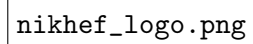


Notes and work progress LISA

Master Project

Ester Abram

January 10, 2019

A rectangular box containing the text 'nikhef_logo.png', which serves as a placeholder for the logo image.

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 be 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 separate functions. The class `la()` contains various functions for vector calculus which are used in the separate 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.

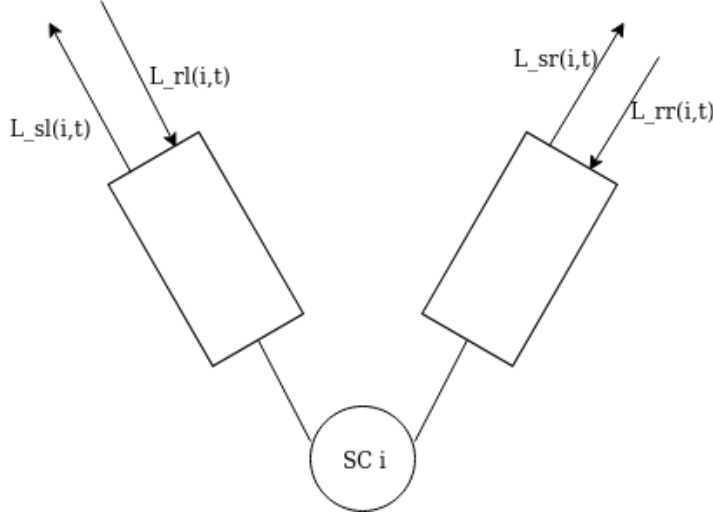
`func_pos(OBJ,i)`

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'` returns `dt` from

¹If this is `False`, the package used to work without `SyntheticLISA`, but this option does not work properly anymore



solving the following equation [1]:

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

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

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

$p(q, t)$ is the position vector of spacecraft q at time t , i is the spacecraft number and i_l and i_r the numbers of the accompanying left and right spacecrafts. c is the speed of light and dt is the armlength 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:

$$\begin{aligned}
v_{send_l}(i, t) &= p(i_l, t + L_{sl}(i, t)) - p(i, t + L_{sl}(i, t)) \\
v_{send_r}(i, t) &= p(i_r, t + L_{sr}(i, t)) - p(i, t + L_{sr}(i, t)) \\
v_{rec_l}(i, t) &= p(i, t - L_{rl}(i, t)) - p(i_l, t - L_{rl}(i, t)) \\
v_{rec_r}(i, t) &= p(i, t - L_{rr}(i, t)) - p(i_r, t - L_{rr}(i, t))
\end{aligned} \tag{1.3}$$

and according to the 'Abram' method this is:

$$\begin{aligned}
v_{send_l}(i, t) &= p(i_l, t + L_{sl}(i, t)) - p(i, t) \\
v_{send_r}(i, t) &= p(i_r, t + L_{sr}(i, t)) - p(i, t) \\
v_{rec_l}(i, t) &= p(i, t) - p(i_l, t - L_{rl}(i, t)) \\
v_{rec_r}(i, t) &= p(i, t) - p(i_r, t - L_{rr}(i, t))
\end{aligned} \tag{1.4}$$

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 $\frac{25000000000}{c}$ m.

