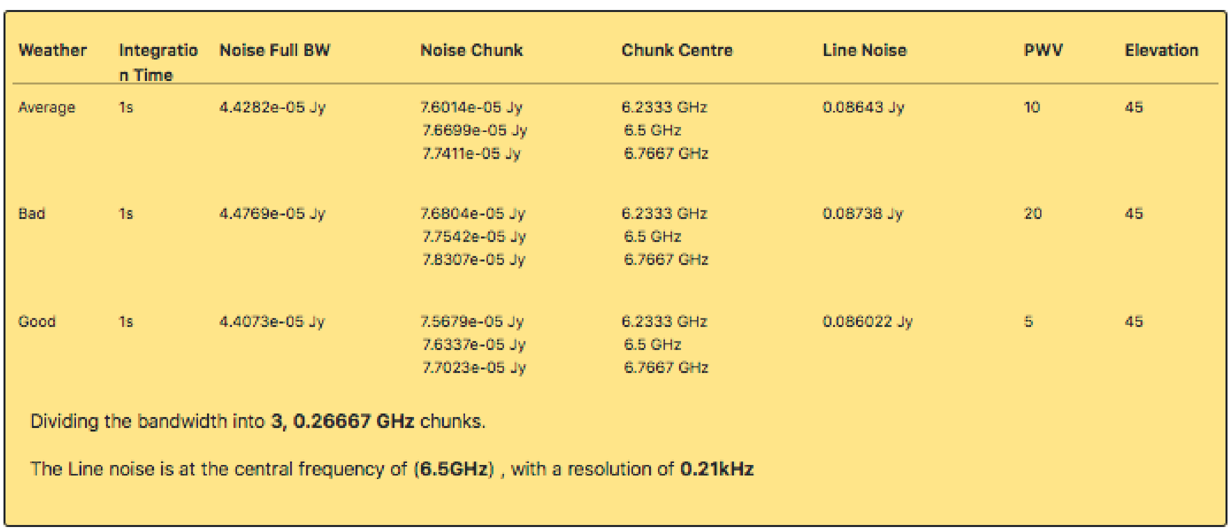



**Figure 2** . Screenshot showing the drop-down tab for `Continuum` mode.

Once configured the Calculator can be used to either calculate the sensitivity for a given on-source integration time, by entering the time and clicking `calculate`, or calculate the integration time required to reach a given sensitivity, by entering the sensitivity and clicking `calculate`. Fig. 3 shows an example report provided to the user when this is done.



| Weather | Integration Time | Noise Full BW | Noise Chunk | Chunk Centre | Line Noise | PWV | Elevation |
|---|---|---|---|---|---|---|---|
| Average | 1s | 4.4282e-05 Jy | 7.6014e-05 Jy<br>7.6699e-05 Jy<br>7.7411e-05 Jy | 6.2333 GHz<br>6.5 GHz<br>6.7667 GHz | 0.08643 Jy | 10 | 45 |
| Bad | 1s | 4.4769e-05 Jy | 7.6804e-05 Jy<br>7.7542e-05 Jy<br>7.8307e-05 Jy | 6.2333 GHz<br>6.5 GHz<br>6.7667 GHz | 0.08738 Jy | 20 | 45 |
| Good | 1s | 4.4073e-05 Jy | 7.5679e-05 Jy<br>7.6337e-05 Jy<br>7.7023e-05 Jy | 6.2333 GHz<br>6.5 GHz<br>6.7667 GHz | 0.086022 Jy | 5 | 45 |

Dividing the bandwidth into **3, 0.26667 GHz** chunks.

The Line noise is at the central frequency of (**6.5GHz**) , with a resolution of **0.21kHz**

**Figure 3** : Screenshot showing the report for the total continuum noise with 3 chunks for a hypothetical observation. No weather PWV (Precipitable Water Vapour) was specified, so results for 3 canonical conditions are shown.

## 1.1 Inputs

The calculator inputs can be categorised by the observing mode they fall under. **Universal** inputs are those that apply regardless of the selected observing mode.

### 1.1.1 Universal

- **Observing Band** The selected band to use for the observation:
  - Band 1: 0.35GHz - 1.05GHz
  - Band 2: 0.95GHz - 1.76GHz
  - Band 5a: 4.6GHz - 8.4GHz
  - Band 5b: 8.4GHz - 15.4GHz
- **Right Ascension** and **Declination** The equatorial coordinates of the observed source. The sensitivity is calculated for the time at which the target reaches its maximum elevation, crossing the meridian.
- **Array Configuration** Preset list of array configurations. Click on the tab to choose from:
  - full: all SKA1 and MeerKAT antennas
  - core: just the MeerKAT antennas
  - extended: just the SKA1 antennas
  - custom: activates the nSKA and nMeer fields where the user can enter the number of SKA and MeerKAT antennas directly.
- **Weather PWV** If no value is set in the weather PWV (Precipitable Water Vapour) field then results will be given for 3 canonical conditions; “Good” (PWV=5mm), “Average” (PWV=10mm) and “Bad” (PWV=20mm). The PWV is used in the calculation of the atmospheric brightness temperature, $T_{atm}$. Since $T_{sys}$ is dependent on $T_{sky}$ and therefore $T_{atm}$ and the weather conditions, if the user decides to manually edit $T_{sys}$ in ‘commisioning mode’, or any of the variables it depends on, the option to set the PWV will be removed.
- **Elevation** The user can use this field to specify the elevation at which the target will be observed. If no value is set then a default of 45 degrees is assumed. If the given elevation is never reached by the target, then the target’s zenith elevation will be used. The actual elevation assumed for the sensitivity calculation is reported in the result table.
- **Integration Time Override** This is an optional input, which can be left blank. If a value is entered, it will take precedence over any integration time inputs for any of the observing modes. This is useful if the user wants to test one integration time for multiple observing modes at once (so they don’t have to edit each one individually). It may be good to have the other integration time inputs disabled when a value is entered here.
- **‘Commissioning Mode’ Inputs** By activating the ‘toggle commissioning mode’ switch the user is given access to some of the parameters used in the sensitivity calculation, as shown in Fig. 4. The calculator front-end automatically enables/disables inputs to avoid conflicts as the user selects which one they want to edit. These are passed to the calculator back-end as hard-coded values which will be used in place of automatically calculating values for those variables.

