Noisy Duck Documentation

Release 0.2.1

Nathan Wukie

May 20, 2019

Contents

CHAPTER 1

Stable release

To install Noisy Duck, run this command in your terminal:

```
$ pip install noisyduck
```

This is the preferred method to install Noisy Duck, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

CHAPTER 2

From sources

The sources for Noisy Duck can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/nwukie/noisyduck
```

Or download the tarball:

```
$ curl  -OL https://github.com/nwukie/noisyduck/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use Noisy Duck in a project:

```
import noisyduck
```

3.1 Example: Annular Cylindrical Duct

Numerical eigenvalue/eigenvector decomposition:

```
import noisyduck as nd
eigenvalues, eigenvectors = nd.annulus.numerical.decomposition(omega,m,r,rho,u,v,w,p,
↪gam,filter='acoustic')
```

Analytical eigenvalue/eigenvector decomposition:

```
import noisyduck as nd
eigenvalues, eigenvectors = nd.annulus.analytical.decomposition(omega,m,mach,ri,ro,n)
```

See `cylindrical_annulus_uniform_flow.py`.

3.1.1 Cylindrical Annuluar Duct

3.1.1.1 Analytical

This module provides a functionality for computing the eigenvalue/eigenvector decomposition of a uniform axial flow in an annular cylindrical duct. The decomposition is based on an analytical solution of the convected wave equation for pressure, yielding the acoustic part of the eigen decomposition. The eigenvectors from this decomposition correspond specifically to the acoustic pressure disturbance.

Theory:

The analysis here follows that of Moinier and Giles, "Eigenmode Analysis for Turbomachinery Applications", Journal of Propulsion and Power, Vol. 21, No. 6, 2005.

For a uniform axial mean, linear pressure perturbations satisfy the convected wave equation

$$\left(\frac{\partial}{\partial t} + M\frac{\partial}{\partial z}\right)^2 p = \nabla^2 p \qquad \lambda < r < 1$$

where M is the Mach number of the axial mean flow and r has been normalized by the outer duct radius. Boundary conditions for a hard-walled duct imply zero radial velocity, which is imposed using the condition

$$\frac{\partial p}{\partial r} = 0 \qquad \text{at} \qquad r = \lambda, 1$$

Conducting a normal mode analysis of the governing equation involves assuming a solution with a form

$$p(r, \theta, z, t) = e^{j(\omega t + m\theta + kz)} P(r)$$

This leads to a Bessel equation

$$\frac{1}{r}\frac{d}{dr}\left(r\frac{dP}{dr}\right) + \left(\mu^2 - \frac{m^2}{r^2}\right) P = 0 \qquad \lambda < r < 1$$

where $\mu^2 = (\omega + Mk)^2 - k^2$. The solution to the Bessel equation is

$$P(r) = aJ_m(\mu r) + bY_m(\mu r)$$

where J_m and Y_m are Bessel functions. Applying boundary conditions to the general solution gives a set of two equations

$$\begin{bmatrix} J'_m(\mu\lambda) & Y'_m(\mu\lambda) \\ J'_m(\mu) & Y'_m(\mu) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = 0$$

which has nontrivial solutions as long as the determinant is zero. Solving for the zeros of the determinant give μ, at which point the quadratic equation above can be solved for the axial wavenumbers k.

Example:

```
eigenvalues, eigenvectors =
noisyduck.annulus.analytical.decomposition(omega,m,mach,r,n)
```

noisyduck.annulus.analytical.**compute_eigenvalues**(*omega*, *mach*, *zeros*)

This procedure compute the analytical eigenvalues for the convected wave equation. A uniform set of nodes is created to search for sign changes in the value for the eigensystem. When a sign change is detected, it is known that a zero is close. The eigensystem is then passed to a bisection routine to find the location of the zero. The location corresponds to the eigenvalue for the system.

Parameters

- **omega** (*float*) – temporal wavenumber.
- **m** (*int*) – circumferential wavenumber.
- **mach** (*float*) – Mach number.
- **ri** (*float*) – inner radius of a circular annulus.

- **ro** (*float*) – outer radius of a circular annulus.
- **n** (*int*) – number of eigenvalues to compute.

Returns An array of the first 'n' eigenvalues for the system.

noisyduck.annulus.analytical.**compute_eigenvector**(*r*, *sigma*, *m*, *zero*)
Return the eigenvector for the system.

