Inverse design of 2D wave devices with optimization

Egor E. Nuzhin[1]†, I. B. Minin[1, 2] and A. I. Boyko[1]

[1]Computational and Data-Intensive Science and Engineering, Skolkovo Institute of Science and Technology, Skolkovo Innovation Center, Building 3, Russian Federation

[2]Department of Physical and Quantum Electronics, Moscow Institute of Physics and Technology, 9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russian Federation

(Received xx; revised xx; accepted xx)

The optimization techniques used in coupled with ab-initio solver for wave physics to obtain optimal shape of a device. The simplified version of the problem, showing applicability of optimization methods to inverse design of some optical devices has been considered.

Key words: Topology optimization, solver, generalized minimal residual method, numerical linear algebra, gradient, Jacobi matrix, complex functions.

---

1. Team

• **Egor**: He has implemented two topology optimization solvers based on the non-gradient (broadcasting on the edge of the homogeneous substance) and the gradient implementations. Both solvers use Greedy algorithm. He has tried to implement this problem as locally convex and to resolve this problem using CVXPY but it hasn't been successful. He has implemented Jacobi matrix for computing the mask gradient of maximum of field vector. He has produced and programmed (in Python) easily computed formula for calculating **row** of Jacobi matrix using 2D FFT-matvec. He has implement topology optimization problem using built-in *scipy*−optimizer.

• **Iurii**: He has designed main idea of stable version of topology optimization solver and described it theoretically. He has produced formula for the easily-computable diagonal of Jacobi matrix and programmed it in Python. He has implemented solver of linear equations systems based on GMRES-solver, organized this function with features that have user-friendly interface with flags of outputing data, for example, on convergence of solution, residuals, etc. He has tested program and organized output results using .png format, has collected output data, created this report.

• **Alexey**: He has provided group with formulas of field vector distribution in 2D space, using Helmholtz equation, Hankel function, Green function, debugged some errors that appeared during the programming. He has checked results of program for physical principles compliance.

† Email address for correspondence: egor.nuzhin@skoltech.ru

2. How should it be used?

Optimization of geometry of acoustic, photonic or nanooptical devices numerically, as opposed to analytically, is a modern direction in engineering in acoustics, optics, photonics and metamaterial science.

It is practical because new generation of computational devices will require photonics components on silicon chips. To make such devices to operate properly one needs to do careful and precise design. The full version of it is a promising way to design actual nano-photonic devices.

The results of work is the visualisation of processes of interaction of electromagnetic waves with conductive medium that is the area of physics that is interesting during studying this processes for students at physics classes in secondary and higher educationals (one may make simulations with field and see the results that depend on these simualtions). Radio photonics is also the area, where such research are often used. In addition, the main application area of the results is topology optimization of size and performance of photonics devices. This application area is spread among such companies as Huawei, Cisco, Intel, Samsung, IBM. They preform such studies intensively.

Creation of a software capable to return an optimal shape of a certain type of wave - device, for example, a light reflector, optical lens or a wave splitting grating. Due to the novelty of the problem it is not known in advance which types of devices may be effectively optimized using standard techniques.

3. Problem formulation

Problem to be resolved is the modeling and the representation of the interaction of electromagnetic waves in a conductive medium as well as the topology optimization of material for wave-based devices. In addition, to mathematically formulate an optimization problem, and especially topology optimization problem based on the generic functional obtained from an integral equation is one of the main part of this research. Furthermore, to explore a few various optimization methods and formulations worked in conjunction with FFT-GMRES Volume Integral Equation solver.

3.1. *Existing methods to resolve this problem:*

This problem may be resolved using the Volume Integral Equation solver. There are attempts to resolve this problem using differential form of equations. In addition, this problem is easily computable using using numerical methods and experimental observing because to compute field analytically for complex forms is very hard. However, this problem is solved numerically more accurately than experimental observing.

3.2. *Data*

Input parameters - some 2D distribution of material as a binary mask. Output parameters - complex-valued electric acoustic field distribution in the device, obtained from physics solver. Cost function - some functional of the quality of the device of the electric field, depending on the device. For example, in case of simple devices such as lens, parabolic reflectors or optical systems the cost function will the the focus power.

4. Main wave integral equations

4.1. *Helmholtz equation*

This research is based on the integral form of Helmholtz equation that represents dependence of electric-field vector $E(x)$ on the dielectric constant distribution $\varepsilon(x)$.

$$E(x) - k^2 \int green\_fun(x - \tilde{x})[\varepsilon(\tilde{x}) - 1]E(\tilde{x})d\tilde{x} = F(x), \tag{4.1}$$

where $E(x)-$is required field, $green_fun(x-\tilde{x})-$is the Green's function, $k-$ is the wave number, $\varepsilon(\tilde{x})-$is the dielectric constant, $F(x)-$ is the electric-field vector distribution in a vacuum.

