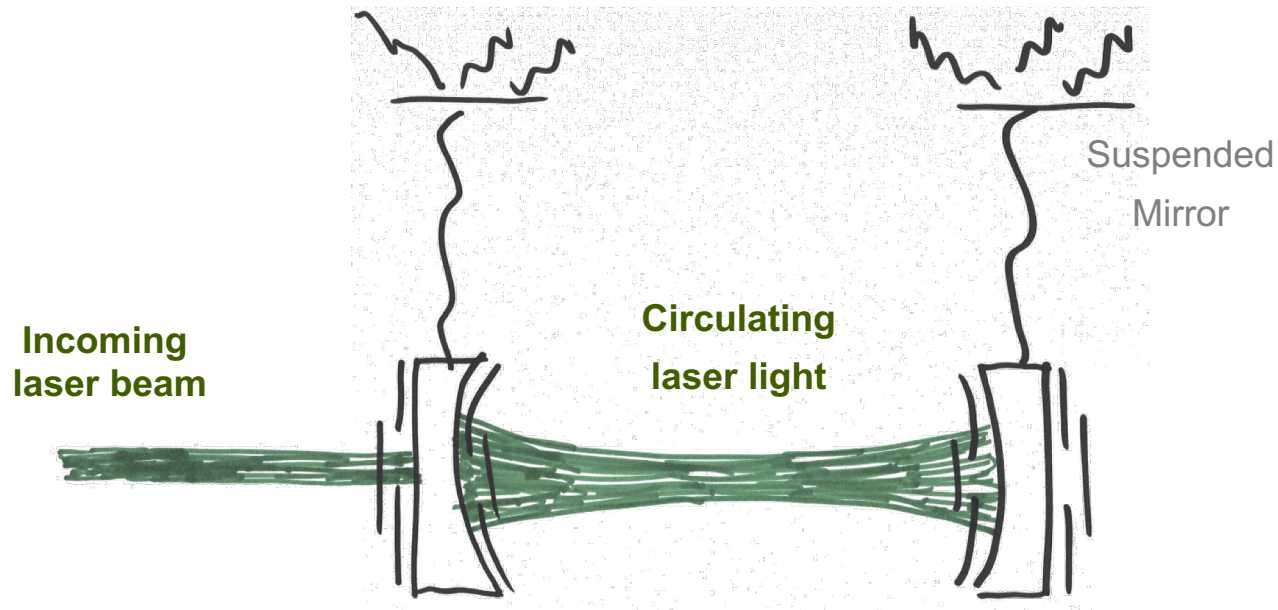
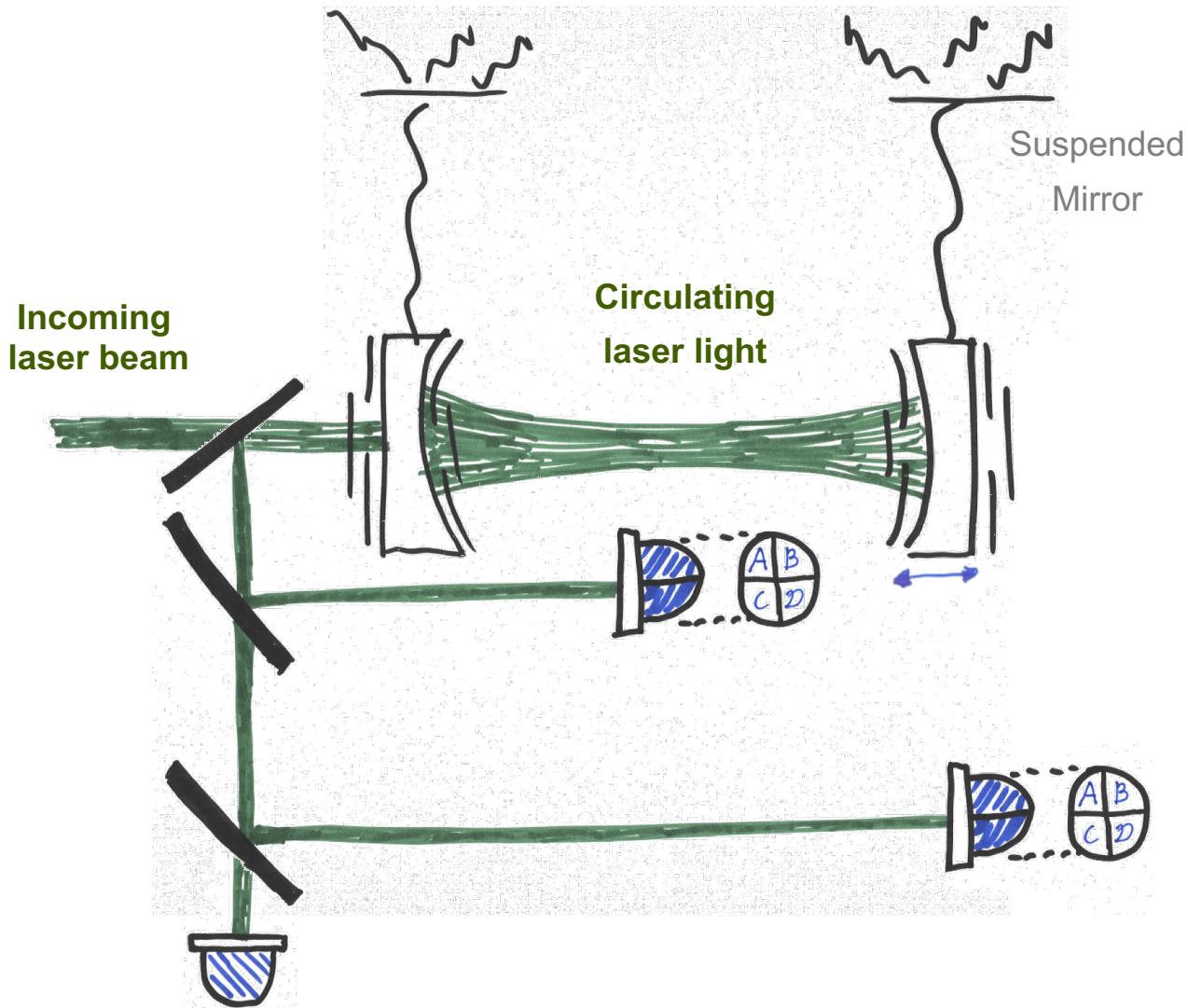




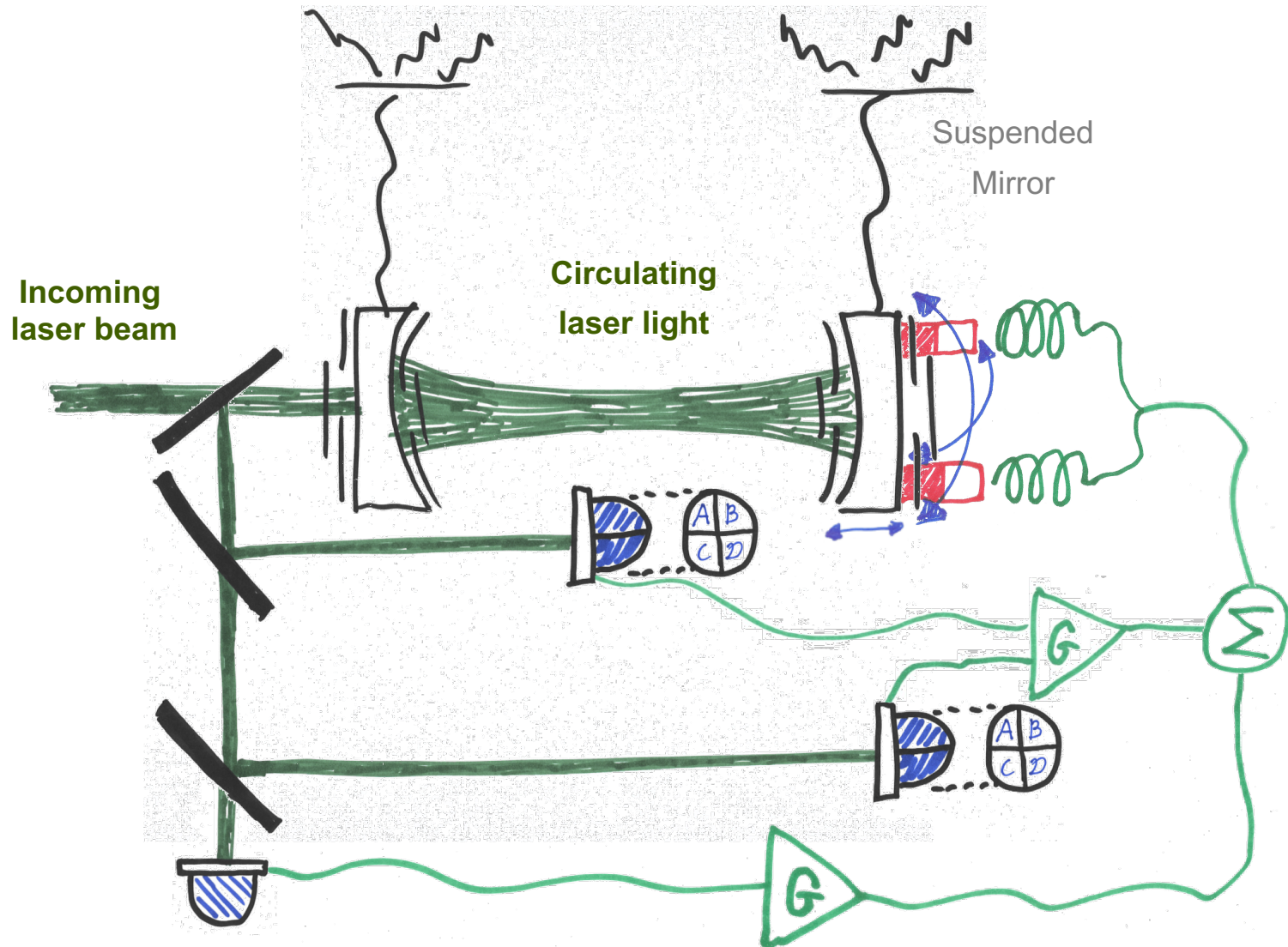
# The traditional control of a suspended optical cavity



