Optickle: Function Reference

M. Evans

December 5, 2007

Abstract

Optickle is a general model for the electro-opto-mechanical part of an interferometric GW detector. It ventures into mechanics only as far as is necessary to include radiation pressure effects, and into electronics only far enough to produce demodulation signals, and into optics only up to first order. There are many other tools that do all these things in greater detail. Optickle is for quick, rough, but essentially complete interferometer design studies.

1 Introduction

As discussed elsewhere (to be written), Optickle can, in principal, simulate any interferometer. But to do so one must know how to configure it, and how to run it. This document provides a list of functions, ordered functionally for each class. All but two of these are types of optics used in Optickle, the other two are the Optickle class and the OpHG class. A brief description of what each function does is also given, for a more complete description use the Matlab *help* command.

2 The Optickle Class

An instance of the Optickle class represents a particular interferometer configuration. To simulate a Fabry-Perot cavity, for example, you start by creating an Optickle instance, which you then build into a cavity by adding a source, mirrors, etc. Once built, an Optickle instance contains the constituents of the system it represents. These are the optics which make up the system, the links that tie them together, and the probes which define how information is to be extracted from the simulation. Some demonstrations of how this is be done can be found in the function library discussed later in this document.

An optic is a general optical component (mirror, lens, etc.). Each type of optic has a fixed number of inputs and outputs and a set of parameters which characterize a given instance. The various types of optics are:

- BeamSplitter - a beam-splitter
- GouyPhase - an abstract telescope
- Mirror - a general curved mirror
- Modulator - audio frequency phase and amplitude modulation
- RFModulator - radio frequency phase and amplitude modulation
- Telescope - a lense or set of lenses
- Sink - a field sink, used for detectors
- Source - a field source

These optics are connected to one another by links. Technically speaking, a link is not a class, but rather a simple organizational structure used within Optickle. The members of this structure are:

- sn = serial number of this link
- snSource, portSource = serial number and port number of source optic
- snSink, portSink = serial number and port number of sink (destination)
- len = the length of the link

Probes define the output of Optickle. They can simulate simple intensity (a DC signal), or an RF demodulated signals. The output of a probe is in Watts, so to convert it into a photodiode voltage factors for the efficiency and transimpedance should be added by the user. Like links, probes are data structures internal to the Optickle class. The probes are characterized by:

- name, sn = name and serial number of this probe
- nField = index of the sampled field
- freq = demodulation frequency
- phase = demodulation phase offset (degrees)

2.1 Initializing a New Model

Creating a new Optickle model is easy. Like any other class, you need only call the class constructor, which is a function with the class as its name.

opt = Optickle(vFrf, lambda)	
vFrf	vector of RF component frequencies
lambda	carrier wave length (default 1064 nm)
opt	empty Optickle model

2.2 Adding Optics

After creating a shiny, new, empty, Optickle, you will need to fill it with optics. For each of the various classes of optics there is a function for adding it to your model. These functions work by returning a model which is like the one you pass them as the first argument, except for the added optic. The add functions also return the serial number of the new optic, which you can use for future reference instead of the optic's name. This is fine if you are concerned about computational efficiency, but I don't recommend it as you will probably not notice the speed difference and it may make your code slightly less readable. The general function for adding an optic is

[opt, sn] = addOptic(opt, obj)	
opt	initial Optickle model (as argument)
obj	the new optic
opt	new Optickle model with optic (as return value)
sn	serial number of the new optic

But you will likely never need to use this function, as there are add functions for each type of optic. These are provided simply to shorten the process of creating an optic and adding it to the model to a single line. All add functions have the same first two arguments. Their return values are as for *addOptic*.

[opt, sn] = addXXX(opt, name, ... optic specific arguments ...)	
opt	initial Optickle model (as argument)
name	name of the new optic

Beyond the first two arguments, the format of the add functions depends on the type of optic being added. Most of these arguments are optional. The various types of optics are discussed below, each of which has a constructor function with an argument list which matches the add functions listed below (less the Optickle model as first argument).

