Topic 4: Transfer function models and digital filters

1. Transfer function models in s domain
   1.1. Pole zero representation
   1.2. Rational representation
   1.3. Partial fraction representation

1. Transformation between representations

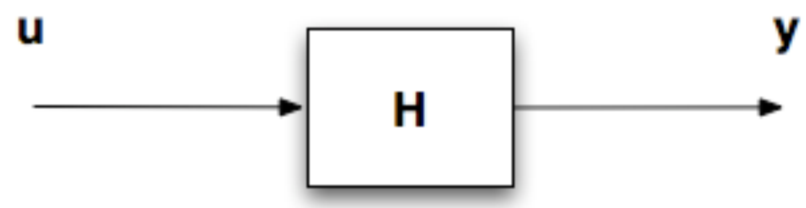
2. Modelling a system

3. Filtering data
   3.1. discretizing a model
   3.2. setting filter properties

4. IFO/Temperature example

Overview

- The general scheme: input, output and a transfer function



- Aim of this topic:
  - How to model the transfer function H in continuous domain, H = H(s)
  - How to discretize our model H(s) -> H(z)
  - How to filter data with H(z)
  - How to define H(z) from filter properties

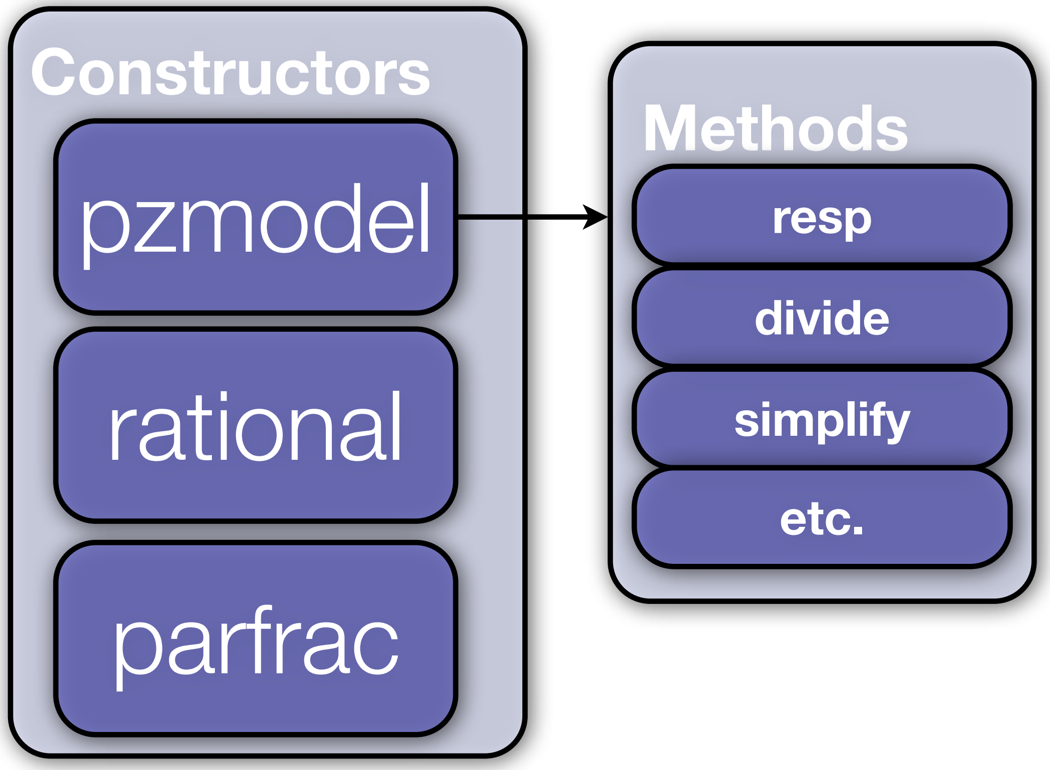
Tools used here

Tools used here

1. Continuous domain

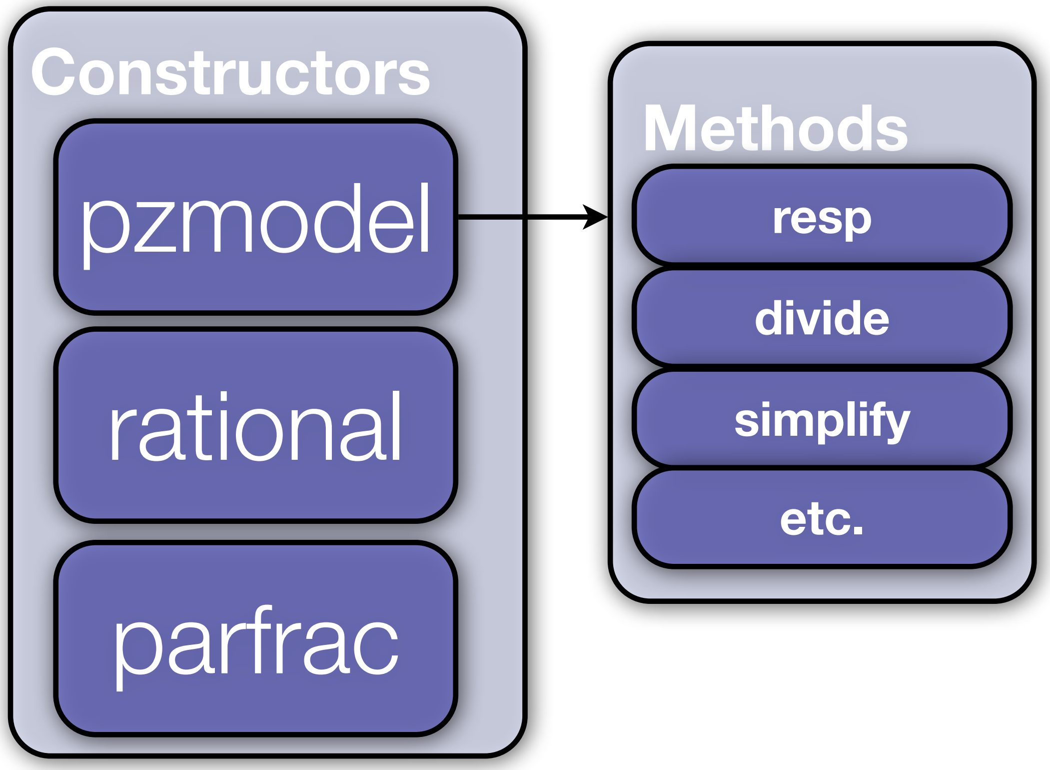
Tools used here

1. Continuous domain



Tools used here

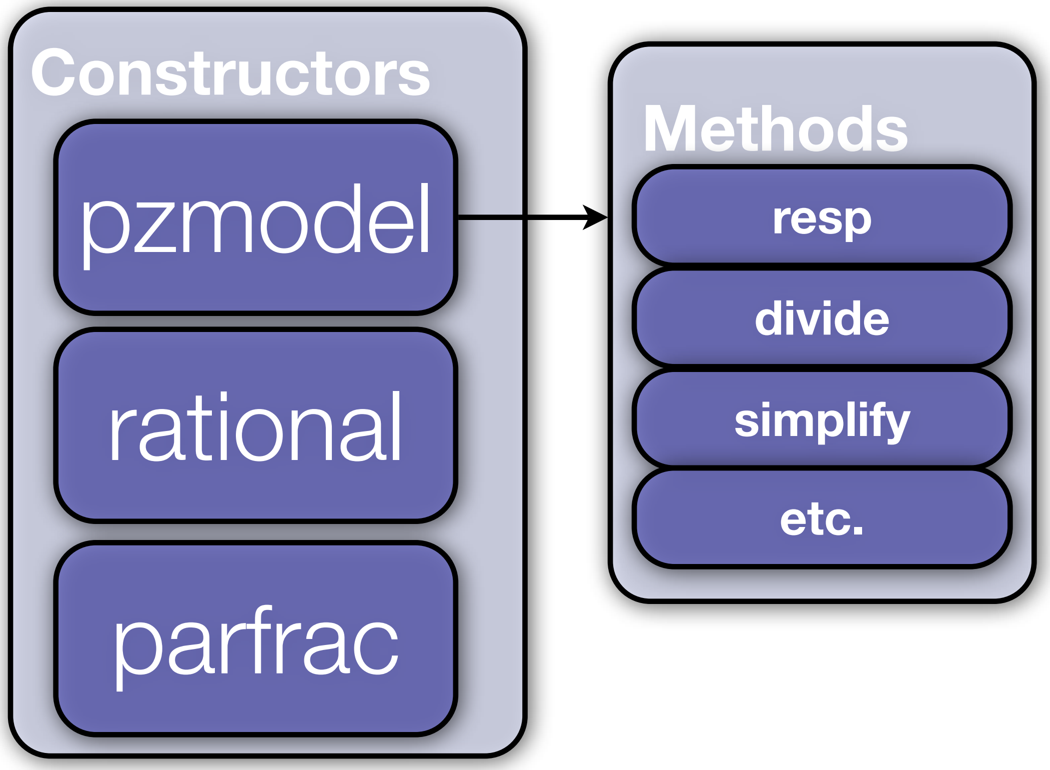
1. Continuous domain



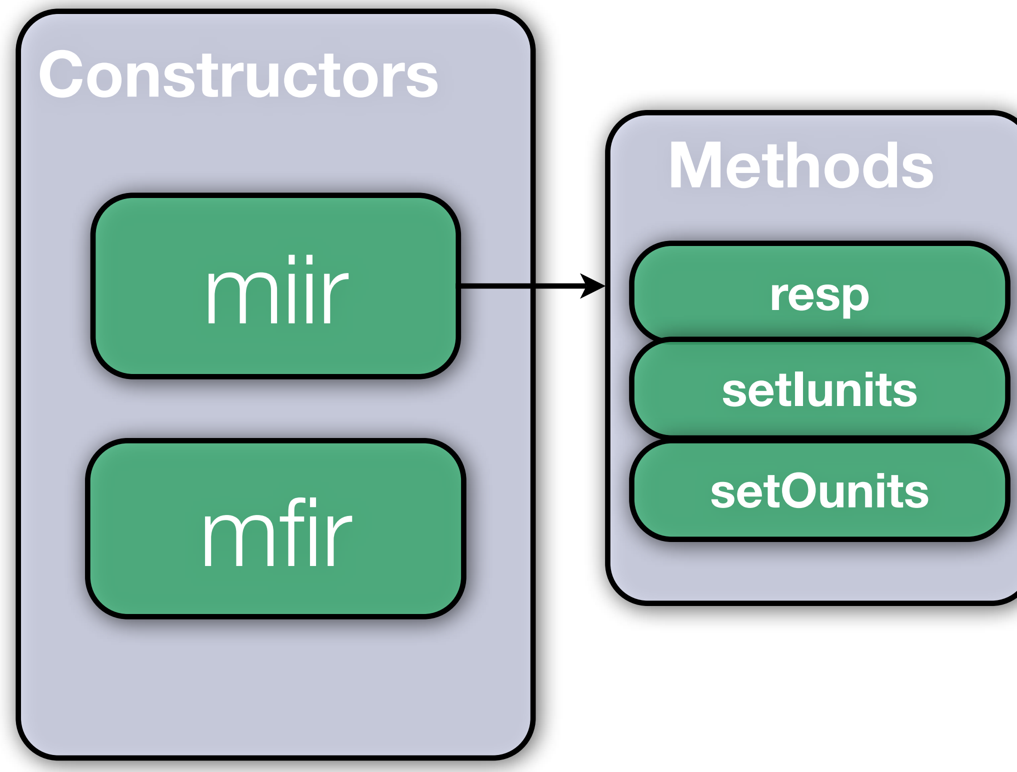
2. Discrete domain

Tools used here

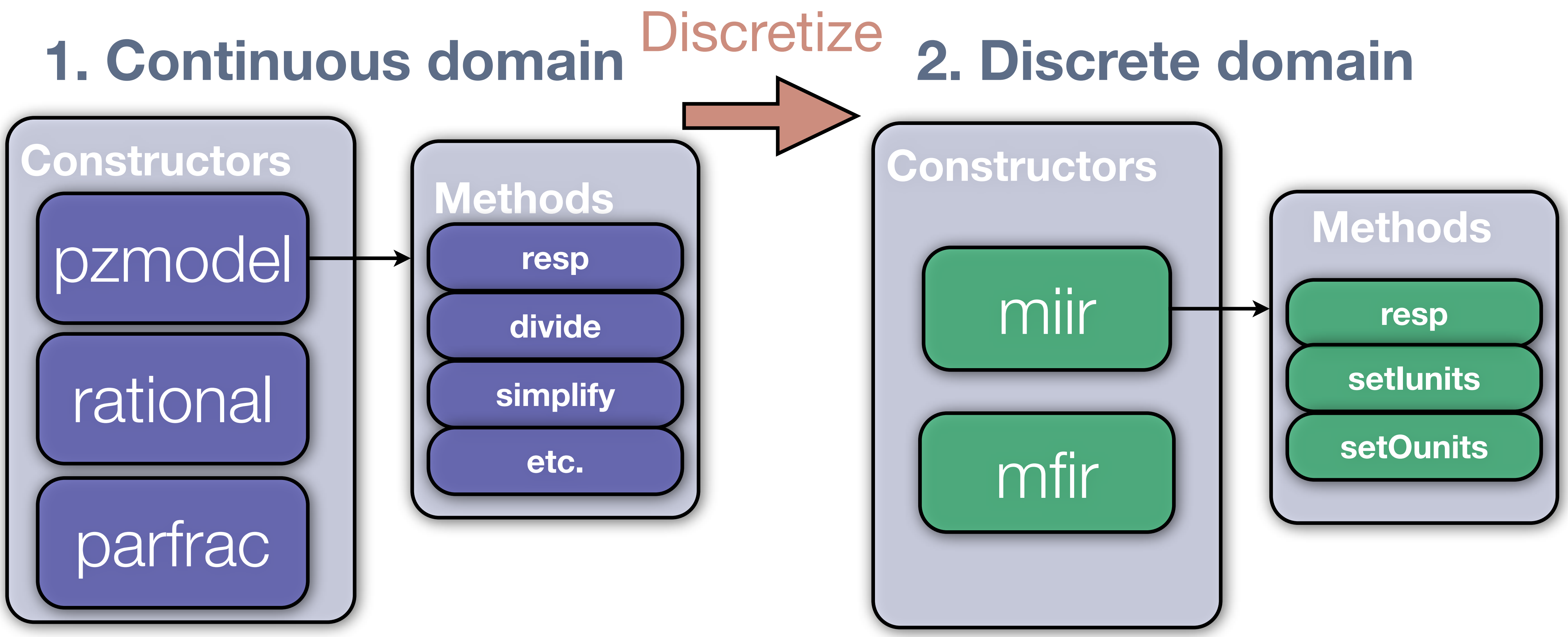
1. Continuous domain



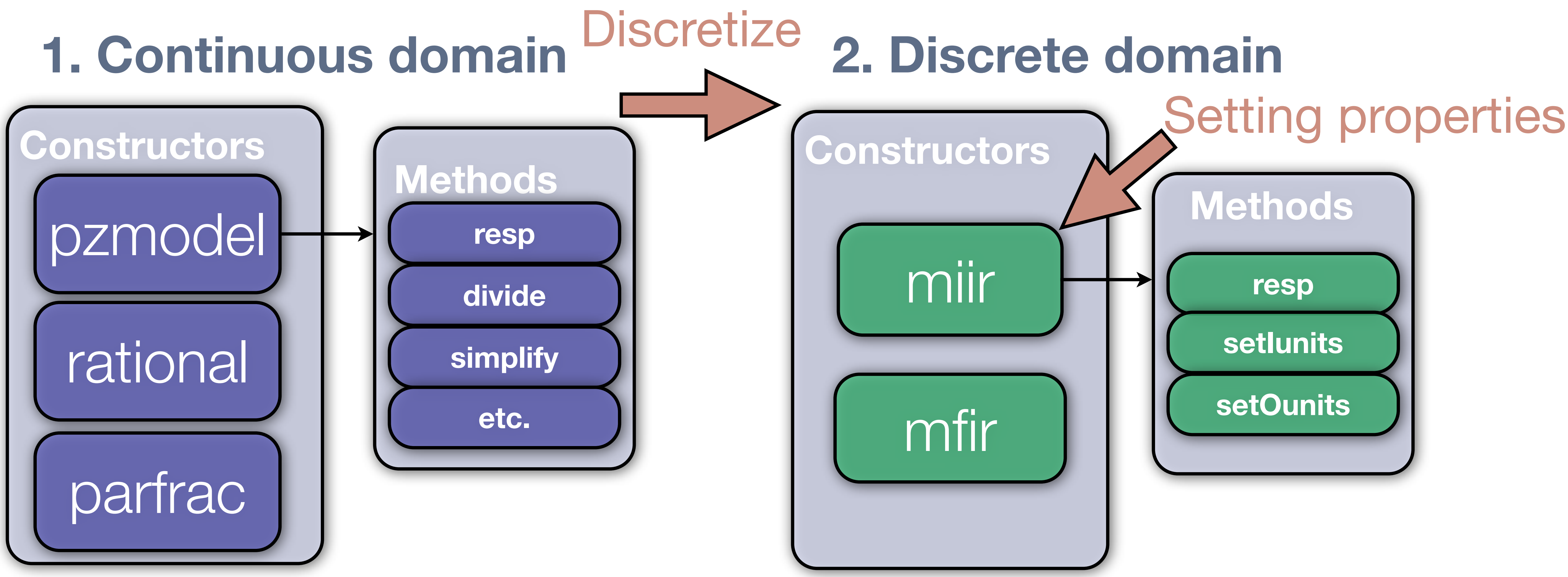
2. Discrete domain



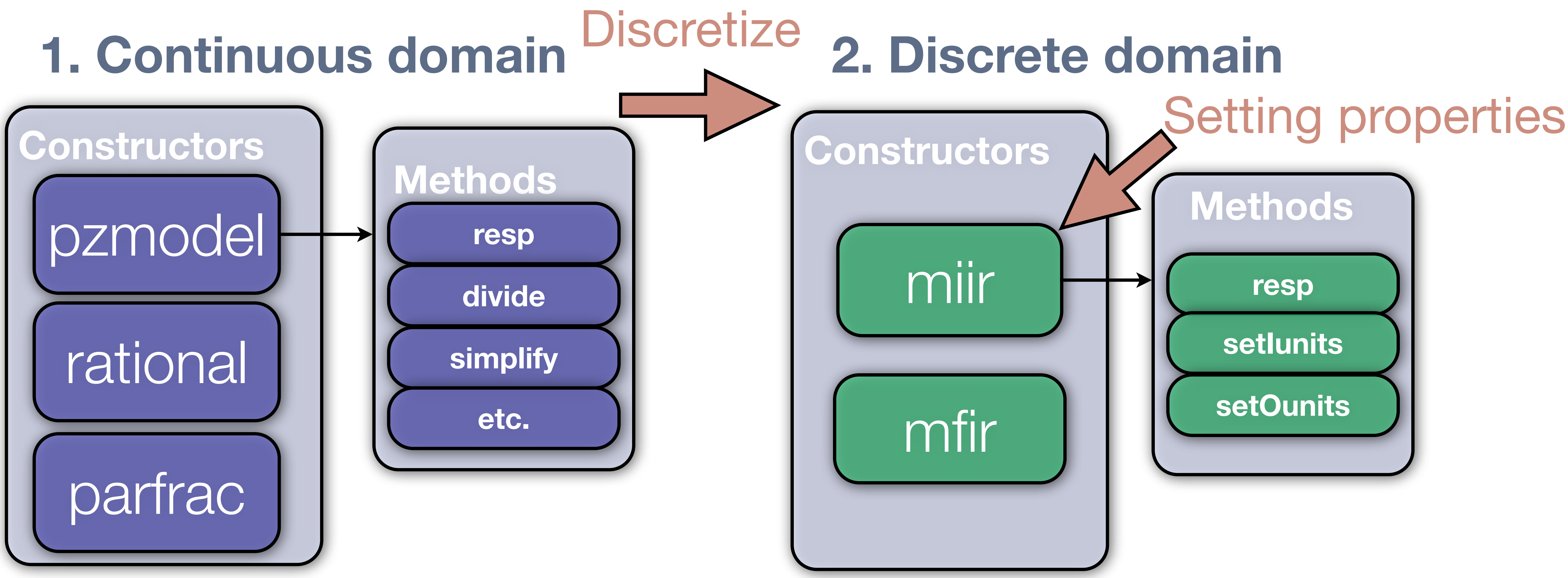
Tools used here



Tools used here

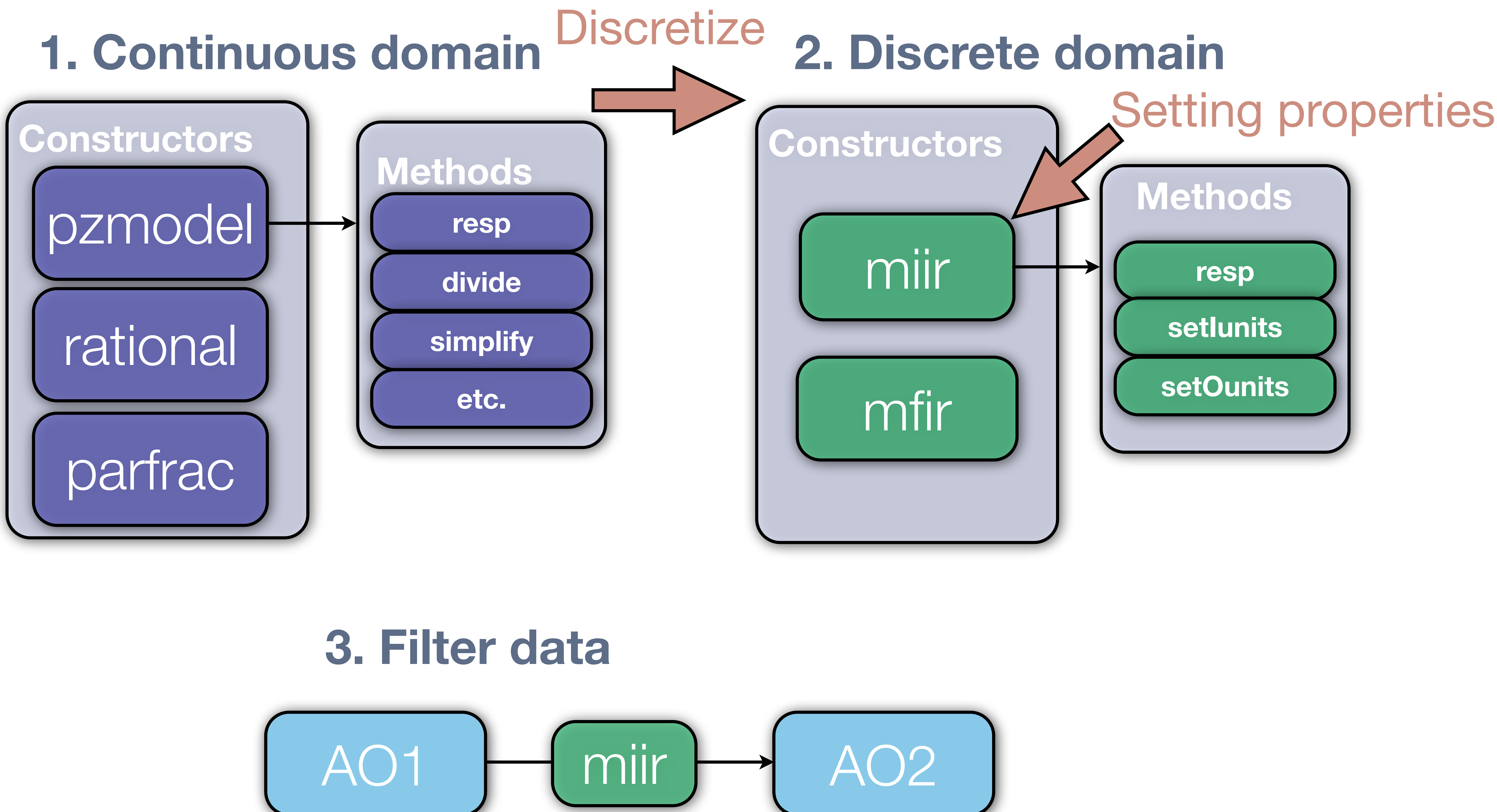


Tools used here



3. Filter data