The $green\_fun(r)$ is represented by the Hankel-function H_α that is the complex superposition of Bessel functions of of the first kind J_α and of the second kind Y_α, where α is complex number, the order of the Bessel function. The $\alpha = 0$ with first kind of Hankel-function $H_0^{(1)}$ has been used in this research:

$$H^{(1)}_{\alpha=0} = J_{\alpha=0} + iY_{\alpha=0} \Leftrightarrow H_0^{(1)} = J_0 + iY_0 \tag{4.2}$$

The $green\_fun$ has been represented using the Hankel-function in the following way:

$$green\_fun(r) = \frac{i}{4} H_0^{(1)}(k \cdot r) \tag{4.3}$$

Equivalenly, in Python-code:

```
def green_fun(self,r):
        return (1j/4)*scipy.special.hankel1(0,self.k*r)
```

5. Wave equation discretization (Vacuum with continuous and homogeneous discrete substances are considered separately)

The equivalent formulation of the integral equation (4.1) is the linear equations system that resolves numerically in this research in Python-code:

$$A(m) \cdot x(m) = f(m), \tag{5.1}$$

where

$$\begin{cases} A(m) = I - k^2 \cdot (\varepsilon - 1) \cdot G \cdot diag(m) \\ x(m) = vect(E(m)) \\ f(m) = vect(F(m)) \end{cases}, \tag{5.2}$$

where G is matrix $size^2 \times size^2$ that corresponds to 4-dimensional cubic $size \times size \times size \times size$ Green's matrix tensor G^{4D}, m is the $size^2$-dimensional vector to be vectorized 2D $size \times size$ mask M that corresponds to the presence of the substance in the 2D space that in this research case is square grid with sizes: $size \times size$: $M_{ij} = 1$ means the presence of the substance in the i, j-th pixel of grid, but $M_{ij} = 1$ means the lack of the substance in the i, j-th pixel of the grid. $m \in T$ is considered, where T in the LP problem with the respect to m is represented by continuous segments in $size^2$-dimensional space limited by 0-os in the left and by 1-es in the right in each axis: $T_{continuous} \equiv C \equiv [0, 1]^{size^2}$ is

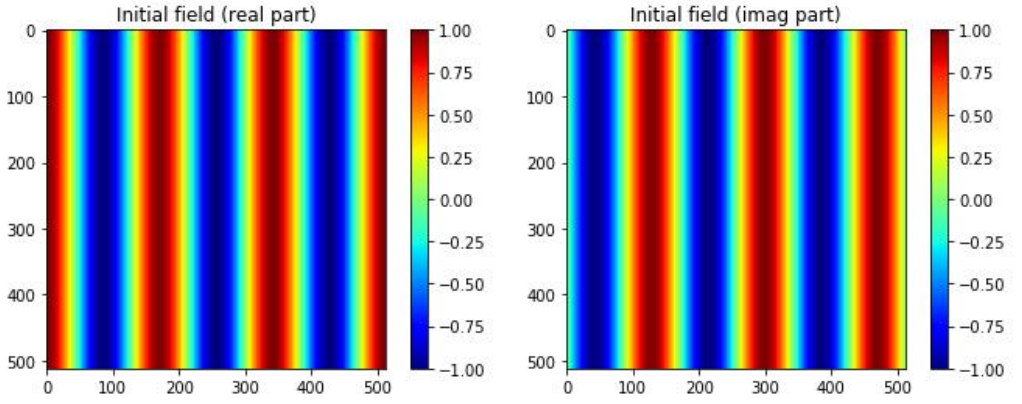


FIGURE 1. Initial field vector (in a vacuum)

the $size^2$-fold Cartesian product, but in the ILP problem with the respect to m, T is represented by discrete values in $size^2$-dimensional space with 0-os and 1-es in each axis: $T_{discrete} \equiv D \equiv \{0,1\}^{size^2}$.

5.1. *Field vector $f(m)$*

Field vector $f(m)$ (Figure 1) is the $size^2$-dimensional vector to be the vectorized matrix $size \times size$ $F(m)$ of the electric-field distribution:

$$F_{ij} = F_{amplitude} \cdot \exp\{k\,[\cos(\phi) \cdot j + \sin(\phi) \cdot i]\}, \forall i \in \overline{1, size}, \forall j \in \overline{1, size}, \tag{5.3}$$

where ϕ is the angle between the j-axis of grid and the direction of wave propagation, $k = \frac{2\pi}{\lambda}$ is the wave number, $\lambda = \frac{\textbf{wavelength per domain}}{size}$ is the wavelength. In Python:

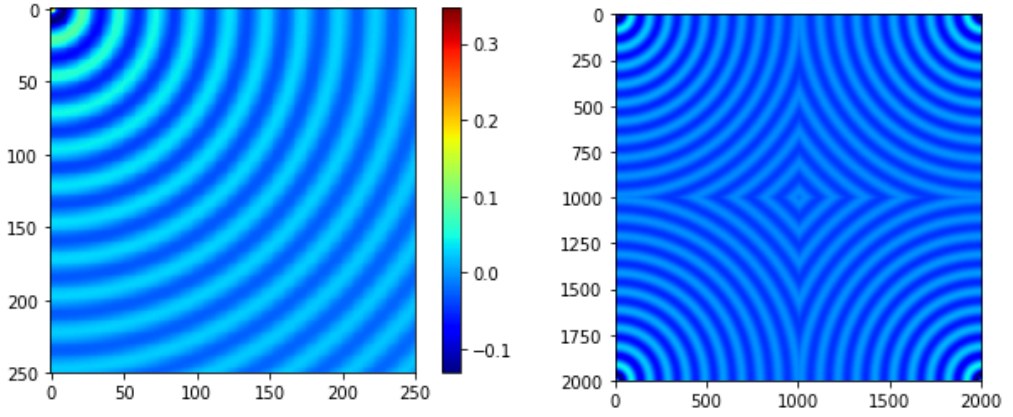
```
def get_f_for_vacuum(size,wave_length_per_domain = 1,
                     deg_angle = 0, F_amplitude = 1):
    y,x = np.mgrid[:size, :size]
    k  = 2*np.pi/(size/wave_length_per_domain)
    rad_angle = np.pi*deg_angle/180
    F = F_amplitude*np.exp(-1j*k*
                (x*np.cos(rad_angle)+y*np.sin(rad_angle)))
    return F.reshape(-1)
```