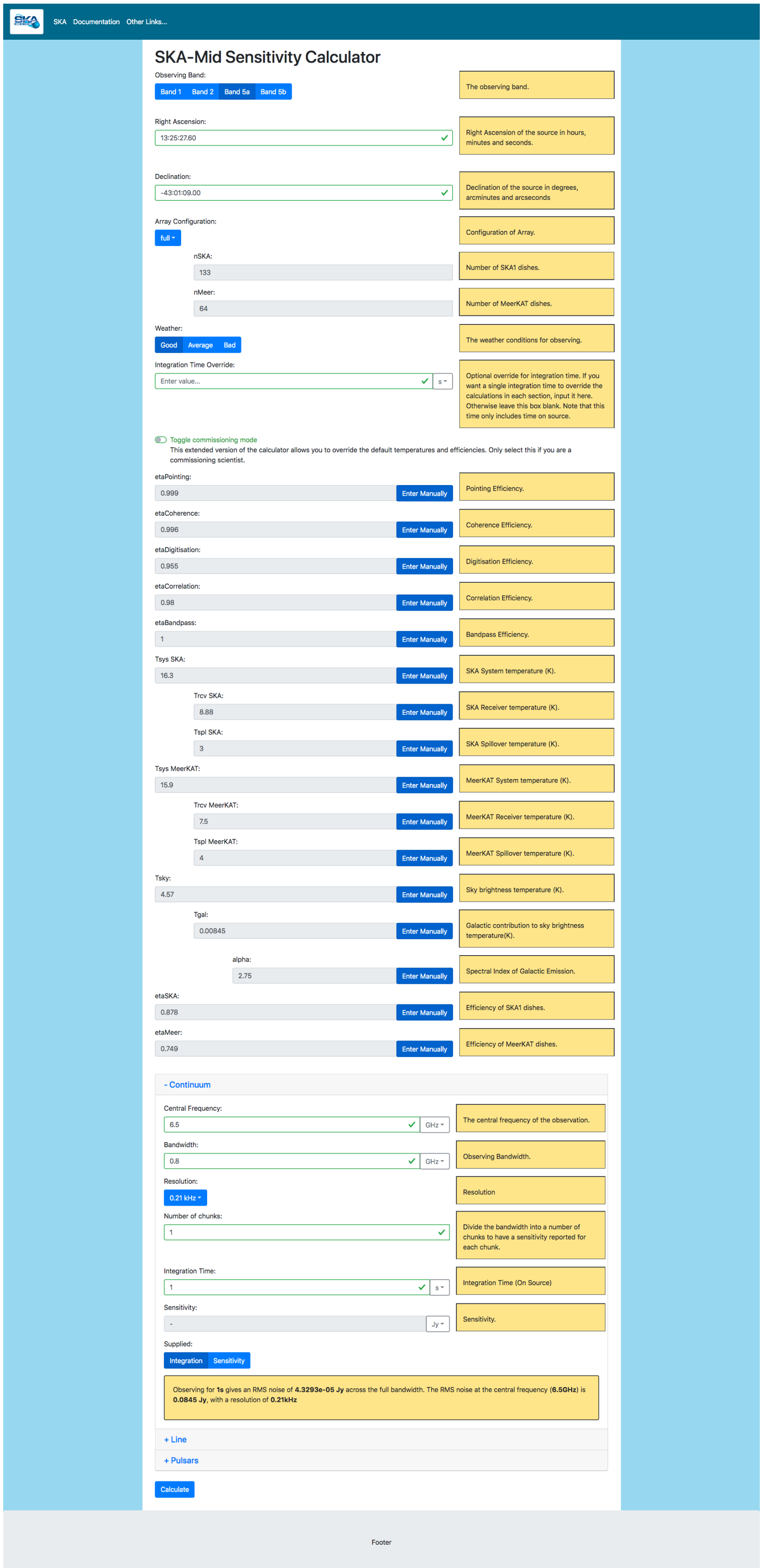

SKA Documentation Other Links...
SKA-Mid Sensitivity Calculator
Observing Band:
Band 1 Band 2 Band 5a Band 5b
The observing band.
Right Ascension:
13:25:27.60
Right Ascension of the source in hours, minutes and seconds.
Declination:
-43:01:09.00
Declination of the source in degrees, arcminutes and arcseconds
Array Configuration:
full
Configuration of Array.
nSKA:
133
Number of SKA1 dishes.
nMeer:
64
Number of MeerKAT dishes.
Weather:
Good Average Bad
The weather conditions for observing.
Integration Time Override:
Enter value...
Optional override for integration time. If you want a single integration time to override the calculations in each section, input it here. Otherwise leave this box blank. Note that this time only includes time on source.
Toggle commissioning mode
This extended version of the calculator allows you to override the default temperatures and efficiencies. Only select this if you are a commissioning scientist.
etaPointing: 0.999 Enter Manually Pointing Efficiency.
etaCoherence: 0.996 Enter Manually Coherence Efficiency.
etaDigitisation: 0.955 Enter Manually Digitisation Efficiency.
etaCorrelation: 0.98 Enter Manually Correlation Efficiency.
etaBandpass: 1 Enter Manually Bandpass Efficiency.
Tsys SKA: 16.3 Enter Manually SKA System temperature (K).
Trcv SKA: 8.88 Enter Manually SKA Receiver temperature (K).
Tspl SKA: 3 Enter Manually SKA Spillover temperature (K).
Tsys MeerKAT: 15.9 Enter Manually MeerKAT System temperature (K).
Trcv MeerKAT: 7.5 Enter Manually MeerKAT Receiver temperature (K).
Tspl MeerKAT: 4 Enter Manually MeerKAT Spillover temperature (K).
Tsky: 4.57 Enter Manually Sky brightness temperature (K).
Tgal: 0.00845 Enter Manually Galactic contribution to sky brightness temperature(K).
alpha: 2.75 Enter Manually Spectral Index of Galactic Emission.
etaSKA: 0.878 Enter Manually Efficiency of SKA1 dishes.
etaMeer: 0.749 Enter Manually Efficiency of MeerKAT dishes.
- Continuum
Central Frequency: 6.5 GHz The central frequency of the observation.
Bandwidth: 0.8 GHz Observing Bandwidth.
Resolution: 0.21 kHz Resolution
Number of chunks: 1 Divide the bandwidth into a number of chunks to have a sensitivity reported for each chunk.
Integration Time: 1 s Integration Time (On Source)
Sensitivity: - Jy Sensitivity.
Supplied:
Integration Sensitivity
Observing for 1s gives an RMS noise of 4.3293e-05 Jy across the full bandwidth. The RMS noise at the central frequency (6.5GHz) is 0.0845 Jy, with a resolution of 0.21kHz
+ Line
+ Pulsars
Calculate
Footer

**Figure 4** . Expanded view of an example use-case for the additional inputs on the 'commissioning mode' version of the calculator.

### 1.1.2 Line

- **Zoom Frequency** For each zoom, the user can input a frequency for that zoom. When a value is entered, the next zoom becomes enabled, allowing a value to be entered. It can however be left blank, and the calculation will only be done for zooms which have a set frequency. This way, the user can select how many zooms they want (up to a maximum, currently 4).
- **Zoom Resolution** For each zoom, the user can set a line resolution.
- **Integration Time** The integration time of the observation. Used when calculating the sensitivity that observing for this amount of time will achieve.
- **Sensitivity** The sensitivity for the observation. Used when calculating the integration time necessary to achieve this sensitivity.
- Supplied **Toggle** allowing the user to swap between integration time and sensitivity as the input (giving the other as the output).

### 1.1.3 Continuum

- **Central Frequency** The central frequency for the observation. Must be within the selected band.
- **Bandwidth** The bandwidth for the observation. Must be fully contained within the selected band.
- **Resolution** The line resolution.
- **Number of chunks** The user can select an integer number of chunks to split the bandwidth up into. If they do, the output report will show the sensitivity (or integration time) for each chunk.
- **Integration Time** The integration time of the observation. Used when calculating the sensitivity that observing for this amount of time will achieve.
- **Sensitivity** The sensitivity for the observation. Used when calculating the integration time necessary to achieve this sensitivity.
- **Supplied** Toggle allowing the user to swap between integration time and sensitivity as the input (giving the other as the output).

CHAPTER

TWO

# SENSITIVITY MODEL

The 'system equivalent flux density' (SEFD) for a single dish is given by:

$$SEFD_{dish} = \frac{2kT_{sys}}{\eta_A A}$$

where:

- $k$ is the Boltzmann constant so that $kT_{sys}$ measures the power received from background emission and all other sources of unwanted signal within the system, that is $T_{sys} = T_{spl} + T_{sky} + T_{rcv} + T_{cmb} + ...$
- $\eta_A$ is the dish efficiency
- $A$ is the geometric dish area.

The SEFD for an interferometer array made up of two types of dish is given by:

$$SEFD_{\mathrm{array}} = \frac{1}{\sqrt{\frac{n_{\mathrm{SKA}}(n_{\mathrm{SKA}}-1)}{SEFD_{\mathrm{SKA}}^2} + \frac{2n_{\mathrm{SKA}}n_{\mathrm{MeerKAT}}}{SEFD_{\mathrm{SKA}}SEFD_{\mathrm{MeerKAT}}} + \frac{n_{\mathrm{MeerKAT}}(n_{\mathrm{MeerKAT}}-1)}{SEFD_{\mathrm{MeerKAT}}^2}}}$$

where $n_{\mathrm{SKA}}$ is the number of SKA antennas, $n_{\mathrm{MeerKAT}}$ is the number of MeerKAT antennas, $SEFD_{\mathrm{SKA}}$ is the SEFD computed for an individual SKA antenna, and $SEFD_{\mathrm{MeerKAT}}$ is the SEFD computed for an individual MeerKAT antenna.

We define the telescope sensitivity here as the minimum detectable Stokes I flux (1 $\sigma$). This is equal to the noise on the background power, obtained using the radiometer equation $\sigma = SEFD/\sqrt{2Bt}$, corrected for atmospheric absorption:

$$\Delta S_{min} \exp(-\tau_{atm}) = \frac{SEFD_{array}}{\eta_s \sqrt{2Bt}} Jy$$

where:

- $\Delta S_{min}$ is the source flux density above the atmosphere
- $\eta_s$ is the efficiency factor of the interferometer
- $B$ is bandwidth
- $t$ is integration time
- $\tau_{atm}$ is the optical depth of the atmosphere towards the target

See *Implementation* for more details.