# The traditional control of a suspended optical cavity

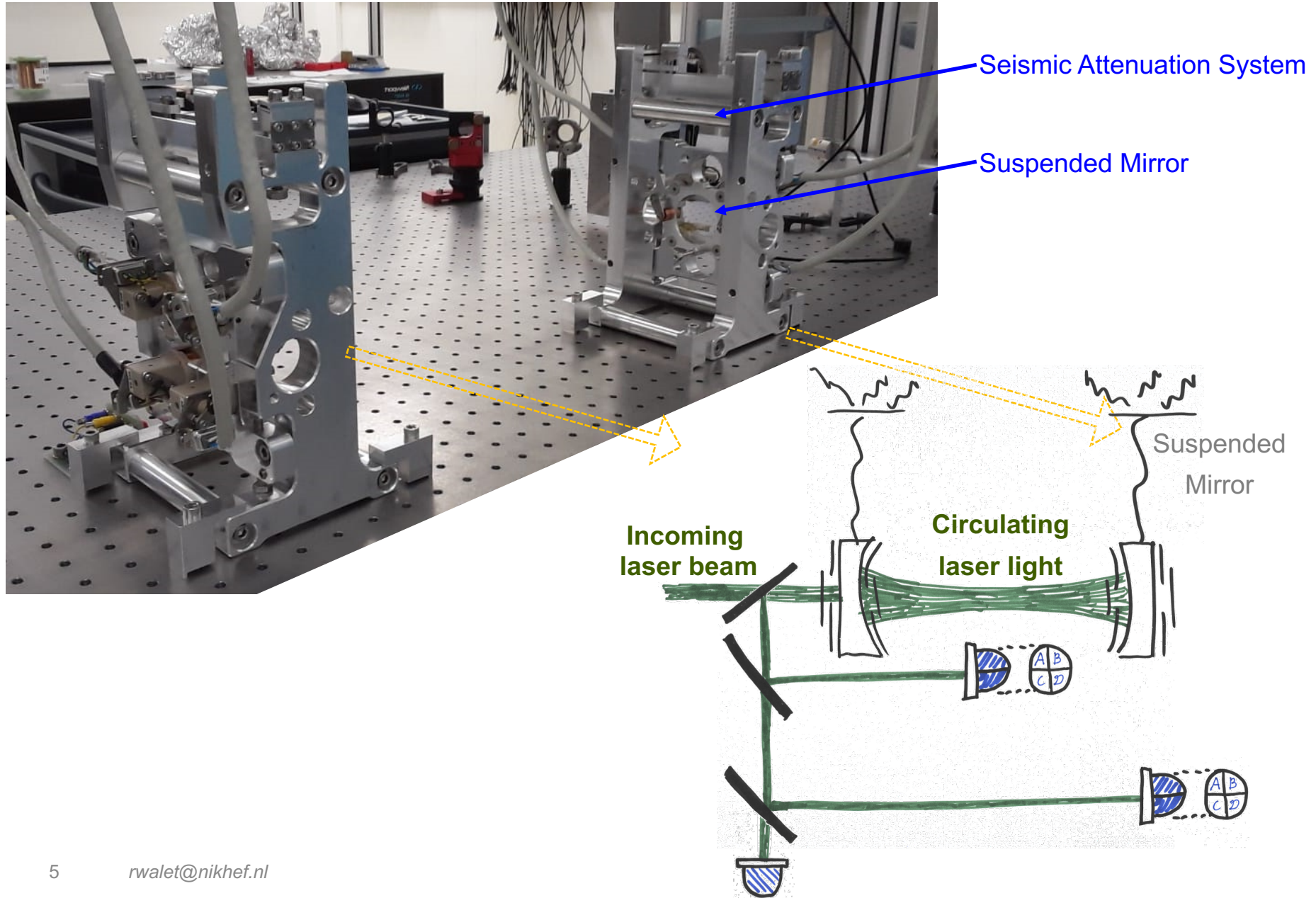


# The traditional control of a suspended optical cavity

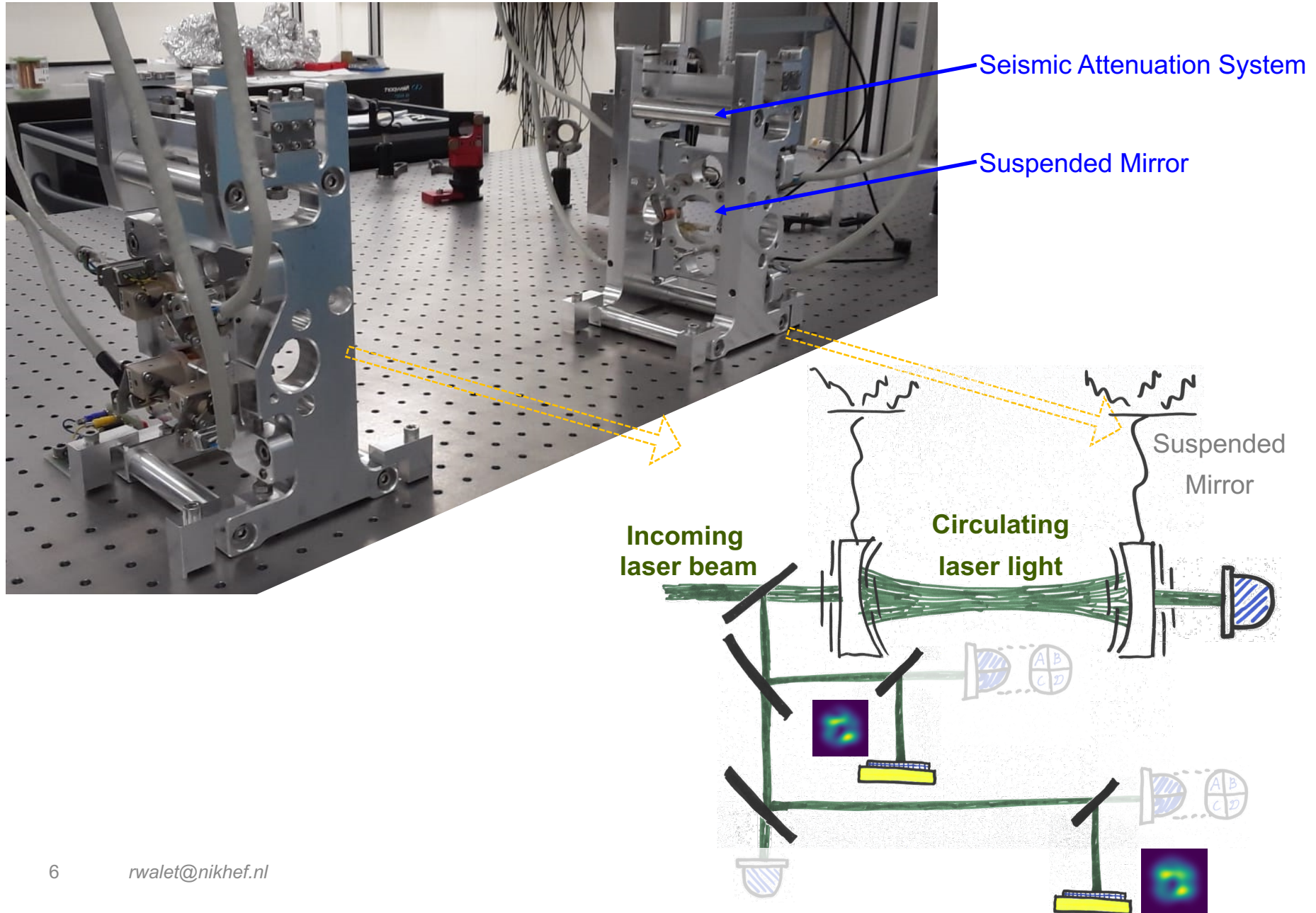




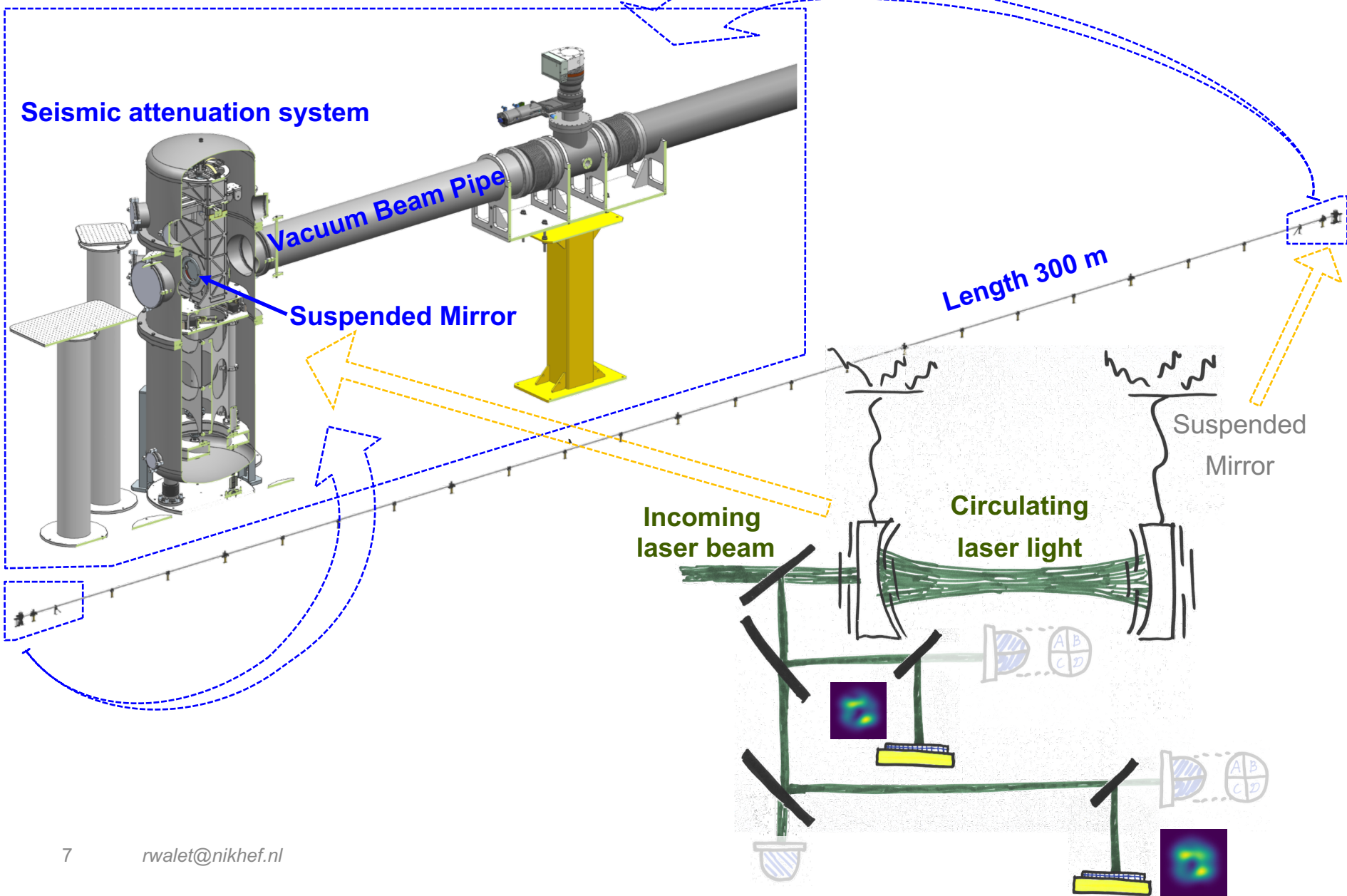
# ML bases controls of a suspended optical cavity



# ML bases controls of a suspended optical cavity



# ML bases controls of a 300m suspended optical cavity



**Questions?**

# Backup



# Single cavity - Longitudinal Control

"Let's focus on a single arm cavity. A resonance cavity can not be simply locked by the carrier field. Phase modulation of the carrier is applied to create error signals needed to keep the system at resonance"

## Pound-Drever-Hall (PDH) technique

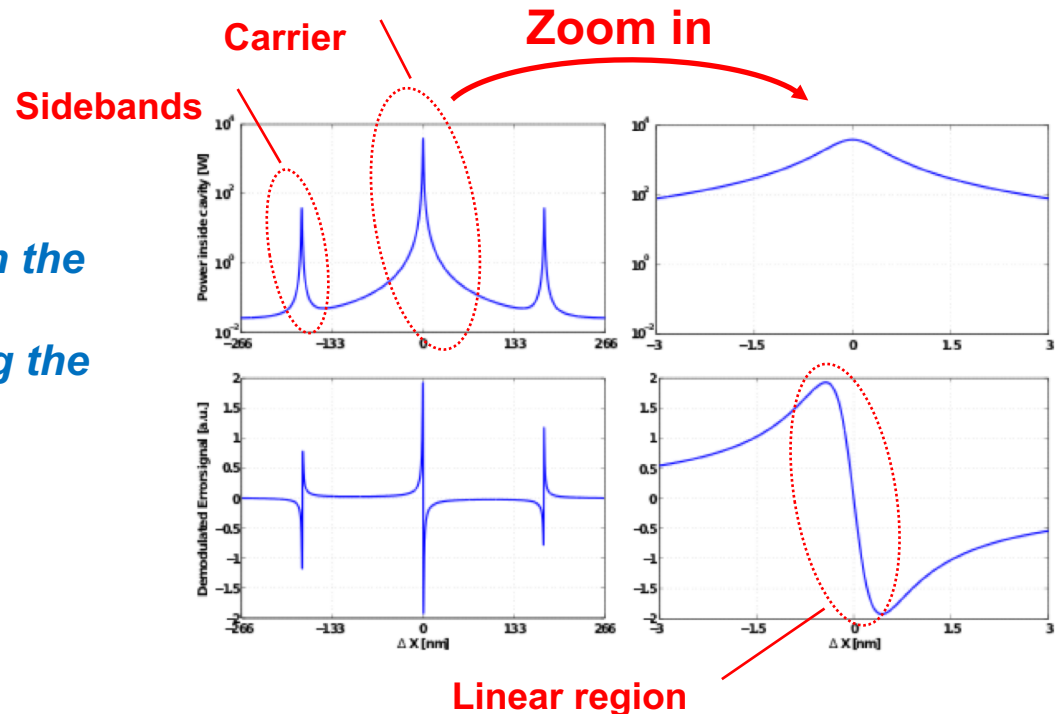
