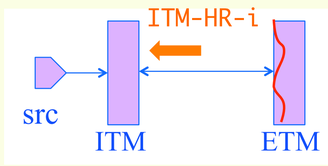
A modular FFT-based Interferometer Simulation Package

Introduction

- FOGPrime13
  - » Objet oriented FFT-based IFO simulation and analysis package on matlab
    - – FP to full aLIGO with input and output MCs, transMon
  - » FOGPrime13 = FOG + SIS + e2e + twiddle
    - – FOG as the field calculation engine
    - – SIS as base of the user interface design and support package
    - – e2e as base of the object oriented package infrastructure
    - – twiddle for setting the initial condition of fields, especially for coupled cavities
- Based on matlab
  - » Matlab functions, built-in and user provided, can be easily integrated
- No setting of Wfft and Nfft
  - » User defines optics quantities and resolution of maps

Code defining an IFO




```
%% create FP cavity based on FFTIFO
classdef FPIFO < FFTIFO

    methods
        %% this defines the IFO configuration
        function defineIFO( obj )
            %% add optics
            % laser source
            obj.addLaser( 'src' );

            % ITM
            obj.addMirror( 'ITM', 'invRoC', 1/1937.9, 'Aperture', 0.326, 'Thick', 0.2, ...
                'T', 1.48e-2, 'Lhr', 10.4e-6, 'Lar', (164+42)*1e-6, 'removeTilt', true );
            % ETM
            obj.addMirror( 'ETM', 'invRoC', 1/2239.7, 'Aperture', 0.326, 'Thick', 0.2, ...
                'T', 3.7e-6, 'Lhr', 10.9e-6, 'resolution', 2e-3, 'removeTilt', true  );

            % detector behind ETM
            obj.addDetector( 'trans' );

            %% define connections among optics
            obj.addProp( '', 'src', 'ITM-AR' );
            obj.addProp( 'FPprop', 'ITM-HR', 'ETM-HR', 3994.5);
            obj.addProp( '', 'ETM-AR', 'trans' );

            %% use the measured maps
            % ITM HR surface
            obj.setHRfiles( 'ITM', './Data/ITM04_CITZygo.dat', '', 0.16, 1937.9 );
            % ITM transmission maps
            obj.setTRfiles( 'ITM', './Data/itm04_sptwe.dat', '', 0.16, 302.41e3);
            % ETM HR surface
            obj.setHRfiles( 'ETM', './Data/ETM07_CIT.dat', '', 0.16, 2239.7 );

        end
    end
end
```

User functions to define an IFO

- addLaser, addDetector
- addMirror(‘mirrorName’, ‘var1Name’, val1, ‘var2Name’, val2, …)
  - » invRoC, invRoCAR, Thickness, T, Lhr, Lar, resolution, optTheta, optPhi , removeTilt, incAngle, invFocal, WfftUser, virtualAperture
- Set[HR,TR,AR]files(‘fileName’, formula, ApertureFit, RoCFit, rotMap, vararin)
  - » ApertureFit and RoCFit are the aperture used to measure the RoC and the measured value
  - » formula = 'scale * map + (x.^2+y.^2)/(2*RoC)’
  - » varargin = ‘scale’, 1, ‘RoC’, 1e20 : used as default values
  - » formula can call any function, e.g., use COMSOL output using
    - – mphinterp(COMSOLmodel, symbol, 'coord', xxx)
- addProps(‘propName’, ‘fromPort’, ‘toPort’, macroLeng, microLeng)
  - » Calls addOneProp(from -> to) and addOneProp(to -> from)

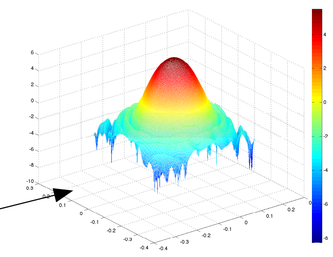
Code analyzing an IFO

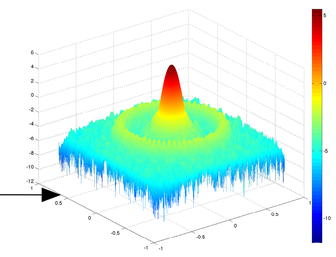
```
% define FP setup
fp = FPIFO;

% lock FP
fp.lock;

% get complex amplitude and coordinate of a field on ITM, HR side, going into ITM
[E0, x0] = fp.getField('ITM-HR-i');
mesh( x0, x0, log(abs(E0))); colorbar;

% The default window size is enough for almost all.
% we want to see the ring pattern scattered by the spiral Pattern by the coating
% the radius of the ring on the ITM plane is 55cm, so make the window size larger, 1.4m
% The second argument specifies the window size of the field, now 1.4m
[E1, x1] = fp.getField('ITM-HR-i', 1.4);
mesh( x1, x1, log(abs(E1))); colorbar;
```





Building an IFO object

- defineIFO
  - » User provides the definition of cavities
- analyzeCavity
  - » Analyze the cavity structures and field flows
- setupBaseMode
  - » Find the input field mode and propagate through out the cavity
- Bookkeeping about dimensional quantities
  - » beam size, optics size -> Wfft, resolution -> Nfft
- setupMisc
  - » Calculate propagators, transmission and reflection maps
- twiddle
  - » calculate powers of all fields using analytic formula to setup initial conditions
- startFFT
  - » Setup the input field and get ready for the simulation

properties calculated internally

- Fields
  - » [E,x,y] = getField(fieldID, viewSize)
  - » Misc tools viewing data - analyzeFIeld, power, HGCoef, LGCoef
- Modes
  - » Mode of the input field is calculated automatically or by user specification
  - » Mode base is propagated to all fields, to define modes for mode expansion
- Cavities
  - » Closed cavities and Interaction among cavities
  - » Adjustable parameters - Lock conditions, Convergence parameters
- Data loader
  - » 3 data formats - MetroPro binary data, LMA .col, SIS _W_N format
  - » Properly incorporate measured RoC
  - » Loaded data are chased in global FOGPRIME13DATA

Analysis functions

- Lock
  - » Set all internal cavities resonant and calculate stationary fields
- Calc
  - » Using the given cavity lengths, calculate stationary fields
- getField(‘fieldName’, Wfft)
  - » Calculate the complex amplitude of the field with the reuested window size
- gaussFit(;fieldName’)
  - » Fit the field by a product of 1D gaussian forms to calculate the effective beam center, size and curvature.
- HGCoef(‘fieldName’, m, n), LGCoef(‘fieldName’, p, l)
  - » Calculate the HG and LG amplitudes of the field

Internal details

- Use R_getguessFin by Richard/Gabrielle for stationary field calculation in each closed loop
- Round trip phases of closed loop are set to assigned values
  - » 0 for resonant cavity, pi for anti-resonant cavity
- Dark port
  - » If there is ‘BS’ and HR side y port and AR side x port are connected, the AR side y port power is minimized
- Interaction among cavities are defined by pump in and pump out between cavities
- User actions
  - » User can override a function to control the locking at the end of each automated adjustment