[opt, sn] = addBeamSplitter(opt, name, aio, Chr, Thr, Lhr, Rar, Lmd, Nmd)

aio	angle of incidence (in degrees)
Chr	curvature of HR surface (Chr = 1 / radius of curvature)
Thr	power transmission of HR surface
Lhr	power loss on reflection from HR surface
Rar	power reflection of AR surface
Nmd	refractive index of medium (1.45 for fused silica, SiO2)
Lmd	power loss in medium (one pass)

[opt, sn] = addGouyPhase(opt, name, phi)

phi	Gouy phase (in radians)

[opt, sn] = addMirror(opt, name, aio, Chr, Thr, Lhr, Rar, Lmd, Nmd)

aio	angle of incidence (in degrees)
Chr	curvature of HR surface (Chr = 1 / radius of curvature)
Thr	power transmission of HR surface
Lhr	power loss on reflection from HR surface
Rar	power reflection of AR surface
Nmd	refractive index of medium (1.45 for fused silica, SiO2)
Lmd	power loss in medium (one pass)

[opt, sn] = addModulator(opt, name, cMod)

cMod	modulation coefficient (1 for amplitude, i for phase)

[opt, sn] = addRFmodulator(opt, name, fMod, aMod)

fMod	modulation frequency
aMod	modulation index (imaginary for phase, real for amplitude)

[opt, sn] = addSink(opt, name, loss)

loss	power loss from input to output (default = 1)

[opt, sn] = addSource(opt, name, vArf, z0, z)

vArf	amplitudes of each RF component
z0	Rayleigh Range is $(waistsize)^2 * \pi/\lambda$
z	distance to waist (negative if beam is converging)

[opt, sn] = addTelescope(opt, name, f, df)

f	focal length of first lens
df	distances and focal lengths for lenses after the first

There are also two “compound” add functions, which are merely common combinations of the above. As with other Optickle functions, use the Matlab *help* command for a detailed description of what these functions do. You should feel free to use the Matlab *type* command, as they are quite simple and provide examples of how the other add commands can be used.

opt = addReadout(opt, name, fphi, names)

fphi	demod frequency and phase (Nx2)
names	optional cell array of probe suffixes (instead of 1:N)

opt = addReadoutTelescope(opt, name, f, df, ts, ds, da, db)

f	focal length of first lens
df	distances and focal lengths for lenses after the first
ds	distance from telescope to splitter
da	distance from splitter to sink A
db	distance from splitter to sink B

2.3 Adding Links and Probes

In addition to optics, an Optickle model is not complete without links to connect the optics and probes to extract information. The add functions for these are

[opt, snLink] = addLink(opt, from, out, to, in, len)

from	serial number or name of the source optic (field origin)
out	number or name of the output port (e.g., 1, 'fr', etc.)
to	serial number or name of the sink optic (field destination)
in	number or name of the input port (e.g., 2, 'bk', etc.)
len	length of the link

[opt, snProbe] = addProbeIn(opt, name, to, in, freq, phase)

name	name of the new probe
to	serial number or name of the sink optic
in	number or name of the input port
freq	demodulation frequency
phase	demodulation phase (in degrees)

[opt, snProbe] = addProbeOut(opt, name, from, out, freq, phase)

name	name of the new probe
from	serial number or name of the source optic
out	number or name of the output port
freq	demodulation frequency
phase	demodulation phase (in degrees)

The last of these is technically problematic and should be avoided, as it can always be replaced by *addProbeIn*, but sometimes it is convenient (see *help addProbeOut* for more information).

2.4 Model Modification: *set* Functions

There are some functions which allow you to modify optics after their addition to a model. The boundary between these functions and parameters to the add functions is not well defined, but the general idea is that these allow you to modify an existing model rather than create a new one (not that they cost much). Since these functions modify a model, they all share the same first argument (the model to modify) and they all return the modified model.

opt = setCavityBasis(opt, name1, name2)	
name1, name2	names of Mirros that make a simple Fabry-Perot cavity

opt = setGouyPhase(opt, name, phi)	
name	name of the GouyPhase optic
phi	new phase value (in radians)

opt = setLinkLength(opt, snLink, len)	
snLink	serial number of a link
len	new length of this link

opt = setMechTF(opt, name, mechTF, nDOF)	
name	name of the optic
mechTF	mechanical transfer function (see LTIMODELS)
nDOF	1 for position (default), 2 for pitch

opt = setMinQuant(opt, minQuant)	
minQuant	minimum loss used for evaluating quantum noise

2.5 Information Retrieval: *get* Functions

Many functions help to get information about an Optickle model. Much of this information is just what we put in when constructing the model, and some of it has to do with the way Optickle arranges its components internally. Since these functions interrogate a model, they all share the same first argument.

