Contents

About

What is IOpy?

IOpy is a python based package for solving the equations of motion of coupled oscillators which are in contact with thermal baths. These equations which are in the form of Langevin equations (or quantum Langevin equations in quantum limit) apear in many research areas in physics. Specifically, in optomechanics, which is the study of interaction of light with mechanical oscillators, this problem forms the essence of the theory. Finding the soultions of Langevin equations in different optomechanical system can result into really novel phenomena. However, the procedure to find these solutions is almost the same for all of them (input-output formalism, which the name IOpy is also coming from it) and the essential

difference between them is the difference between the physical setups. Moreover, in complex setups the calculations for finding the solutions can be really hard and tedious to do by hand (for example inverting matriced with large dimensions).

On the other hand, in many problems in optomechanics there are a lot of elements which are involved in the dynamics. For people who are not porfessional in the field, like students who want to study optomechanics, as a young field of research it can be confusing for them to distinguish different effects of different elements involved in the dynamics. Looking for a solution to resolve the two mentioned issues was the motivation for writing this code.

With IOpy, you can define your physical setup very fast and without the need for going through the details. Further more you can visualise the results in way can help people to test their theoretical results and also help begginers to grasp the elements of the optomechanics. For example to see the spectrum of hot optical resonator you can define your optical mode in a single line:

```
a = Mode(name = 'a', omega = 5e9 *2*np.pi)
```

And then defining the thermal bath and the driving field each in a single line:

```
a_inex = Input(name = 'ex', a, kappa = 0.2e6 *2*np.pi, kind = 'drive', omega_drive = 5e9 *2*
a_in0 = Input('0', a, kappa = 0.3e6 *2*np.pi, kind = 'bath', bath_temp = 10e-3)
```

And finally defining the system, output port and the spectrum:

```
sys_cav = System([a], [a_in0,  a_inex], [])
a_outex = Output(sys_cav, a_inex)
spec = me.spectrum(omegas, me.PowerMeasurement(a_outex), components = False, plot = True)
```

And the result would be:

A more detailed explenation of this example as well as more examples for optomechanics are availabe on the Examples page.

Structure of IOpy

IOpy is made of four scripts which each of them has a special purpose:

`elements`: For defining different components of the physical system (modes, couplings, input-output field etc.)

`DCnonlinearities`: For calculation of DC shifts in the system variables due to nonlinear effects.

`measurement`: For calculations on the output fields like linear response and spectrum.

`plots`: For visualising the measurements results on graphs.

Installation

<!— ### Comments of Nick In general I would try to make a story around these two usecases. IOpy aims for: * testing and visulizing of theorectical models * being an educational tool to learn about the classical effects in optomechanics As an eyecatcher, it maybe makes sense to show a very short example of IOpy on the about page. (You can use the simple cavity example. But don't add a lot of explanation. This you will do in the examples section) Usually people (including me =)) that want to use the software for the first time, just look at the first page to start. Also try to make a short comment about the structure of IOpy. What are the important files? And link to the section that gives a more detailed description. At the end you should have some links to installation and further examples. Installation: git pull and also name all the packages that have to installed to use iopy: numpy, scipy, matplotlib (These are all the classics)

Theory

Input-output formalism

In this section, the theory behind the IOpy calculations is introduced. There are essentially two main approaches through modeling of open quantum systems. First is density matrix approach and using master equations and second is using the Langevin equations, which is also known as input-output theory. Input-output theory allows us to directly model the quatum fluctuations injected from any couplings port into the system. Quantum Langeving equations are formulated on the level of Heisenberg equations of motion describing the time evolution of any operator of the system. Moreover, in case of optical cavities any coherent laser drive that may be present can be taken into account. For more details see Gardiner and Zoller (2004) and Clerk et al. (2010).

