Notes and work progress LISA

Master Project

Ester Abram

December 7, 2018

nikhef_logo.png

Chapter 1

PAA_LISA package

1.1 Orbit class

First in the 'Orbit' class (`class_orbit.py`) orbitfiles will be read. A lot of functions in this file are not used anymore (some calculations and plotting), which is done by the PAA class (`calc2.py`). It returns a LISA object which is called `self.lisa_obj` which is a SampledLISA object.

1.2 functions.py

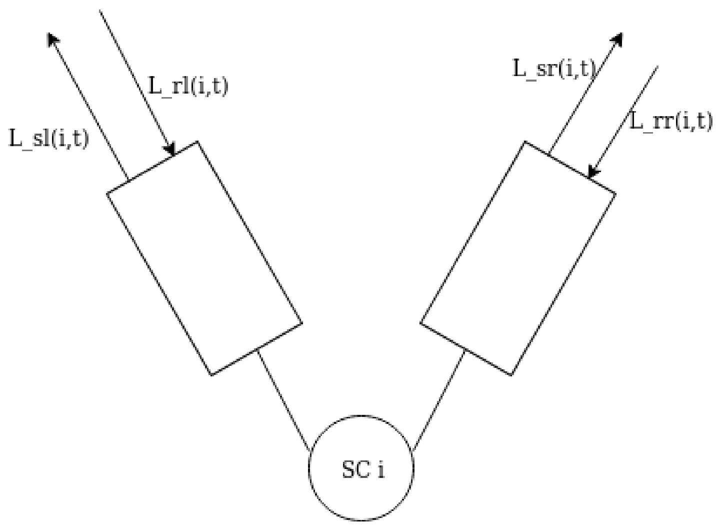
In `functions.py` various functions can been found which are used to perform (additions) calculations needed to compute the point ahead angle (PAA). In this file one class is defined and several seperate dunctions. The class `la()` contains various functions for vector calculus which are used in the seperate functions to compute the PAA.

`LISA_obj(OBJ,type_select='cache')`
This function select which kind of LISA object is being used. The default value is CacheLISA. The Orbit class creates a SampledLISA object which this function converts to either a ChachedLISA or PyLISA object (or keep using the Sampled LISA). The LISA object will be written to OBJ.LISA[1].

`func_pos(OBJ,i)`
This function returns the (absolute) position of spacecraft i as a function of time.

`solve_L_PAA(OBJ,t,pos_OBJ,pos_left,pos_right,select='sl',calc_method='Waluschka')`
This function returns the traveling time of a photon between two spacecrafts at time 't'. 'select' can be either, sl, sr, rl, or rr which stands for 'send left' (send from the left spacecraft), 'send right, 'received left' and 'received right' respectively (see figure ??). posOBJ, posleft and posright are functions of the positions of the spacecrafts.
`calc_method` can either be set on 'Waluscka' or 'Abram'. 'Waluschka' retuns dt from

[1] If this is False, the package used to work without SyntheticLISA, bit this option does not work properly anymore



solving the following equation [1]:

$$\begin{aligned}
|p(i_l,t+dt)-p(i,t+dt)|-c\cdot dt=0 \;\; \texttt{(for 'sl')}\\
|p(i_r t+dt)-p(i,t+dt)|-c\cdot dt=0 \;\; \texttt{(for 'sr')}\\
|p(i,t-dt)-p(i_l,t-dt)|-c\cdot dt=0 \;\; \texttt{(for 'rl')}\\
|p(i,t-dt)-p(i_r,t-dt)|-c\cdot dt=0 \;\; \texttt{(for 'rr')}
\end{aligned} \tag{1.1}$$

and 'Abram' returns the value for dt solved by the next set of equations:

$$\begin{aligned}
|p(i_l t+dt)-p(i,t)|-c\cdot dt=0 \;\; \texttt{(for 'sl')}\\
|p(i_r t+dt)-p(i,t)|-c\cdot dt=0 \;\; \texttt{(for 'sr')}\\
|p(i,t)-p(i_l,t-dt)|-c\cdot dt=0 \;\; \texttt{(for 'rl')}\\
|p(i,t)-p(i_r,t-dt)|-c\cdot dt=0 \;\; \texttt{(for 'rr')}
\end{aligned} \tag{1.2}$$