vBasis = getAllFieldBases(opt)	
vBasis	a Nfldx2 matrix of complex numbers (see also @OpHG/apply)

n = getDriveIndex(opt, name, driveType)	
name	name of the optic
driveType	name of the drive (e.g., 'pos' or 'amp') (optional)
n	index of this drive (e.g., in sigAC returned from tickle)

strList = getDriveNames(opt)	
strList	cell array of drive names for display, OpticName.DriveName

n = getFieldIn(opt, name, inName)

name	name of the optic
inName	name of an input to the optic
n	index of input field (e.g., in fDC returned from tickle)

n = getFieldOut(opt, name, outName)

name	name of the optic
outName	name of an output from the optic
n	index of output field

n = getFieldProbed(opt, name)

name	name of the probe
n	index of probed field

str = getInputName(opt, name, inNum)

name	name of the optic
inNum	input index
str	display name for this input

vDist = getLinkLengths(opt)

vDist	lengths of each link (used with library function getGouyPhase)

n = getLinkNum(opt, nameSource, nameSink)

nameSource	name of source optic (link start)
nameSink	name of sink optic (link end)
n	array of link serial numbers

obj = getOptic(opt, name)

name	name of the optic
obj	copy of the optic with this name

name = getOpticName(opt, sn)

sn	number of the optic (or vector, or cell array, or nothing)
name	name of the optic (or cell array of names)

par = getOptParam(opt)

par	a structure of data about this Optickle model

str = getOutputName(opt, name, outNum)

name	name of the optic
inNum	output index
str	display name for this output

name = getProbeName(opt, snPrb)

snPrb	number of the probe (or vector, or cell array, or nothing)
name	name of the probe (or cell array of names)

n = getProbeNum(opt, name)

name	name of the probe
n	index of probed (e.g., in sigDC returned from tickle)

phase = getProbePhase(opt, snPrb)

snPrb	number or name of probe (or vector, or cell array, or nothing)
phase	phase of probe (or vector of phases)

sn = getSerialNum(opt, name)

name	name of the optic
sn	serial number of the optic

[str, sn, port] = getSinkName(opt, snLink)

snLink	serial number of a link
str	display name of the sink (see getInputName)
sn	serial number of sink optic
port	port number of sink

[name, sn, port] = getSourceName(opt, snLink)

snLink	serial number of a link
str	display name of the source (see getOutputName)
sn	serial number of source optic
port	port number of source

[vFrf, vSrc] = getSourceInfo(opt)

vFrf	source frequency vector (Nrf x 1)
vSrc	source amplitude vector (Nfld x 1) (Nrf x 1)

2.6 Setting the Working Point

The optics in Optickle have two kinds of inputs: fields and drives. Fields, or beams, originate with *Source* optics and are routed from one optic to the next with the previously mentioned links. Other excitations to the system arrive in the form of a drive to an optic. A mirror, for instance, has a single drive, namely its location, while an RF modulator has two drives, amplitude and phase of the oscillator. Sources and sinks, on the other hand, have no drives.

The static value of a drive is called its "position", which makes sense for mirrors, but is a bit of a stretch for modulators. The following are functions which allow you to manipulate the positions of the optics.

opt = addDriveOffset(opt, nDrv, pos)

nDrv	drive index
pos	addition to zero position for this drive (see setDriveOffset)

opt = addPosOffset(opt, name, pos)

name	name or serial number of optic
pos	addition to zero position for this optic (see setPosOffset)

opt = setDriveOffset(opt, nDrv, pos)
pos = getDriveOffset(opt, nDrv)

nDrv	drive index (or indices)
pos	zero position for this drive (or drives)

opt = setPosOffset(opt, name, pos)
pos = getPosOffset(opt, name)

name	name or serial number of optic
pos	zero position for this optic

[opt, pos] = setOperatingPoint(opt, mDrive, mSense, vOffset)
[opt, pos] = setOperatingPoint(opt, mDrive, nameErrFunc)

mDrive	drive matrix
mSens	sensing matrix
vOffset	offset vector ($vErr = mSense \times sigDC - vOffset$)
nameErrFunc	name of error function

The last of these, *setOperatingPoint*, is a simple solver that attempts to zero a set of error signals by moving some optics (i.e., to "lock" the interferometer). Usually locking is not necessary as error signals tend to be zero when cavities are on resonance, but when working with detuned cavities locking may be necessary (see *help setOperatingPoint* for details).