For example for an optical cavity which is driven by a coherent laser field with the detuning Δ from the cavity resonance frequency, via a coupling of κ_{ex} and is also coupled to the enviroment through the dissipation rate of κ_0 the equations of motion can be written as:

$$\dot{a} = -\frac{\kappa}{2}a + i\Delta a + \sqrt{\kappa_{ex}}a_{in} + \sqrt{\kappa_0}f_{in}$$

Where $\kappa = \kappa_0 + \kappa_{ex}$ is the total dissipation rate. a_{in} and f_{in} are field operators of the driving field and the stochastic thermal field respectively. This equation can also be written in terms of quadrature operators:

$$\dot{X} = \Delta Y - \frac{\kappa}{2}X + \sqrt{\kappa_{ex}}X_{in,ex} + \sqrt{\kappa_0}X_{in,0}$$

$$\dot{Y} = -\Delta X - \frac{\kappa}{2}Y + \sqrt{\kappa_{ex}}Y_{in,ex} + \sqrt{\kappa_0}Y_{in,0}$$

According to the input-output theory of open quantum systems the output field (which in case of optical cavity can be the reflected or transmitted light through the cavity) is given by:

$$a_{out} = a_{in} - \sqrt{\kappa_{ex}}a$$

Or in terms of quadratures:

$$X_{out,ex} = X_{in,ex} - \sqrt{\kappa_{ex}}X$$
$$Y_{out,ex} = Y_{in,ex} - \sqrt{\kappa_{ex}}Y$$

In general, any set of Langevin equations can in principle be linearized around the stationary solutions and can be written in a simillar way:

$$\dot{Z} = \mathbf{M}Z + \mathbf{L}Z_{in} \tag{1}$$

With the input-output relations:

$$Z_{out} = Z_{in} - \mathbf{L}^T Z \tag{2}$$

Where Z is a vertical vector containing the quadreture operators of different oscillatros involved in the dynamics and Z_{in} and Z_{out} are vectors containing the qudreture operatos of the input and output fields. For the optical cavity of the example above we have:

$$Z = \begin{pmatrix} X \\ Y \end{pmatrix}, \quad Z_{in} = \begin{pmatrix} X_{in,ex} \\ Y_{in,ex} \\ X_{in,0} \\ Y_{in,0} \end{pmatrix}, \quad Z_{out} = \begin{pmatrix} X_{out,ex} \\ Y_{out,ex} \\ X_{out,0} \\ Y_{out,0} \end{pmatrix}$$

And:

$$\mathbf{M} = \begin{pmatrix} -\frac{\kappa}{2} & \Delta \\ -\Delta & -\frac{\kappa}{2} \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} \sqrt{\kappa_{ex}} & 0 & \sqrt{\kappa_0} & 0 \\ 0 & \sqrt{\kappa_{ex}} & 0 & \sqrt{\kappa_0} \end{pmatrix}$$

Due to the linearity of the equations one can take the Fourier transform of the equations and define a scattering matrix which relates the output fields to the input fields:

$$Z_{out}(\omega) = \mathbf{S}(\omega)Z_{in}(\omega) \tag{3}$$

Where:

$$\mathbf{S} = \mathbb{1} + \mathbf{L}^T(i\omega + \mathbf{M})^{-1}\mathbf{L} \tag{4}$$

Measurements

In IOpy calculations there are two ways to measure the output fields, which are essentially same as what we do in practice. First is the linear response of the system, which is simillar to output of a VNA. And second is the spectrum of the output field, which is the same as the ouput of the spectrum analyser. ### Linear response According to the Equation (3), a quadrature pair of any output port i can be related to the quadrature pair of any input port j using a submatrix of the scattering matrix:

$$\begin{pmatrix} X_{out,i} \\ Y_{out,i} \end{pmatrix} = \mathbf{S}^{(ij)} \begin{pmatrix} X_{in,j} \\ Y_{in,j} \end{pmatrix} \tag{5}$$

Then the output field operator can be written as:

$$a_{out,i} = \frac{X_{out,i} + iY_{out,i}}{\sqrt{2}} = (\mathbf{S}_{11}^{(ij)} + i\mathbf{S}_{21}^{(ij)})\frac{X_{in,j}}{\sqrt{2}} + (\mathbf{S}_{22}^{(ij)} - i\mathbf{S}_{12}^{(ij)})\frac{iY_{in,j}}{\sqrt{2}}$$

In many cases the input field is a coherent drive with constant amplitude. Therefore, $X_{in,j}$ and In many cases the input field is a coherent drive with constant amplitude. Therefore, $X_{in,j}$ and

$X_{in,j}$ and $Y_{in,j}$ are cosine part and sine part of the input field. In many cases the input field is a coherent drive with constant amplitude. Therefore, choosing any arbitrary values for the quadratures, as long as they satisfy the identity $X_{in,j}^2 + Y_{in,j}^2 = 1$, will only impose a total phase shift to the response. By choosing $Y_{in,j}$ to be zero, we consider this phase shift to be zero and $a_{in,j} = X_{in,j}/\sqrt{2}$. as a result, any phase response in the output is being calculated with respect to the input field.