# CHAPTER THREE

# INTRODUCTION

Calculating the sensitivity of SKA Mid-Frequency Aperture Array will be important for SKA scientists and engineers during the construction and operation of the telescope. This document describes the current Calculator implementation, explaining the decisions behind its design, and links to a space where future work can be planned.

CHAPTER

FOUR

# MAKE TARGETS

This project uses Docker containers to make the results independent of host environment.

This project contains a Makefile which acts as a UI for building Docker images, testing images, starting and stopping containers, etc. The following make targets are defined:

| Makefile target | Description |
|---|---|
| build | Build a new application image |
| build_wheel | Build the Python wheel |
| down | stop all containers launched 'make up' |
| help | show a summary of the makefile targets |
| lint | lint the application (static code analysis) |
| pull | Download the application image from the Docker registry |
| push | Push the application image to the Docker registry |
| test | Test the application image |
| up | launch the Calculator container service |

CHAPTER

FIVE

# DESIGN

## 5.1 Goals and Considerations

A prime consideration in the design of the calculator is exactly how the user will interface with it. Some users may prefer a simple, accessible interface, e.g. a web page, while others may prefer to be able to download a GUI to their own device. Some may even prefer to access the calculator API directly to use with their own code. Importantly, however, the sensitivity calculator will ultimately be part of some larger observing tool. The calculator is expected to provide the user with a report for attachment to their observing proposal supporting their use of the telescope. After speaking to the developer of the ALMA Observing Support Tool, it became clear that the vast majority of their users would use the web-based tool where possible and nearly always have internet access when wanting to use the tool. Therefore it was decided that the prototype calculator should use a web-based interface to demonstrate functionality, but feature a distinct front- and back-end, allowing the interface to be modified, or for other interfaces to be added (if needed) as the project evolves. Because the calculator is publicly available via the SKA GitLab, anyone who wants to directly interact with the source code can do so.

The scientific model behind the sensitivity calculation will be updated as the life-cycle of the telescope progresses. At the time of writing it is based on the calculation framework of 'SKA1: Design Baseline Description - SKA-TEL-SKO-0001075' and 'SKA1 System Performance Assessment Report - SKA-TEL-SKO-0001089', with some additional information from the earlier document Anticipated SKA1 Science Performance. Once the telescope is live, we should be able to actively record, for example, the system temperature. This, among other possible developments, will change how the sensitivity calculator functions. In addition, with the current calculator being a prototype, a number of features will certainly be added as time goes on. With all of this considered, it is sensible to maintain a modular design for the calculator back-end, where functionality is separated into independent modules. This means that if, say, the model describing the receiver temperature for SKA1 dishes is changed, the relevant code can easily be modified and the rest of the program should still run without any issues.

One of the desired features of the calculator is to be able to both calculate sensitivity, given an on-source integration time (and the other required parameters) and calculate the integration time required to reach a given sensitivity. The user should also be offered a range of different observing modes, so they can calculate e.g. total continuum noise, line noise, etc. These observing modes are not mutually exclusive, however. A user may be interested in performing a continuum observation with a number of zooms and would therefore want to know the sensitivity they could obtain in each case. Allowing for this while also allowing the user to swap between calculating sensitivity and integration time potentially makes both the front- and back-end design quite complicated. The solution ultimately was to separate out the different observing modes into individual tabs as shown in the *User Guide*.

CHAPTER

SIX

# IMPLEMENTATION

## 6.1 Theoretical Background

### 6.1.1 Reference Documents

| | |
|---|---|
| RD1 | Anticipated SKA1 Science Performance |
| RD2 | 'SKA1 Performance Assessment Report' SKA-TEL-SKO-0001089 |

### 6.1.2 Applicable Documents

| | |
|---|---|
| AD1 | An improved source-subtracted and destriped 408 MHz all-sky map |

An overview of the theoretical performance of SKA Mid comprising SKA1 and MeerKAT dishes is given in RD1. A more detailed analysis is given in RD2 for an SKA Mid made up of only SKA1 dishes. We are grateful to Songlin Chen for help in navigating and understanding the documentation.

The Mid Sensitivity Calculator (SC) was originally implemented following the theoretical framework of RD1 but is moving to the more rigorous framework of RD2, though some details remain simplified.

### 6.1.3 Dish SEFD

The 'system equivalent flux density' (SEFD) for a single dish is the flux density of a source that produces a signal equal to the background power of the system:

$$SEFD_{dish} = \frac{2kT_{sys}}{\eta_A A}$$

where:

- $k$ is the Boltzmann constant so that $kT_{sys}$ measures the power received from background emission and all other sources of unwanted signal within the system, that is $T_{sys} = T_{spl} + T_{sky} + T_{rcv} + T_{cmb} + \ldots$
- $\eta_A$ is the dish efficiency
- $A$ is the geometric dish area.
- The 2 is there because a radio telescope measures only one polarization and it is assumed for this purpose that the other polarization has the same strength.

### 6.1.4 Array SEFD

SKA Mid is an interferometer that works by combining the signal from multiple dishes. There are 2 types of dishes involved, SKA1 and MeerKAT, with distinct characteristics. It can be shown, by adding up the signals from each baseline, that the array SEFD is given by:

$$SEFD_{\mathrm{array}} = \frac{1}{\sqrt{\frac{n_{\mathrm{SKA}}(n_{\mathrm{SKA}}-1)}{SEFD_{\mathrm{SKA}}^2} + \frac{2n_{\mathrm{SKA}}n_{\mathrm{MeerKAT}}}{SEFD_{\mathrm{SKA}}SEFD_{\mathrm{MeerKAT}}} + \frac{n_{\mathrm{MeerKAT}}(n_{\mathrm{MeerKAT}}-1)}{SEFD_{\mathrm{MeerKAT}}^2}}}$$

where:

- $n_{\mathrm{SKA}}$ is the number of SKA antennas
- $n_{\mathrm{MeerKAT}}$ is the number of MeerKAT antennas
- $SEFD_{\mathrm{SKA}}$ is the SEFD computed for an individual SKA antenna
- $SEFD_{\mathrm{MeerKAT}}$ is the SEFD computed for an individual MeerKAT antenna.
- and the assumption has been made (?) that all baselines are equally efficient.

### 6.1.5 Array Sensitivity

The 'sensitivity' of a radio telescope is an overloaded term. For the purpose of the SC we define the sensitivity as the minimum detectable Stokes I flux (1 $\sigma$). This is equal to the noise on the background power, obtained using the radiometer equation $\sigma = SEFD/\sqrt{2Bt}$, corrected for atmospheric absorption:

$$\Delta S_{min} \exp(-\tau_{atm}) = \frac{SEFD_{array}}{\eta_s \sqrt{2Bt}} Jy$$

where:

- $\Delta S_{min}$ is the source flux density above the atmosphere
- $\eta_s$ is the efficiency factor of the interferometer
- $B$ is bandwidth
- $t$ is integration time
- $\tau_{atm}$ is the optical depth of the atmosphere towards the target
- the formula applies to the centres of fields-of-view where the dish aperture response is unity.

### 6.1.6 Dependency Tree

The devil is in the detail of calculating $T_{sys}$ and the efficiency factors $\eta_A$ and $\eta_s$. Fig.1 shows how these values depend on other factors that must be estimated.