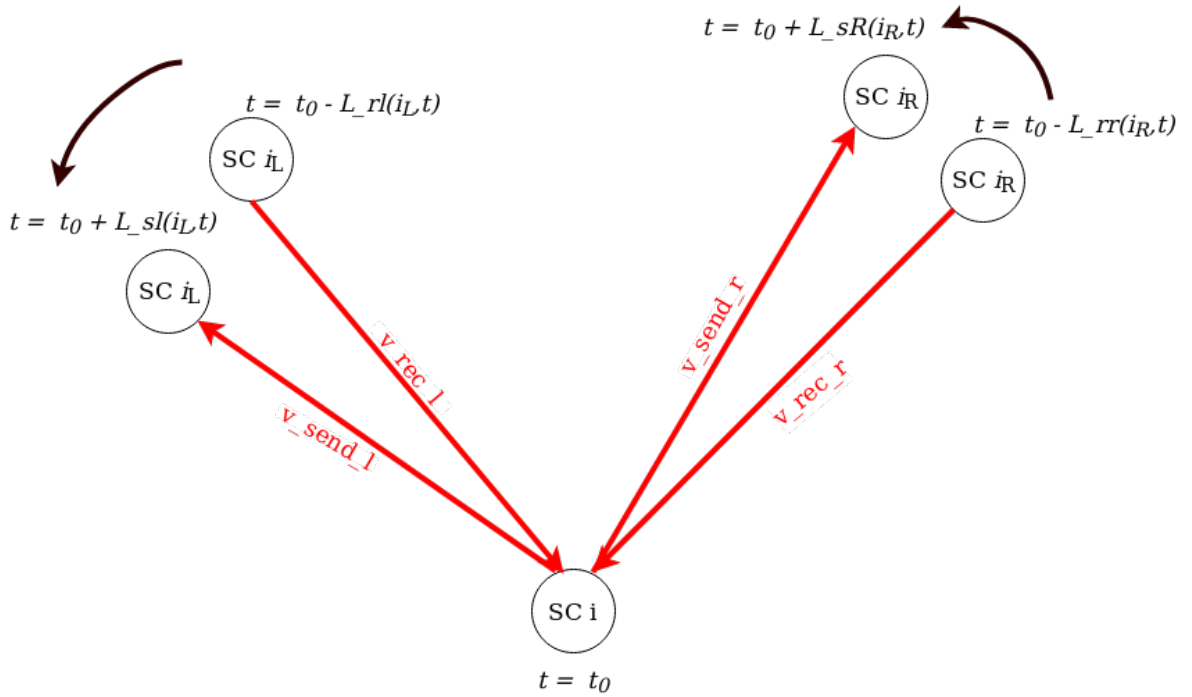


Figure 1.1: The four vectors received or emitted by spacecraft i at $t = t_0$. There are in total 12 vectors to be considered concerning every spacecraft.

`calc_PAA_lin(OBJ, i, t)`, `calc_PAA_lout(OBJ, i, t)`, `calc_PAA_rin(OBJ, i, t)` and `calc_PAA_rout(OBJ, i, t)`. These functions are to adjust the incoming beams for aberration effect (...*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 vectors².
- `self.ang_breathing_stat` and `self.ang_breathing_din` are the static and dynamic breathing 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}_{lstat} and \mathbf{v}_{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 *v_{rec}*; and *v_{send}*; respectively.

³In 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.

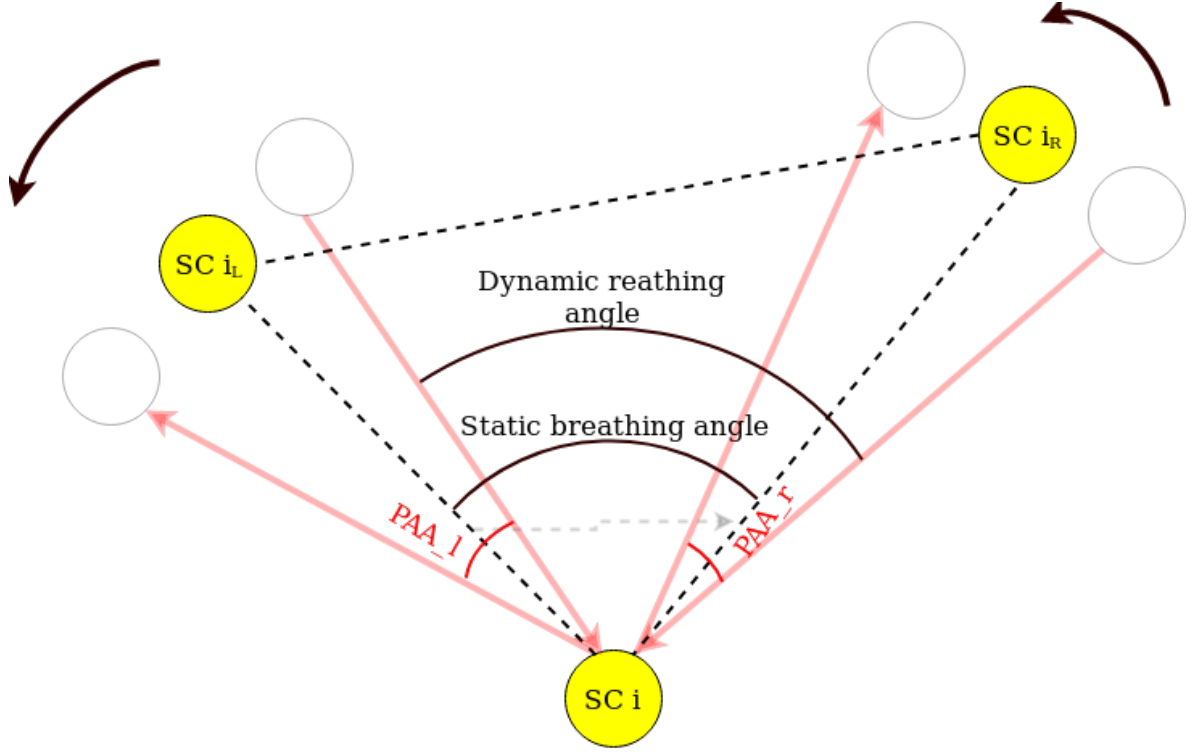


Figure 1.2: The geometric definition of the angels mentioned. The yellow circles are the spacecrafts at the same moment in time. The point ahead angle is the angle between the incoming and outgoing beam of either the left or right side (per telescope). The static breathing angle is the angle between 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.