$p(q,t)$ is he position vector op spacecraft q at time t, i is de spacecraft nummer and 1_l and i_r the numbers of the accompanying left and right spacecrafts. c is the speed of light and dt is the armlengthe in seconds, which is the traveling time of a photon between two spacecrafts.

`L_PAA(OBJ,pos_OBJ,pos_left,pos_right,calc_method='Walushka')`
This function calls function `solve_L_PAA` to calculate the armlength and returns it as a function over time for the spacecraft with position `pos_OBJ`. This is done for all four laserbeams which results in `[L_sl,L_sr,L_rl,L_rr]`, which are the armlengths in seconds for sl, sr, rl, rr respectively.
`send_func(OBJ,i,calc_method='Waluschka')`
This function uses `L_PAA` (the armlengths) to compute the a function of the beam vectors `v_send_l`. `v_send_r`, `v_rec_l` and `v_rec_r`. The geometric definitions are shown in figure ??. According to the 'Waluschka' method they hold the following

equations:

$$
\begin{aligned}
v_send_l(i,t) &= p(i_l, t + L_{sl}(i,t)) - p(i, t + L_{sl}(i,t)) \\
v_send_r(i,t) &= p(i_r, t + L_{sr}(i,t)) - p(i, t + L_{sr}(i,t)) \\
v_rec_l(i,t) &= p(i, t - L_{rl}(i,t)) - p(i_l, t - L_{rl}(i,t)) \\
v_rec_r(i,t) &= p(i, t - L_{rr}(i,t)) - p(i, t - L_{rr}(i,t))
\end{aligned} \tag{1.3}
$$

and according to the 'Abram' method this is:

$$
\begin{aligned}
v_send_l(i,t) &= p(i_l t + L_{sl}(i,t)) - p(i,t) \\
v_send_r(i,t) &= p(i_r, t + L_{sr}(i,t)) - p(i,t)) \\
v_rec_l(i,t) &= p(i,t) - p(i_l, t - L_{rl}(i,t)) \\
v_rec_r(i,t) &= p(i,t) - p(i, t - L_{rr}(i,t))
\end{aligned} \tag{1.4}
$$

When OBJ.delay is set on False, the beam propagation time is set to 0 and if it is equal to 'constant' it is set on $\frac{2500000000\ m}{c}$.