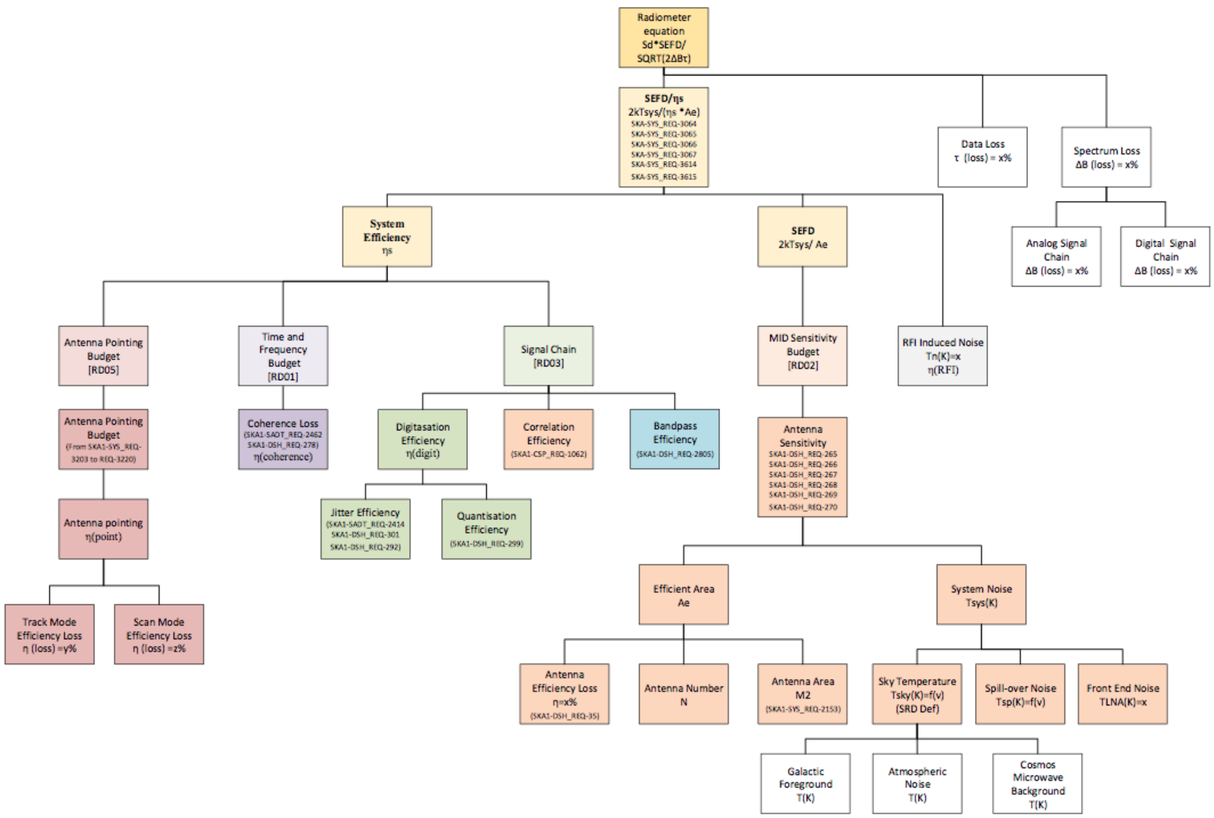



**Figure 1** . The dependency tree for factors in the sensitivity calculation (from RD2).

Currently, the SC does not incorporate all the dependency factors. Those that are included are described in the following sections.

### 6.1.7 System Temperature

The system temperature is given by:

$$T_{sys} = T_{spl} + T_{sky} + T_{rcv}$$

where:

$$T_{sky} = T_{CMB} + T_{gal} + T_{atm}$$

and:

- $T_{spl}$ is the spillover temeprature, measuring power from the ground reaching the receiver. Currently this is set to 3K for SKA1 dishes and 4K for MeerKAT.
- $T_{rcv}$ measures noise from the receiver and electronics, depending on band and dish type.
- $T_{sky}$ is the total emission from the sky.
- $T_{CMB}$ is the cosmic microwave background, 2.73K.
- $T_{gal}$ is the Galactic astronomical emission in the target direction. $T_{gal} = T_{408}(0.408/\nu_{GHz})^{alpha}$ K, where $T_{408}$ is the Galactic emission at 408MHz whose estimation is described in *Brightness at 408MHz*.
- $T_{atm}$ measures the brightness of the atmosphere, which depends on weather, observing frequency and elevation. $T_{atm}$ and $\tau_{atm}$ at the zenith are interpolated from lookup tables of results from the CASA atmosphere module,

run for a grid of frequencies and weather PWVs. $T_{atm}$ at the target elevation is estimated by relating it to the physical temperature by $T_{phys} \sim T_{atm}(1 - \exp(-\tau_{atm}))$, where $\tau_{atm}$ varies as $\sec(z)$.

### 6.1.8 Brightness at 408MHz

The brightness of the astronomical background signal at 408MHz is estimated using the all-sky non source-subtracted HEALPix map described by AD1 (Fig.2). The brightness seen by a dish is calculated by multiplying map pixels that lie under the beam by the beam profile. The beam is assumed to be Gaussian, truncated at a radius equal to the FWHM.

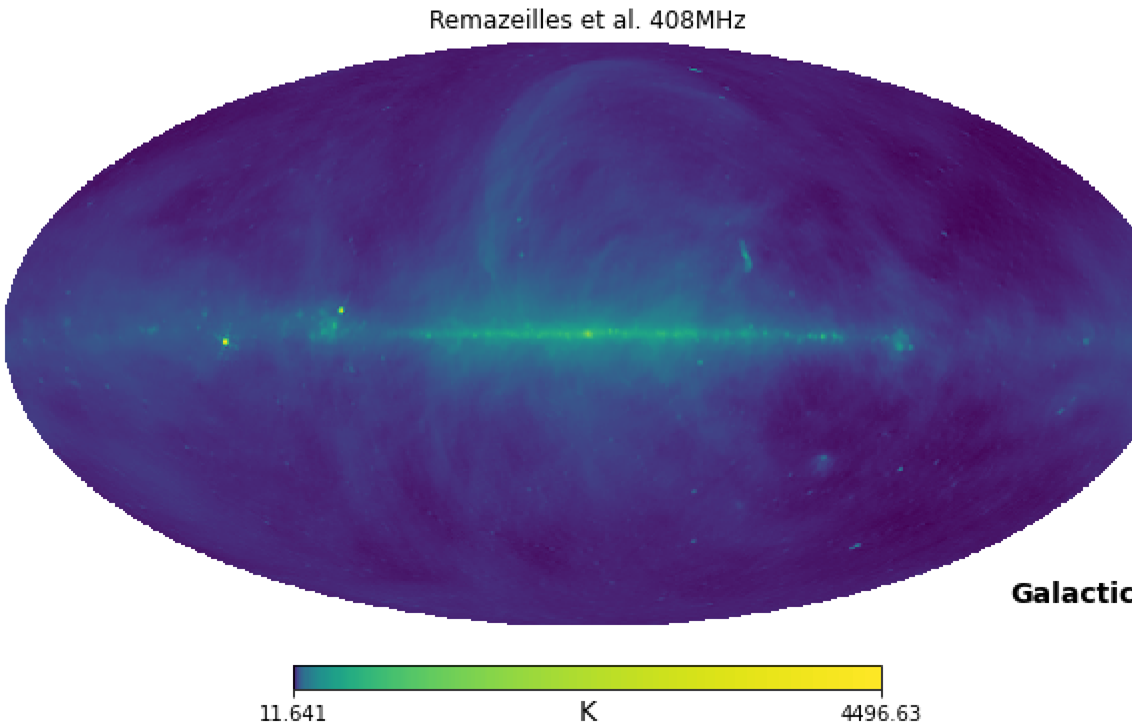



**Figure 2** . The all-sky 408Mhz map from AD1, used to calculate $T_{408}$.

### 6.1.9 Efficiencies

#### Aperture

Following RD2, the aperture efficiency $\eta_A$ is given by:

$$\eta_a = \eta_{dish}\eta_{feed}$$

where:

$$\eta_{dish} = \eta_{block}\eta_{transp}\eta_{surface}\eta_{rad.r}$$
$$\eta_{feed} = \eta_{rad.f}\eta_{ill}$$

and:

- $\eta_{dish}$ accounts for the efficiencies attributable to the dish optics
- $\eta_{block}$ accounts for physical aperture blockage
- $\eta_{transp}$ accounts for losses by transmission through the reflector surface

- $\eta_{surface}$ accounts for all losses due to incoherent propagation through the optics, including panel roughness, systematic deformation and mis-alignment;
- $\eta_{rad.r}$ accounts for the Ohmic dielectric and scattering losses in the reflector system only
- $\eta_{feed}$ accounts for the efficiencies attributable to the feeds
- $\eta_{rad.f}$ accounts for feed mismatches and losses
- $\eta_{ill}$ is the efficiency due to the actual illumination pattern

Currently, the SC follows RD1 and calculates an overall $\eta_{dish}$ from estimates of $\eta_{ill}$, $\eta_{surface}$ and $\eta_{diffraction}$ (?).