$$\vec{r} = \frac{m_{i_l} \cdot \vec{v_{lstat}} + m_{i_r} \cdot \vec{v_{rstat}}}{m_i + m_{i_l} + m_{i_r}} \quad (1.5)$$

$$\vec{n} = \frac{\vec{v_{rstat}} \times \vec{v_{lstat}}}{|\vec{v_{rstat}}| |\vec{v_{lstat}}|} \quad (1.6)$$

...

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.

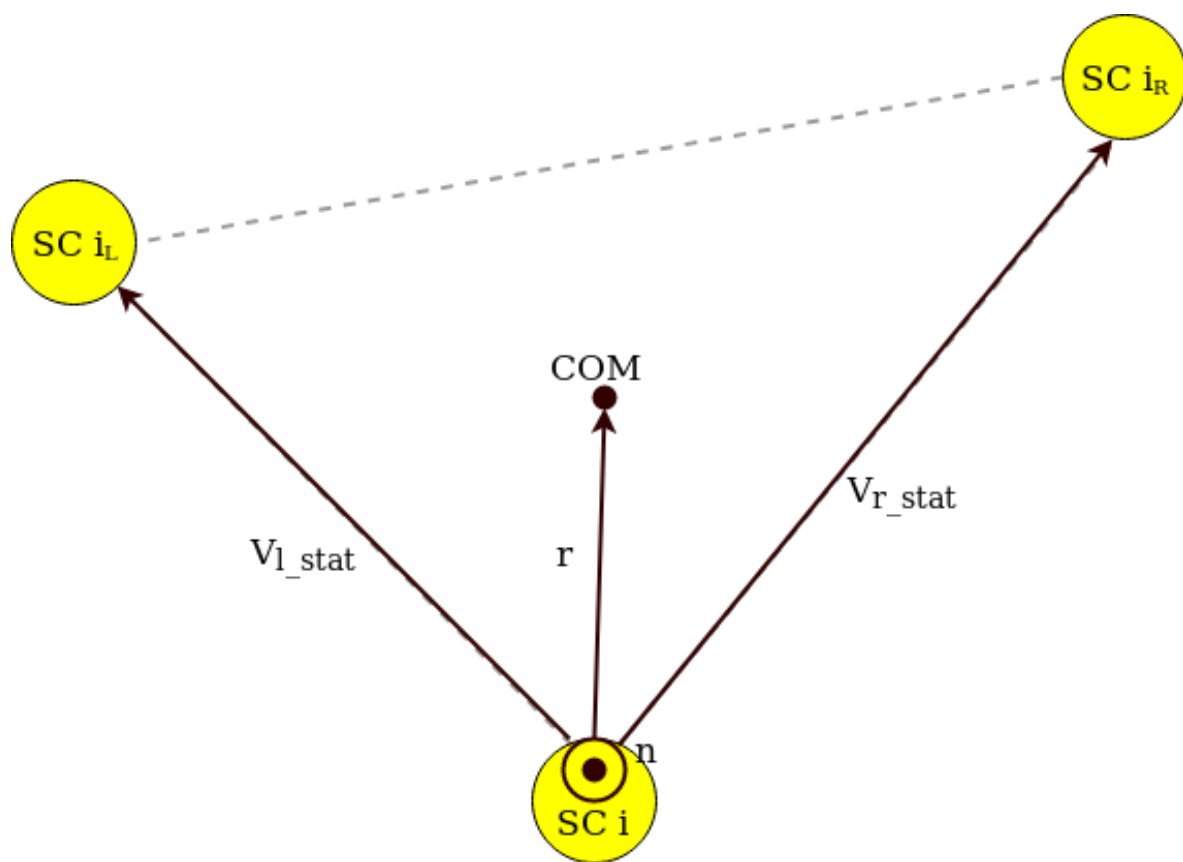


Figure 1.3: Drawing of the

Chapter 2

NOISE_LISA

In the `NOISE_LISA` package, the TDI variables are calculated and different Noise sources are simulated. It also 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 (`Noise.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 \vec{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 considered.