5.2. *4-dimensional cubic Green's matrix tensor G^{4D}*

Green's matrix tensor G^{4D} is determined by the following formula:

$$\begin{gathered} G^{4D}_{i_1,i_2,j_1,j_2} = green\_fun(r_{i_1,i_2,j_1,j_2}), \\ r_{i_1,i_2,j_1,j_2} = ((correction + i_1 - i_2)^2 + (j_1 - j_2)^2)^2 = r_{i_1-i_2,j_1-j_2} \Rightarrow \\ G^{4D}_{i_1,i_2,j_1,j_2} = G^{4D}_{i_1-i_2,j_1-j_2}, \forall i_1 = \overline{1, size}, \forall i_2 = \overline{1, size}, \forall j_1 = \overline{1, size}, \forall j_2 = \overline{1, size} \end{gathered} \tag{5.4}$$

where *correction* in this research paper has been equal to $-\frac{1}{2}$. Thus, G^{4D} is the multilevel Toeplitz matrix. The 2D space of $\Delta i, \Delta j$-th G-matrix values dependent on $\Delta i \equiv i_1 - i_2$ and $\Delta j \equiv j_1 - j_2$ is represented in Figure 2(a). In Python:



(a) (b)

FIGURE 2. The visualization for G^{2D}-matrix (a) and for the extended $G^{2D}_{extended}$-matrix to solve FFT-matvec problem(b)

```
def get_green_matrix(self):
    j,i = np.mgrid[:self.size, :self.size]
    correction = −1/2
    r = np.sqrt((i+correction)**2+j**2)
    green_mat = self.green_fun(r)
    return green_mat
```

5.3. *Fast (FFT) matvec linear operator for G matrix by vector multiplication*

The G-matrix by arbitrary $size^2$-dimensional vector $x_{arbitrary}$ multiplication problem is the following:

$$G_{multiplication} = G \cdot x_{arbitrary}, \tag{5.5}$$

where $G_{multiplication}$ is the result of this multiplication to be $size^2 \times size^2$ matrix.

In this case usual matrix-by-vector multiplication requires $\mathcal{O}(size^4)$, but the same results may be obrained using 2D $size \times size$ fast (FFT) matvec for multilevel Toeplitz matrix G^{4D}. For this 2D FFT matvec one have to do $size \times 1D$ FFTs in each axis so that's $\mathcal{O}(size^2 \cdot \log size)$. In addition, in the case of usual G-matrix by vector multiplication there is a storage problem with matrix G. For example, for $size = 64$: G - matrix storage requires $size^4$ float elements storage that corresponds to 2 gigabytes of Random-Access Memory while mentioned FFT matvec requires to storage $(2 \cdot size - 1) \times (2 \cdot size - 1)$ $G^{2D}_{extended}$ instead of G and $(2 \cdot size - 1) \times (2 \cdot size - 1)$ $X_{arb.ext}$ instead of $x_{arbitrary}$. Thus, both these problems have been resolved using 2D FFT matvec linear operator $\widehat{\text{FFT\_matvec}}\{G, (\cdot)\}$ for matrix G by vector $x_{arbitrary}$ multiplication in this research.

Firstly, this linear operator creates $size \times size$ G^{2D}-matrix, such that $G^{2D}_{i,j} \equiv G_{\Delta i, \Delta j}, \forall i = \overline{\Delta i = 1, size}, \forall j = \overline{\Delta j = 1, size}$, where $G_{\Delta i, \Delta j}$ has been represened in (5.4). Secondly, the linear operator extend G^{2D}-matrix (Figure 2(a)) to $G^{2D}_{extended} = \begin{bmatrix} G^{2D} & G^{2D}_{-2} \\ G^{2D}_{-1} & G^{2D}_{-1,-2} \end{bmatrix}$ (first matrix of the 4-dimensional circulant tensor represented in Figure 2(b)), where -1 means to eliminate first row and to inverse row order while

-2 means to eliminate first column and to inverse the column order. Thirdly, the linear operator converts $size^2$-element $x_{arbitrary}$ to $size \times size$ $X_{arbitrary}$ that, next, extends to $X_{arb.ext} = \begin{bmatrix} X_{arbitrary} & 0_{size\times(size-1)} \\ 0_{(size-1)\times size} & 0_{(size-1)\times(size-1)} \end{bmatrix}$, where $0_{m\times n}$ is the zero $m \times n$ matrix. Finally, the result of this linear operator is computed by using the following formula:

$$\begin{gathered} G^{4D} \cdot X_{arbitrarty} = \left[\mathrm{IFFT}_{2\mathrm{D}}\left\{\mathrm{FFT}_{2\mathrm{D}}\left\{G^{2D}_{extended}\right\} \cdot \mathrm{FFT}_{2\mathrm{D}}\left\{X_{arb.ext}\right\}\right\}\right]_{\overline{1,size},\overline{1,size}} \equiv \\ \equiv G^{2D}_{multiplication}, \\ vect(G^{2D}_{multiplication}) = G_{multiplication}, \\ \textbf{Thus, } \widehat{\mathrm{FFT\_mat}}\mathrm{vec}\{G, x_{arbitrary}\} \equiv \\ \equiv vec\left(\left[\mathrm{IFFT}_{2\mathrm{D}}\left\{\mathrm{FFT}_{2\mathrm{D}}\left\{G^{2D}_{extended}\right\} \cdot \mathrm{FFT}_{2\mathrm{D}}\left\{X_{arb.ext}\right\}\right\}\right]_{\overline{1,size},\overline{1,size}}\right) \end{gathered} \tag{5.6}$$