### Array

The system efficiency $\eta_s$ is the result of multiplying together the following factors:

- **eta_bandpass** This factor describes the loss of efficiency due to the departure of the bandpass from an ideal, rectangular shape. At present the value is set to 1.0.
- **eta_coherence** This factor desribes the loss of efficiency due to coherence loss on a baseline.

  $$\eta_{coherence} = \exp - \frac{< \phi_\epsilon^2(t) >}{2} = \exp -2\pi^2 \nu_0^2 < \tau_\epsilon^2(t) >$$

  We take the coherence loss at 1s integration time, which is white phase-noise dominated. The total phase delay is due to the sum in quadrature of the phase delay of the clock and signal path on both receptors:

  $$< \tau_\epsilon^2 > = < \tau_{clk,i}^2 > + < \tau_{clk,j}^2 > + < \tau_{dsh,i}^2 > + < \tau_{dsh,j}^2 >$$

  The signal path depends on the environment (atmosphere, gusty wind) and the calibration quality, which is quite complicated to estimate in practice. For now we adopt a value of $\eta_{coherence} = 0.98$ at $\nu_0 = 15.4 GHz$ as coherence loss for the worst case, and scale it to the frequency of observation using the given formula.
- **eta_correlation** This factor describes the loss of efficiency due to imperfection in the correlation algorithm, e.g. truncation error. Analysis described in "SKA CSP SKA1 MID array Correlator and Central beamformer sub element Signal Processing Matlab Model" (311-000000-007) shows that the CSP correlation efficiency is almost 100% in the case of zero RFI, and better than 98% in the case of strong RFI (defined as <10% RFI in the outside visibility ?query, what does this mean).

  Currently the efficiency value is set to 0.98.
- **eta_digitisation** This factor describes the loss of efficiency due to quantization during signal digitisation. The process is independent of the telescope and environment, and depends only on the 'effective number of bits' (ENOB) of the system, which depends in turn on digitiser quality and clock jitter, and on band flatness.

  The values used for each band are as follows:

  | Band | ENOB | Band Flatness (dB) | $\eta$ |
  |---|---|---|---|
  | Band 1 | 8 | 6.5 | 0.999 |
  | Band 2 | 8 | 6.5 | 0.999 |
  | Band 3 | 6 | 6.5 | 0.998 |
  | Band 4 | 4 | 6.5 | 0.98 |
  | Band 5a | 3 | 4 (in any 2.5GHz BW) | 0.955 |
  | Band 5b | 3 | 4 (in any 2.5GHz BW) | 0.955 |

- **eta_point** This factor describes the loss of efficiency due to dish pointing errors. Here we currently use an approximate formula:

  $$\eta_{point} \sim \frac{1}{1 + 8 ln2 \frac{\sigma_\theta^2}{FWHM^2}}$$

where FWHM is the beam full-width at half maximum power for the dish, given by the approximate formula $FWHM \sim 66\lambda/D$ (degrees), and $\sigma_\theta$ is the RMS pointing error.

### Design Independent

This section lists efiiciency factors that are independent of the telescope design.

- **eta_rfi** This factor describes the loss of efficiency due to parts of the spectrum that are lost due to strong RFI noise corrupting the astronomical signal. Currently set to 1.
- **eta_data_loss** This describes the loss of observing time due to the need for calibration, time spent moving to source, etc.

  It is currently not used in the calculator, so implicitly set to 1.

### Sensitivity Degradation due to RFI

The effect of RFI is currently removed from the system efficiency budget because of the complexity of the RFI impact. Estimates for the impact of RFI are not solid and work continues to understand them.

- **Strong RFI** Strong RFI which results in saturation in the analogue chain or clipping in digitisation will be flagged. The data loss and spectrum loss are instrument independent.
- **Moderate RFI** Moderate RFI is not flagged but contributes significant input power and might induce extra noise in the digitisation and correlation processes.
- **Weak RFI** Weak RFI, or the high-order intermodulation components of strong and moderate RFI, contribute to the sensitivity in the form of additive system noise.

## 6.2 Back-end

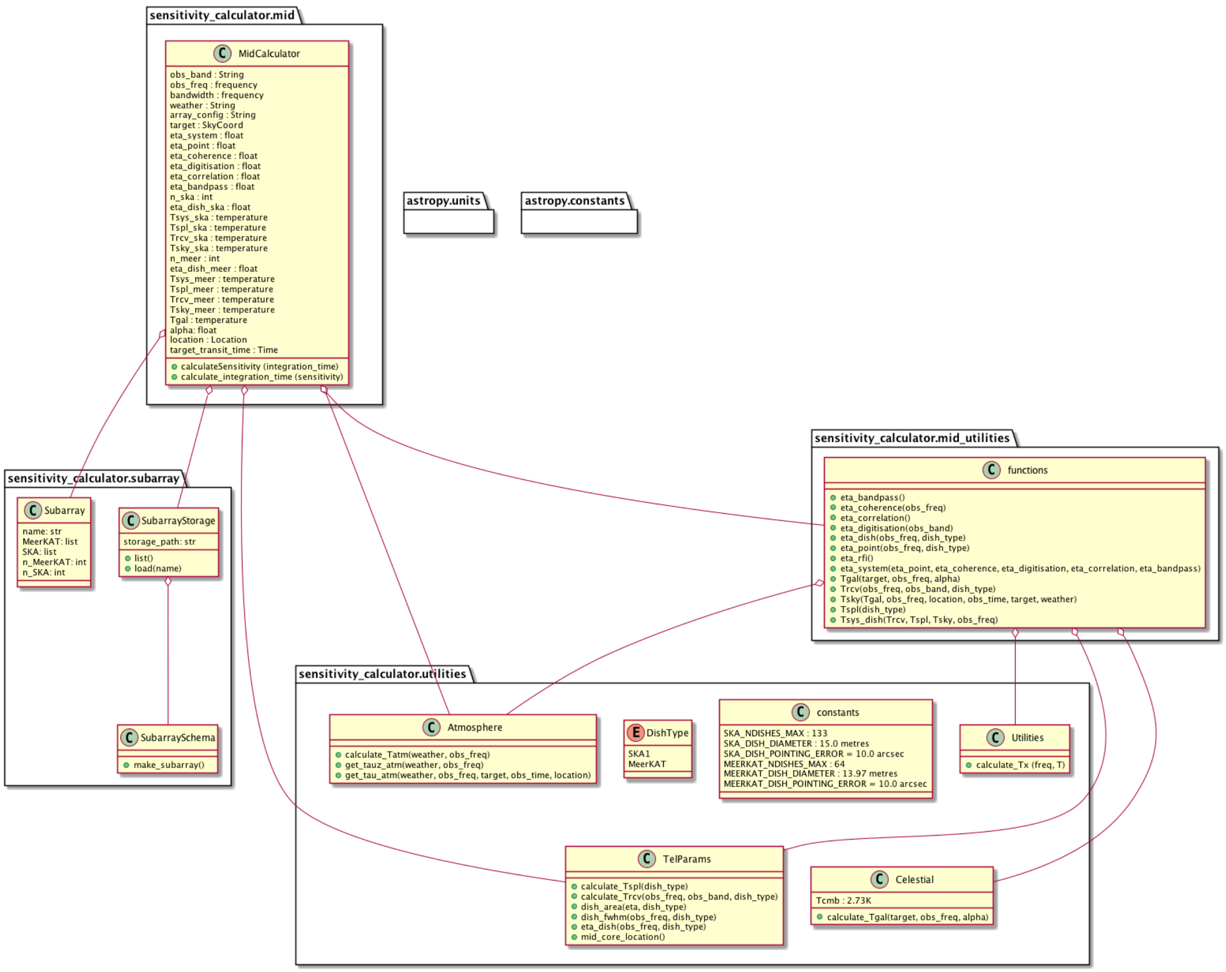



**Figure 3** . Class diagram of the Sensitivity Calculator back-end.

The back-end is written in Python 3.7 and the class diagram is shown in Fig.3. The class *MidCalculator* has 2 public methods: *calculate_sensitivity* to get the array sensitivity in Jy for the given integration time, and *calculate_integration_time* to get the integration time required for the given sensitivity.

The *MidCalculator* constructor has a number of required parameters that define the observing configuration, target and weather. The rest default to None, in which case their values will be calculated automatically. The automatic values can be overriden by setting them here.

All parameters, internal variables and results that describe 'physical' measures are implemented as astropy Quantities to prevent mixups over units.

The calculator is modular in design. There are separate functions for deriving each element of the calculation, which can be easily modified as the sensitivity model is updated.

## 6.3 Front-end

### 6.3.1 Public and 'Expert' Users

The calculator is intended for two types of user.

The first type is the ordinary observer who will use the calculator to simply calculate the performance of the telescope when looking at their target object.