`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:

$$\alpha = \frac{1}{2}PAA \cdot MAGNIFICATION \quad (2.1)$$

The magnification of the telescope is $135\times$ (...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 wavefront is (almost) spherical. The diffraction integral is (...source):

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

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 the telescopes.

2.3 Wavefront Error

As mentioned earlier, accurate pointing of the beam is very important and a great challenge when designing LISA. According to Sasso (...) the noise power density has to be below $1/pm^2Hz^{-1}$ which results in a requirement of the noise power density of the of the interference-signal phase of $1\mu rad^2Hz^{-1}$, so therefore one has to look at the phase stability of the received wavefront. Therefore the beam properties, transmitted wavefront and its far-field propagation were simulated to investigate its phasenoise (and intensity).

2.3.1 Beam properties

It can be assumed that the beams exiting the telescopes are gaussian (...source). Therefore its waist follows 2.3, radius of curvature 2.4.

$$w(z) = w_0 \sqrt{1 + \frac{z^2}{z_R^2}} \quad z_R = \frac{\pi w_0^2}{\lambda} \quad (2.3)$$

$$R(z) = z \left(1 + \left(\frac{z}{z_R} \right)^2 \right) \quad (2.4)$$

At the transmitted telescope aperture, the beam intensity looks flat and at the receiving telescope ($\approx 2.5 \cdot 10^9/m$ distance) it can be assumed that received wavefront is spherical (with a radius approximately equal to traveled distance of the beam. Although the telescope aperture is relatively small ($D = 0.24 \text{ m}$ (...source)). Due to the pointing of the PAAM's and telescopes (see subsection 2.3.4) the beam is not perfectly aligned with the transmitting and receiving telescope aperture. This will result in an effective 'tilt' in both telescopes, which will be expressed in first order zernike polynomials according to equation 2.5 ¹.

$$z_{11} = |\vec{\alpha}| \theta_{11} = \angle \vec{\alpha} \quad (2.5)$$

where $\vec{\alpha} = [\alpha_x, \alpha_y]$, so the vector composed of the angles of the tilt around the y and x-axis respectively ².

2.3.2 Transmitted wavefront

To a first approximation (see...jitter), the tilt of the transmitted wavefront $\vec{\alpha}_t$ only has an out of plane component due to the PAAM³. The transmitted wavefront error (w_0) in radians will be equal to:

$$w_0 = w(0, \vec{\xi}) = \frac{2\pi}{\lambda} \vec{\xi} \cdot \begin{bmatrix} \sin \alpha_x \\ \sin \alpha_y \end{bmatrix} \quad (2.6)$$

For every point on the receiving aperture, one has to integrate over the transmitted wavefront to calculate the far-field amplitude of the transmitted beam $u(\vec{r}, z)$ and therefore obtain the received wavefront error and intensity. The paraxial propagation of light in free space is given by the Rayleigh-Sommerfield [] equation. Because of the great distance between the telescopes $k\xi^2/(2z) \ll w_0(\vec{x})$

$$u(\vec{r}, z) \approx \frac{ike^{-\frac{ikr^2}{2z}}}{2\pi z} \int_A e^{\frac{ik\vec{r} \cdot \vec{\xi}}{z}} u_0 \xi e^{i w_0(\xi)} d\xi \quad (2.7)$$

where $u_0(\vec{\xi}) = e^{-\frac{r^2}{w^2}}$ is the complex amplitude of the monochromatic optical field propagation of a Gaussian beam⁴.

2.3.3 Received wavefront

To obtain $u(\vec{r}, z)$ one has to obtain proper values for $\vec{r} ([x, y])$ and z first.

The photons that will hit the aperture further away from the center, will be 'delayed' because of the effective difference in distance as is shown in figure ?? (marked with

¹These expressions for the zernike polynomials are different from the one used in [?]

²Note: α_x is the angle rotated around the y-axis and α_y around the x-axis.

³There will also be an inplane component due to the (PAAM) jitter between the telescope and optical bench (PAAM).

⁴Monochromatic optical field propagation between spacecrafts: $E(\vec{r}, z; t) = u(\vec{r}, z) e^{-i(kz - \omega t)}$.

z_1 and z_2). When the receiving aperture is perpendicular to the beam it receives, this difference is appeared to be negligible ($< 10^{-64} \text{ rad}$). However, the beam is not perfectly aligned with the receiving telescope due to the pointing of the telescopes and PAAM (see 2.3.4). Therefore one can say that the receiving wavefront is 'tilted', which can be expressed in zero 2.8 and first order zernike polynomials 2.5.