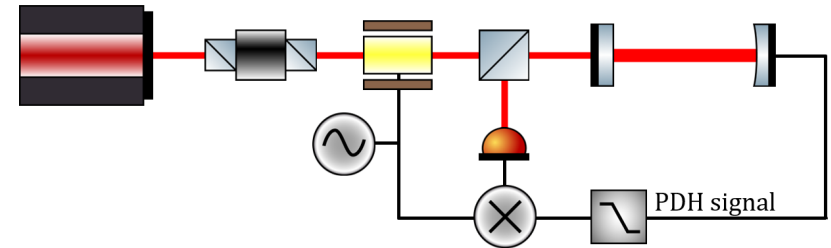
Electro Optical Modulator (EOM)  
create sidebands around the carrier

The carrier field resonates inside the cavity while the sidebands are anti-resonant, therefore they are reflected

The PDH error signal is given by the beat note between the carrier and the sidebands (used as phase reference, carrying the cavity length information)

Phase modulation

$$E_{EOM} = E_0 \cdot e^{(i\omega_0 t + m \cos(\Omega t))}$$

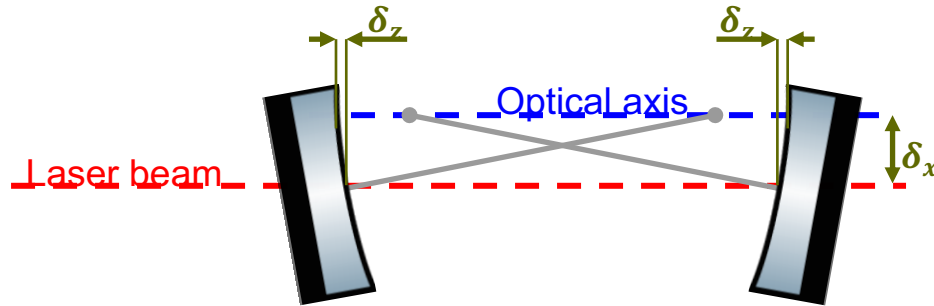




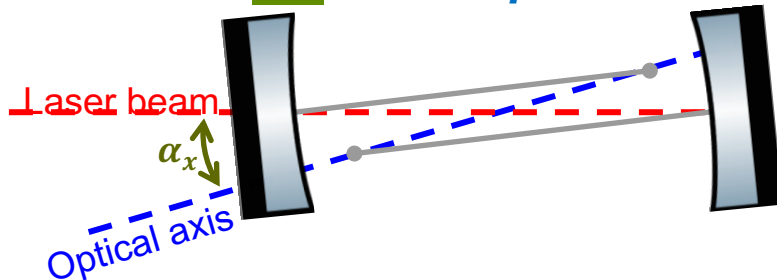
# Single Arm Cavity – Angular Alignment

“The input Gaussian beam and the optical axis of a cavity need to be aligned to prevent the occurrence of higher order modes (HOMs)”

