Quasinormal Modes and Electromagnetic Eigenvalue Problems – Single Dipole

$$\mathbf{p}(\omega) = \overleftrightarrow{\alpha}(\omega) \cdot \mathbf{E} \qquad\qquad \overleftrightarrow{\alpha}^{-1}(\omega) \cdot \mathbf{p}(\omega) = \mathbf{E}$$

The <u>self-sustaining</u> solutions are the solution of the <u>homogeneous</u> problem:

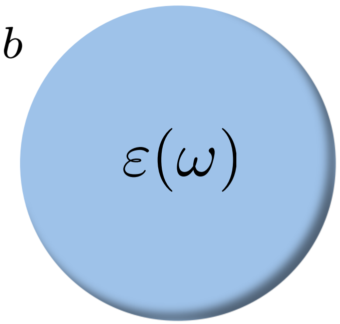
$$\overleftrightarrow{\alpha}^{-1}(\omega) \cdot \mathbf{p}(\omega) = \mathbf{0}$$

$$\det\left[\overleftrightarrow{\alpha}^{-1}(\omega)\right] = 0$$

Eigen frequencies are determined by the complex poles of the NP polarizability

The Electromagnetic Eigenvalue Problem – Single Dipole

ε_b



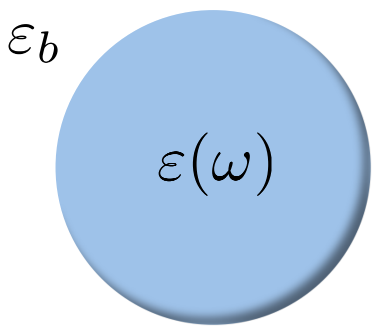
$$\varepsilon(\omega) = \varepsilon_\infty - \frac{\omega_{\mathrm{p}}^2}{\omega^2 + i\gamma\omega}$$

$$\alpha(\omega) = \xi \frac{\varepsilon(\omega) - \varepsilon_b}{\varepsilon(\omega) + 2\varepsilon_b} \quad = \xi \left(1 - \frac{3\varepsilon_b}{\varepsilon(\omega) + 2\varepsilon_b} \right)$$

Poles are the roots of the denominator:

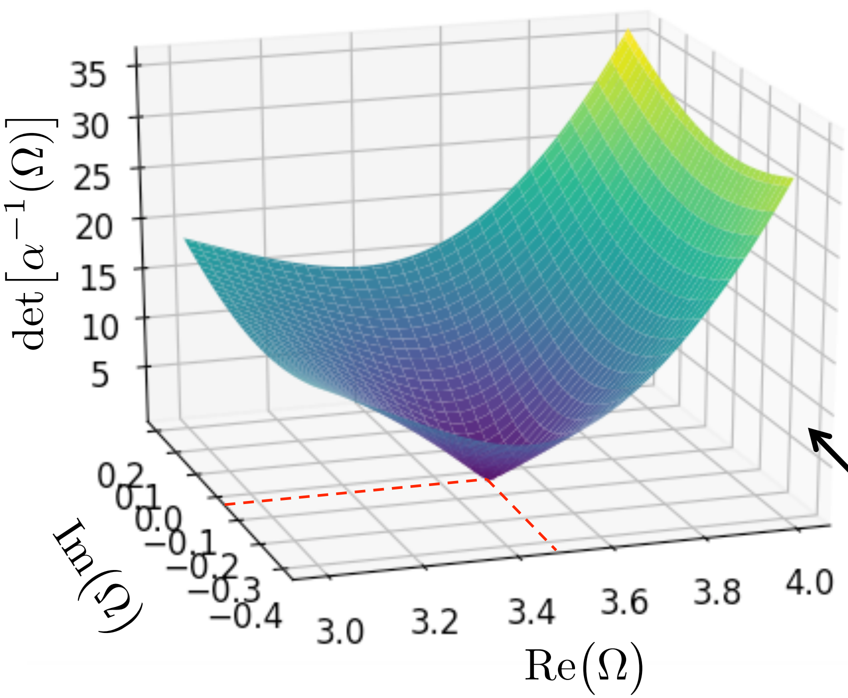
$$\varepsilon_\infty - \frac{\omega_{\mathrm{p}}^2}{\omega^2 + i\gamma\omega} + 2\varepsilon_b = 0 \qquad \Omega = \sqrt{\frac{\omega_{\mathrm{p}}^2}{\varepsilon_\infty + 2\varepsilon_b} - \left(\frac{\gamma}{2}\right)^2} - i\frac{\gamma}{2}$$