Parameters

- **r** (*np.array(float)*) – array of radial locations.
- **sigma** (*float*) – ratio of inner to outer radius, ri/ro.
- **m** (*int*) – circumferential wavenumber.
- **zero** (*float*) – a zero of the determinant of the convected wave equation

Returns the eigenvector associated with the input 'm' and 'zero', evaluated at radial locations defined by the input array 'r'. Length of the return array is len(r).

noisyduck.annulus.analytical.**compute_zeros**(*m*, *mach*, *ri*, *ro*, *n*)
This procedure compute the zeros of the determinant for the convected wave equation.

A uniform set of nodes is created to search for sign changes in the value of the function. When a sign change is detected, it is known that a zero is close. The function is then passed to a bisection routine to find the location of the zero.

Parameters

- **m** (*int*) – circumferential wavenumber.
- **mach** (*float*) – Mach number
- **ri** (*float*) – inner radius of a circular annulus.
- **ro** (*float*) – outer radius of a circular annulus.
- **n** (*int*) – number of eigenvalues to compute.

Returns An array of the first 'n' eigenvalues for the system.

noisyduck.annulus.analytical.**decomposition**(*omega*, *m*, *mach*, *r*, *n*)
This procedure computes the analytical eigen-decomposition of the convected wave equation. The eigenvectors returned correspond specifically with acoustic pressure perturbations.

Inner and outer radii are computed using min(r) and max(r), so it is important that these end-points are included in the incoming array of radial locations.

Parameters

- **omega** (*float*) – temporal wave number.
- **m** (*int*) – circumferential wave number.
- **mach** (*float*) – Mach number.
- **r** (*float*) – array of radius stations, including rmin and rmax.
- **n** (*int*) – number of eigenvalues/eigenvectors to compute.

Returns a tuple containing an array of eigenvalues, and an array of eigenvectors evaluated at radial locations.

Return type (eigenvalues, eigenvectors)

noisyduck.annulus.analytical.**eigensystem**(*b*, *m*, *ri*, *ro*)

Computes the function associated with the eigensystem of the convected wave equation. The location of the zeros for this function correspond to the eigenvalues for the convected wave equation.

The solution to the Bessel equation with boundary conditions applied yields a system of two linear equations. .. math:

```
A x =
\begin{bmatrix}
    J'_m(\mu \lambda)    &    Y'_m(\mu \lambda) \\
    J'_m(\mu)            &    Y'_m(\mu)
\end{bmatrix}
\begin{bmatrix}
    x_1 \\ x_2
\end{bmatrix}
= 0
```

This procedure evaluates the function .. math:

```
det(A) = f(b) = J_m(b*ri)*Y_m(b*ro)  -  J_m(b*ro)*Y_m(b*ri)
```

So, this procedure can be passed to another routine such as numpy to find zeros.

Parameters

- **b** (*float*) – coordinate.
- **m** (*int*) – circumferential wave number.
- **ri** (*float*) – inner radius of a circular annulus.
- **ro** (*float*) – outer radius of a circular annulus.

3.1.1.2 Numerical

This module provides a functionality for computing the numerical eigenvalue/eigenvector decomposition of a uniform axial flow in an annular cylindrical duct. The decomposition is based on a normal mode analysis of the three-dimensional linearized Euler equations, which yields an eigensystem that is discretized and solved numerically.

Theory:

Filtering:

Example:

```
eigenvalues, eigenvectors_l, eigenvectors_r =
    noisyduck.annulus.numerical.decomposition(omega,
                                              m,
                                              r,
                                              rho,
                                              vr,
                                              vt,
                                              vz,
                                              p,
                                              gam,
                                              filter='acoustic')
```

noisyduck.annulus.numerical.**construct_numerical_eigensystem_general**(*omega*, *m*, *r*, *rho*, *vr*, *vt*, *vz*, *p*, *gam*, *perturb_omega=True*)

Constructs the numerical representation of the eigenvalue problem associated with the three-dimensional linearized euler equations subjected to a normal mode analysis.

NOTE: If perturb_omega=True, a small imaginary part is added to the temporal frequency to facilitate determining the propagation direction of eigenmodes based on the sign of the imaginary part of their eigenvalue. That is: $\omega = \omega - 10^{-5}\omega j$. See Moinier and Giles[2].