where $[(\cdot)]_{\overline{1,size},\overline{1,size}}$ means to take only first i,j-th elements of $(\cdot)$-matrix, where $i = \overline{1,size}, j = \overline{1,size}$; $G^{2D}_{multiplication}$ is $size \times size$ matrix. In Python:

```
    def get_G_for_fft(self):
        toeplitz_block_G = self.get_green_matrix()
        G_for_fft = get_circulant_matrix(toeplitz_block_G)
        return G_for_fft

    def get_fft2_right_matrix(self,vec):
        top_left = vec.reshape((-1,self.size))
        top = np.hstack((top_left,self.top_right_zeros))
        mat = np.vstack((top,self.bottom_zeros))
        return mat

    def G_matvec(self,vec):
        mat = self.get_fft2_right_matrix(vec)
        mat_result = np.fft.ifft2(np.fft.fft2(self.G_for_fft)
                                              *np.fft.fft2(mat))
        result = mat_result[:self.size,:self.size].reshape(-1)
        return result
```

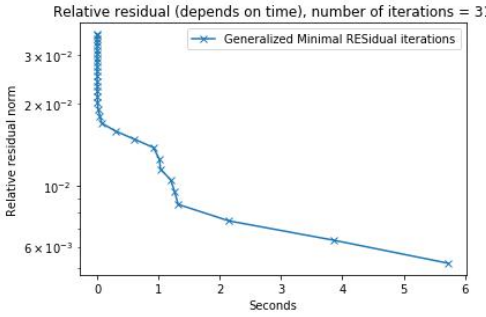
5.4. *Linear operator $\widehat{A(m)}$ for $A(m)$-matrix*

$\widehat{A}$ is based on $\widehat{\mathrm{FFT\_mat}}\mathrm{vec}\{G, (\cdot)\}$, according to (5.2) and (5.6):

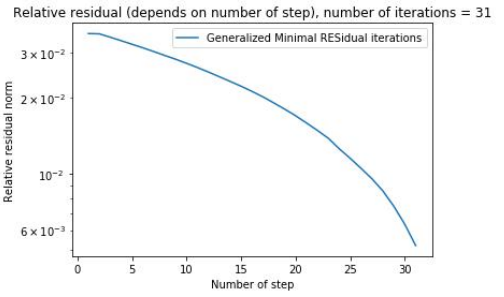
$$\begin{gathered} \widehat{A(m)} \cdot x_{arbitrary} = \widehat{I} - k^2 \cdot (\varepsilon - 1) \cdot \widehat{\mathrm{FFT\_mat}}\mathrm{vec}\{G, \mathrm{diag}(m) \cdot x_{arbitrary}\} \equiv \\ = \widehat{I} - \chi \cdot \widehat{\mathrm{FFT\_mat}}\mathrm{vec}\{G, m \circ x_{arbitrary}\}, \\ \chi = k^2 \cdot (\varepsilon - 1) \end{gathered} \tag{5.7}$$

where $\circ-$is the Hadamard product, $\widehat{I}-$ is the linear operator that corresponds to the identity matrix I; $\widehat{\mathrm{diag}(m)}-$is the linear operator that corresponds to the diagonal matrix $\mathrm{diag}(m)$. In Python:

```
    def get_eps(self):
        raise RunTimeError()
```



Depend on time



Depend on the step number

FIGURE 3. Convergence of GMRES Solver (grid 512×512). Relative residuals graphs. Iterations number = 31

```
    def A_matvec(self,vec):
        eps = self.get_eps()
        chi = self.k**2*(eps - 1)
        return vec-self.G_matvec(vec*chi)
```

5.5. *Linear equation system solver usage*

In this research built-in GMRES algorithm of *scipy.sparse.linalg*-library has been used. The stop of this algorithm comes (Figure 3) when $\|r_{k+1} - r_k\| \leqslant tol$, where r_k is the kth iteration residual of this alogorithm, *tol* is the required absolute tolerance of this algorith, that is, by-default, $= 10^{-5}$. In Python:

```
def plot_relative_residuals_norms(time, residuals, b):
    norm_b = np.linalg.norm(b)
    plt.suptitle("Relative residuals (GMRES), number of
                iterations =%i" % len(residuals),size = 16)

    plt.subplot(1,2,1)
    plt.semilogy(time, residuals/norm_b, 'x-')
    plt.title("Depends on time")
    plt.xlabel('Seconds')
    plt.ylabel('Relative residual norm')

    plt.subplot(1,2,2)
    plt.semilogy(np.arange(len(residuals)),
                                        residuals/norm_b,'x-')
    plt.title("Depends on number of step")
    plt.xlabel('Number of step')
    plt.ylabel('Relative residual norm')

    plt.subplots_adjust(left=0.2, wspace=0.8, top=0.8)
    plt.show()

def plot_convergence(A, b, **kwargs_for_gmres):
```

```
gmres_residuals_with_time = []
def write_residuals_callback(residual):
    gmres_residuals_with_time.append(
            [np.linalg.norm(residual), time.time()])

t0 = time.time()
solution, info =  spla.gmres(A, b,callback =
            write_residuals_callback,**kwargs_for_gmres)
if info != 0:
    raise RunTimeError("GMRES doesn't converge!")

if len(gmres_residuals_with_time)>=1:
    gmres_residuals, gmres_full_time = np.array(
                            gmres_residuals_with_time).T
    gmres_time = gmres_full_time - t0
    plot_relative_residuals_norms(gmres_time,
                                  gmres_residuals, b)
```