The Electromagnetic Eigenvalue Problem – Single Dipole

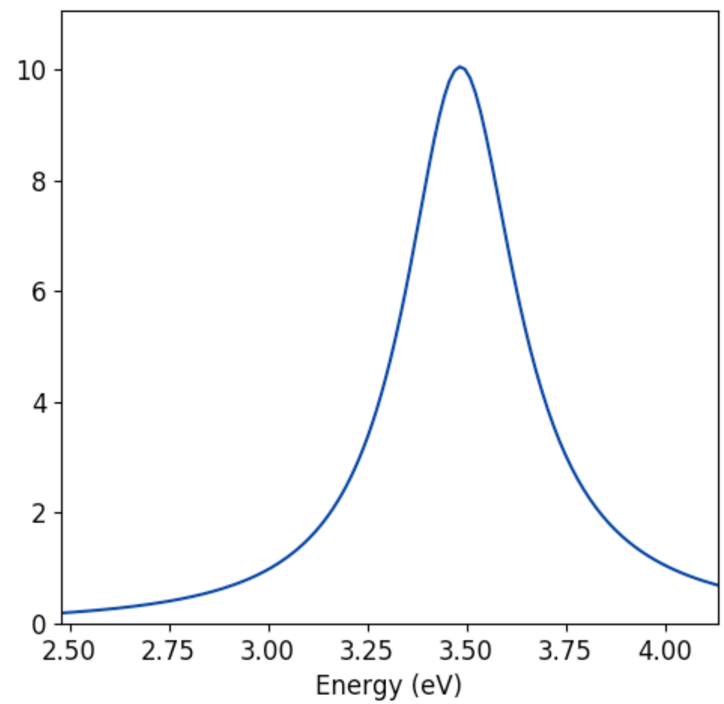


$$\varepsilon(\omega) = \varepsilon_\infty - \frac{\omega_{\mathrm{p}}^2}{\omega^2 + i\gamma\omega}$$