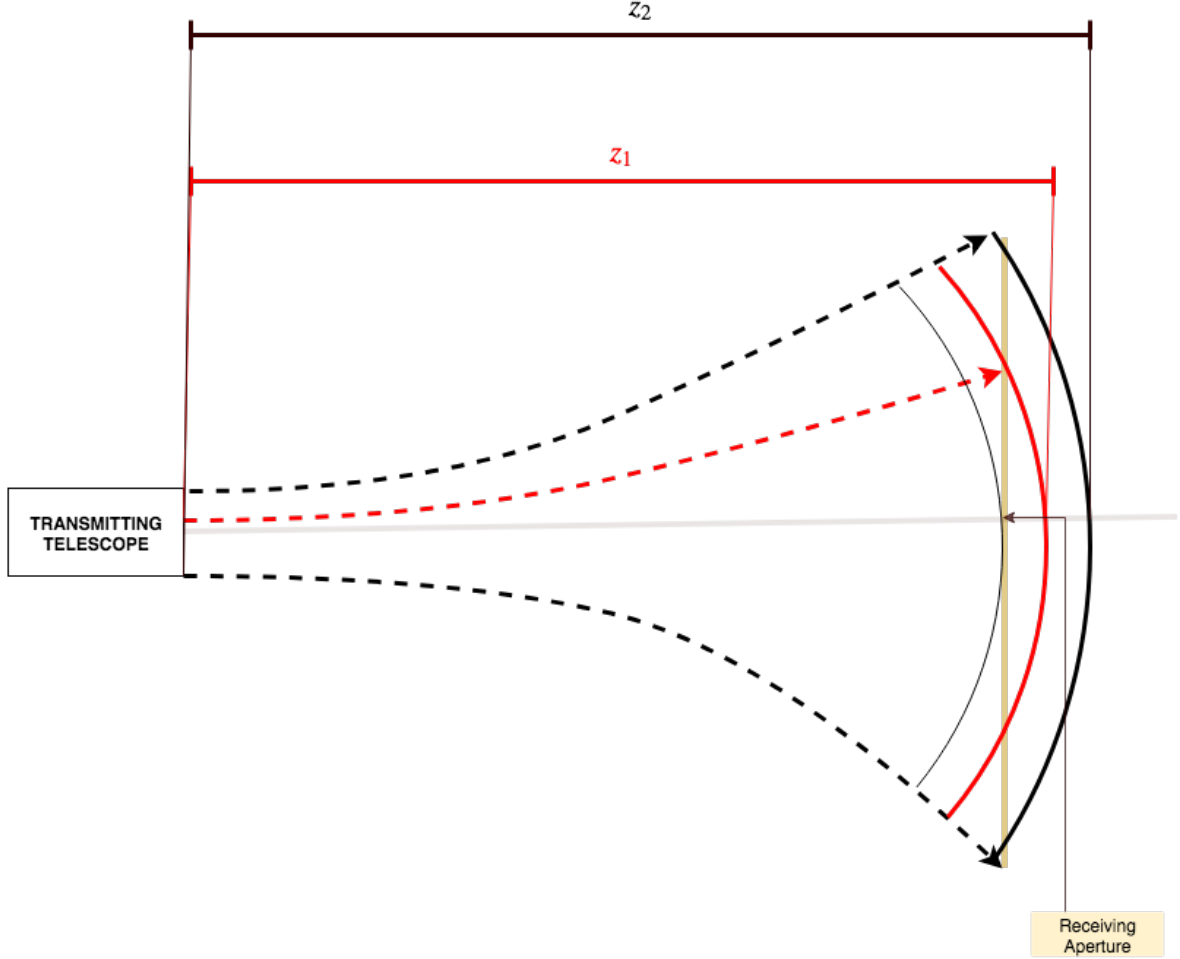


Figure 2.1: Sketch of the wave front. Even when the telescope is perfectly aimed, there is a phase difference because of the 'flat' aperture compared to the spherical beam. At distances as far as the inter spacecraft distance, these phase differences are negligible.

$$z_{00} = z \quad (\text{piston}) \quad (2.8)$$

To obtain the aperture coordinates $\vec{r} = [x, y]$ from the initial coordinates (x_0, y_0, z_0) ,

the following equations were used (see also figure ??)

$$x = x_0 \cdot \cos(\alpha_x) \quad (2.9)$$

$$y = y_0 \cdot \cos(\alpha_y) \quad (2.10)$$

$$z = z_0 + x_0 \cdot \sin(\alpha_x) + y_0 \cdot \sin(\alpha_y) \quad (2.11)$$

So now z_{00} can be obtained from z and \vec{r} from x and y .