## Shift of the optical axis



## Tilt of the optical axis



Four angular degrees of freedom / cavity

Misalignments introduce HOM

HOMs decrease the power of the fundamental mode

Misalignments change the cavity lengths

## Approximation

for  $\delta_x \ll \omega_o$

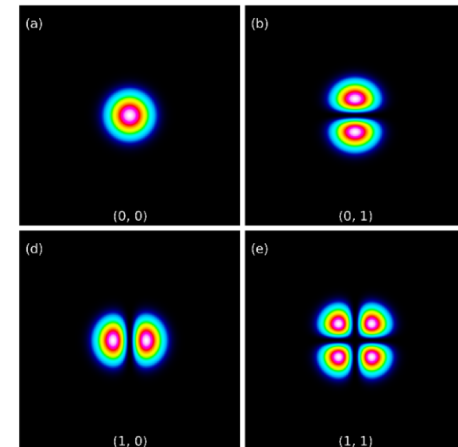
$$E(x + \delta_x) \approx A \left[ H_0(x) + \frac{\delta_x}{\omega_o} H_1(x) \right]$$

/ Beam divergence

for  $\alpha_x \ll \theta_d$

$$E(x + \delta_x) \approx A \left[ H_0(x) + \frac{\alpha_x}{\theta_d} H_1(x) \right]$$

## Hermite-Gaussian modes (with degrees, $n$ and $m$ )



# Single Arm Cavity – Angular error signals

## Dithering line (mechanical modulation)

Angular oscillation  $\omega_{mij}$  of the two rotational DOF of the mirror

Misaligned optics create a cavity length change  $\delta_z$

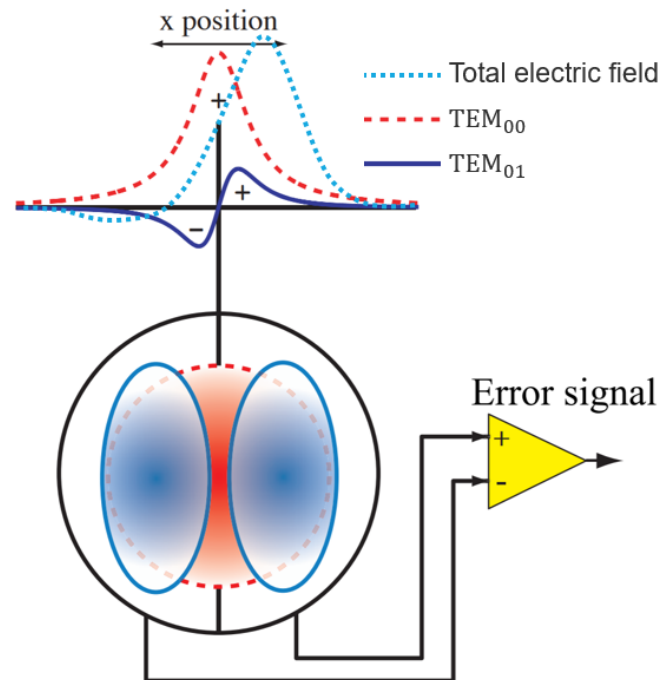
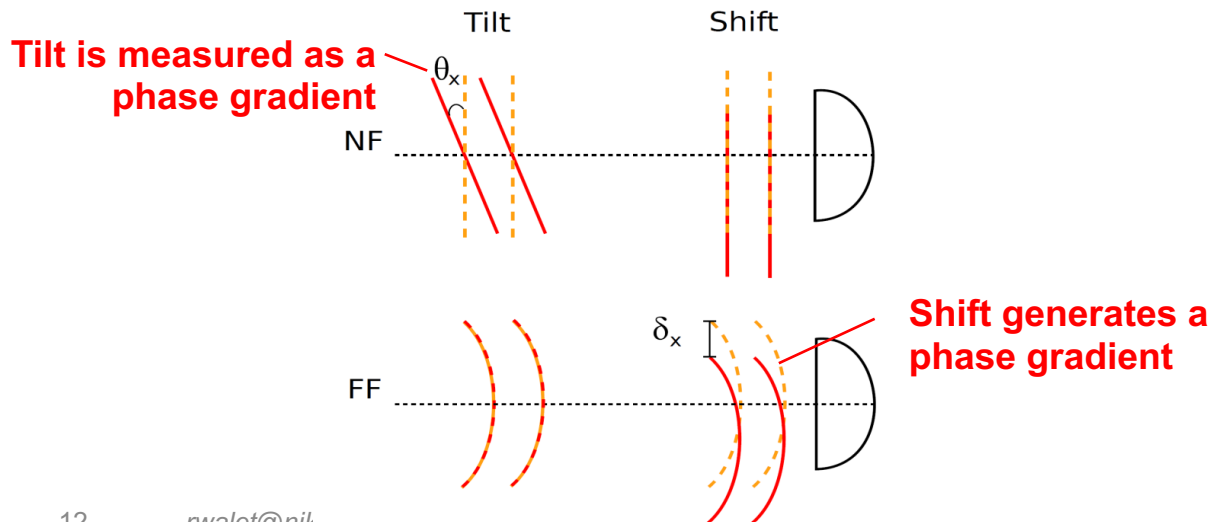
Error signal created by demodulating the longitudinal signal by  $\omega_{mij}$

## Ward technique (Phase modulation)

Usage of the longitudinal modulation frequency by EOM

Tip and tilt are both mixed in one error signal

Two\* QPDs in reflection of the cavity measures the  
spatial beam phase distribution  
(\*near and far field)



# VIRGO fact sheet

Test-masses longitudinal degrees of freedom 4 (5)

Test-masses angular degrees of freedom 16 (18)

Seismic Attenuation Systems  
for all optical components

Mirror Alignment  $10^{-9}$  [rad/min]

Mirror Positioning  $10^{-12}$  [m/min]

But there is more

Real time

~10.000 control loops  
ranging from a few Hz up to 200 kHz

Complex

Error signals are strongly nonlinear functions of the mirror positions  
and can only be linearized in a very small fraction of phase space

Lots of complex tricks and manual actions  
are involved to enable classical controls

Making the commissioning a continuous challenge relying on well  
trained, highly experienced people

...M€ systems

GW Observatory → one can not afford downtime

Trade-off between interventions and the risk of downtime

Simulations and test facilities are key to drive  
break-through innovations

## Longitudinal DOF

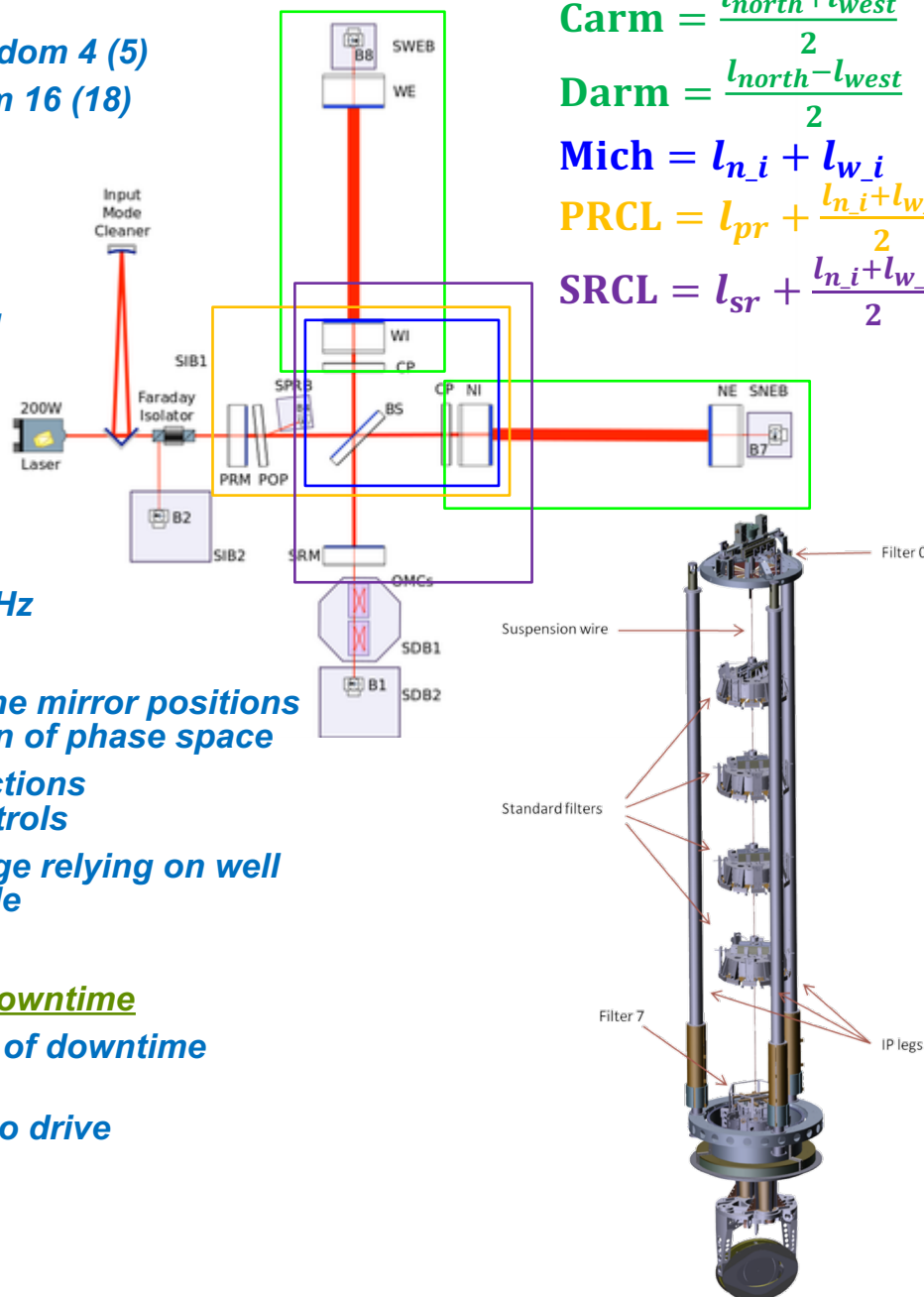
$$C_{arm} = \frac{l_{north} + l_{west}}{2}$$