$$\Omega_m = \omega_m - i\frac{\gamma_m}{2}$$

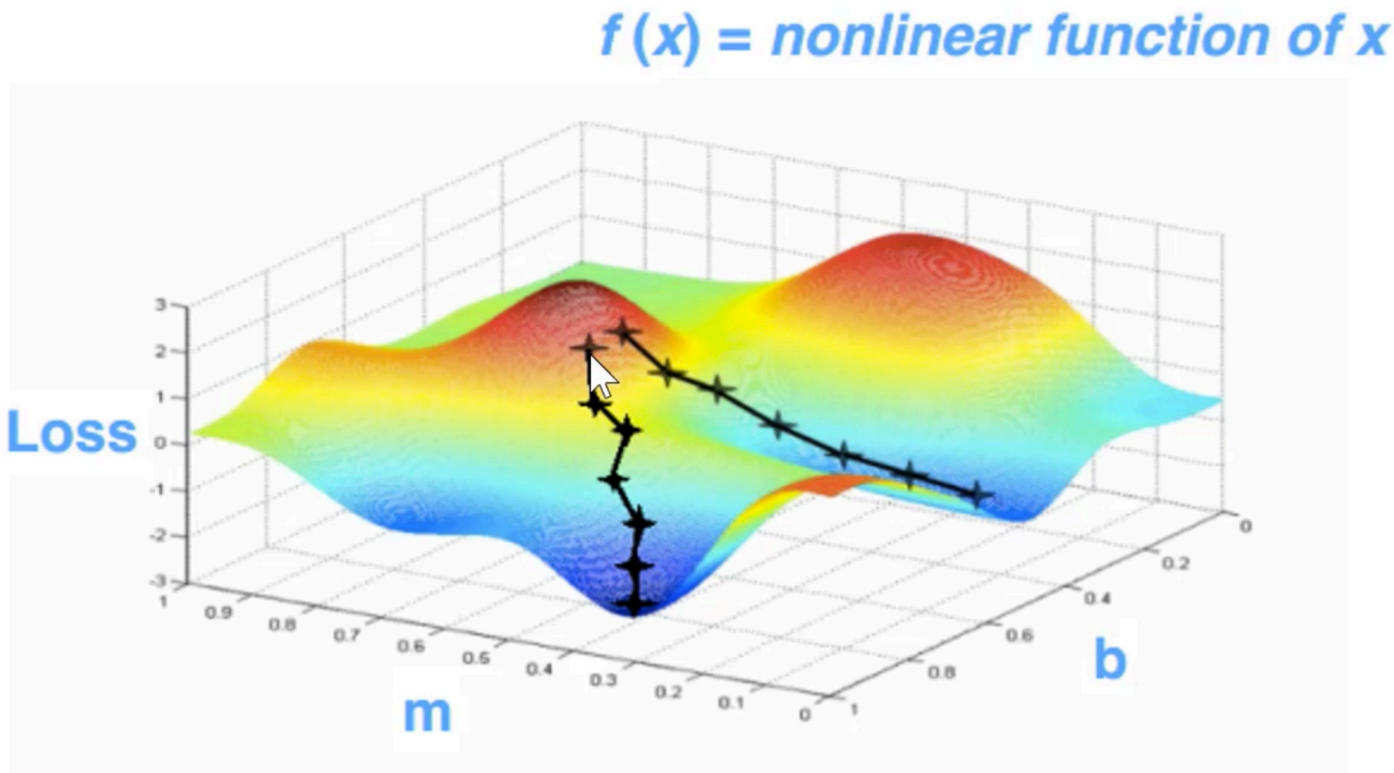


R = 25 nm; n = 1.0



SLRpy_1d/singleNP.py (run_type = “grid_search”)

Minimization Problem = Gradient Descent?



In our case, *f (x)* is:

- Nonlinear (sensitive to initial guess)
- In General it is challenging/impossible to compute gradients of *f*(*x*)

IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION, VOL. 66, NO. 12, DECEMBER 2018

Global Complex Roots and Poles Finding Algorithm Based on Phase Analysis for Propagation and Radiation Problems

Piotr Kowalczyk

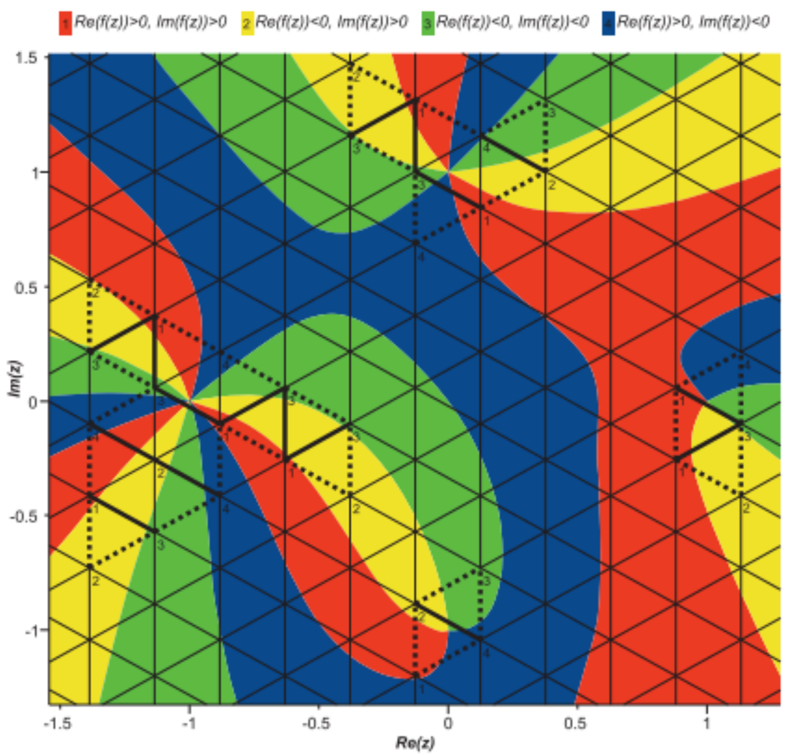


Fig. 1. Preliminary estimation algorithm applied for function $f(z) = (z-1)(z-i)^2(z+1)^3/(z+i)$. The numbers (colors): 1 (red), 2 (yellow), 3 (green), and 4 (blue) represent the quadrants in which the function values lie. Thick black lines: candidate edges. Black dotted lines: boundaries of the candidate regions.