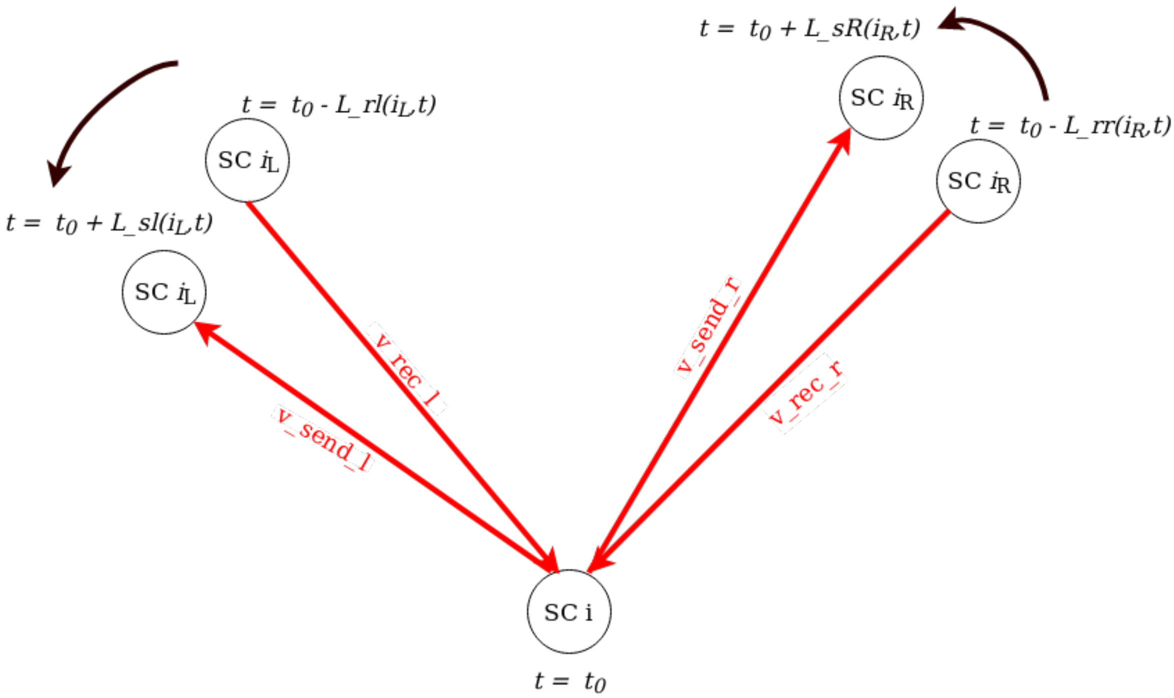


Figure 1.1: The four vectors received or emitted bij spacecraft i at $t = t_0$. There are in total 12 vectors to be considered concertning every spacecraft.

`calc_PAA_lin(OBJ,i,t)`, `calc_PAA_lout(OBJ,i,t)`, `calc_PAA_rin(OBJ,i,t)` and `calc_PAA_rout(`
These functions are to adjust the incoming beams for abberation efffect (*...This function does not work properly at this moment!*
There are more functions defined in `functions.py` which are not mentioned before, because they are straight forward or not that important. However there are also some functions which calculate the velocity of the spacecraft (absolute and relative to each other. Which gives a great geometrical overview and can be used to check if some off

the calculations using the beam vectors match those of its estimates obtained by using the velocities.

1.3 calc2.py

In `calc2.py`, the Orbit class is called which creates a LISA object. Together with some functions defined in `functions.py` it calculates some properties like the point ahead angle and obtains those as functions of time (and spacecraft).

`calc2.py` consists of a class `PAA()` which holds all calculated property functions:

- `self.L_sl_func_tot(i,t)`, `self.L_sr_func_tot(i,t)`, `self.L_rl_func_tot(i,t)` and `self.L_rr_func_tot(i,t)` are the time delay (armlength/propagation time) functions.

- `self.v_l_func_tot(i,t)`, `self.u_l_func_tot(i,t)`, `self.v_r_func_tot(i,t)` and `self.u_r_func_tot(i,t)` are the beam vectors[2].

- `self.ang_breathing_stat` and `self.ang_breathing_din` are the static and dynamic breating angles (see figure **??**).

- `self.PAA_func_val` are the calculated point ahead angles. This is a dictionary with the keys '`l_in`', '`l_out`', '`r_in`' and '`r_out`' which is the point ahead angle decomposed in a inplane and out-of-plane part (see figure **??**)

- `self.n_func{i,t}` and `self.r_func{i,t}` are the normal vector and teh vector spanned between the center of mass (COM) of the constellation and spacecraft i. These vectors are used to decompose vectors in an inplane and out op plane component (see figure **??**).

- `self.X_Y_Z_func_tot` where X is either `v` or `u`, `Y` is `l` or `r` and `Z` is `in` or `out`. These are the beam vectors decomposed in inplane and out of plane components.