$$D_{arm} = \frac{l_{north} - l_{west}}{2}$$

$$Mich = l_{n_i} + l_{w_i}$$

$$PRCL = l_{pr} + \frac{l_{n_i} + l_{w_i}}{2}$$

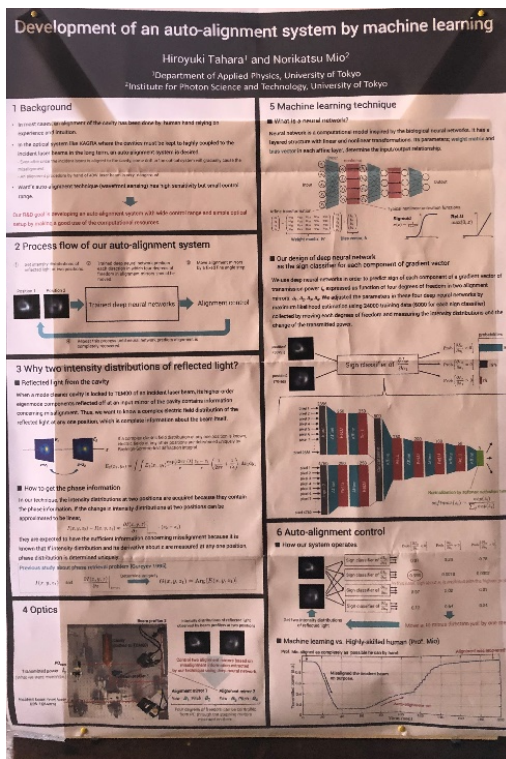
$$SRCL = l_{sr} + \frac{l_{n_i} + l_{w_i}}{2}$$



# ML in the GW field

*No (public) experimental success stories yet! → not a continuous development*

## University of Tokyo



**Experimental Results!!**  
**But for rigid optic**

## University of Birmingham

UNIVERSITY OF BIRMINGHAM  
SCHOOL OF PHYSICS AND  
ASTRONOMY



Fourth Year Project Report  
March 2019

Alignment of LIGO Fabray-Perot Interferometer using  
Machine Learning

Robert Beesty  
Student ID: 1526199  
In Collaboration with Emma Morley

Supervisors: Prof. A Freise, Dr. D Martynov  
With Thanks to G Smetana

**Only theory!!**  
**Just one mirror tilted**

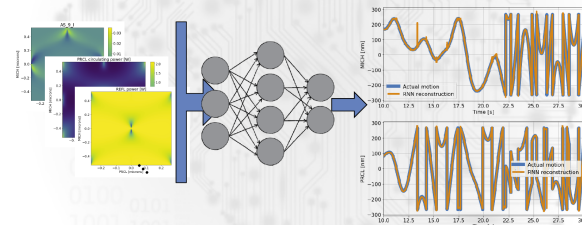
## University of Caltech/MIT

- Analytical model of fields, plane wave approximation, implemented in python
- Neural Network training using TensorFlow / PyTorch
  - About 150000 trajectories for the training
  - Training a full DNN in about one day on **4 GPUs**, using existing LIGO computing resources
  - Forget about training models on CPUs, they're hundreds of times slower!



**LIGO** Non-linear estimator of IFO d.o.f.s

- Inputs:** optical error signals (POP\_DC, REFL\_DC, POP\_1F\_I/Q, etc...)
- Outputs:** MICH, PRCL, etc.. positions



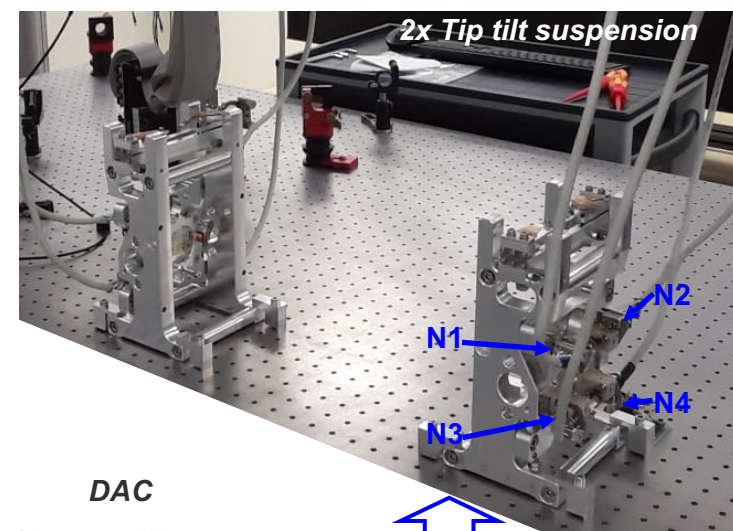
We are dealing with time series of signals: the instantaneous values are not enough to predict the MICH/PRCL positions.  
We need to feed the network some past history of optical error signals:  
**Recurrent Neural Networks (RNN) maintain an internal memory**

**Simulations are great!**  
**Network is trained! WORKING**  
**NO experimental results!!**

**40m Prototype too complex to start with?!**



# Table-top suspended FP cavity



2x Tip tilt suspension

N1  
N2  
N3  
N4

8x dsub 9

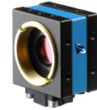


2x Break out box

VC 4x BNC SS 4x BNC

ML Error Signals

DMK 42AUC03



2x CCD



ADC

16 bit – 40Mps

32 channels

Differential



DAC

16 bit – 500ksps

16 channels

Differential

8x BNC

30x BNC

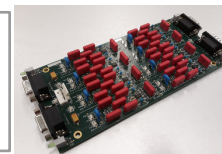
AA Crate

32 channels

Differential

3x AA board

6 x dsub 9 (4ch)



AI Crate

16 channels

Differential

2x AI board

4 x dsub 9 (4ch)

8ch

7x [dsub9 (DF) to 4xbnc (SE)]

8 ch

2x [dsub9 (DF) to 4xbnc (SE)]

2x dsub9(DF)

6 ch

1x Break out box  
16 channels  
Single ended

6 ch spare

2x [dsub25 to 2x dsub9]

16 ch

4 ch

bnc to 1x bnc

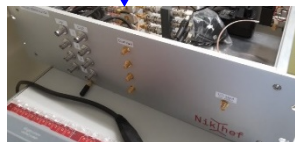
8 ch

2x QPDio-base



4x bnc 1x dsub 25

2x QPD

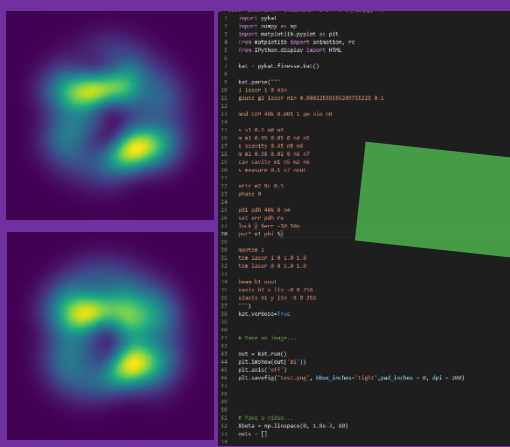


2x PD

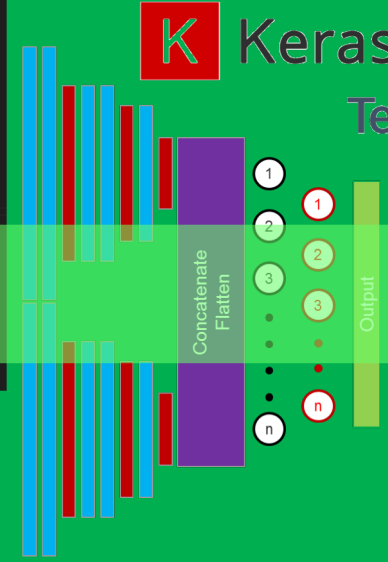


## Software Architecture – auto alignment system

## Pykat Finisse trainings data

[illegible]

## Convolutional Recurrent Neural Network



Keras  TensorFlow

## Realtime C-Code

[illegible]