5.6. *Example of computing on circular mask*

In Python (Figure 4):

```
mask = Mask(size).get_quadratic_mask
matvec = DiscreteMatvec(size,e0,mask)
f = get_E_for_plane_wave(size)
plot_initial_condition(f, mask)
plot_convergence(matvec.A,f.reshape(-1))
```

6. Topology optimization problem solvers

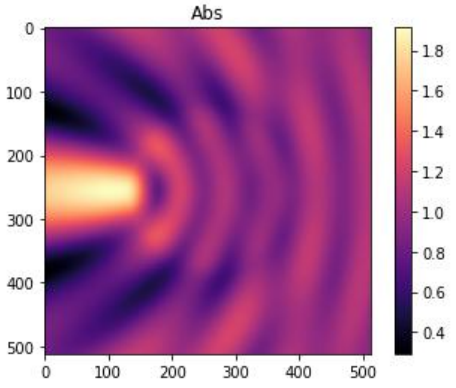
6.1. *Topology optimization problem statements*

The topology optimization problem is to maximize the maximal power of electromagnetic wave $E(x)$ in 2D space by varying the mask of the 2D space m. According to (5.2), there are two optimization problem statement: for discrete ($m \in D$) mask:

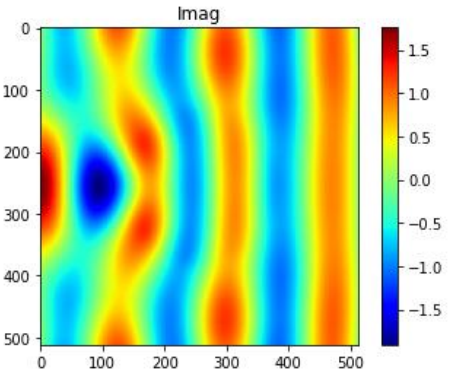
$$\max_m \|x(m)\|_\infty$$
$$\textbf{Subject to} \begin{cases} A(m) \cdot x(m) = f(m) \\ A(m) = I - k^2 \cdot (\varepsilon - 1) \cdot G \cdot diag(m) \\ m \in D \end{cases} \tag{6.1}$$

and for continuous ($m \in C$) mask:

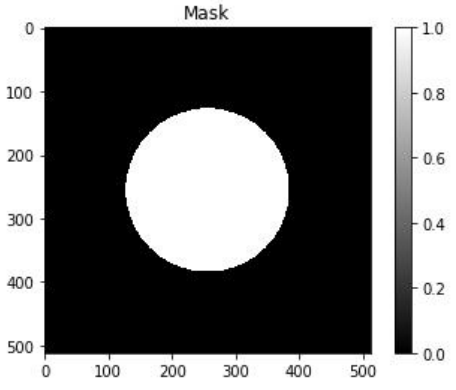
$$\max_m \|x(m)\|_\infty$$
$$\textbf{Subject to} \begin{cases} A(m) \cdot x(m) = f(m) \\ A(m) = I - k^2 \cdot (\varepsilon - 1) \cdot G \cdot diag(m) \\ m \in C \end{cases} \tag{6.2}$$



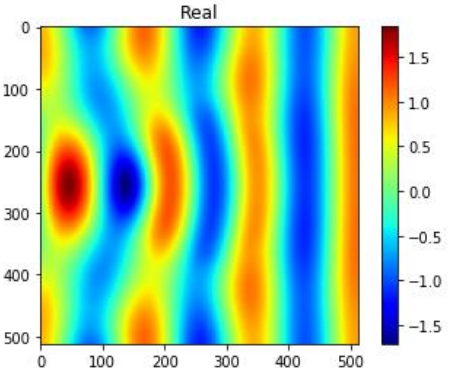
Absolute value on field vector x



Imaginary part of solution (field vector x)



Circular mask M



Real part of solution (field vector x)

FIGURE 4. Circular mask ($\varepsilon = 1.5$)

General formulation of this problem is non-convex because a lot of local optimums may be achieved in this statement because of big freedom degree of 2D space, but the convex problem may be formulated using decreasing the freedom degree using heuristic constraints required for practiacal implementations to be devices. One of them is almost connected set. It means that result of this resarch should be mask to be one, two or three connected sets. In this research it has been achieved by using neighbors bypassing idea. This idea is based on process of adding of neighbor mask pixels that corresponds to the best increasing of electromagnetic power in the pixel of maximal power electromagnetic power.