The point ahead mechanism in every telescope should compensate for the PAA, but only compensates it for the out of plane component. The telescope will be actuated in line with the incoming beam, but only in the inplane direction.
The inplane components are defined by defining a plane (see figure **??**). This plane is spanned by vector $\mathbf{v}_{\mathbf{l_{stat}}}$ and $\mathbf{v}_{\mathbf{rstat}}$. Its normal $\mathbf{n}$ and the inplane vector $\mathbf{r}$ are used as reference vectors for the orientation of the telescope, PAAM (and spacecraft)[3].

---

[2] u represents an incoming and v an outgoing beam. Compared to figure **??** those correspond to $\mathrm{v}_rec; and \mathrm{v}_send; respectively.$

[3] In reality the inplane is spanned by the telescope pointing of each spacecraft (so the plane can be different for each spacecraft). The difference in this plane and the plane defined here is unknown and perhaps negligible. It would be useful to also get the pointing orientation from the imported orbit files.

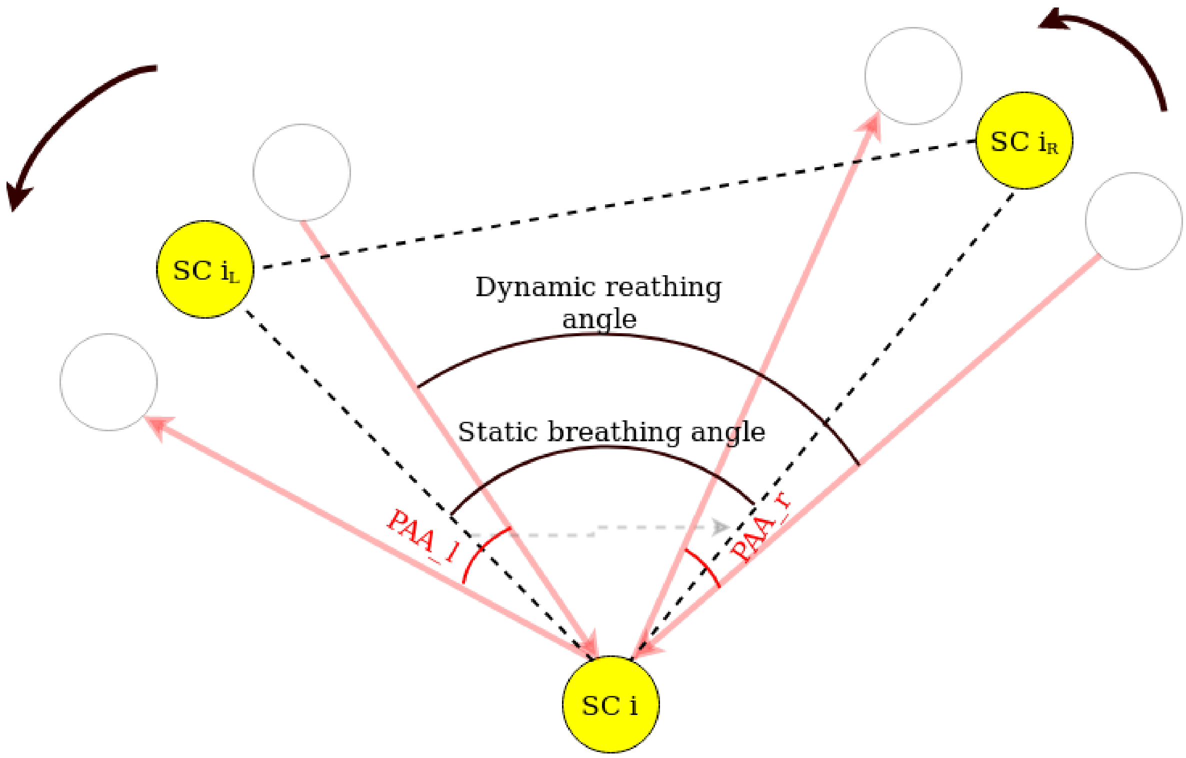


Figure 1.2: The geometric definition of the angels mentioned. The yellow circles are the spacecrafts at the same moment in time. The point ahead angle is the angle between the incoming and outgoing beam of either the left or right side (per telescope). The static breathing angle is the angle between the position vectors v_1 and v_2 (see figure **??**. The dynamic breathing angle is the angle between the incoming beam on one of the two telescopes with the other.