## Optimize memory by spare matrices??

```

// 1. Program for Sparse Matrix Representation
// using Linked lists
#include<iostream.h>
using namespace std;

// Node to represent sparse matrix
struct Node
{
    int value;
    int row_position;
    int column_position;
    struct Node *next;
};

// Function to create new node
void create_new_node(struct Node** start, int row_index_element,
                    int row_index, int column_index)
{
    struct Node *temp, *r;
    if (*start == NULL)
    {
        // Create new node dynamically
        temp = (struct Node*) malloc (sizeof(struct Node));
        temp->value = row_index_element;
        temp->row_position = row_index;
        temp->column_position = column_index;
        temp->next = NULL;
        *start = temp;
    }
    else
    {
        while (temp->next != NULL)
            temp = temp->next;

        // Create new node dynamically
        r = (struct Node*) malloc (sizeof(struct Node));
        r->value = row_index_element;
        r->row_position = row_index;
        r->column_position = column_index;
        temp->next = r;
    }
}

// This function prints contents of linked list
// starting from start
void printLinkedList(struct Node* start)
{
    struct Node *temp, *r;

    if (start == NULL)
        return;

    while(temp != NULL)
    {
        printf("%d\t", temp->row_position);
        temp = temp->next;
    }
    printf("\n");
    while(r != NULL)
    {
        printf("%d\t", r->column_position);
        r = r->next;
    }
    printf("\n");
    while(temp != NULL)
    {
        printf("%d\t", temp->value);
        printf("\n");
        temp = temp->next;
    }
}

// Driver of the program
int main()
{
    // Assume del operator is
    int sparseMatrix[10][10] =
    {
        {0, 0, 1, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
    };

    // Start with the empty list r
    struct Node* start = NULL;

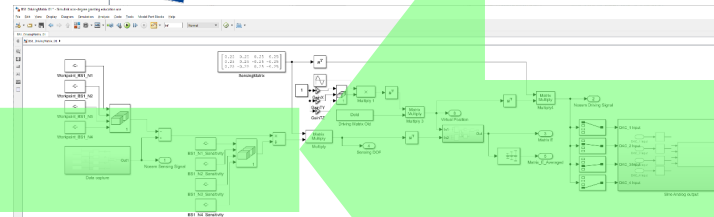
    for (int i = 0; i < 10; i++)
        for (int j = 0; j < 10; j++)
            if (sparseMatrix[i][j] != 0)
                create_new_node(&start, sparseMatrix[i][j], i, j);

    printLinkedList(start);

    return 0;
}

```

## Real data



MATLAB® & SIMULINK®



## Software Architecture – Longitudinal and angular control

## Pykat Finisse trainings data

## Convolutional Recurrent Neural Network



# Keras



# TensorFlow

## Realtime C-Code

```
void mmult_kernel(float in_A[A_ROWS][A_COLS],
                 float in_B[B_ROWS][B_COLS],
                 float out_C[C_ROWS][C_COLS]) {
    // prepare HLS inline self
    // prepare HLS array-partition variables in a block factor=16 dim2
    // prepare HLS array-partition variables in a block factor=16 dim1

    int index_a = index_b = index_d;

    for (index_a = 0; index_a < A_ROWS; index_a++) {
        for (index_b = 0; index_b < B_COLS; index_b++) {
            // prepare HLS PIPELINE II=4
            float result = 0;
            for (index_d = 0; index_d < A_COLS; index_d++) {
                // multiply operation in a block factor=16 dim2 operators
                // so that AutoTSS can infer two FP operations
                float temp = in_A[index_a][index_d] * in_B[index_b][index_d];
                result += temp;
            }
            out_C[index_a][index_b] = result;
        }
    }
}
```

## Optimize memory by spare matrices??

```
// C program for Sparse Matrix Representation
// using linked lists
#include<iostream>
using namespace std;

// Node to represent sparse matrix
struct Node
{
    int value;
    int row_position;
    int col_position;
    struct Node *next;
};

// Function to create new node
void create_new_node(struct Node** start, int row_pos_element,
                    int col_pos_element, int column_index)
{
    struct Node *temp, *r;
    temp = *start;
    if (temp == NULL)
    {
        // Create new node dynamically
        temp = (struct Node *) malloc (sizeof(struct Node));
        temp->value = row_pos_element;
        temp->row_position = row_index;
        temp->col_position = column_index;
        temp->next = NULL;
        *start = temp;
    }
    else
    {
        while (temp->next != NULL)
            temp = temp->next;
        // Create new node dynamically
        r = (struct Node *) malloc (sizeof(struct Node));
        r->value = row_pos_element;
        r->row_position = row_index;
        r->col_position = column_index;
        temp->next = r;
    }
}

// Driver of the program
int main()
{
    // Sparse matrix
    int sparseMatrix[4][5] =
    {
        { 0, 0, 0, 0, 0 },
        { 0, 0, 0, 0, 0 },
        { 0, 0, 0, 0, 0 },
        { 0, 0, 0, 0, 0 }
    };

    // Start with the empty list
    struct Node* start = NULL;

    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 5; j++)
        {
            // Find only those values which are non-zero
            if (sparseMatrix[i][j] != 0)
            {
                create_new_node(&start, sparseMatrix[i][j], i, j);
            }
        }

    // Print the contents of linked list
    // starting from start
    void Print(struct Node* start)
    {
        struct Node *temp, *r;
        temp = *start;
        while (temp != NULL)
        {
            cout << temp->value << " ";
            temp = temp->next;
        }
        cout << endl;
    }
}
```

## Real data

SPACE

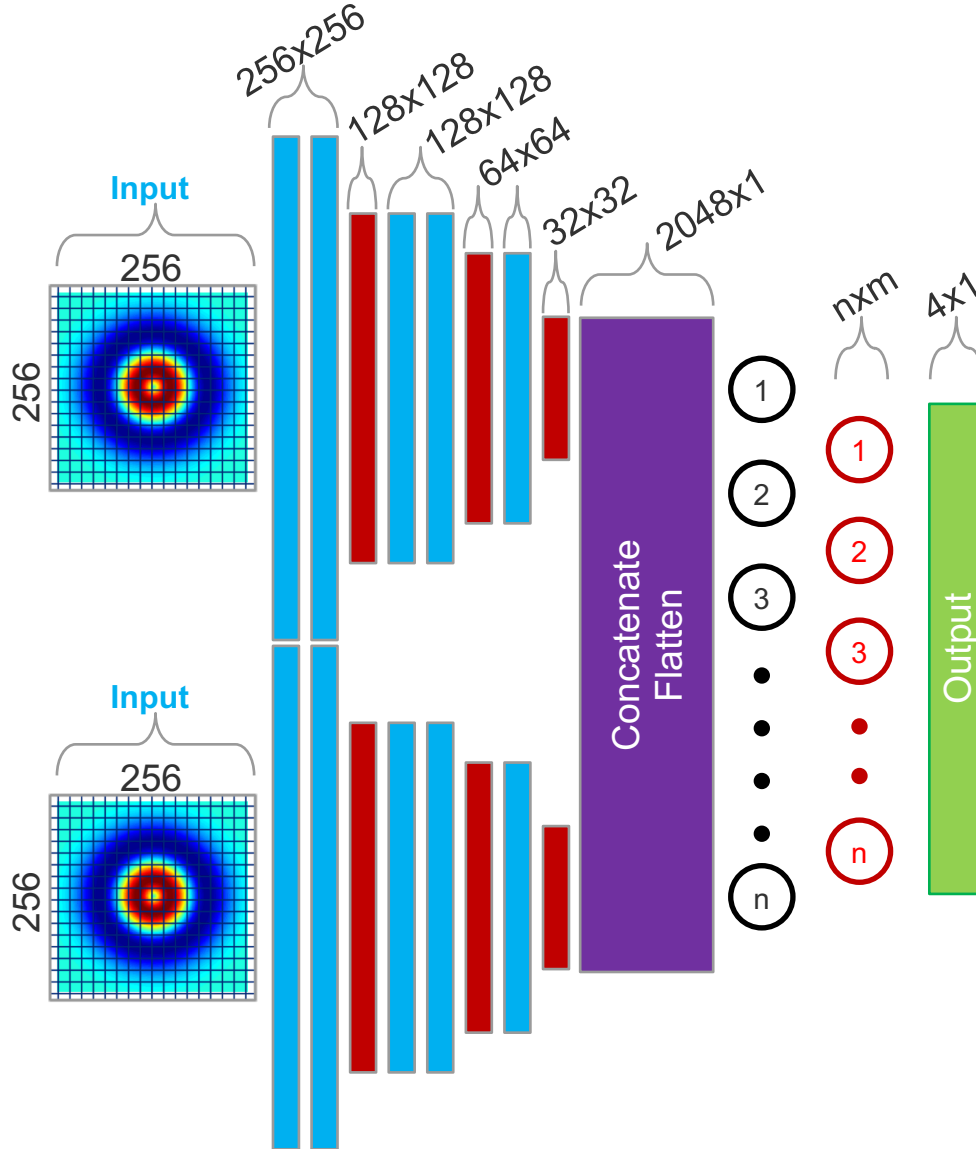


# Convolutional Recurrent Neural Network

CCD Image

CRNN

Mirror position



Convolutional Layer

Containing a RELU activation layer

Pooling Layer

Containing a drop-out to reduce overfitting

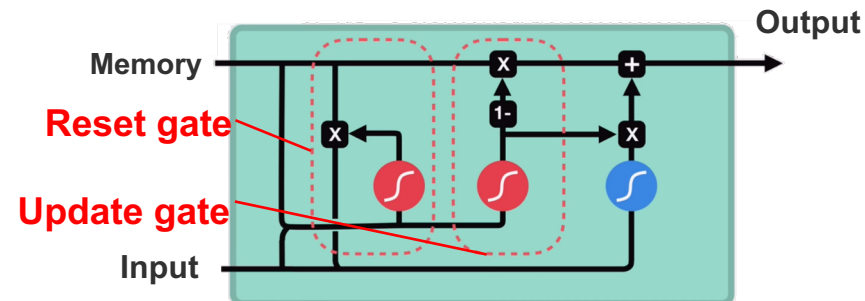
Output

Containing a Sigmoid activation function

*The auto-alignment systems needs to deal with discontinued and time varying signals*

- *The convolutional recurrent neural network will include Long Short-Term Memory (LSTM) like in speech recognition algorithms*
- *The CRNN enable auto alignment of the Fabry-Perrot resonance cavity*
- *(Atmospheric) disturbances can be compensated to maintain the lock*