$$a_{out,i} = (\mathbf{S}_{11}^{(ij)} + i\mathbf{S}_{21}^{(ij)})a_{in,j} \tag{6}$$

or in a more familiar notation:

$$\chi^{(ij)}(\omega) = \mathbf{S}_{11}^{(ij)} + i\mathbf{S}_{21}^{(ij)} \tag{7}$$

Spectra In general for analysing the specrum of some measured signals, we can define a correlator using a measurement matrix Q_{ij}:

$$Q(\tau) = \langle Q_{ij} Z_{out,i}(0) Z_{out,j}(\tau) \rangle \tag{8}$$

The Fourier transform of this correlator would be the spectral density or spectrum.

$$S_{QQ}[\omega] = \int_{-\infty}^{\infty} Q(\tau) e^{i\omega\tau} d\tau \tag{9}$$

It can be easily shown with a simple calculation the we can rewrite the spectrum in the following way:

$$S_{QQ}[\omega] = \frac{1}{2\pi} Q_{ij} \int_{-\infty}^{\infty} \langle Z_{out,i}(\omega_1) Z_{out,j}(\omega) \rangle d\omega_1$$

According to Equation (3), we can write the output signals in terms of input signals:

$$Z_{out,i}(\omega) = S_{ik}(\omega) Z_{in,k}(\omega)$$

$$S_{QQ}[\omega] = \frac{1}{2\pi} Q_{ij} \int_{-\infty}^{\infty} S_{ik}(\omega_1) S_{jl}(\omega) \langle Z_{in,k}(\omega_1) Z_{in,l}(\omega) \rangle d\omega_1 \tag{10}$$

In general, the input signals can be correlated to each other and have complecated statistical behaviours, but based on what realy happens in experiments it's a good approximation to (1) consider different sources to be uncorrelated and (2) consider each input source a white noise source. More precisely:

$$\langle Z_{in,k}(\omega_1) Z_{in,l}(\omega) \rangle = 2\pi \delta_{kl} \mathcal{S}_k^{in}(\omega) \delta(\omega_1 + \omega) \tag{11}$$

Where $\mathcal{S}_k^{in}(\omega)$ is the spectral density of the kth input signal. Now we can simplify Equation (10) :

$$S_{QQ}[\omega] = Q_{ij} S_{ik}(-\omega) S_{jk}(\omega) \mathcal{S}_k^{in}(\omega) \tag{12}$$

This is the final result that we use in our calculations. Here we make a remark on the summations by rearanging the terms in the following way:

$$S_{QQ}[\omega] = \sum_k \mathcal{S}_k^{in}(\omega) (\sum_{i,j} Q_{ij} S_{ik}(-\omega) S_{jk}(\omega)) = \sum_k c_k(\omega) \mathcal{S}_k^{in}(\omega)$$

In this way, the measured spectrum can be seen as summation of different contributions from different noise sources.

Library

Library

In this section different classes, functions and other commands of IOpy are introduced. IOpy has 4 main scripts including `elements`, `measurement`, `DCnonlinearaity` and `plots`. ### elements Objects and functions in this

script are used for defining the oscillating modes, input-output ports, couplings between different modes and finally the whole system of coupled oscillators. ##### Class `Mode` Harmonic oscillators of any kind can be defined using this class.

```
Attributes:
    name: name of the mode.
    omega: resonance frequency of the mode in rad/sec.
    kappa: mode total dissipation rate in rad/sec.
    omega_rot: frequency at which the mode frame is rotating in rad/sec.
    driven: flag indicates whether the mode is driven or not.

Properties:
    omega_d():
       returns the frequency of the field which is driving the mode in rad/sec.
```

Class `Input`

Input field coupled to a `Mode`. Inputs can be coherent drives (pumps) or thermal baths.

```
Attributes:
    name: name of the input field.
    mode: the mode which the input is coupled to.
    kind: flag which indicates the input is a pump or a thermal bath.
          'drive' for a pump and 'bath' for a thermal bath.
    kappa: coupling rate to the mode in rad/sec.
    omega_drive: frequency of the pump fields in rad/sec.
    bath_temp: tempreture of the bath or the pump in Kelvins.
    nbar: average number of thermal photons in the input field.
```