$$\vec{r} = \frac{m_{i_l} \cdot \overrightarrow{v_{lstat}} + m_{i_r} \cdot \overrightarrow{v_{rstat}}}{m_i + m_{i_l} + m_{i_r}} \tag{1.5}$$

$$\vec{n} = \frac{\overrightarrow{v_{rstat}} \times \overrightarrow{v_{lstat}}}{|\overrightarrow{v_{rstat}}||\overrightarrow{v_{lstat}}|} \tag{1.6}$$

...

1.4 runfile.py

In runfile.py the intire package can be runned, including a plot option. It returns an object called `data` which is a dictionary including all calculated variables and function mentioned before per key. Each key belongs to one orbit file.

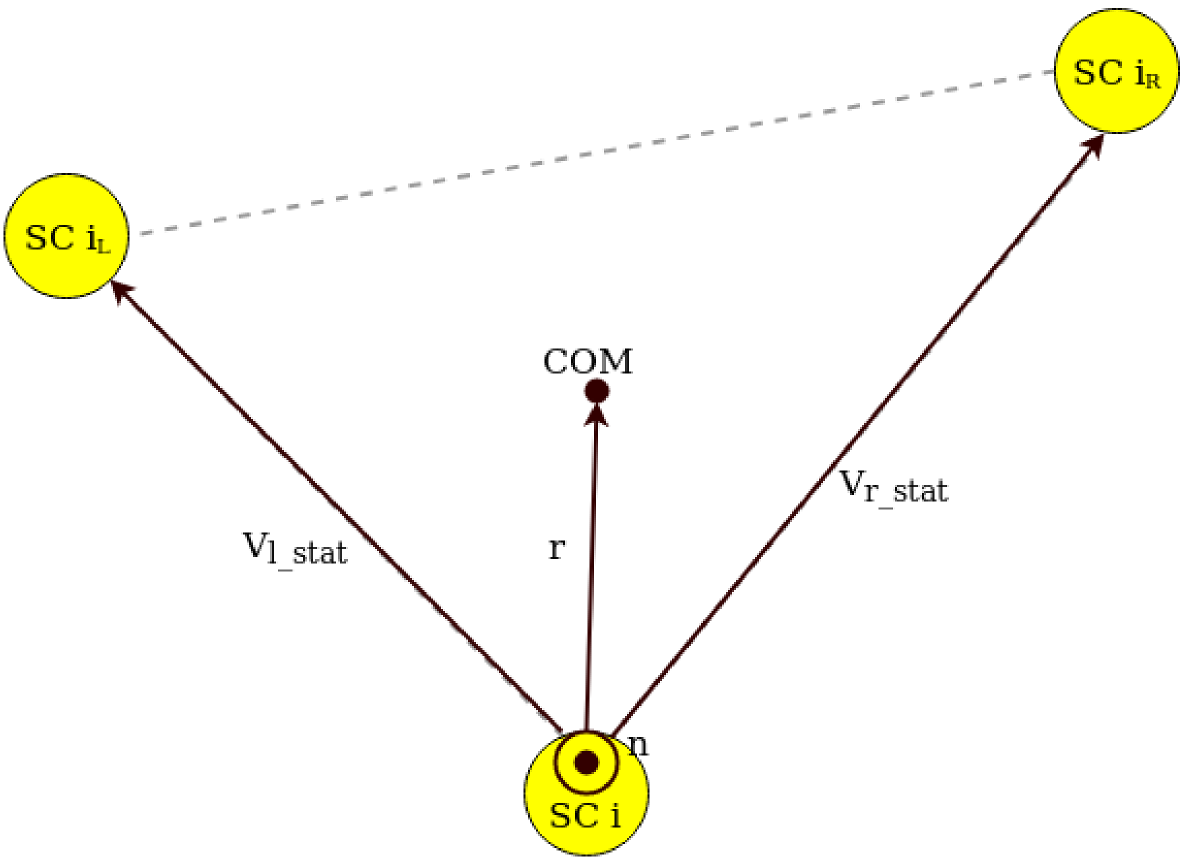


Figure 1.3: Drawing of the

Chapter 2

NOISELISA

In the `NOISE_LISA` package, the TDI variables are calculated and different Noise sources are simulated. Also it includes functions that define the controlling of the PAAM and the telescopes.

2.1 calc.py

In `calc.py` two classes can be found: `Noise` and `TDI`

2.1.1 class Noise

In this class different noise sources are being simulated/sampled. Because of the controlling of the PAAM (and telescope) this noise will vary over time. Therefore the laser and shotnoise is obtained from getting their PSD (`Nose.Noise(self,f0,f_max,N,psd,unit='freq')`) and sample from it (`Noise.Noise_time(self,f0,f_max,N,psd,t_stop,unit='freq',t=False)`. In `Noise.lasernoise(self,PSD,f0=1e-6,f_max=1e-3,N=4096)` and `Noise.shotnoise(self)` the y and z variables (used to compute the various TDI-variables) of those noise sources are obtained (...insert source and equations, y and z are phases).

(`Noise.tele_control(self,i,time,option='full control',dt=1,side='l')`)// This function obtains the position (aim) of the telescopes. It can be set to different control methods:

- 'full control'. This option aligns the telescope with the incoming beam. Therefore the beam will always be centered and perpendicular to the telescope.
- 'no control'. Here the telescopes will not move (be actuated) and will stay at a $30°$ angle from $\overrightarrow{r}$