Tools used here



1.1 Pole zero model

- A pole zero model is defined by:
  - Gain, poles, zeros, delay

$$H(s) = G\frac{(s-z_1)(s-z_2)\ldots(s-z_n)}{(s-p_1)(s-p_2)\ldots(s-p_m)}\,e^{-i\omega\tau}$$

- LTPDA constructor: PZMODEL
  - PZMODELs can be multiplied and divided
  - Delay is added or subtracted in such a case
  - Can read LISO files

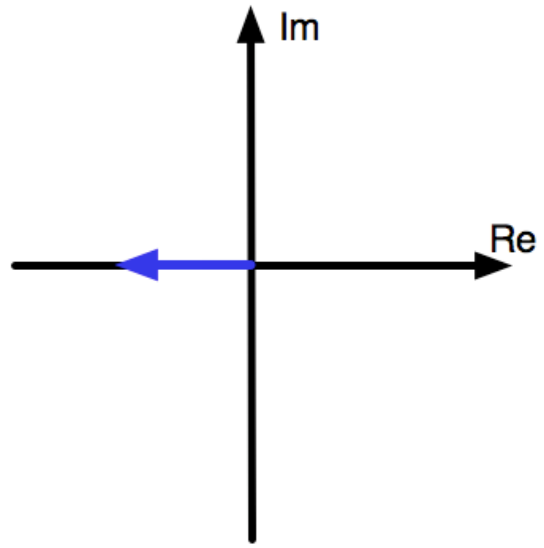
About poles (and zeros) notation

About poles (and zeros) notation

- Simple pole: f = 1 Hz

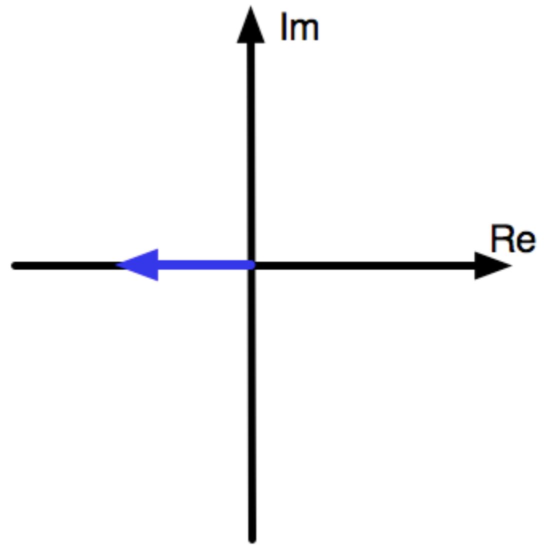
About poles (and zeros) notation

- Simple pole: f = 1 Hz



About poles (and zeros) notation

- Simple pole: f = 1 Hz
- Pole pairs: (f = 1 Hz, Q)