7. Penalization of optimization problem

$$\begin{gathered} \max_m \|x(m)\|_\infty - c \cdot m^T(1-m) \\ \textbf{Subject to} \begin{cases} A(m)x(m) = f(m) \\ A(m) = I - k^2 \cdot (\varepsilon - 1) \cdot G \cdot diag(m) \\ 0 \leqslant m \leqslant 1 \end{cases} \end{gathered} \tag{7.1}$$

8. Acceleration of optimization problem provided by Jacobi matrix

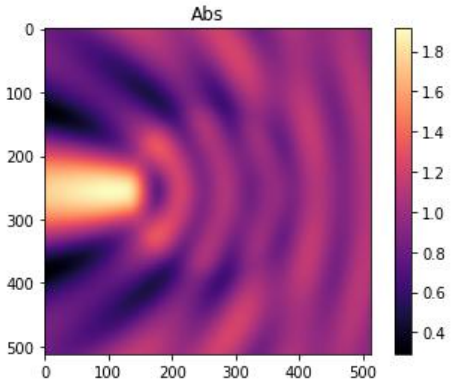
To solve this problem fast we have coupled Jacobian:

$$\begin{aligned} A(m) \cdot J_m(x) &= B(m) \\ A(m) &= I - k^2 \cdot (\varepsilon - 1) \cdot G \cdot diag(m) \\ B(m) &= k^2 \cdot (\varepsilon - 1) \cdot G \cdot diag(m) \end{aligned} \tag{8.1}$$

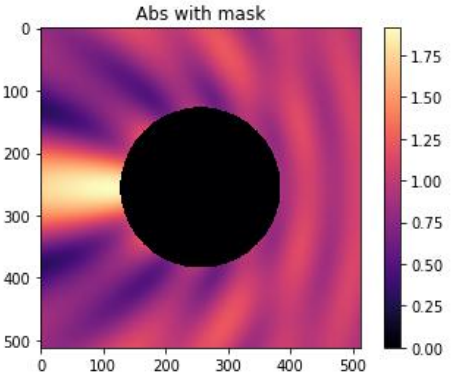
9. Examples of work of the solver

The solver is in Python (programming language). In initial vector of wave intensity that corresponds to the case when there is now homogeneous substance and there is only vacuum in all $n \times n$ 2D space is represented by its real part and its imaginary part:

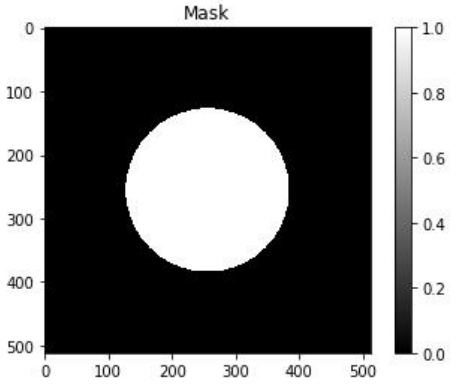
The **number of periods of electromagnetic wave** on this planes= $\frac{\textbf{the length of the plane}}{\textbf{the wave length}}$



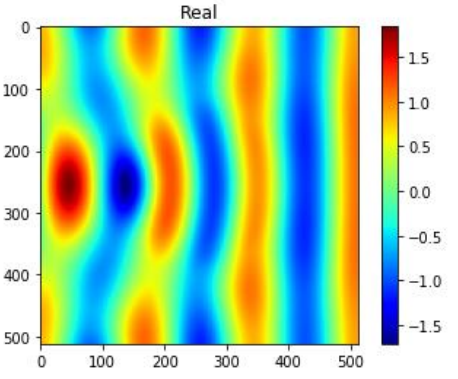
Absolute value on field vector x



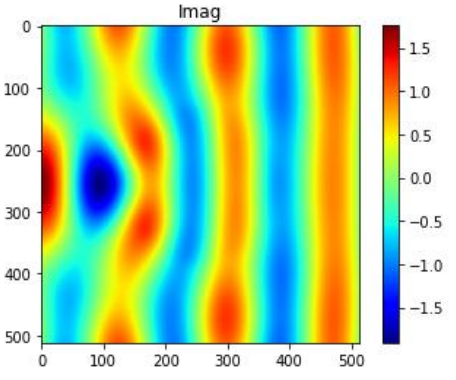
Absolute value on field vector x with mask m



Circular mask m of grid. Black means $m_b = 0$, white means $m_w = 1$

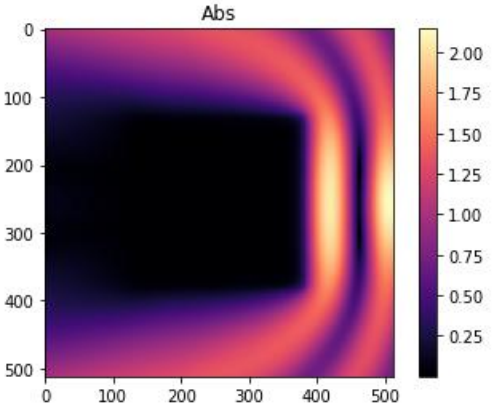


Real part of solution (field vector x)

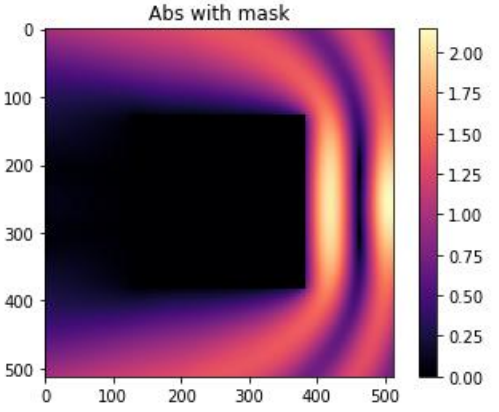


Imaginary part of solution (field vector x)