`nbar` is calculated using the formula:

$$\bar{n} = \frac{1}{e^{\frac{\hbar\omega}{kT}} - 1}$$

Methods:

```
spectrum(self, omegas):
    spectrum of the input field. Here we approximatley take the thermal spectrum to be flat
    Args:
        omegas: the frequencies vector at which we want to calculate the spectrum.

    Returns:
            spectrum of the input field in units of number of photons.
```

Class `Coupling`

Couplings between two `Modes` using the coupling vector. The coupling vector, V_g, is a 4-dimensional vector which is defined in a way that the interaction Hamiltonian for two coupled modes would be:

$$H_{int} = 4\hbar(V_{g,1}q_1q_2 + V_{g,1}q_1p_2 + V_{g,1}p_1q_2 + V_{g,1}p_1p_2)$$

For example for optomechanics the couling vector is:

$$V_g = (g, 0, 0, 0)$$

```
Attributes:
    mode1: first mode.
    mode2: second mode.
    vg: coupling vector. a 4-d real vector.
```

Methods:

```
contains_mode(self, mode):
    indicates if the mode is involved in this coupling or not.

    Args:
        mode: the mode we want to look for.

    Returns:
        'True' if the mode is involved and 'False' for other wise.
```

Class `System`

Defines the complex system made of coupled `Modes` and `Inputs`. It's most important purpose of it is for calculating the scattering matrix.

```
Attributes:
    modes: an array of modes in the system.
    inputs: an array of inputs of the system.
    couplings: couplings between the modes of the system.
    M: the M matrix in the relation dZ/dt = M*Z + L*Z_in.
    L: the L matrix in the relation dZ/dt = M*Z + L*Z_in.
```

Refer to Equation (1) for more about M and L matrices.

Methods:

```
add_mode(self, mode):
    Adding a mode to the system.

    Args:
        mode: the mode we want to add.
```

```
add_input(self, inp):
    Adding an input to the system.

    Args:
        inp: the input we want to add.

add_coupling(self, coup):
    Adding a coupling to the system.

    Args:
        coup: the coupling we want to add.

make_ML(self):
    Constructing M and L matrices of the system.

SMatrix(self, omegas):
    Constructing the scattering matrix of the system for a frequency range.

    Args:
        omegas: frequencies vector in rad/sec.

    Returns:
        Ss: the scattering matrix.
```

Refer to Equation (4) for more about scattering matrix. ##### Class `Output` The output field of the system with respect to an input field (in terms of input-output formalism).

```
Attributes:
    system: the complex system of coupled modes and inputs.
    input: the input field which we want to define its output field (in terms of input-outpu
    mode: the mode which these input and output fields are coupled.
```

measurement

Objects and functions in this script are used for measuring the output fields. Different measurement schemas can be used and linear response or spectrum of the output fields can be seen as results. ##### Class `MeasurementOperator` (it seems this one is useless!) ##### Class `PowerMeasurement` A power measurement scheme object. The correlator function and measurement matrix in this scheme are:

$$Q(\tau) = \langle q(0)q(\tau) + iq(0)p(\tau) - ip(0)q(\tau) + p(0)p(\tau) \rangle$$

$$[Q] = \begin{pmatrix} 1 & i \\ -i & 1 \end{pmatrix}$$

```
Attributes:
    system: the system which the output field is coming from.
```

```
    omega_d: the driving frequency of the mode which the output field is coming from.
    Q: the measurment matrix.
```

Class `HomodynMeasurement`

A Homodyn measurement scheme object with a theta phase. The correlator function and measurement matrix in this scheme are:

$$Q(\tau) = \langle \cos^2(\theta) q(0)q(\tau) + \sin(\theta)\cos(\theta) q(0)p(\tau) + \sin(\theta)\cos(\theta) p(0)q(\tau) + \sin^2(\theta) p(0)p(\tau) \rangle$$

$$[Q] = \begin{pmatrix} \cos^2(\theta) & \sin(\theta)\cos(\theta) \\ \sin(\theta)\cos(\theta) & \sin^2(\theta) \end{pmatrix}$$

```
Attributes:
    system: the system which the output field is coming from.
    omega_d: the driving frequency of the mode which the output field is coming from.
    Q: the measurment matrix.
```