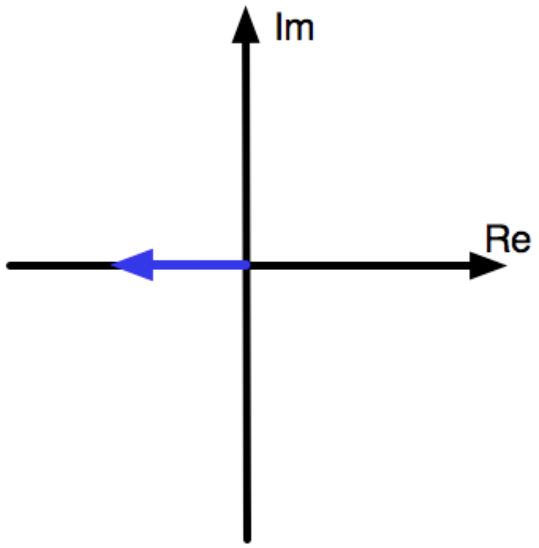


About poles (and zeros) notation

- Simple pole: f = 1 Hz

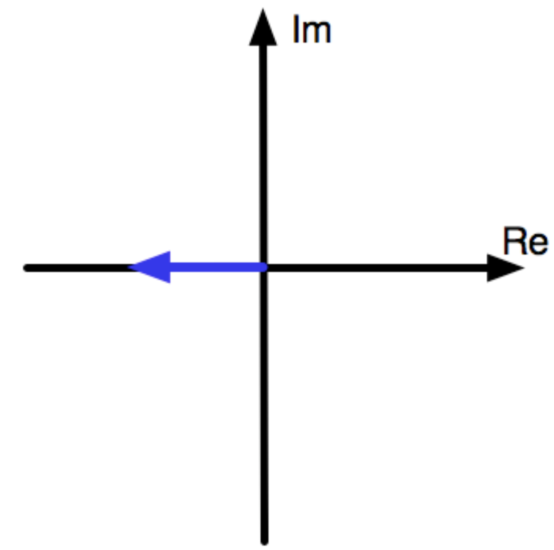
- Pole pairs: (f = 1 Hz, Q)

  Q > 0.5
  (underdamped)



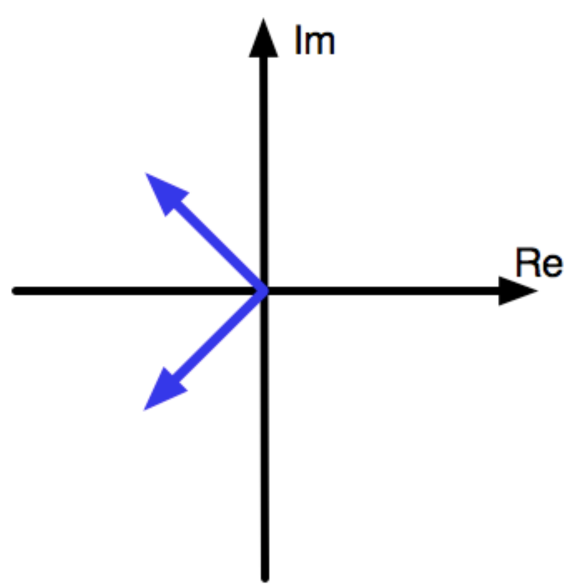
About poles (and zeros) notation

- Simple pole: f = 1 Hz



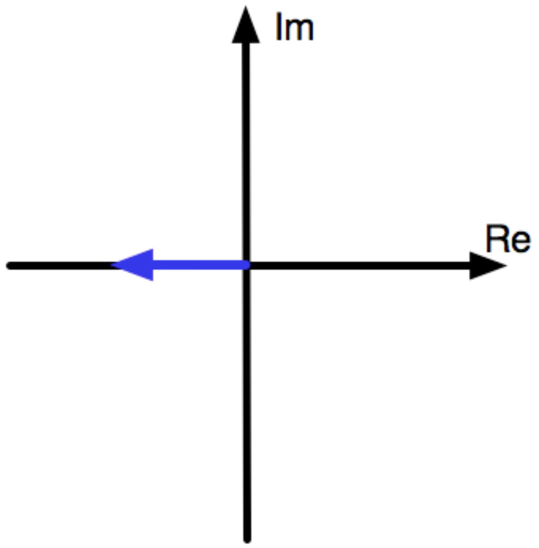
- Pole pairs: (f = 1 Hz, Q)

Q > 0.5
(underdamped)



About poles (and zeros) notation

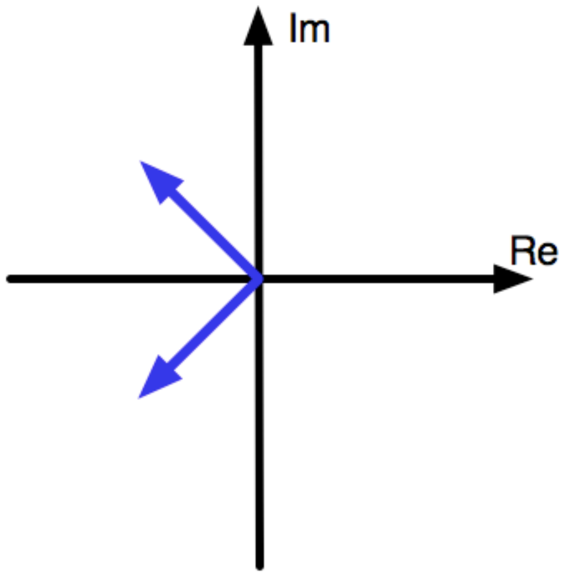
- Simple pole: f = 1 Hz



- Pole pairs: (f = 1 Hz, Q)

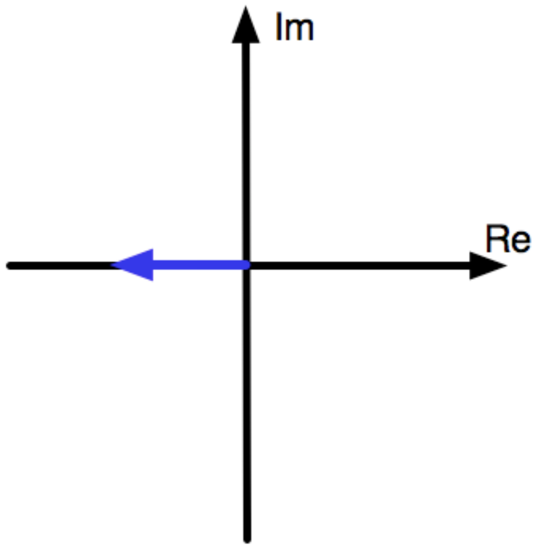
Q > 0.5
(underdamped)

Q = 0.5
(critically damped)



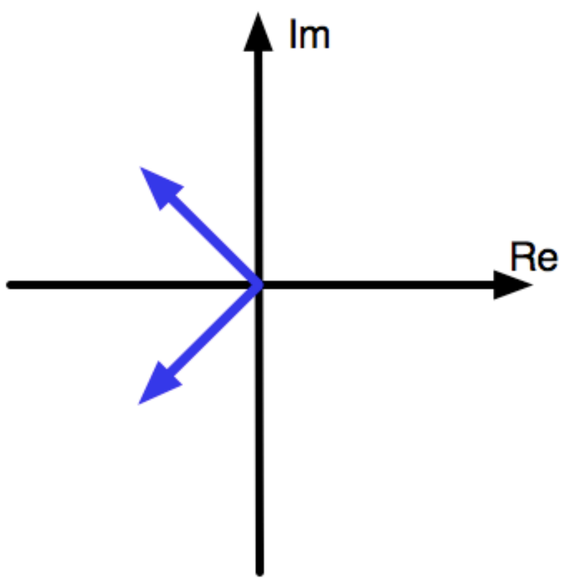
About poles (and zeros) notation

- Simple pole: f = 1 Hz

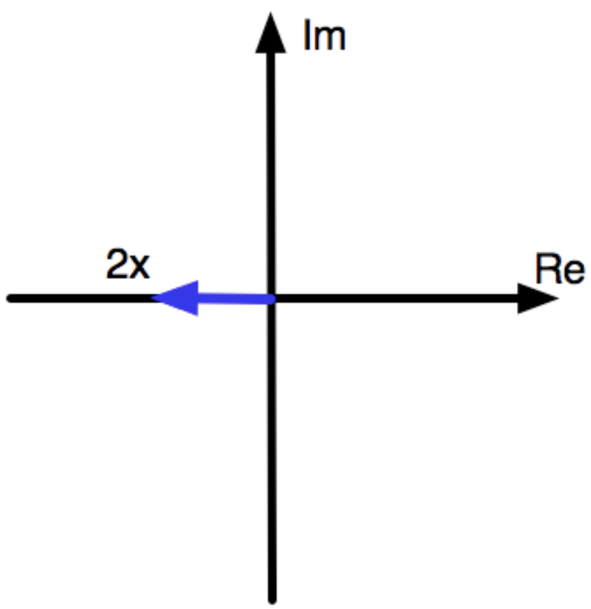


- Pole pairs: (f = 1 Hz, Q)

Q > 0.5
(underdamped)