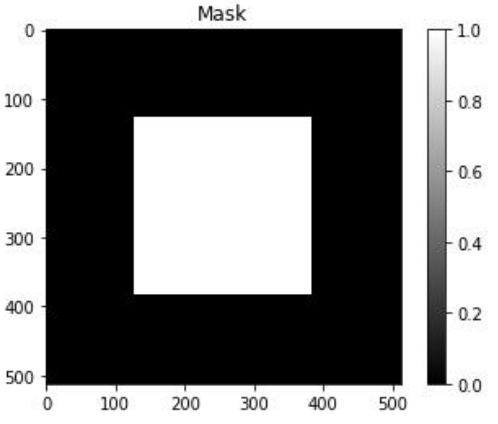
FIGURE 5. Circular mask ($\varepsilon = 1.5$)



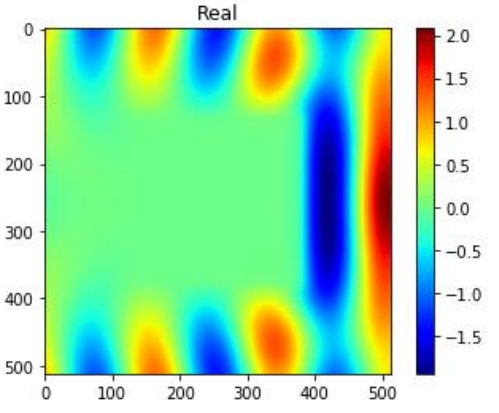
Absolute value on field vector x



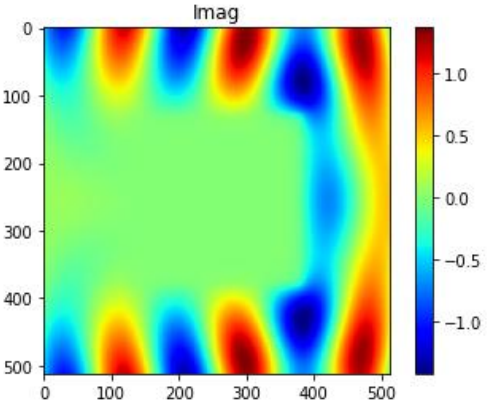
Absolute value on field vector x with mask m



Rectangular mask m of grid. Black means $m_b = 0$, white means $m_w = 1$

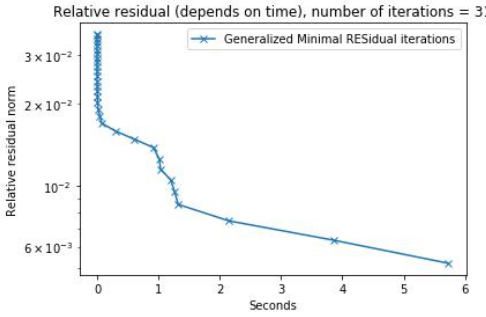


Real part of solution (field vector x)

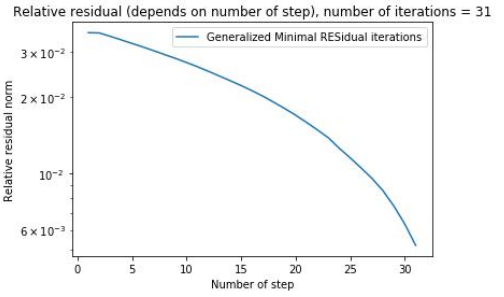


Imaginary part of solution (field vector x)

FIGURE 6. Rectangular mask ($\varepsilon = -10$)

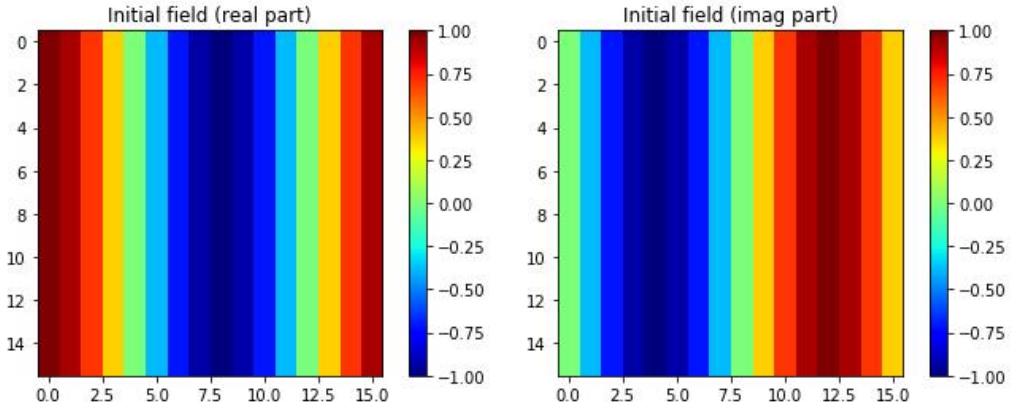


Relative residuals (depend on time), number of iterations = 31



Relative residuals (depend on the step number), number of iterations = 31

FIGURE 7. Convergence of GMRES Solver (grid 512×512)