Gated Recurrent Unit



# Tip-tilt suspension

*Suitable for  
50mm and 2" mirrors  
beam splitter and FP cavities*

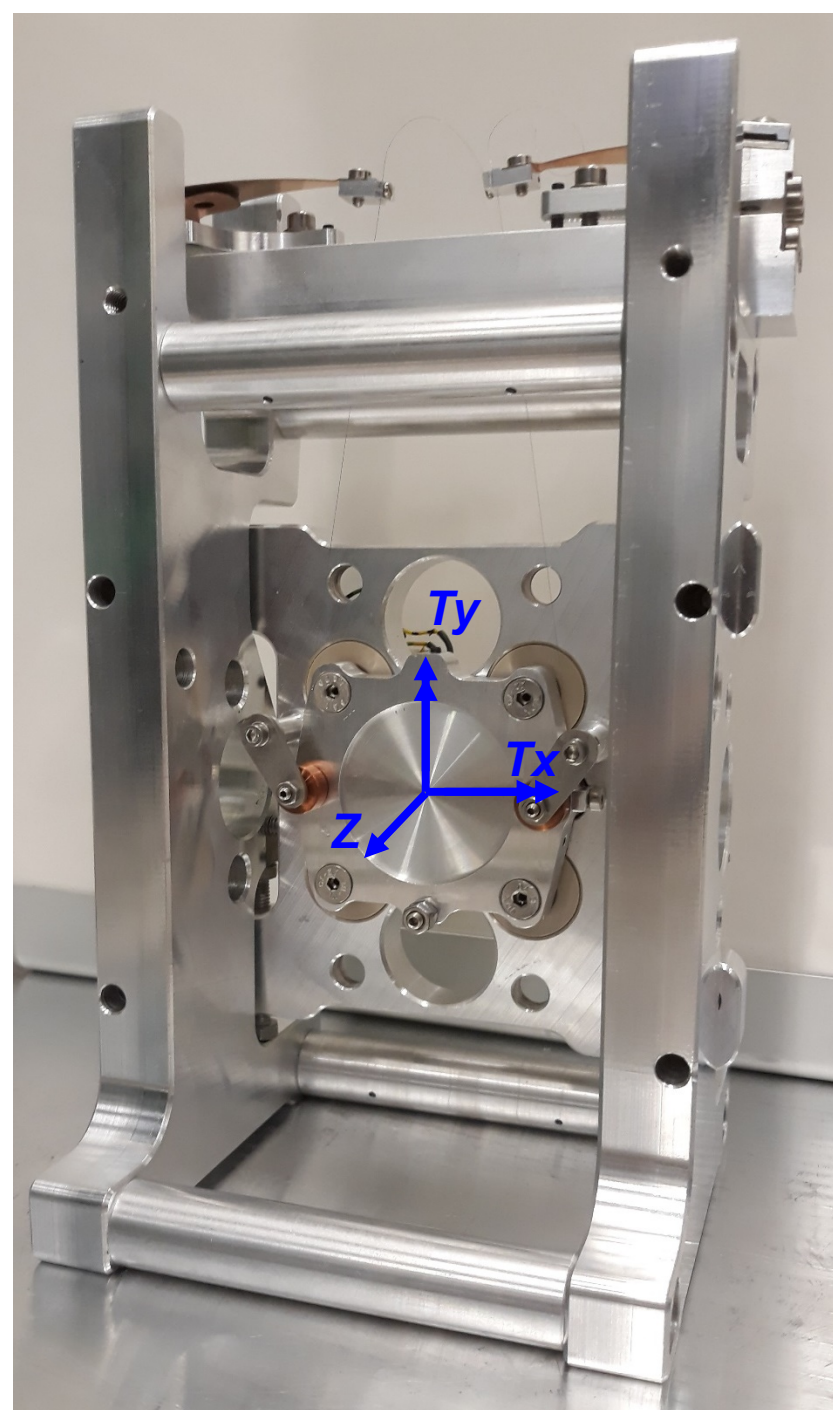
*Three DOF actively controlled  
Other DOF are passively damped*

*L x B x H*

*120 x 140 x 250 mm*

*UGF 200Hz*

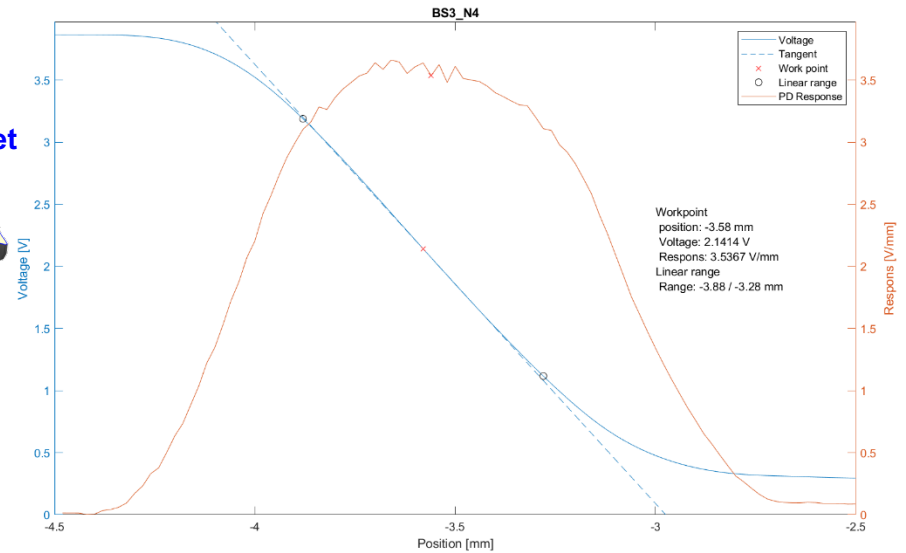
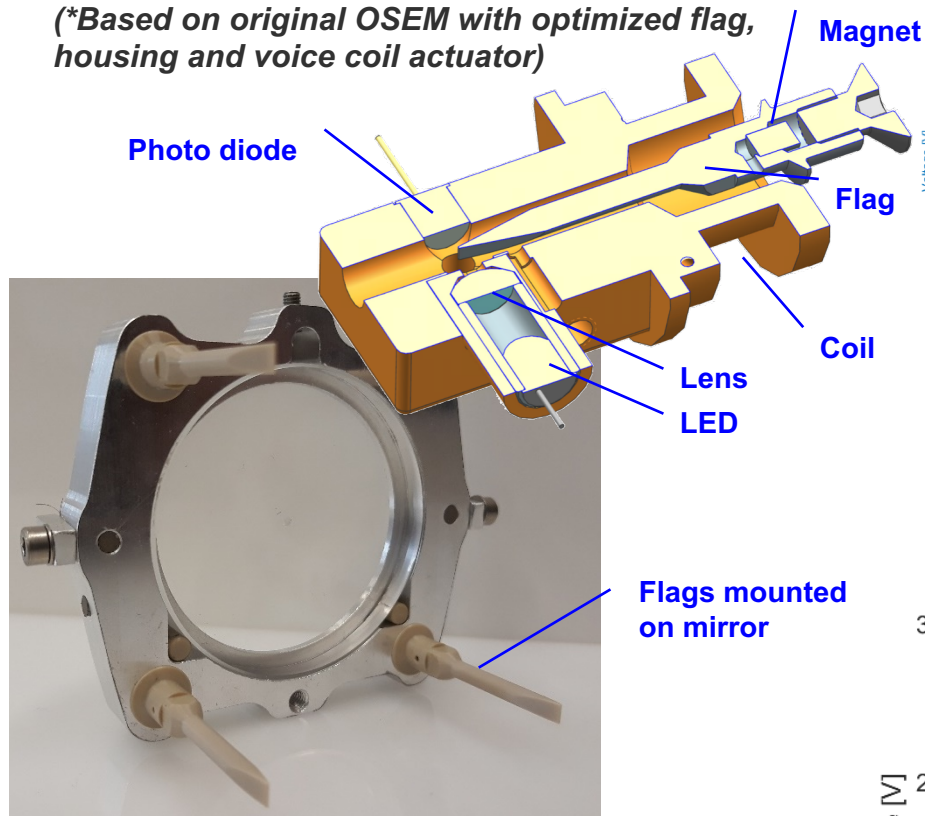
*Seven tip-tilt suspensions  
ready for use*