2.7 Response Evaluation

One the model is build and configured, you will probably want to compute something with it. The number of functions devoted to doing the real work is relatively small, but what happens inside is somewhat more complicated (*type* them at your own risk). The mathematics of these functions is discussed in the document "Optickle inner workings" (T070260).

sCon = convertSimulink(opt, sys, f)

sys	name of Simulink system
f	vector of evaluation frequencies
sCon	control structure to be used with *tickle* or *tickle01*

[fDC, sigDC] = sweep(opt, pos)

pos	optic positions (Ndrive x Npos)
fDC	DC fields at each position (Nlink x Nrf x Npos)
sigDC	DC signals for each probe (Nprobe x Npos)

[pos, sigDC, fDC] = sweepLinear(opt, posStart, posEnd, Npos)

posStart	sweep start positions (Ndrive x 1)
posEnd	sweep end positions (Ndrive x 1)
Npos	sweep points (including posStart and posEnd)
sigDC	DC signals for each probe (Nprobe x Npos)
fDC	DC fields at each position (Nlink x Nrf x Npos)

[fDC, sigDC, sigAC, mMech, noiseAC, noiseMech] = tickle(opt, pos, f)
[fDC, sigDC, sOpt, noiseOut] = tickle(opt, pos, sCon)

pos	optic positions (Ndrive x 1 or empty for zeros)
f	vector of evaluation frequencies (empty for DC only)
sCon	control structure from *convertSimulink*
fDC	DC fields (Nlink x Nrf)
sigDC	DC signals for each probe (Nprobe x 1)
sigAC	transfer matrix (Nprobe x Ndrive x Naf)
mMech	modified drive transfer functions (Ndrv x Ndrv x Naf)
noiseAC	quantum noise at each probe (Nprobe x Naf)
noiseMech	quantum noise at each drive (Ndrv x Naf)
sOpt	response structure
noiseOut	quantum noise at each Simulink output

[sigAC, mMech] = tickle01(opt, pos, f)
sOpt = tickle01(opt, pos, sCon)

pos	optic positions (Ndrive x 1 or empty for zeros)
f	vector of evaluation frequencies (empty for DC only)
sCon	control structure from *convertSimulink*
sigAC	TEM01 transfer matrix (Nprobe x Ndrive x Naf)
mMech	modified pitch drive transfer functions (Ndrv x Ndrv x Naf)
sOpt	response structure

2.8 Internal Data

The members of the Optickle class are:

optic	a cell array of optics
Noptic	number of optics
Ndrive	number of drives
link	an array of links
Nlink	number of links
probe	an array of probes
Nprobe	number of probes
lambda	carrier wave length
vFrf	RF components
h	Plank constant
c	speed of light
k	carrier wave-number
minQuant	minimum loss considered for quantum noise
debug	debugging level (not widely used)

Of these, only some can be reference directly. They are: Noptic, Ndrive, Nlink, Nprobe, lambda, k, c, h, and debug. For others, use *get* functions (see previous section). [1] If an access function does not exist for something you need, cheat and access it directly. This should produce a warning message, which will serve as your reminder either to create an access function to do what you need and send it to me, or to ask me to create an access function and send it to you. Typically these things are quick and easy to make.

3 Optics

This section has a list of types of optics, ordered alphabetically. The general purpose of each type is given, and any special functions related to the type of optic are listed. For more information use the Matlab help function, followed by the name of the class (e.g., help Mirror). At the end of the section, functions common to all types of optics are listed for reference, though casual Optickle users are not expected to need to know about them

[1]The reason for this is to protect Optickle users from developers who like to change the internal structures of things. The access functions define an interface that developers can try to maintain, even if the internals change.

3.1 BeamSplitter

The beam-splitter class is really a special type of mirror (see Mirror class). The thing that makes it special is that it has 4 input beams and 8 output beams (4 primary outputs, and 4 pick-off outputs). Despite the name, beam-splitters should not be used to split an input beam into two output beams, as a normal mirror can do this without problems. In fact, beam-splitters are only useful when you need to simulation a mirror at non-normal incidence with beams entering and exiting along the same paths (e.g., the beam-splitter in a Michelson interferometer).