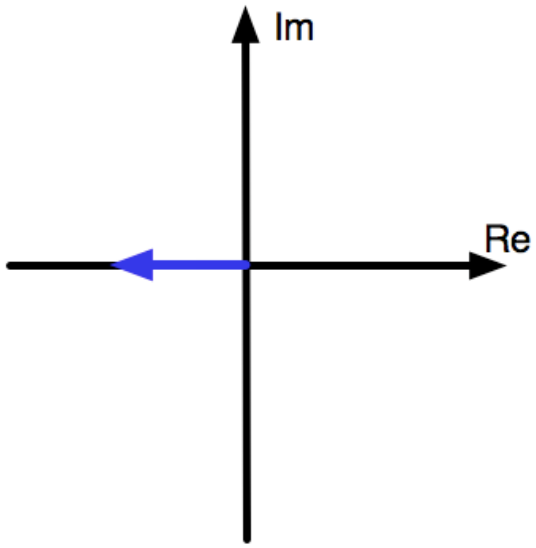


Q = 0.5
(critically damped)



About poles (and zeros) notation

- Simple pole: f = 1 Hz

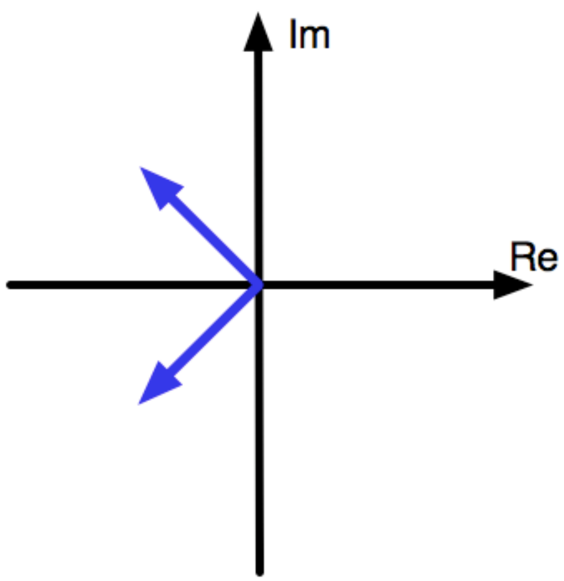


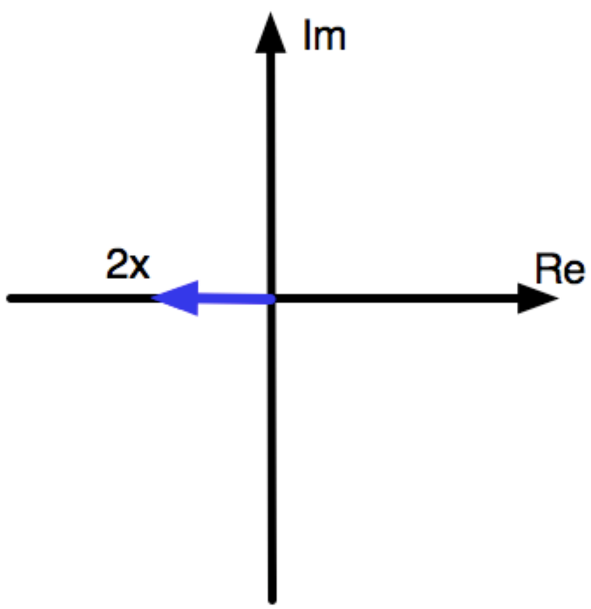
- Pole pairs: (f = 1 Hz, Q)

Q > 0.5
(underdamped)

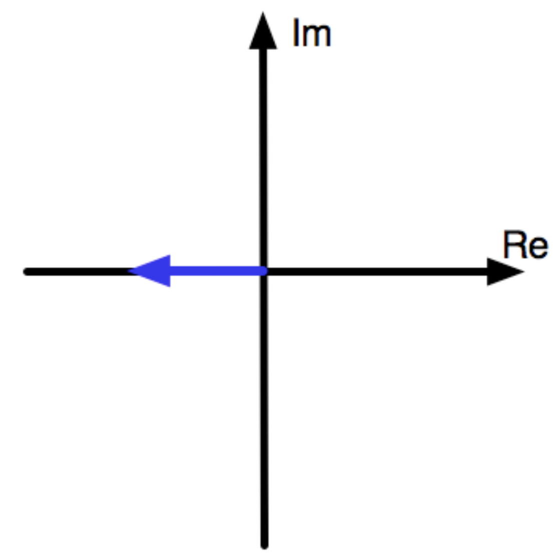
Q = 0.5
(critically damped)

Q < 0.5
(overdamped)



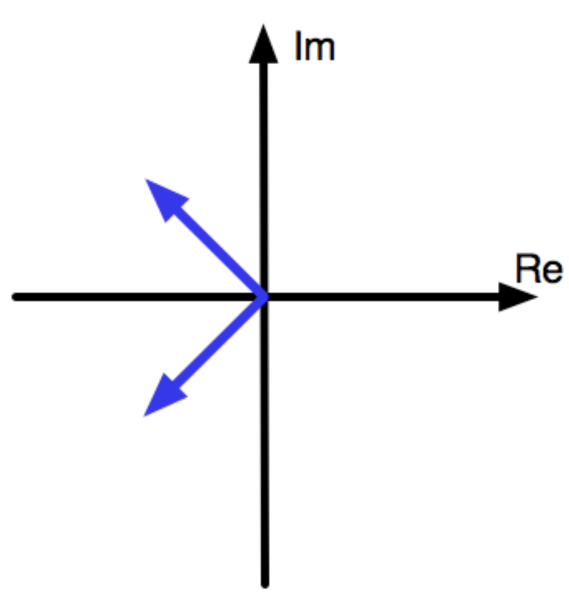


About poles (and zeros) notation

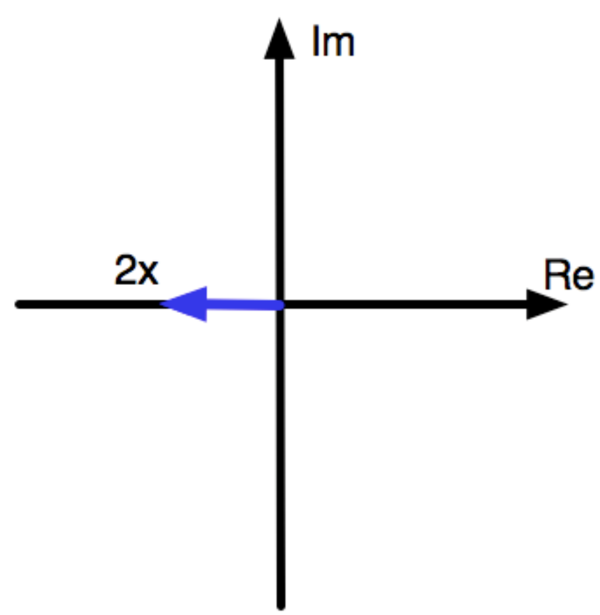


- Simple pole: f = 1 Hz
- Pole pairs: (f = 1 Hz, Q)

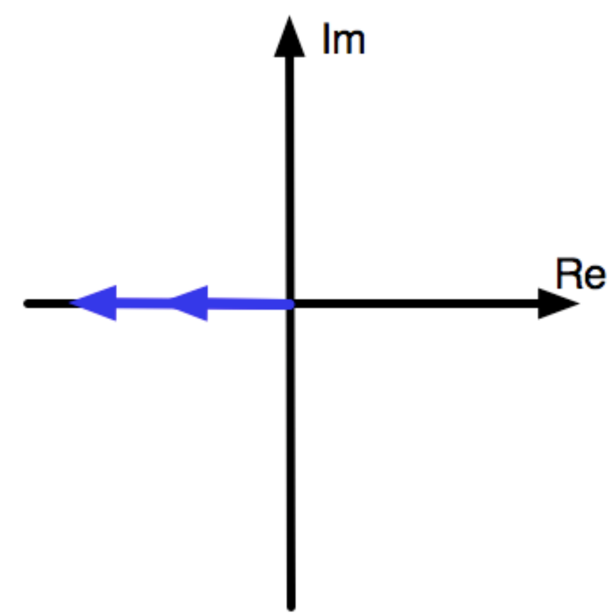
Q > 0.5
(underdamped)



Q = 0.5
(critically damped)