Mesh Refinement Algorithm

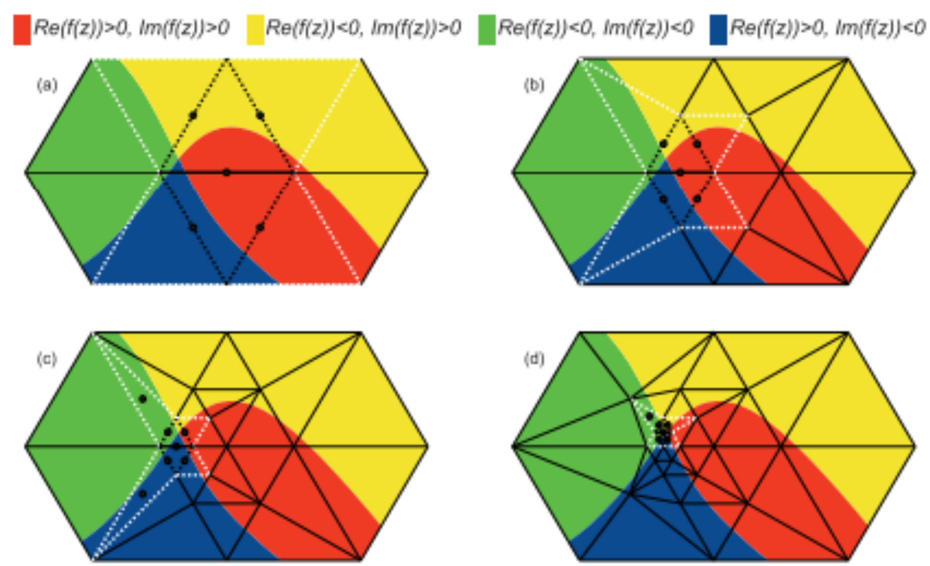


Fig. 2. Simple example of the mesh refinement process. (a)–(d) Four consecutive iterations. Thick black lines: candidate edges. Black dotted lines: boundary of the candidate regions. White dotted line: boundary of the extra zone.

DOI: 10.1109/TAP.2018.2869213

https://github.com/PioKow/GRPF

Modified GRPF – GRPF_python Overview

main.py

- set run parameters
- start Matlab engine
- for each k vector in sweep:
  - generate Matlab input files
  - run GRPF to find roots and poles of function f defined in pyfunc.py
  - write identified root and pole info to respective files
- stop Matlab engine
- exit

GRPF_python.m

- slightly modified version of GRPF code
- modified to define a Matlab function that could be run by the Matlab engine in main.py

pyfunc.py

- all definitions of python functions related to calculating things related to NP and QE properties/ coupling
- function with name "f" is what GRPF tries to locate the roots and poles of

see SLRpy_1d/GRPF_python/NOTES.txt for additional information

*Seems to ONLY Run with python 2.7

Some Important Notes:

```
% main program GRPF
%
close all;
clear;
clc;
format long;

%
% Set up ability to work with Python
%
clear classes;
if count(py.sys.path, '') == 0
    %insert(py.sys.path,int32(0),'/home/mrb179/Projects/EigenvalueProblem/GRPF/PYTHON_GRPF/');
    insert(py.sys.path,int32(0),'/home/mrb179/Projects/SLRpy_1D/GRPF_python/');
end
modu = py.importlib.import_module('pyfunc');
py.reload(modu);
```

Need to specify proper path!!

```
%% general loop
it=0;
while it<ItMax&&NrOfNodes<NodesMax
    it=it+1;

    NodesCoord=[NodesCoord ; NewNodesCoord];

    disp(['Evaluation of the function in ',num2str(size(NewNodesCoord,1)),' new points...'])

    for Node=NrOfNodes+1:NrOfNodes+size(NewNodesCoord,1)
        z=NodesCoord(Node,1)+1i*NodesCoord(Node,2);
        disp("Made it this far");
        FuntionValues(Node,1)= py.pyfunc.f(real(z), imag(z), RNP, Nind); % <------- If you change definition of f
in pyfunc.py, need to change argument list here!!

```

Changes in argument of pyfunc.f() must be reflected in GRPF_python.m

- Advantages of GRPF – locates all poles within a region
- It is not clear to me that GRPF is always better than simplex optimization (see singleNP.py)