In many respects, a BeamSplitter object is like two Mirror object which share the same position and the same optical parameters (see Mirror, again). In terms of inputs and outputs, if call these two virtual mirrors A and B, then the I/O names for a beam-splitter are the same as those for these two mirrors (e.g., fr and bk) with A and B appended: frA, bkA, frB and bkB.

[FIGURE OF BEAMSPLITTER]

3.2 GouyPhase

The GouyPhase object is an abstraction of an optical telescope (see Telescope class) which simplifies the process of building an angular sensing system. The abstraction used is to ignore the lenses and distances that would actually be used to change the Gouy phase of a TEM01 mode, and simply add the desired phase. This can save time in the initial stages of interferometer design when the details of the optical chain are not critical.

This object has two special functions associated with it. They are used to interact with its unusual (and unphysical) ability to change TEM01 phase.

getPhase return the phase added to the TEM01 mode by this optic (in radians)

setPhase assign the phase added to the TEM01 mode by this optic (in radians)

3.3 Mirror

Mirrors are the bread and butter of optics. A Mirror object in Optickle has two input beams and 4 output beams (2 primary and 2 pick-off).

[FIGURE OF MIRROR]

Mirrors can be used for everything from core interferometer optics to simple 1-inch steering mirrors. The optical parameters used to describe a Mirror are

aio angle of incidence (in degrees)

Chr curvature of HR surface (Chr = 1 / radius of curvature)

Thr power transmission of HR suface

Lhr power loss on reflection from HR surface

Rar power reflection of AR surface

Nmd refractive index of medium (1.45 for fused silica, SiO2)

Lmd power loss in medium (one pass)

3.4 Modulator

Modulators are used to make audio frequency modulation. Often, this means phase and amplitude noise. The only argument **cMod** is the modulation coefficient. To made a phase modulator which produces with one radian of phase per unit of drive, set **cMod = i**. For an amplitude modulator $\frac{dAmp}{drive} = Amp \times \mathbf{cMod}$, with **cMod** real.

3.5 RFmodulator

An RFmodulator is used to produce optical frequency components of different frequencies. Physically, this object is the same as modulator, but computationally they are quite different as an RF modulator results in RF sidebands while an audio modulator makes audio sidebands. RF modulators have 2 drives, "amp" for oscillator amplitude noise and "phase" for oscillator phase noise.

vMod modulation frequencies and amplitudes, Nmod x 2

Nmod number of RF modulation frequencies

3.6 Sink

A Sink is a beam dump or attenuator. Sinks are often used in the place of a physical detector, but they can also be used as attenuators by setting the **loss** parameter to something less than 1.

3.7 Source

A Source produces the carrier field which excites the optical system (i.e., PSL, laser, fiber, etc.) Sources have no inputs and no drives.

vArf amplitudes of the RF field components in the model

z0 Rayleigh Range is $(waistsize)^2 * \pi/\lambda$ (optional)

z distance to waist (negative if beam is converging) (optional)

3.8 Telescope

Telescopes are used for changing the beam size to match cavities, or to add Gouy phase for readout. A telescope is made up of N + 1 lenses with N distances between them.

The audio phase accumulated while propagating through a telescope is NOT included. Typically this phase is very small (e.g., for a 1 meter telescope, a 300Hz audio SB should gain 1e-6 radians of phase). In special cases where the audio phase is of interest, break the telescope into individual lenses and link them together with links of the correct distances.

f focal length of first lens

df distances from previous lens and lens focal length for lenses after the first

3.9 Optic Internals: Optical Properties

These functions are common to all optics, though each optic defines them differently. They represent each optic's input to output relations, and thus determine how the optics behave in the system.

getBasisMatrix basis transformation matrices (see OpHG)

getDriveMatrix drives and input fields are combined to make output audio sidebands

getDriveMatrix01 as getDriveMatrix, but the outputs are TEM01

getFieldMatrix input field to output field matrix

getFieldMatrix01 input field to output field matrix for a TEM01 field

getNoiseMatrix quantum noise inputs for this optic, as seen at its outputs

getReactMatrix input field to force matrix

getReactMatrix01 as getDriveMatrix, but the outputs are torques

3.10 Optic Internals: Matlab Mechanics

I'll only list these here, as they are the same for each class and of little interest.

display display information about an instance

get get member data

set set member data

subsasgn handle indexed assignment (set)

subsref handle indexed retrieval (get)