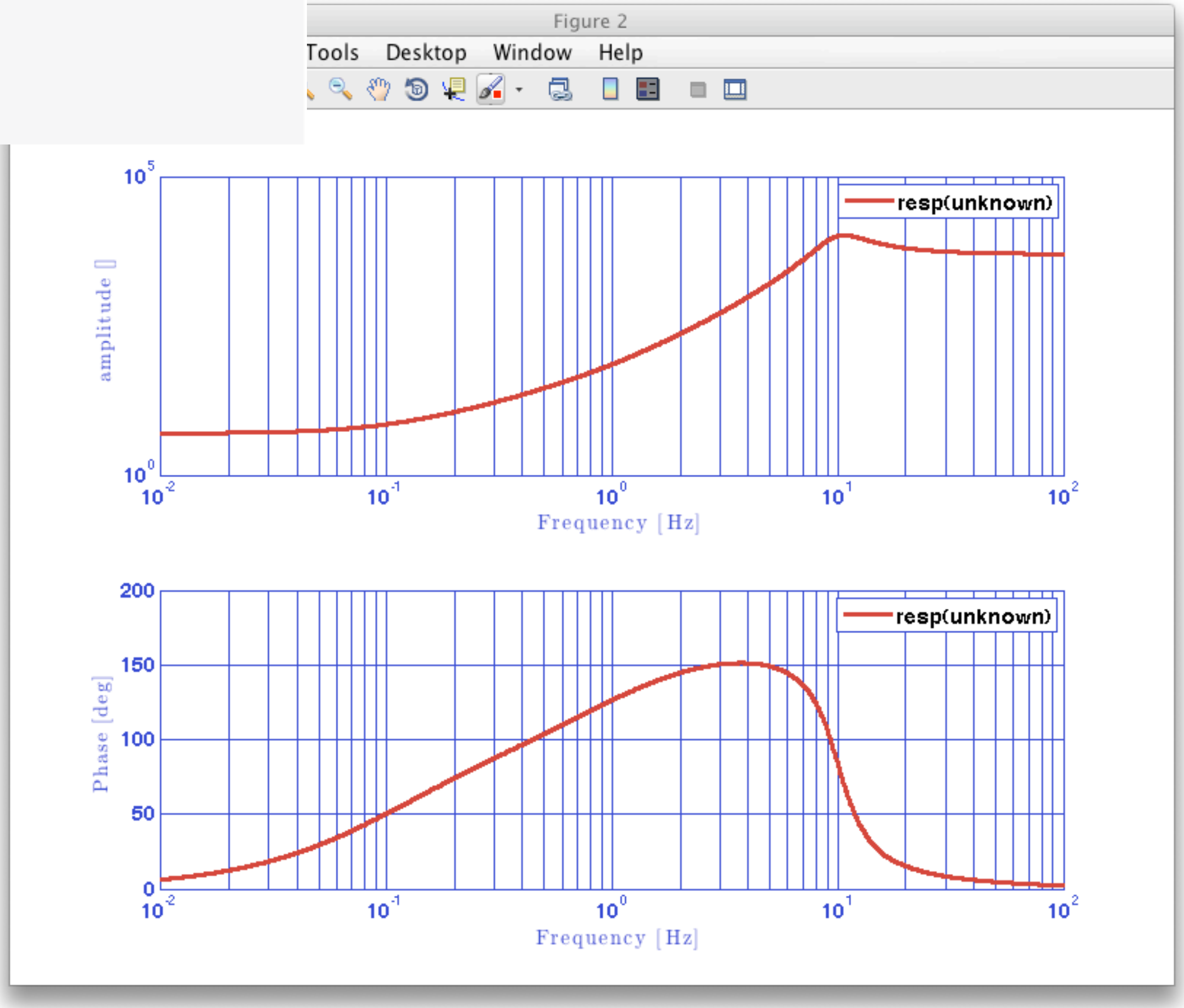
Q < 0.5
(overdamped)



1.1 Pole zero model

- Working example: Compute pole zero response
  - Topic 4 > Create transfer func... > Create pole zero model

Key	Value
GAIN	5
POLES	(f = 1 Hz, Q = 2)
ZEROS	(f = 1 Hz), (f = 0.1 Hz)



1.2 Rational model

- A rational model is defined by:
  - Num. and den. coefficients

$$H(s) = \frac{a_1\, s^m + a_2 s^{m-1} + \ldots + a_{m+1}}{b_1\, s^n + b_2\, s^{n-1} + \ldots + b_{n+1}}$$

- LTPDA constructor: RATIONAL
  - RATIONALs can NOT be multiplied and divided

- Working example: Compute rational response
  - Topic 4 > Create transfer func... > Create rational model

1.3 Partial fraction model

- A partial fraction model is defined by:
  - Poles, residues and direct terms

$$H(s) = K(s) + \sum_{i=1}^{N} \frac{R_i}{s - p_i}$$

- LTPDA constructor: PARFRAC
  - PARFRACs can NOT be multiplied and divided

- Working example: Compute par. frac. response
  - Topic 4 > Create transfer func... > Create par. frac. model

1.4 Transforming models

- Only rational <-> pzmodel translation is implemented in v2.0
  - Works inserting object into constructor, e.g. rat = rational(pzm)

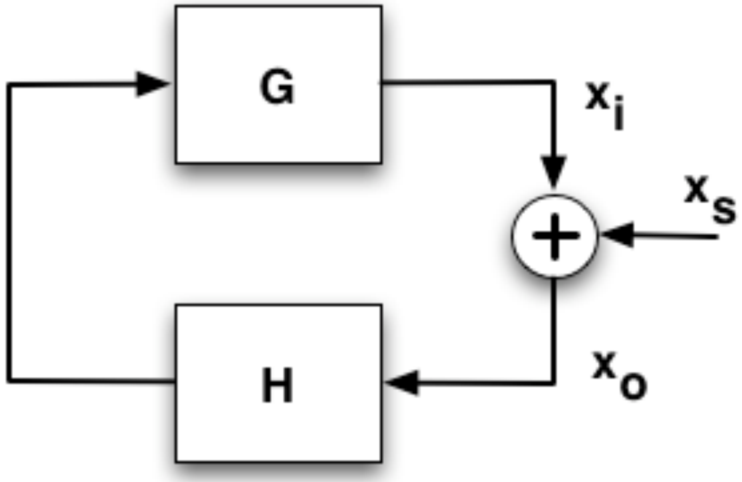
	Pole/Zero	Rational	Partial Fraction
Pole/Zero		√	X
Rational	√		X
Partial Fraction	X	X	

- Working example: pzmodel -> rational -> pzmodel
  - Topic 4 > Transforming models between representations

2. Modelling a system

- Modelling a closed loop system with pzmodels
  - Basic pzmodel operations

- Our system:



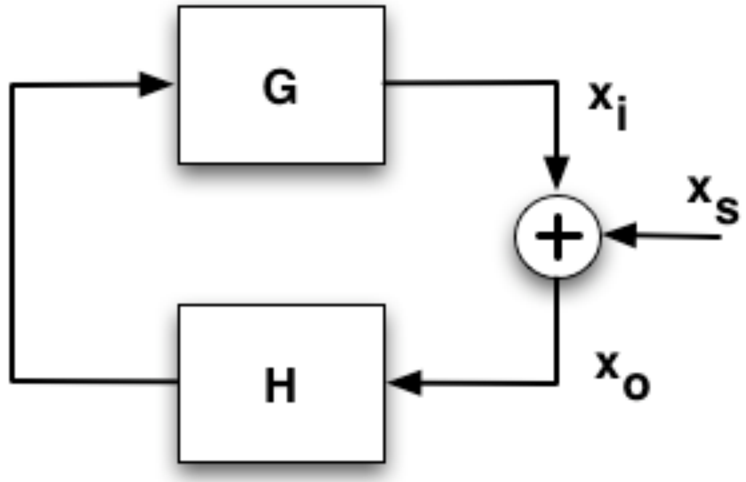
$$\frac{x_o}{x_s} = \mathrm{CLG} = \frac{1}{1 - \mathrm{OLG}} = \frac{1}{1 - \mathrm{H} \cdot \mathrm{G}}$$

- Stating the problem
  - Assuming OLG and H known, determine H and CLG

2. Modelling a system

- Step-by-step

  1. G = OLG/H
     (G is a pzmodel)
  2. Operate on G: setName, simplify ...
  3. CLG = 1/(1-OLG)
     (CLG is NOT a pzmodel)
  4. Repeat loading H with delay



$$\frac{x_o}{x_s} = \mathrm{CLG} = \frac{1}{1 - \mathrm{OLG}} = \frac{1}{1 - \mathrm{H} \cdot \mathrm{G}}$$

- Working example: Modelling a system
  - Topic 4 > Modelling a system

3. Entering the discrete domain

- The LTPDA toolbox allows you to build digital filters...
  - Discretizing your model
    - Example: find the filters for H,G, OLG in our closed loop
  - Defining filter properties
    - Example: Design a bandpass filter to evaluate power spectrum in a bandwidth

- Filter constructors in LTPDA
  - MIIR (IIR filters)

$$y[n] = \sum_{k=0}^{N} b[k]\, x[n-k] - \sum_{k=1}^{M} a[k]\, y[n-k]$$

  - MFIR (FIR filters)

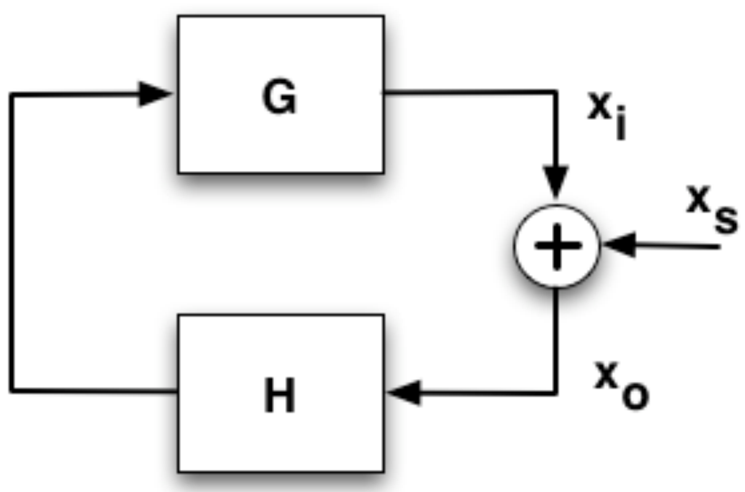
$$y[n] = \sum_{k=0}^{M} b[k]\, x[n-k]$$

3.1 By discretizing a transfer function

- Syntax: insert pzmodel into constructor

```
Gd = miir(G,plist('fs',10));
```

- miir → Constructor
- G → pzmodel obj.
- Gd → filter obj.