Function `linear_response`

The linear response (susceptibility) of the system from one specific input port to an output port in frequency domain:

$$a_{out} = \chi a_{in}$$

Refer to Equation (7) for more about linear response.

```
Args:
    Omegas: the frequencies vector we want to claculate the linear response for them, in fra
            rotating frame)
    system: the system which we want to measure its response.
    output: the output port.
    Input: the input port.
    plot: flag, indicates to plot the susceptibilities or not.

Returns:
    omegas_out: the frequencies vector we want to claculate the linear response for them, in
            rotating frame)
    a: the susceptibility we want to measure.
```

Function `spectrum`

The spectrum of an output field. Refer to spectra section for more about the spectra.

```
Args:
    omegas: the frequencies vector we want to claculate the spectrum for them, in frame of t
            rotating frame)
```

```
    measurement: the measurement scheme, of kinds PowerMeasurement or HomodynMeasurement.
    components: flag, indicates to calcuate different contributions of noise sources or just
    plot: flag, indicates to plot the spectra or not.

Returns:
    spec: the spectrum of the output field.
```

DCnonlinearities

Functions in this script are used for calculating the DC shifts resulting form nonlinear effects. ##### Function `Kerr_effect_nbar` This function findes the steady state average number of photons in an optical cavity with kerr type nonlinearity. It findes the smallest real route of a third order polynomial equation:

$$(\frac{-\kappa_{ex}P_{in}}{\hbar\omega_{drive}})\bar{n}^3 + (\Delta^2 + (\frac{\kappa}{2})^2)\bar{n}^2 + (2K\Delta)\bar{n} + (K^2) = 0$$

```
Args:
    P_in: input power in Watts.
    kappa_0 = cavity intrinsic dissipation rate in rad/sec.
    kappa_ex = input coupling rate in rad/sec.
    omega_c = cavity resonance frequency in rad/sec.
    omega_drive = frequency of the input field in rad/sec.
    K = nonlinearity coefficient in rad/sec.

returns:
    smallest real route of the third order polynomial equation.
```

Function `optomechanics`

This function findes the steady state average number of photons in an optomechanical cavity and also finds the DC shift cavity resonance frequency. It uses the `Kerr_effect_nbar()` function to solve the third order equation:

$$\bar{n}(\frac{\kappa^2}{4} + (\Delta - (\frac{2g_0^2}{\Omega_m})\bar{n})^2) = \kappa_{ext}\frac{P_{in}}{\hbar\omega_{drive}}$$

```
Args:
    P_in: input power in Watts.
    kappa_0 = cavity intrinsic dissipation rate in rad/sec.
    kappa_ex = input coupling rate in rad/sec.
    omega_c = cavity resonance frequency in rad/sec.
    omega_drive = frequency of the input field in rad/sec.
    omega_m = resonance frequency of the mechanical oscillator in rad/sec.
    g_0 =  vacuum optomechanical coupling rate in rad/sec.
```

```
returns:
    omega_c = modified cavity resonance frequency in rad/sec.
    g = optomechanical coupling rate in rad/sec.
```

plots

Functions in this script are for plotting the linear responses and spectrums. ##### Function `plot_linear_response` This function is for plotting the linear response functions. It plots the absolute value and phase of the linear response as well as plotting the linear response in complex space.

```
    Args:
        omegas: the vector containing the frequecnies in rad/sec (not in a rotating frame).
        A: the linear response function.
        system: the system which the linear response is from.
        output: the output field we that the linear response is calculated for.
        input: the input field we that the linear response is calculated for.
```

Function `plot_spectrum`

This function is for plotting the spectra.

```
    Args:
        omegas: the vector containing the frequecnies in rad/sec (not in a rotating frame).
        spec: the spectrum.
        componenets: flag, indicates to plot different contributions of noise sources in the
        spectrum.
        system: the system which the spectrum is from.
```

Examples

Examples

In this section a set famous phenomena in optomechanics context are presented. These calculations are first as examples of using IOpy in simulating Langevin equations, and second are benchmarks which results of IOpy can be compared against results which are theoretically developed.