The second type is the 'expert' user, who understands the telescope design and wants to test the effect of tweaking some aspect of it. This mode of use is intended for SKA staff. It allows the user to manually edit some of the values which are usually calculated automatically as part of the sensitivity calculation. Say a user wanted to test out how a different array configuration might affect the sensitivity of a given observation. They could manually edit the number of SKA1 and MeerKAT dishes in the array and these would override the numbers that the calculator uses and use the new values in the sensitivity calculation. The diagram in Fig.4 shows the dependencies between the variables used in the sensitivity calculation. The variables coloured green are the ones which can be edited by the user in 'expert' mode; blue are calculated, and pink not yet implemented. The overall pattern matches that of the dependency tree in Fig.1, with some small differences which wil be eliminated in time.

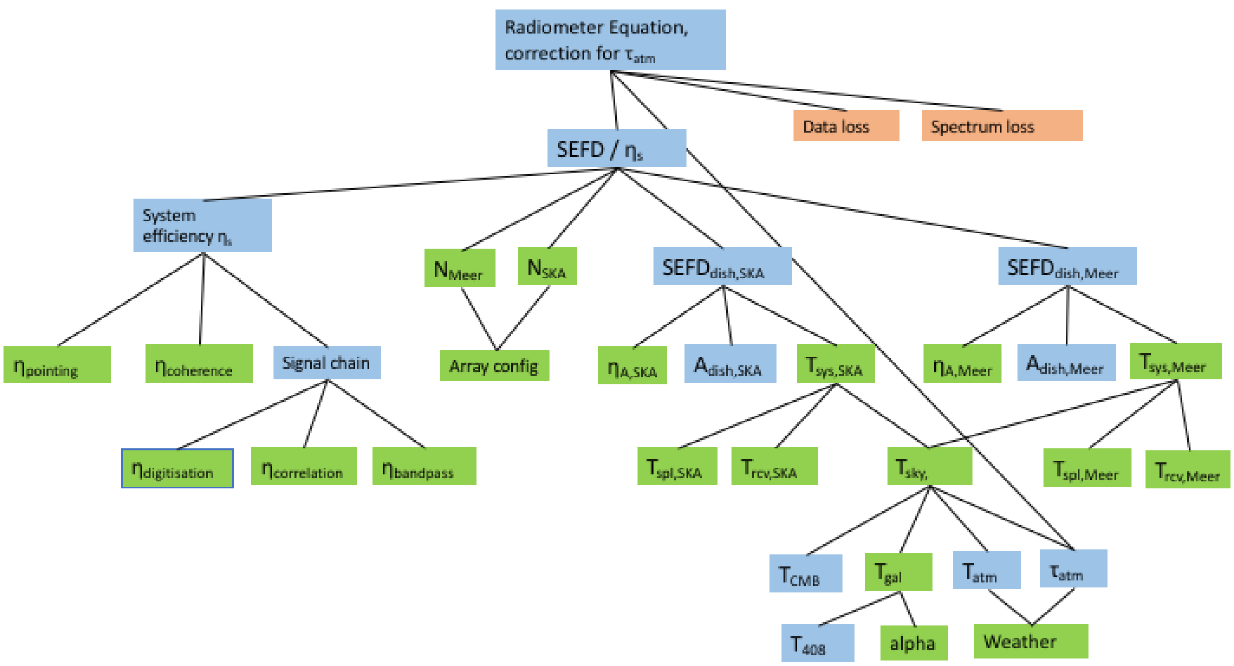



**Figure 4** . Flowchart diagram showing the dependencies of the variables used in the sensitivity calculation.

### 6.3.2 Technologies

The calculator front-end is designed as a Flask web application, using Bootstrap 4 for a responsive design.

Flask is a WSGI web application framework, allowing for very simple interaction with the Python 3 back-end code. It also ensures that the sensitivity calculations are performed on the server-side, rather than the client-side. Currently the calculations are not computationally intense but as the calculator goes on to be developed and more features are added, there's a good chance that this will change. In any case, having this setup minimises the amount of data to transfer between the user and the server, since, for example, when the program needs to access a lookup table hosted on the server, this file doesn't need to be transferred to the user's device. This will improve the speed of the calculator's response and reduce the server load.

Bootstrap 4 is the most recent version of the Bootstrap toolkit - an open-source HTML, CSS and JS library allowing for quick and clean deployment of responsive web applications. Responsive design is extremely important to make sure that the webpage functions and looks good, regardless of the size/orientation of the user's device. The calculator uses the Bootstrap 4 CDN (Content Delivery Network), which means that nothing needs to be installed on the server. When the user loads a page which uses the CDN, they may already have the required files cached on their device after visiting another website using the CDN. Since Bootstrap is the world's most popular web framework, it is likely this will be the case. If, however, the user does not have the files cached, they will be retrieved from the closest server to the user that is part of the network. This means that there may be a little extra load-time when the user first visits the website, but overall load-time will be reduced from then on.

Typescript is the main client-side language. Along with some jQuery and AJAX to send RESTful requests to and from the server. Once the page is loaded there are two 'entry points' for the Typescript code to run. The first is when the "Calculate" button at the bottom of the page is clicked. Code will then run to read the information from the form, perform some validation (checking the inputs are formatted correctly, within some allowable ranges, etc.) before using AJAX to send the data using a GET request to the server-side Flask code. By performing this validation on the client side, we limit the number of unnecessary requests to the server, i.e. sending inputs which would not be allowed. The Flask code will parse the inputs and, based on the data sent, call the calculator back-end code to perform the necessary calculation, then return the results to the client-side, which will output them to the user's screen. The other 'entry point' into the client-side code is when the user modifies one of the inputs. Once a value is changed and that input is deselected, some Typescript code will execute to perform similar validation. This helps make it clear what the actual values are that go into the calculator when the "Calculate" button is pressed.

There is also some custom CSS used to style the site. While bootstrap takes care of a lot of this, there are some tweaks which are made, such as setting the colours of the webpage to match those laid out in the SKA Brand Guidelines.

## 6.3.3 Inputs

The calculator inputs can be categorised by the observing mode they fall under. **Universal** inputs are those that apply regardless of the selected observing mode.

### Universal

- **Observing Band** The selected band to use for the observation:
  - Band 1: 0.35GHz - 1.05GHz
  - Band 2: 0.95GHz - 1.76GHz
  - Band 5a: 4.6GHz - 8.4GHz
  - Band 5b: 8.4GHz - 15.4GHz
- **Right Ascension** and **Declination** The equatorial coordinates of the observed source. The sensitivity is calculated for the time at which the target reaches its maximum elevation, crossing the meridian.
- **Array Configuration** Preset list of array configurations. Click on the tab to choose from:
  - full: all SKA1 and MeerKAT antennas
  - core: just the MeerKAT antennas
  - extended: just the SKA1 antennas
  - custom: activates the nSKA and nMeer fields where the user can enter the number of SKA and MeerKAT antennas directly.
- **Weather PWV** If no value is set in the weather PWV (Precipitable Water Vapour) field then results will be given for 3 canonical conditions; "Good" (PWV=5mm), "Average" (PWV=10mm) and "Bad" (PWV=20mm). The PWV is used in the calculation of the atmospheric brightness temperature, $T_{atm}$. Since $T_{sys}$ is dependent on

$T_{sky}$ and therefore $T_{atm}$ and the weather conditions, if the user decides to manually edit $T_{sys}$ in 'commisioning mode', or any of the variables it depends on, the option to set the PWV will be removed.

- **Elevation** The user can use this field to specify the elevation at which the target will be observed. If no value is set then a default of 45 degrees is assumed. If the given elevation is never reached by the target, then the target's zenith elevation will be used. The actual elevation assumed for the sensitivity calculation is reported in the result table.
- **Integration Time Override** This is an optional input, which can be left blank. If a value is entered, it will take precedence over any integration time inputs for any of the observing modes. This is useful if the user wants to test one integration time for multiple observing modes at once (so they don't have to edit each one individually). It may be good to have the other integration time inputs disabled when a value is entered here.
- **'Commissioning Mode' Inputs** As shown in Fig.5, the calculator in 'commissioning' mode allows the user to modify some of the variables used in the sensitivity calculation. The calculator front-end automatically enables/disables inputs to avoid conflicts as the user selects which one they want to edit. These are passed to the calculator back-end as hard-coded values which will be used in place of automatically calculating values for those variables.