[1] Kousen, K. A., "Eigenmodes of Ducted Flows With Radially-Dependent Axial and Swirl Velocity Components", NASA/CR 1999-208881, March 1999.

[2] Moinier, P., and Giles, M. B., "Eigenmode Analysis for Turbomachinery Applications", Journal of Propulsion and Power, Vol. 21, No. 6, November-December 2005.

Parameters

- **omega** (*float*) – temporal frequency.
- **m** (*int*) – circumferential wavenumber.
- **r** (*float*) – array of equally-spaced radius locations for the discretization, including end points.
- **rho** (*float*) – mean density.
- **vr** (*float*) – mean radial velocity.
- **vt** (*float*) – mean tangential velocity.
- **vz** (*float*) – mean axial velocity.
- **p** (*float*) – mean pressure.
- **gam** (*float*) – ratio of specific heats.
- **perturb_omega** (*bool*) – If true, small imaginary part is added to the temporal frequency. Can help in determining direction of propagation.

Returns

left-hand side of generalized eigenvalue problem, right-hand side of generalized eigenvalue problem.

Return type (M, N)

noisyduck.annulus.numerical.**decomposition**(*omega*, *m*, *r*, *rho*, *vr*, *vt*, *vz*, *p*, *gam*, *filter='None'*, *alpha=1e-05*, *equation='general'*, *perturb_omega=True*)

Compute the numerical eigen-decomposition of the three-dimensional linearized Euler equations for a cylindrical annulus.

Parameters

- **omega** (*float*) – temporal frequency.
- **m** (*int*) – circumferential wavenumber.

- **r** (*float*) – array of equally-spaced radius locations for the discretization, including end points.
- **rho** (*float*) – mean density.
- **vr** (*float*) – mean radial velocity.
- **vt** (*float*) – mean tangential velocity.
- **vz** (*float*) – mean axial velocity.
- **p** (*float*) – mean pressure.
- **gam** (*float*) – ratio of specific heats.
- **filter** (*string, optional*) – Optional filter for eigenmodes. values = ['None', 'acoustic']
- **alpha** (*float, optional*) – Criteria governing filtering acoustic modes.
- **equation** (*string, optional*) – Select from of governing equation for the decomposition. values = ['general', 'radial equilibrium']
- **perturb_omega** (*bool*) – If true, small imaginary part is added to the temporal frequency. Can help in determining direction of propagation.

Returns a tuple containing an array of eigenvalues, an array of left eigenvectors evaluated at radial locations, an array of right eigenvectors evaluated at radial locations.

Return type (eigenvalues, left_eigenvectors, right_eigenvectors)

Note: The eigenvectors being returned include each field $[\rho, v_r, v_t, v_z, p]$. The primitive variables can be extracted into their own eigenvectors by copying out those entries from the returned eigenvectors as:

```
res = len(r)
rho_eigenvectors = eigenvectors[0*res:1*res,:]
vr_eigenvectors  = eigenvectors[1*res:2*res,:]
vt_eigenvectors  = eigenvectors[2*res:3*res,:]
vz_eigenvectors  = eigenvectors[3*res:4*res,:]
p_eigenvectors   = eigenvectors[4*res:5*res,:]
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/nwukie/noisyduck/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

Noisy Duck could always use more documentation, whether as part of the official Noisy Duck docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/nwukie/noisyduck/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *noisyduck* for local development.

1. Fork the *noisyduck* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/noisyduck.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenv installed, create a new virtual environment (here we assume your virtual environments are located in ~/.virtualenv. You can create this directory if it doesn't already exist.) and activate it. This is how to set up a virtual environment, install noisyduck and start developing:

```
$ virtualenv ~/.virtualenv/noisyduck
$ source ~/.virtualenv/noisyduck/bin/activate
$ cd noisyduck/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 noisyduck tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.5, 3.6, and 3.7. Check https://travis-ci.org/nwukie/noisyduck/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To manuall run tests, you can execute py.test in the noisyduck project directory:

```
$ py.test
```

CHAPTER 5

Credits

5.1 Development Lead

- Nathan Wukie <nathan.wukie@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2018-02-02)

- First release on PyPI.

6.2 0.1.5 (2018-03-12)

- New governing equations
- Tests for uniform axial mean flow
- New API + documentation

Python Module Index

n

Index

C

D

E

N