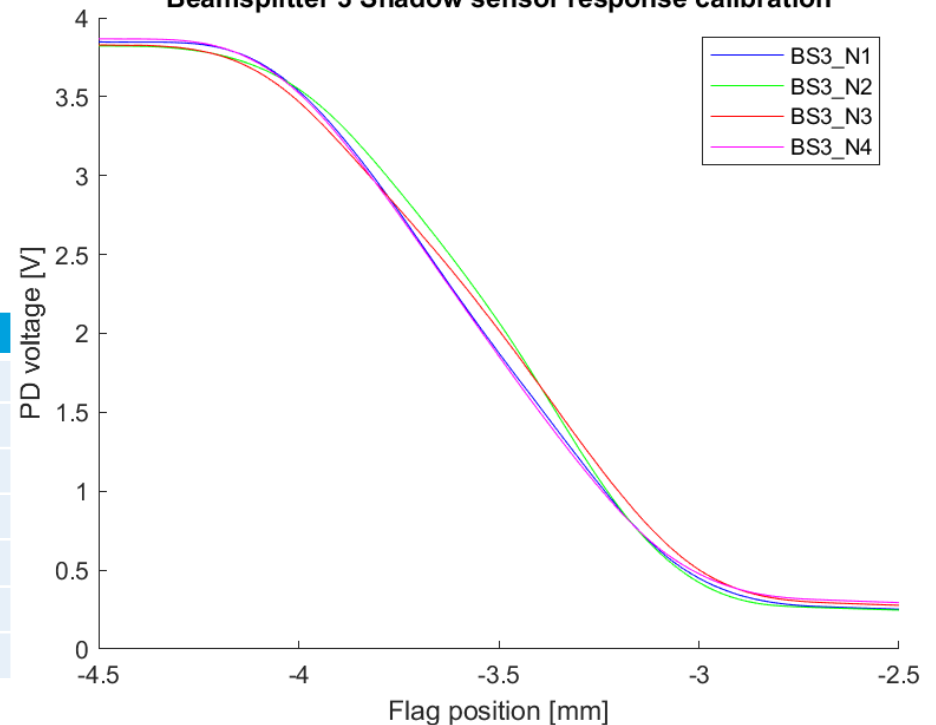
# NOSEM\*

**Nikhef\* Optical Sensor and Electro-Magnetic actuator**

(\*Based on original OSEM with optimized flag, housing and voice coil actuator)



**Beamsplitter 3 Shadow sensor response calibration**



Specifications	Value	Unit
Linear range	>0.7	mm
Horizontal coupling	2.3	%
Vertical coupling	0.8	%
Angular coupling	0.5	%
Noise @ 10Hz	2.6E-05	V <sub>rms</sub> /Hz <sup>1/2</sup>
Linearized Response	3.6	V/mm
Resolution @ 10 Hz	7.5	Nm/Hz <sup>1/2</sup>

# Controls

## Real-Time controls

- dSPACE MicroLabBox (COTS)
- C/C++, Matlab (Simulink) & python
  - Simulink used for modeling, control and validation
- FPGA Xilinx® Kintex®-7 XC7K325T
  - CLBs slices 50950, 4Mb
  - DSP slices 840



## Analog input;

- 8x 14-bit channels, 10 Msps, differential
- 24x 16-bit channels, 1 Msps, differential
- Voltage range  $\pm 10$  V

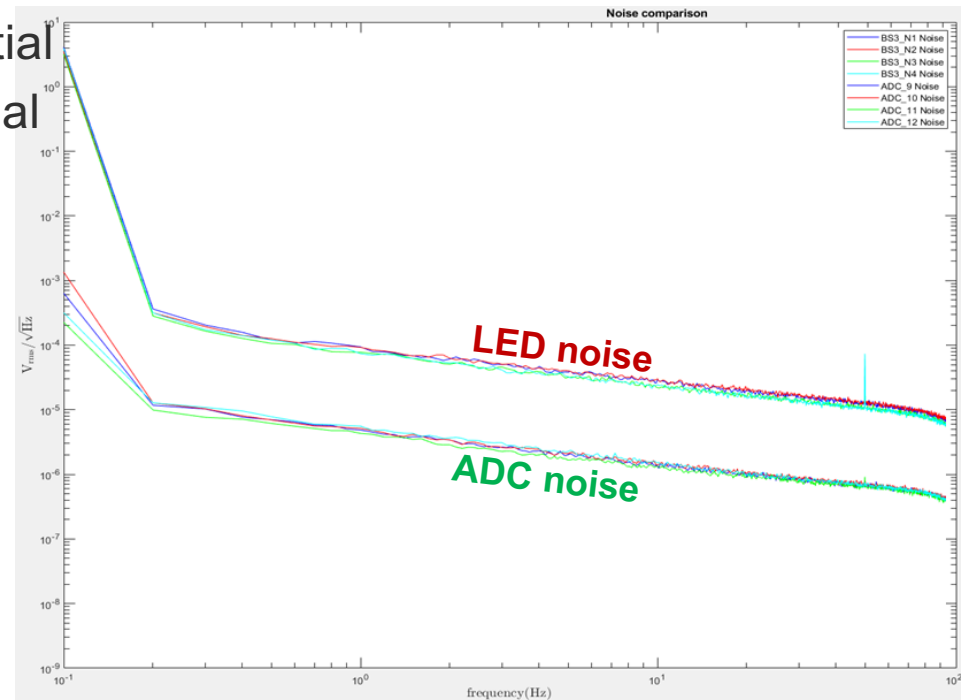
## Analog output;

- 16x 16-bit channels, 1 Msps
- Voltage range  $\pm 10$  V

## Digital I/O

- 48 bidirectional channels

1kHz bandwidth bin 0.01Hz





# Laser frequency stabilization

*FP cavity (at resonance) used to stabilize the laser frequency by Pound-Drever-Hall locking*

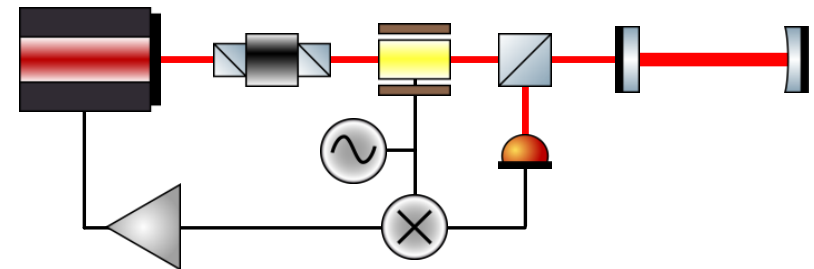
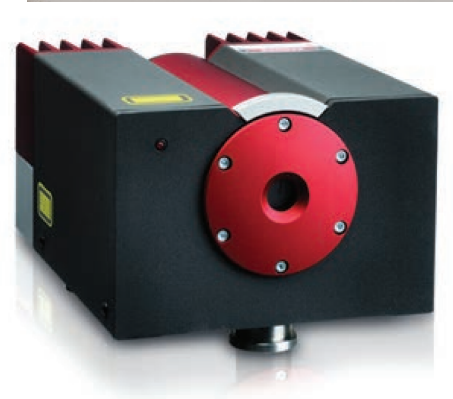
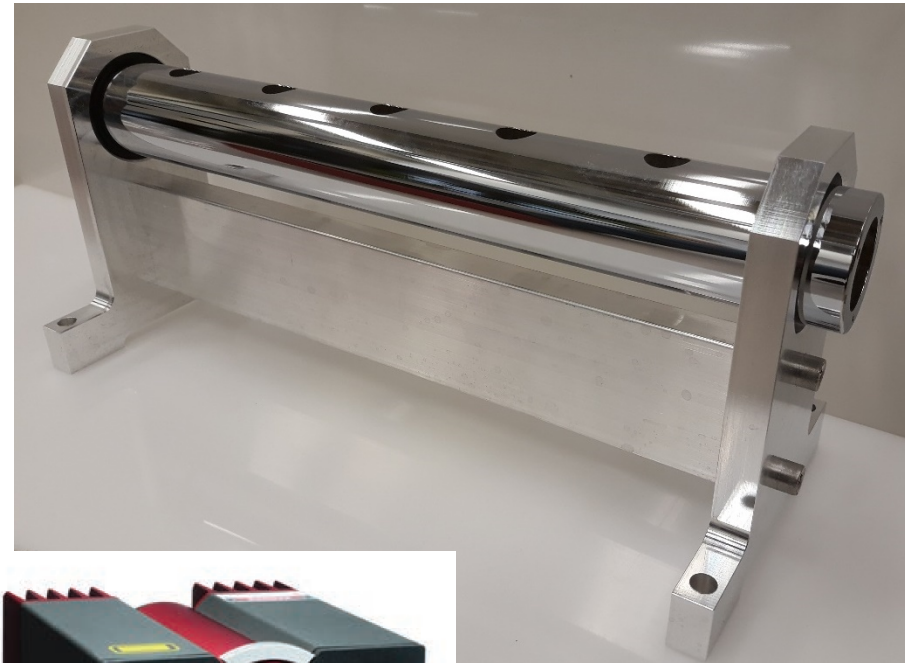
## Laser

- **Mephisto 1064 nm**
- **Tunable frequency by thermal tuning of the ND:YAG crystal (slow control) and integrated PZT (fast controls)**

## Reference cavity

- **Rigid Fabry-Perot cavity**
- **Plano-concave mirrors**
- **PDH Lock**
- **Invar CTE  $1.7 \times 10^{-6} \text{ K}^{-1}$**
- **UGF 100kHz**

$$\frac{\Delta \nu}{\nu} = \frac{\Delta \lambda}{\lambda} = \frac{\Delta L}{L} = \frac{510 \text{ nm/K}}{0.3 \text{ m}} = 1.7 \text{ ppm/K}$$





DS6201 Digital I/O Board



# AI Filter\_1MHz Notch\_v2

Goal; "Optimize LIGO Filter for DAC running at 1Msp/sec and a control loop UGF of 50 kHz"

Requirements;

- Cut of frequency  $\rightarrow$  300 kHz  $\rightarrow$  times 6
- Notch filter 1Mhz

Performance;

- 10kHz Magnitude -6dB phase -3.6deg
- 20kHz Magnitude -6dB phase -7.1deg
- 50kHz Magnitude -7dB phase -17.7deg
- 1MHz Notch -65dB

Remarks;

- Resistors replaced by 23.4k ohm (1206)