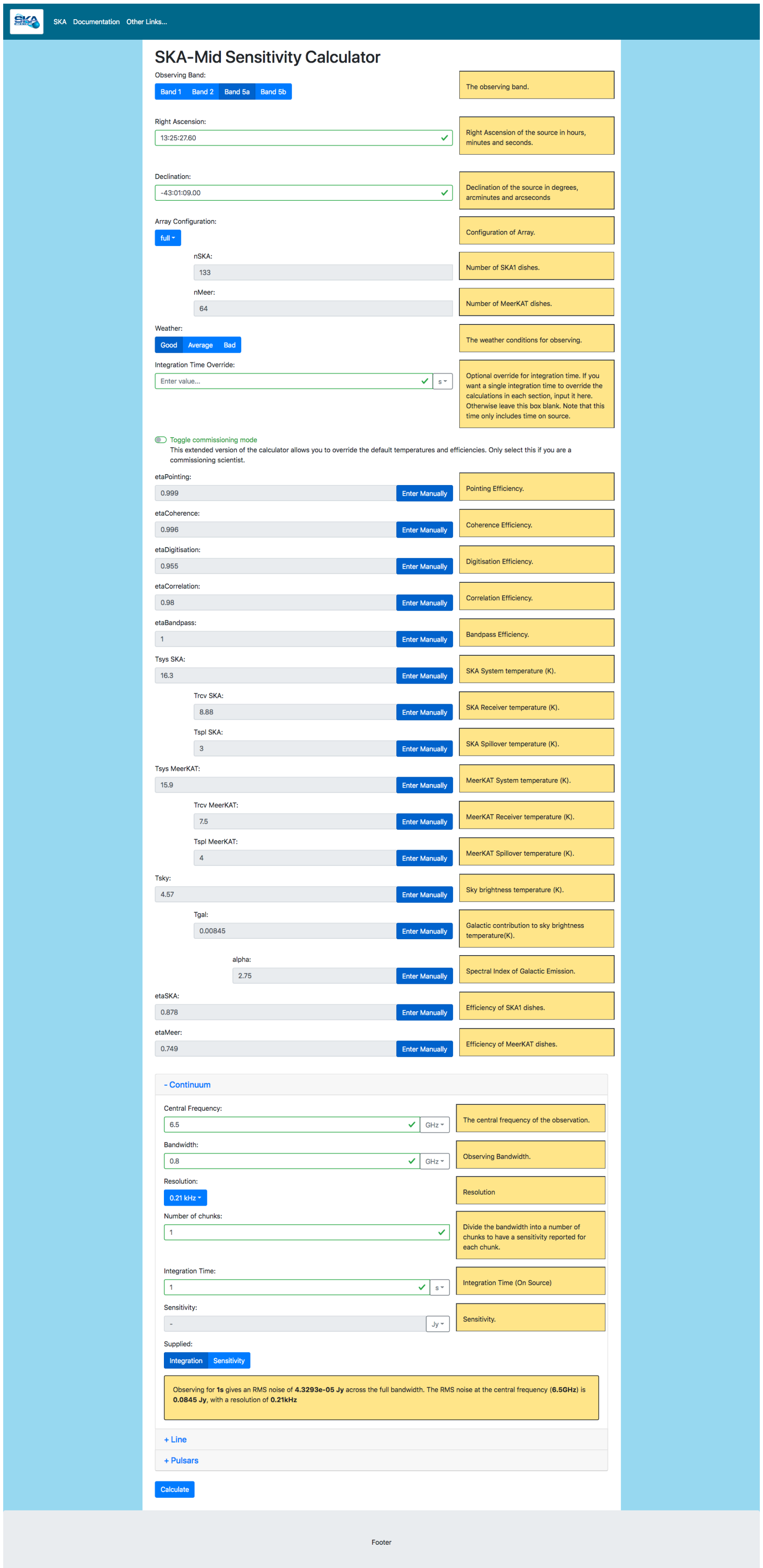

SKA Documentation Other Links...
SKA-Mid Sensitivity Calculator
Observing Band:
Band 1 Band 2 Band 5a Band 5b
The observing band.
Right Ascension:
13:25:27.60
Right Ascension of the source in hours, minutes and seconds.
Declination:
-43:01:09.00
Declination of the source in degrees, arcminutes and arcseconds
Array Configuration:
full
Configuration of Array.
nSKA:
133
Number of SKA1 dishes.
nMeer:
64
Number of MeerKAT dishes.
Weather:
Good Average Bad
The weather conditions for observing.
Integration Time Override:
Enter value...
Optional override for integration time. If you want a single integration time to override the calculations in each section, input it here. Otherwise leave this box blank. Note that this time only includes time on source.
Toggle commissioning mode
This extended version of the calculator allows you to override the default temperatures and efficiencies. Only select this if you are a commissioning scientist.
etaPointing:
0.999
Enter Manually
Pointing Efficiency.
etaCoherence:
0.996
Coherence Efficiency.
etaDigitisation:
0.955
Digitisation Efficiency.
etaCorrelation:
0.98
Correlation Efficiency.
etaBandpass:
1
Bandpass Efficiency.
Tsys SKA:
16.3
SKA System temperature (K).
Trcv SKA:
8.88
SKA Receiver temperature (K).
Tspl SKA:
3
SKA Spillover temperature (K).
Tsys MeerKAT:
15.9
MeerKAT System temperature (K).
Trcv MeerKAT:
7.5
MeerKAT Receiver temperature (K).
Tspl MeerKAT:
4
MeerKAT Spillover temperature (K).
Tsky:
4.57
Sky brightness temperature (K).
Tgal:
0.00845
Galactic contribution to sky brightness temperature(K).
alpha:
2.75
Spectral Index of Galactic Emission.
etaSKA:
0.878
Efficiency of SKA1 dishes.
etaMeer:
0.749
Efficiency of MeerKAT dishes.
- Continuum
Central Frequency:
6.5
GHz
The central frequency of the observation.
Bandwidth:
0.8
GHz
Observing Bandwidth.
Resolution:
0.21 kHz
Resolution
Number of chunks:
1
Divide the bandwidth into a number of chunks to have a sensitivity reported for each chunk.
Integration Time:
1
s
Integration Time (On Source)
Sensitivity:
-
Jy
Sensitivity.
Supplied:
Integration Sensitivity
Observing for 1s gives an RMS noise of 4.3293e-05 Jy across the full bandwidth. The RMS noise at the central frequency (6.5GHz) is 0.0845 Jy, with a resolution of 0.21kHz
+ Line
+ Pulsars
Calculate
Footer

**Figure 5** . Expanded view of the additional inputs available in 'commissioning' mode.

#### Line

- **Zoom Frequency** For each zoom, the user can input a frequency for that zoom. When a value is entered, the next zoom becomes enabled, allowing a value to be entered. It can however be left blank, and the calculation will only be done for zooms which have a set frequency. This way, the user can select how many zooms they want (up to a maximum, currently 4).
- **Zoom Resolution** For each zoom, the user can set a line resolution.
- **Integration Time** The integration time of the observation. Used when calculating the sensitivity that observing for this amount of time will achieve.
- **Sensitivity** The sensitivity for the observation. Used when calculating the integration time necessary to achieve this sensitivity.
- Supplied **Toggle** allowing the user to swap between integration time and sensitivity as the input (giving the other as the output).

#### Continuum

- **Central Frequency** The central frequency for the observation. Must be within the selected band.
- **Bandwidth** The bandwidth for the observation. Must be fully contained within the selected band.
- **Resolution** The line resolution.
- **Number of chunks** The user can select an integer number of chunks to split the bandwidth up into. If they do, the output report will show the sensitivity (or integration time) for each chunk.
- **Integration Time** The integration time of the observation. Used when calculating the sensitivity that observing for this amount of time will achieve.
- **Sensitivity** The sensitivity for the observation. Used when calculating the integration time necessary to achieve this sensitivity.
- **Supplied** Toggle allowing the user to swap between integration time and sensitivity as the input (giving the other as the output).

### 6.3.4 Subarray lookup service

There is a subarray lookup service implemented as a REST API. The json files with the different subarray configuration are stored in a local folder in the server. The list of available subarray configurations can be retrieved from **\subarrays**. The idea is to populate the subarray dropdown menu using this service. In the future the service can be expanded to accept custom subarray configurations.

CHAPTER

SEVEN

# LOCAL DEVELOPMENT ENVIRONMENT

Here we describe how to set up a local development environment to test the different parts of the sensitivity calculator. This is in general only required to gain fine grained control over the different development stages.