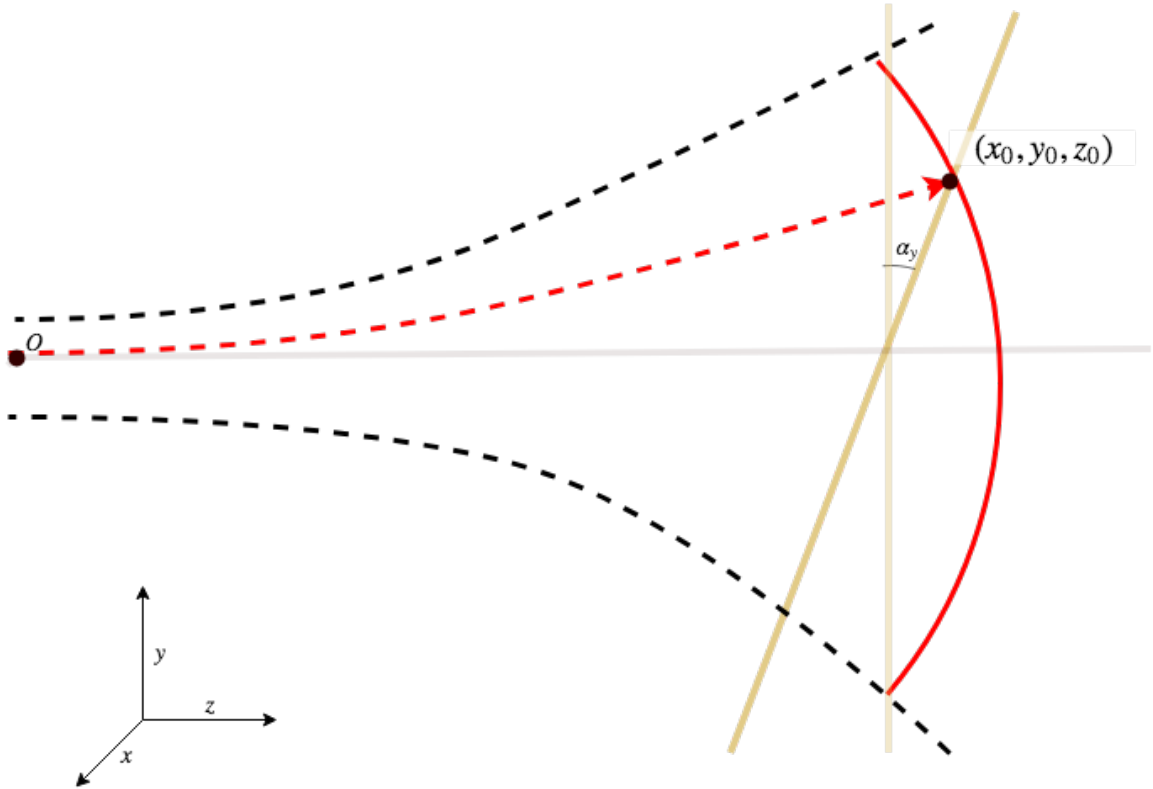


Figure 2.2: Sketch of the received and wave front with a tilted aperture. The z -direction is perpendicular to the beam, the x direction is in the inplane and y out of plane. Here only α_y is drawn but there is an α_x as well.

2.3.4 PAAM and Telescope pointing and control

The PAAM can only be actuated in the out of plane direction and the telescopes only in the plane spanned by the spacecrafts. In PAA_LISA the desired telescope and PAAM angles are obtained.

`class AIM()`

In `control.py` the class `AIM()` is defined. In this class the rotation angles, pointing vectors and coordinate systems of the PAAM and telescopes are obtained.

`tele_aim(method=False,dt=3600*24*10,jitter=False,tau=3600*24*5,mode='overdamped')`

In this function first the angle between \vec{r} and the telescope is obtained according to three options:

- 'full control': Hereby the angle is equal to the angle of the projection of `PAA_NOISE.u_l_func+tot` (or `PAA_NOISE.u_r_func+tot(i,t)` in the plane spanned by the spacecrafts and \vec{r} . This is the optimum pointing angle
- 'no control': There is now actuation of the telescopes. So the angle is set constant ($\pm 30^\circ$)
- 'SS': This is for the step and stair actuation mode. After every `dt` seconds, the angles is set on the full controlled one for another `dt` seconds (`AIM.tele_l_ang_SS(i,t)` or `AIM.tele_r_ang_SS(i,t)`). By the use of `AIM.step_response(function,dt,tau=3600,mode='overdamped')` a method for the simulation of the overdamped SS (step) control is implemented. Each step is modified by:

$$y_{new}(t) = y_{old}(t_1) + (y_{old}(t_0) - y_{old}(t_1)) \cdot e^{-\frac{t-t_0}{\tau}} \quad (2.12)$$

where t_0 is the time at the beginning of the step, t_1 at the end and τ the damp- ing coefficient. y_{old} is the ideal step and stair response and y_{new} the adjusted overdamped one.