- Step-by-step
  1. Discretize G,H,OLG
  2. Compare continuous and digital response
  3. Get filter coefficients
  4. Delay is NOT used in the discretization!!

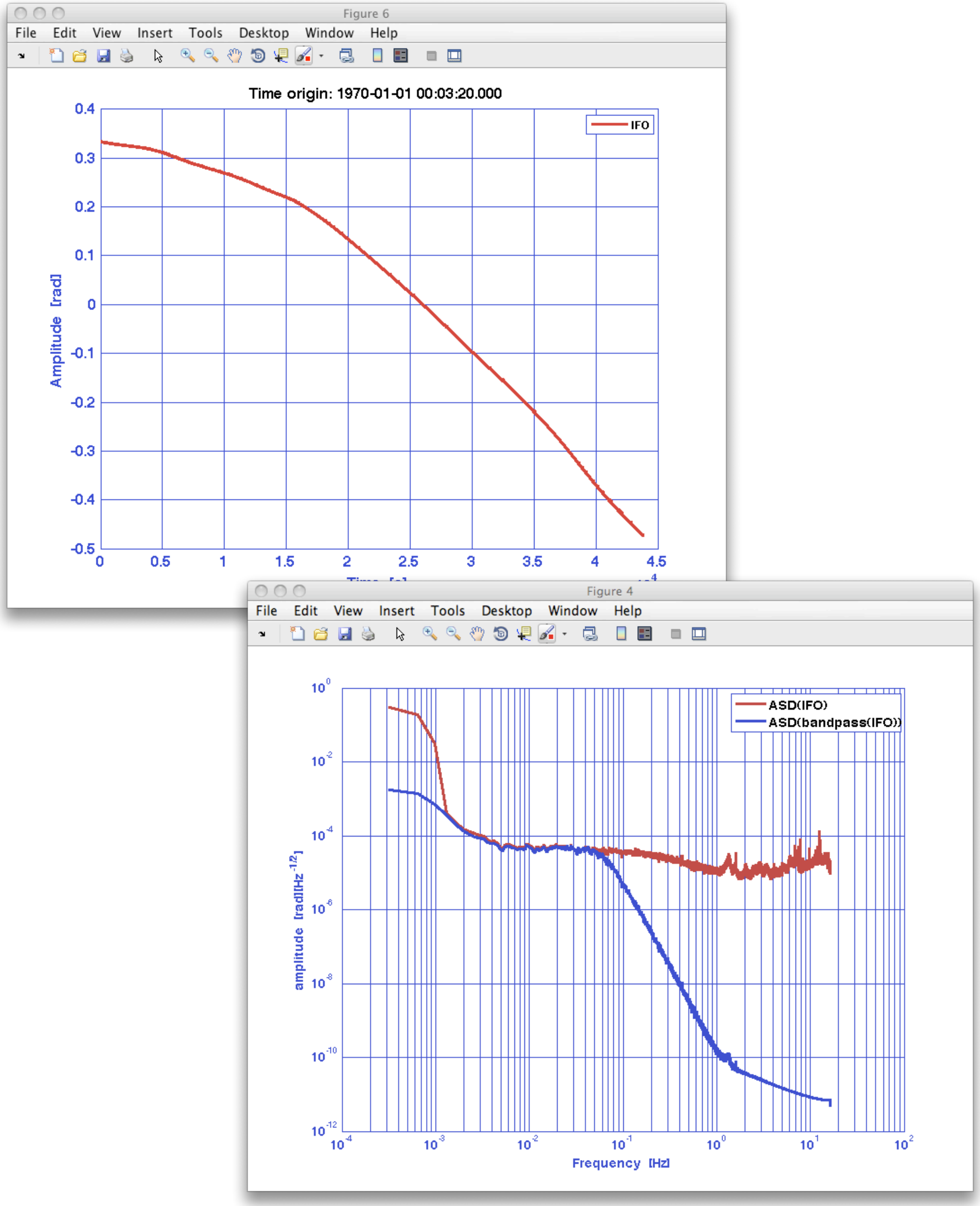
- Working example: Get filters for closed loop pzmodels
  - Topic 4 > How to filter data > By discretizing...

3.2 By defining filter properties

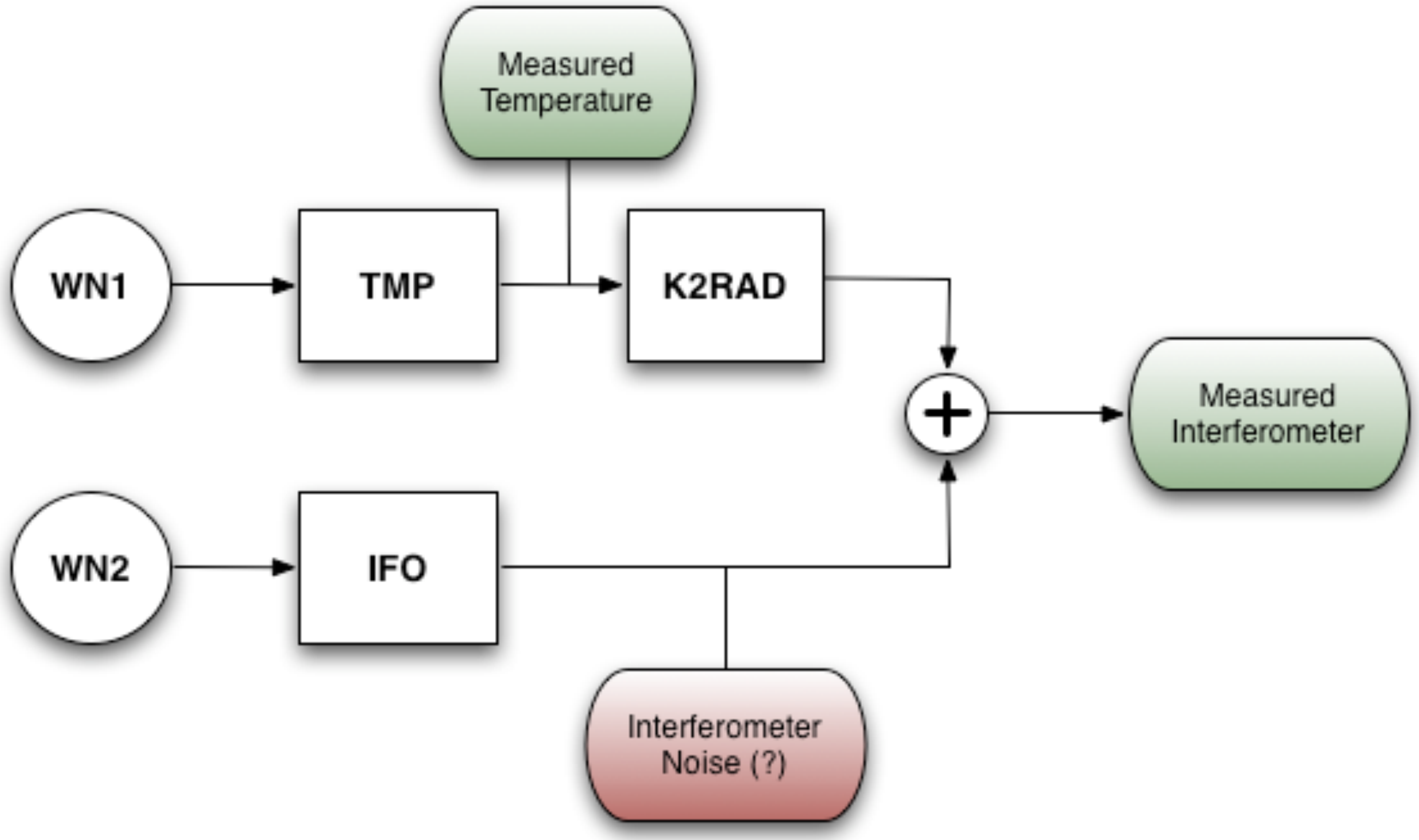
- Design a bandpass filter
  - Standard pre-processing step used in LTP lab
  - Alternative to detrending
- Syntax:

```
Gd = miir(plist('fs',32.47, 'order',...));
```