- In `WFE` there is another function: `tele_control_noise` which obtains a step an stair control of the telescope (with overshoot). Here only movement inplane is concidered.

`PAAM_noise(self,C_func,C_func_star)` In this function the optical path delay (OPD) as function of the PAA angle is calculated. This is coupled to the lasernoise and returns `y_PAAM` (the phase fluctuation due to the OPD differences). It uses the function

`PAA_control(self)` to compute the 'real' PAA angle α the PAAM has to compensate over:

$$\alpha = \frac{1}{2} PAA \cdot MAGNIFICATION \tag{2.1}$$

The magnification of the telescope is 135× (...source).

2.1.2 class TDI

...

2.2 WFE.py

In this file, the wavefront error is obtained. A lot of different jitters and components can influence this which are also simulated in this class.
The beamshape can be assumed to be gaussian. Because it will reach the next telescope over a distance of approximately 2.5 million kilometer, the wafefront is (almost) spherical. The diffraction integral is (...source):

$$A(X,Y,Z) \cdot e^{i\frac{2\pi}{\lambda}R} \cdot e^{i\frac{2\pi}{\lambda}\psi(X,Y,Z)} = \int\int E(x,y,z) \frac{e^{i\frac{2\pi}{lambda}(Z_m+S)}}{S} dxdy \tag{2.2}$$

Where X, Y and Z are the coordinates of receiving and x,y and z of the transmitting wavefront. S is the difference between those points and is coupled to the telescope pointing. Z_m is the relative jitter between telescopes.

2.2.1 Telescope jitter and pointing

`WFE.jitter_tele(self,N,t_end,psd_h,psd_v)`
This function uses two PSD fuctions: One for the inplane and one for the out of plane jitter. It samples over it and returns the jitter in the time domain.
`;WFE.tele_control_noise(self,i,step_max=False,dt=1,side=l`');
This class uses a transfer function, which represents the transfer function of the telescope servo). It simulates the response to a stap and stair control of the telescope. Hereby one can simulate a control error, overshoot and rise time. Because of this the tilt of the telescope compared to the laserbeamns is simulated properly[1].
`WFE.tele_control_ss(self,step_max=False,dt=1)`
...