Simple Cavity

The first example is simulating a hot microwave resonator. Here the temperature of the bath is higher than the temperature of the drive and therefore we would

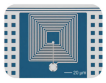

Figure 1: LC

expect to see an emmition shaped like a Lorenzian. To see the fool written example, go to Simple cavity example. first we have to define the cavity mode:

```
omega_c = 5e9*np.pi*2
a = Mode('a', omega_c)
```

Then we have to define input fields. In this example the cavity is driven by a coherent drive and is coupled to a thermal bath.

```
kappa_ex = 0.2e6*np.pi*2
kappa_0 = 0.3e6*np.pi*2
kappa = kappa_ex + kappa_0

T_drive = 2e-5
T_bath = 10e-3

a_inex = Input('ex', a, kappa_ex, kind = 'drive', omega_drive = omega_c, bath_temp=T_drive)
a_in0 = Input('0', a, kappa_0, kind = 'bath', bath_temp=T_bath)
```

And finally the system object and the output field.

```
sys_cav = System([a], [a_in0,  a_inex], [])

a_outex = Output(sys_cav, a_inex)
```

Now for measuring the spectrum of the output field, we have to use the `spectrum` function:

```
omegas = np.linspace( omega_c - 15*kappa, omega_c +  15*kappa, 1001)
spec = me.spectrum(omegas, me.PowerMeasurement(a_outex), components = False, plot = True)

    cavity output spectrum
</p>
```

As we expect, we can see a Lorentzian in the spectrum. If the tempreture of the driving field was higher than the thermal bath, we would see a deep instead of a peak.

The linear response of the system to driving can also be measured with the `linear_response` function:

```
omegas_newex, S_ex = me.linear_response(omegas, sys_cav, a_outex, a_inex, plot = 1)
```

The results can also be compared to the theory. For example in this case the linear response of the system to the drive field is:

$$S_{aa} = 1 - \frac{\kappa_{ex}}{\frac{\kappa}{2} - i(\omega - \omega_c)}$$

The graphs below show the comparison between this equation and IOpy results.

```
    dsf
</p>
```

Basic optomechanics

In this example we simulate an optomechanical system. Here we can see the mechanical sidebands. To see the fool written example, go to Basic optomechanics.

In the basic optomechanical interaction, the cavity resonance frequency shift by a constant value due to DC nonlinearity. Therefore, before difining the modes we have to calculate this DC shift.

```
omega_c = 5e9*np.pi*2     # cavity resonance frequency

kappa_0 = 0.3e6*np.pi*2
kappa_ex = 0.4e6*np.pi*2

kappa = kappa_0 + kappa_ex

omega_m = 5e6*np.pi*2    # mechanical resonance frequency
gamma_m = 100*np.pi*2

P_in = 5e-9

g_0 = 200*np.pi*2

omega_drive = omega_c - 1* omega_m


from DCnonlinearities import optomechanics

omdir = optomechanics(P_in, kappa_0, kappa_ex, omega_c, omega_drive, omega_m, g_0)

g= omdir['g']                # optomechanical coupling rate = sqrt(nbar) * g_0
omega_c = omdir['omega_c']   # new cavity resonance frequency
```

Now we can define the modes, as well as input fields including thermal baths coupled to optics and mechanics and the optical driving field.

```
a = Mode('a', omega_c)
b = Mode('b', omega_m)
```

```
a_inex = Input('ex', a, kappa_ex, kind = 'drive', omega_drive = omega_drive, bath_temp=10e-3
a_in0 = Input('0', a, kappa_0, kind = 'bath', bath_temp=10e-3)

b_in0 = Input('0', b, gamma_m, kind = 'bath', bath_temp=10e-3)
```

Then we should define the coupling between the optical and mechanical modes:

```
g_ab = Coupling(a, b, g * np.array([1,0,0,0]))
```

And finally the whole optomechanical system:

```
sys_om = System([a, b], [a_in0,b_in0 , a_inex], [g_ab])
```

Now just like the previous example we can measure the spectrum and the linear response of the system.

Strong coupling regime

Strong coupling regime

OMIT

OMIT

Frequency conversion using OMIT

Ponderomotive squeezing

Outlook

Time domain simulations

Multidrive issue

References

Clerk, A. A., F. Marquardt, and J. G. E. Harris, 2010, Phys. Rev. Lett. 104, 213603. Gardiner, C. W., and P. Zoller, 2004, Quantum Noise, Springer Series in Synergetics (Springer, Berlin/Heidelberg).