After obtaining the telescope angles (`AIM.tele_l_ang(i,t)/AIM.tele_r_ang(i,t)`), also their pointing vectors (`AIM.tele_l_vec(i,t)/AIM.tele_r_vec(i,t)`) and coordinate systems (`AIM.tele_l_coor(i,t)/AIM.tele_r_coor(i,t)`) are obtained.

`PAAM_control(method=False,dt=3600*24,jitter=False,tau=3600*12,mode='overdamped')`

This function works the same as `tele_aim`. First it wil compute its PAAM angle, which is the out of plane pointing of the PAAM's (`PAA_LISA.PAA_func['l_out'](i,t)/PAA_LISA.PAA_func['l_out'](i,t)`). Again 'full control', 'no control' and 'SS' are the methods to choose from. After obtaining the 'real' (actuated) PAAM angles, the beam pointing vectors (`AIM.beam_l_vec(i,t)/AIM.beam_r_vec(i,t)`) and their coordinate systems (`AIM.beam_l_coor(i,t)/AIM.beam_r_coor(i,t)`) are obtained. This calculation is slightly more complicated because first the telescope has to be pointed and when taking the telescope pointing as a reference frame, the beam is rotated by the 'real' PAAM angle in the out of plane direction (rotation over the \vec{x}).

Beam and telescope misalignment

As discussed in section ??, there is a non negligible variation in the angle spanned by the two telescopes on the same spacecraft. For the Halloin-orbit [], this angle fluctuates over approximately 0.025 radians (so ≈ 0.0125 radians for each telescope with the centered vector \vec{r}). Because the incoming beam is pointed to the center of mass of the receiving spacecraft (...zoek op), this will result in a misalignment (pointing error) between the beam and telescope as can be seen in figure 2.3.