- Working example: Band pass



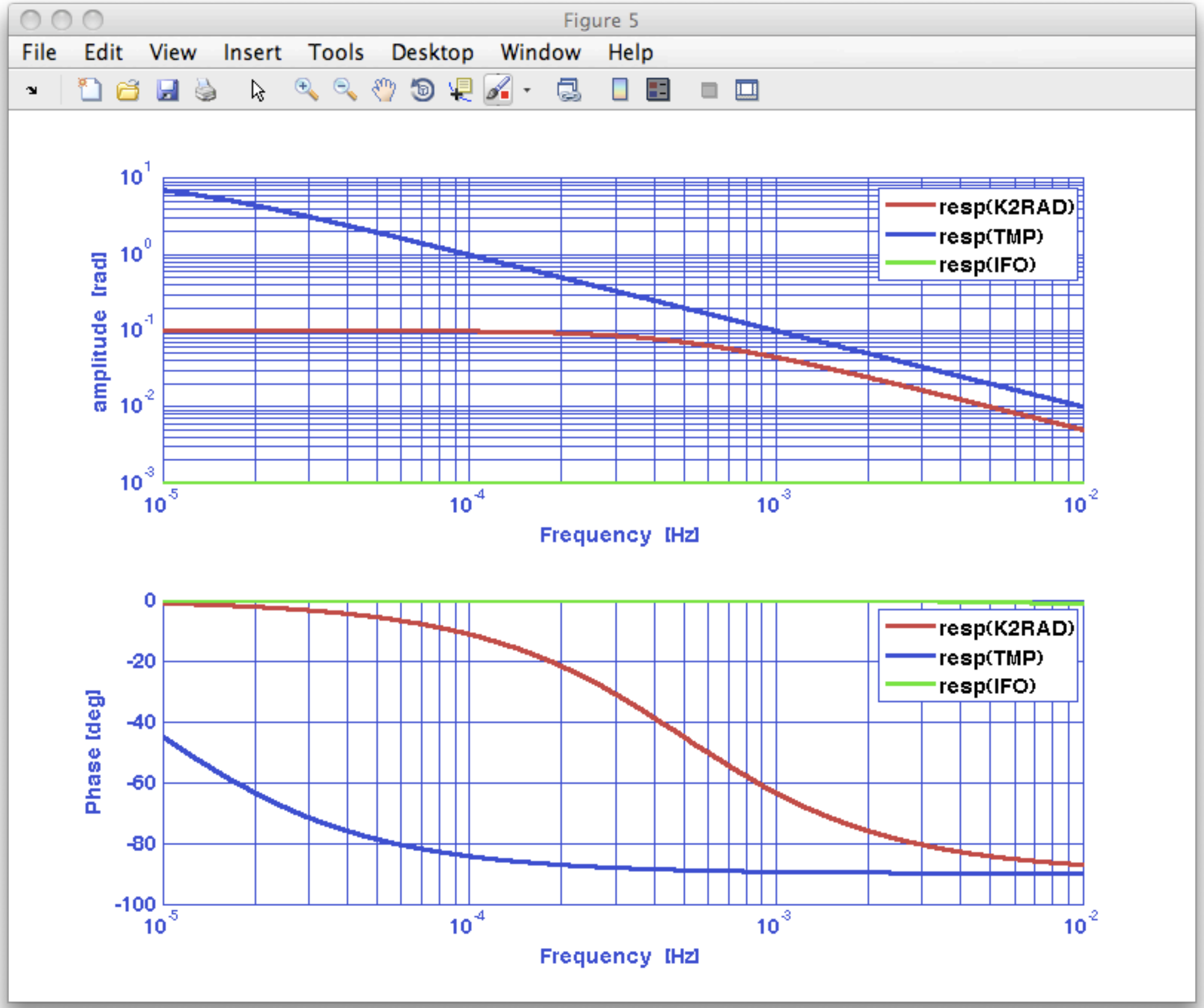
4. IFO/Temperature Example



- Aim: perform the analysis with a toy model
    - Create transfer function models: TMP, IFO,K2RAD
    - Discretize
    - Filter (white noise) data
    - Estimate transfer function (topic 3) with synthetic data

4. IFO/Temperature Example

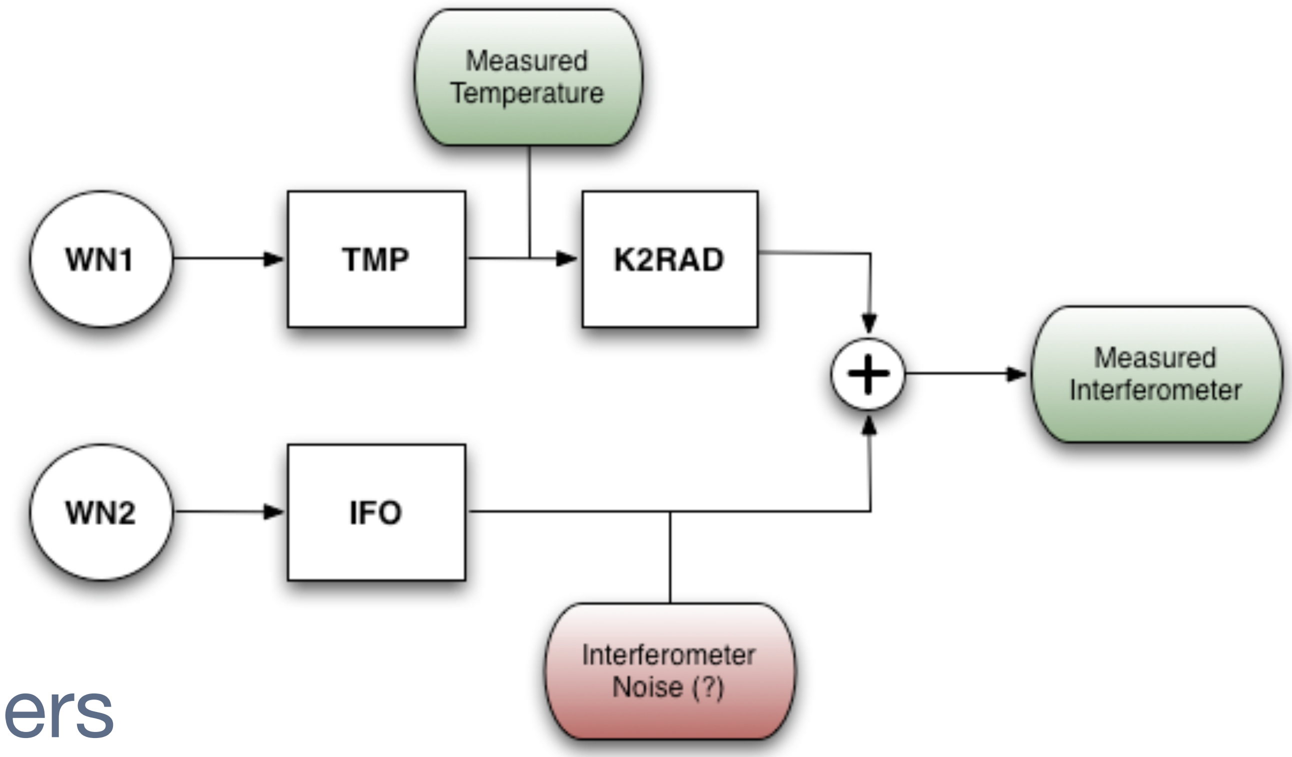
- Our toy models

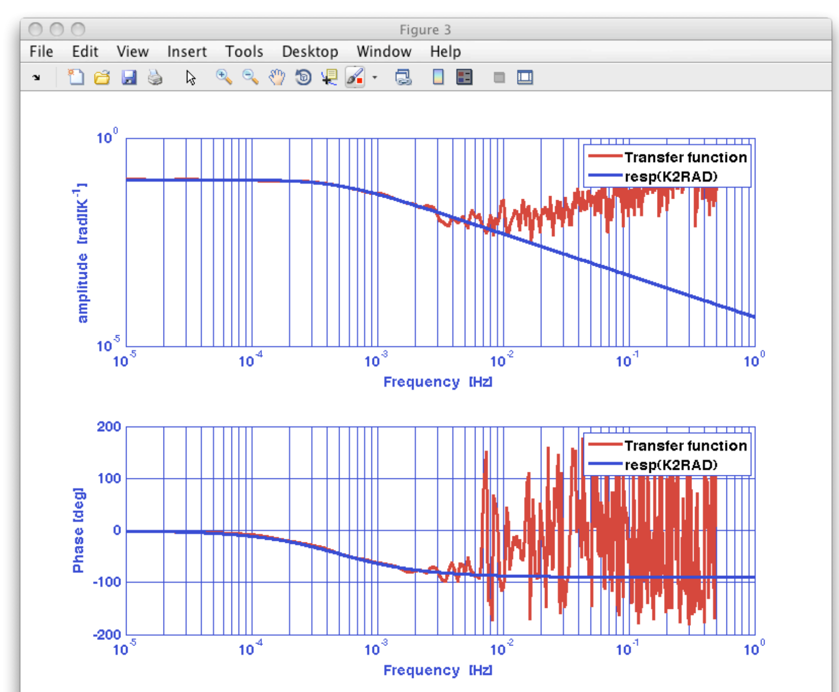


4. IFO/Temperature Example

- Step-by-step

1. Generate models: TMP, IFO, K2RAD
2. Discretize
3. Build two white noise time series
4. Filter with the digital filters
5. Estimate transfer function
6. Project temperature noise





- Working example: IFO/Temperature