The implementation details are described in *Implementation*.

## 7.1 Outline and dependencies

The **backend libraries** have the following Python dependencies:

- `numpy`
- `scipy`
- `astropy`
- `astropy_healpix`

The **frontend server** is implemented using `Flask` and has as an extra dependency `marshmallow`.

The **frontend** uses `bootstrap 4` and custom libraries written in TypeScript. The assets are managed by a Node script that compiles the TypeScript libraries.

The **documentation** uses Sphinx and is written in rst format. The Python dependencies to build the documentation are:

- `sphinx`
- `sphinx_rtd_theme`
- `recommonmark`

## 7.2 Python and Typescript development

The **Python version** can be managed with pyenv which allows the installation and use of a specific Python version for the project. This could be later used to test the code, particularly the backend, in different Python versions. The additional extension pyenv-virtualenv can be used to manage virtual environments in combination with pyenv.

After installing pyenv and pyenv-virtualenv the following commands can be executed from the project root directory to install a Python 3.7 version that will be used locally for the project. For example:

```
pyenv install 3.7.8
pyenv local 3.7.8
```

A .python-version file will be created in the root folder with the Python version that will be used.

The virtual environment and dependencies are managed by Poetry which allows the pinning of dependencies and to resolve possible conflicts. It can also prepare the local virtual environment and manage the building and distribution of the project. The configuration is written in the **pyproject.toml** file (see PEP-518).

We can set poetry to use a local virtual environment with:

```
poetry config virtualenvs.in-project true
poetry config virtualenvs.create true
```

The dependencies and virtual environment can be installed with:

```
poetry install
```

Please, note that Poetry is not using the Python registry defined by the `PIP_INDEX_URL` environment variable at the moment. This parameter is set in the `pyproject.toml` section called `tool.poetry.source`.

After that, all the commands can be run prepending **poetry run** before.

The **frontend assets** are managed with Node.js which is used, at the moment, to install and run the TypeScript tools, the testing environment `cypress`, and the Typescript tests. The Node.js versions can be managed with nvm with the latest stable LTS version being v12.18.4 (Erbium). After installing nvm, the required version of Node.js can be installed with:

```
nvm install v14.16.1
nvm use v14.16.1
```

To make npm work with the SKA Central Artefact Reository npm registry we can set the environment variable to the appropriate value:

```
export npm_config_registry=https://artefact.skao.int/repository/npm-all/
```

The TypeScript dependencies can be installed using:

```
npm install --no-cache
```

The TypeScript assets can be compiled using a Node.js script (defined in the file `package.json`; see for example this link) which can be run with:

```
npm run build
```

This command will run the necessary scripts to transpile the TypeScript code to es6 Javascript.

To run the **Flask server**

```
poetry run flask run
```

## 7.3 Documentation

Before compiling the **documentation** make sure that the libraries required are installed with:

```
poetry install -E docs
```

Then, the **documentation** can be compiled from the root directory running the following command:

```
poetry run make -C ./docs html
```

The documentation will be located in `docs/build/html` and from this directory it can be inspected locally with a command like:

```
python -m http.server 8020
```

and opening a browser tab in the address http://localhost:8020/

## 7.4 Tests with Cypress

Cypress is a testing framework that can be easily installed using Node.js. Cypress API is a chaining API. See: https://docs.cypress.io/api

Install Cypress with:

```
npm install --save-dev cypress
```

Before running cypress, the sensitivity calculator must be running using, for example, `make up`.

To open the cypress GUI:

```
npx cypress open
```

To run in a headless mode like in CI:

```
npx cypress run
```

All the tests are saved in the `cypress` directory.

## 7.5 Base Docker images

The project is installed in base Docker images that are fetched from the Central Artefact Repository (CAR). These images contain the basic dependencies for the project to run or to be tested. They are also used by the CI pipelines which are also used to create them when necessary (in the `build_base_images` step).

The creation of these images is handled by the code located in the directory called `docker`. To trigger the creation and pushing of a new release in the CI pipeline, the version of the release must be updated in the file `.release`. At the moment there are two types of images that can be created: the `production` image with just the basic dependencies, and the `e2e` image that contains some additional dependencies to help the running of the End 2 End tests with Cypress.

The creation of a new base image can be tested locally using the command:

```
make release
```

from the `docker` directory. However, the uploading to the CAR should fail as the credentials will only be available in the CI environment.

CHAPTER

EIGHT

# FURTHER WORK

The calculator, in its current prototype state, is the product of 6 months of work by one developer. It is intended to be a sufficient platform to allow the development of a more sophisticated tool. There are already several suggestions for work that needs to be done in the future. These are grouped below according to their source.

## 8.1 SKA Engineering

It has been pointed out that SKA Engineers and Astronomers define ‘sensitivity’ differently, and there is some question whether the Sensitivity Model used by the Calculator matches that already in use by the Engineers. We need to ensure that we define and use our terms carefully and that the sensitivity models are reconciled.

## 8.2 Ideas from the Prototype

These ideas cropped up during devolopment of the prototype, but were considered too time-consuming or too far off in the future to be added at that stage:

- **Shadowing.** In reality, depending on the pointing direction of the dishes in the array, the dishes are likely to obscure one another, resulting in a loss of effective area. This can be accounted for by determining a shadowing fraction (i.e. what proportion of the total area is shadowed) and reducing the effective area by a proportional amount.
- **Array configuration.** The calculator currently only operates with the full array. There is an option in the calculator to select an array configuration but it currently does nothing. In the future, the user should be able to select from a preset list of configurations.
- **Image weighting.** The type of image weighting used will affect the sensitivity one can achieve with the observation. Incorporating the Briggs robust weighting parameter into the calculation will help reflect this.
- **Beam synthesis.** Running some simulations to synthesise beams would be incredibly useful and open up a lot of other options for functionality for the calculator.
- **Weather.** Currently the user is given the options of “Good”, “Average” and “Bad” weather, corresponding to pwv values of 5.8mm, 10.7mm and 19.2mm respectively. While this is important for the calculation of the atmospheric temperature, $T_{atm}$, it is impossible for the user to predict what the weather is going to be like when their observation gets scheduled. Instead, it may make more sense to give options for different months/seasons, since then the user would at least get an idea what the weather conditions will likely be over the time their observations could be scheduled.
- **Optional smoothing for zooms.** Down the line it is probably a good idea to add a optional line smoothing option for zooms.

- **More observing modes.** The calculator currently sports two observing modes - continuum and line observations. As it is developed, it would be good to have more observing modes added. The prototype has a tab for pulsar observations (and some comments throughout the code), but there is nothing yet implemented for this mode - it is just a placeholder/suggestion.
- **Report resources.** Adding some report of the resources that will be used for the observation (e.g. compute time) would be a valuable addition to the calculator output.
- **Populate inputs from URL.** A handy feature would be if the calculator would parse the query string from the URL and preload the calculator inputs with those values. When combined with a 'link generator' feature which would be fairly straightforward to add, this would allow users to generate links to the calculations they have performed and share them with colleagues. When the colleague clicked the link/pasted it into their address bar, they would be taken to the page and the inputs would be loaded with the same values the first user had used.
- **Other Calculators.** In developing this calculator, it was useful to regularly look at other, similar calculators/tools which exist. These other tools helped inform design and inspire new feature ideas. A list of such calculators follows here, which will hopefully be of use as the calculator is further developed.
  - ALMA Sensitivity Calculator
  - ATCA Sensitivity Calculator
  - e-MERLIN Sensitivity Calculator
  - VLA Exposure Calculator

## 8.3 The Vision Thing

Would it be worth asking some people to write a (very) short story describing how they imagine they would use the SKA 'in the ideal world', especially with reference to the Sensitivity Calculator? Consider different scenarios e.g. standard observing, response to transient triggers, survey planning, whatever you can think of.

CHAPTER

# NINE

# SENSITIVITY_CALCULATOR.MID

CHAPTER

TEN

# SENSITIVITY_CALCULATOR.SUBARRAY

**Todo:**

- Insert todo's here

CHAPTER

ELEVEN

# SENSITIVITY_CALCULATOR.UTILITIES

**Todo:**

- Insert todo's here

CHAPTER

TWELVE

# SENSITIVITY_CALCULATOR.MID_UTILITIES

CHAPTER

# THIRTEEN

# TESTS.SENSITIVITY_CALCULATOR