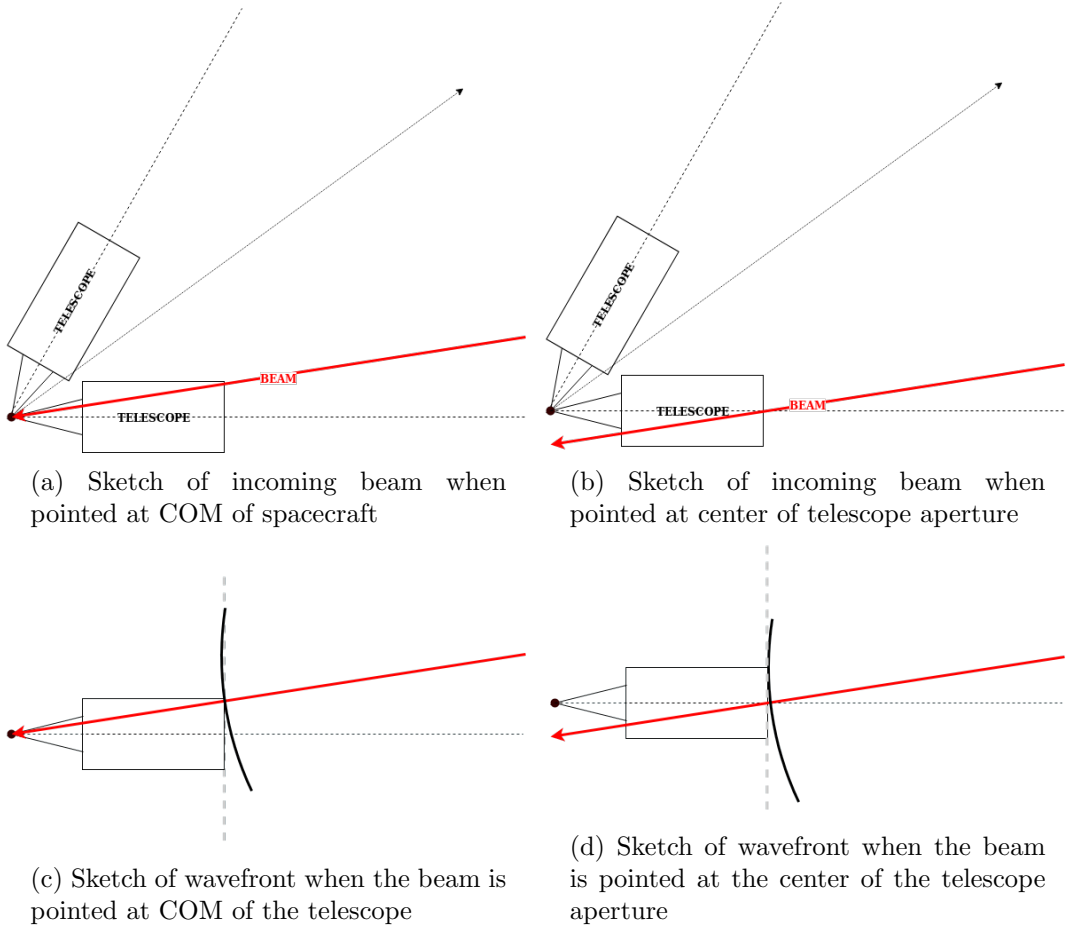


Figure 2.3: A sketch of the wavefront when the beam is pointed at the COM of the spacecraft (?? and ??) and when the beam is pointed at the center of the telescope aperture (?? and ??). As one can see when comparing figure ?? and ??, an extra tilt of the wavefront will occur when the beam is not centered with the telescope aperture.

Due to this pointing 'error', the pointing offset ($[x_{off}, y_{off}, z_{off}]$) will cause an additional tilt and will influence the piston as well. This leads to new (adjusted) zernike polynomials which are calculated in the function `WFE.zern_aim(i,t,side='l')` which computes first the pointing offset according to equations 2.13 and figure ??.

$$\vec{t}_b = \mathbf{B}\vec{t} \quad (2.13)$$

$$\vec{t}_{proj,b} = (\vec{t}_b \cdot \vec{b}) \frac{\vec{b}}{|\vec{b}|^2} \quad (2.14)$$

$$[x_{off}, y_{off}, z_{off}] = \vec{b} + \vec{t}_{proj,b} \quad (2.15)$$

$$\vec{t}_b = [t_r, t_n, t_x], \quad \vec{t}_{b,in} = [t_x, t_r] \text{ and } \vec{t}_{b,out} = [t_n, t_r] \quad (2.16)$$

$$\vec{\alpha} = \begin{bmatrix} \tan^{-1}(t_x/t_r) \\ \tan^{-1}(t_n/t_r) \end{bmatrix} \quad (2.17)$$

The matrix \mathbf{B} is the unit transformation matrix. Multiplying a vector in the COM frame with this matrix will result in the vector in the frame of the beam (transmitting telescope)

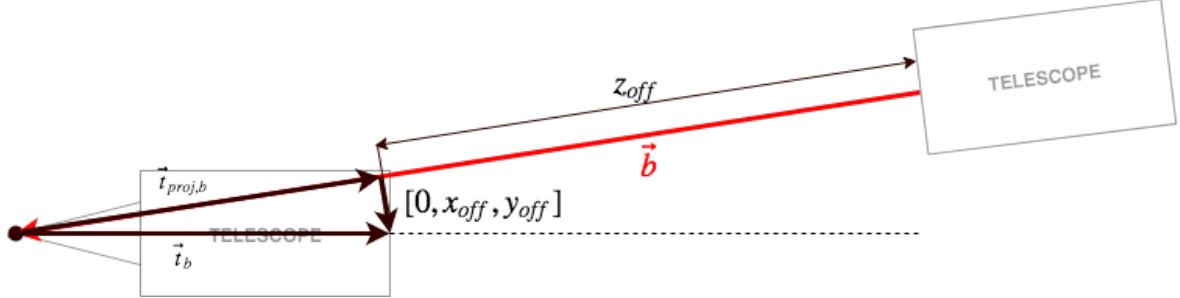


Figure 2.4: Definition of the vectors and coordinates used in equations 2.13 for obtaining the offset of the beam with the center of the telescope aperture

2.3.5 Piston and Tilts

2.3.6 TTL

So by obtaining the pointing of the telescopes and PAAM's, zero and first order zernike polynomials are obtained which results in a piston and two tilt expression according to equation 2.8 and 2.5.

2.3.7 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 properly⁵.

```
WFE.tele_control_ss(self,step_max=False,dt=1)
```

...

2.3.8 Send wavefront

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

WFE properties

⁵Currently the transfer function does not match the real telescope that well. Also it is renewed all the time which is not a proper method

- `WFE.tele_SS_1(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.