2.2.2 Send wavefront

To model outgoing beam, zernike polynomials have been used (...source Sasso).
`WFE` properties

[1] Currently the transfer function does not match the real telescope that well. Also it is renewed all the time which is not a proper method

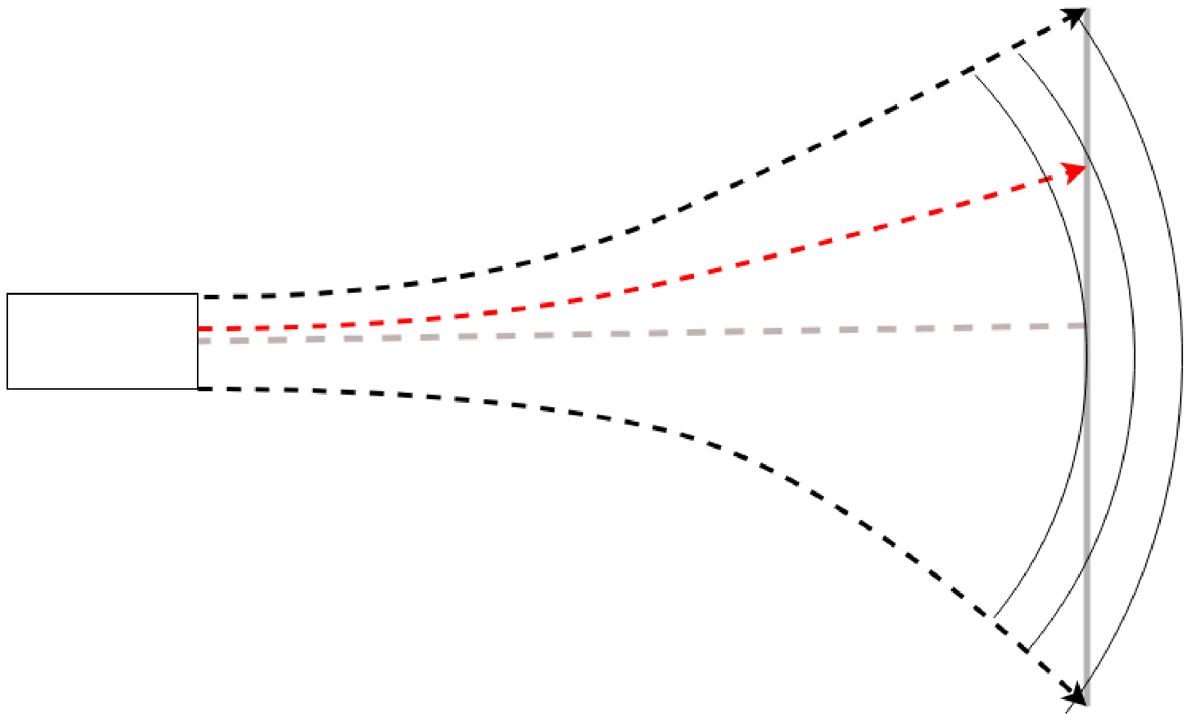


Figure 2.1: Sketch of the wave front. Even when the telescope is perfectly aimed, there is a phase difference becaus of the 'flat' aperture compared to the spherical beam. At distances as far as the inter spacecraft distance, these phase differences are neglectible.

- `WFE.tele_SS_l(i,t)` is the function of the telescope pointing with the step and stair method (at this moment the jitter is added later instead of at this property).
-

Bibliography

[1] Eugene Waluschka. Lisa optics model. *Classical and Quantum Gravity*, 20(10):S171, 2003.