Real part of discretized field vector Imaginary part of discretized field vector

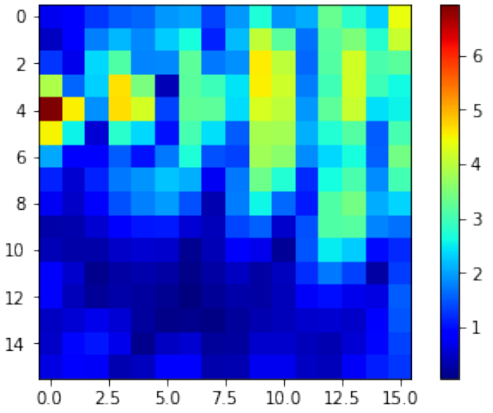
FIGURE 8. Initialization of topology optimization for 16×16 grid with 1 wavelength per domain

10. Results of optimization

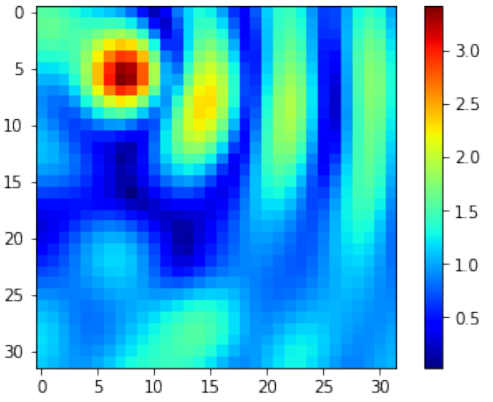
10.1. *Initialization*

For this task grid 16×16 has been used, since optimization is more hard computing task.

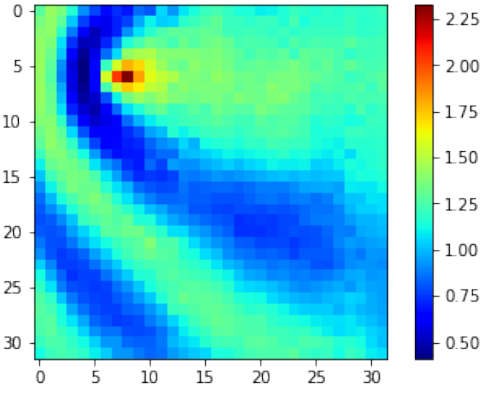
10.2. *Optimization examples on continuous ε grid*



Real part of discretized field vector

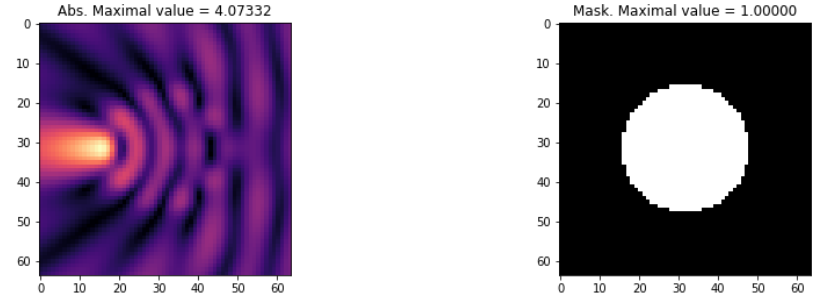


Imaginary part of discretized field vector

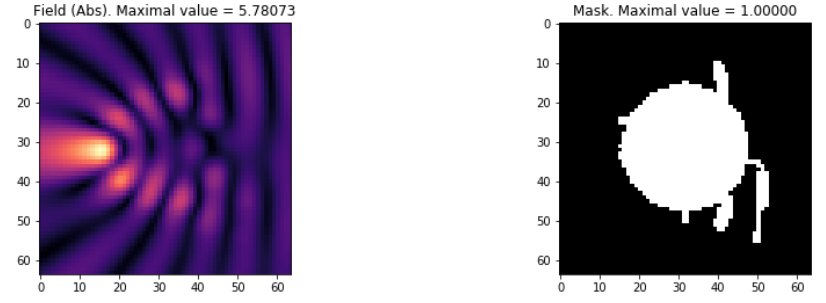


Imaginary part of discretized field vector

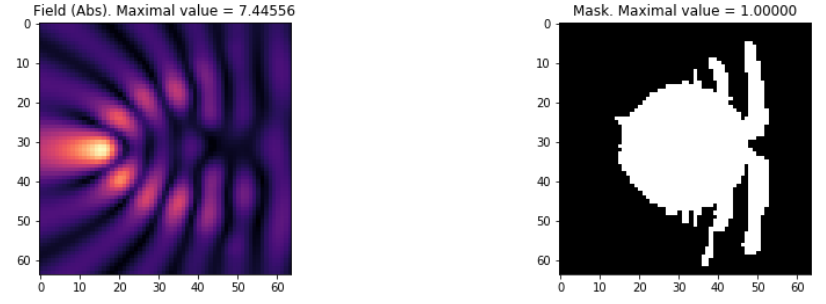
FIGURE 9. Optimization examples on continuous ε grid



Absolute value of field vector Mask



Absolute value of field vector Mask



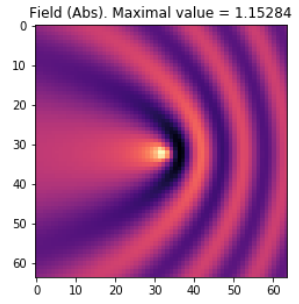
Absolute value of field vector Mask

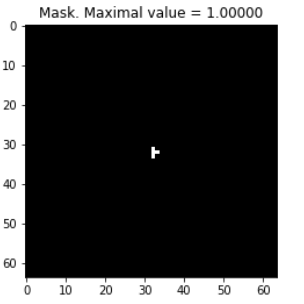
FIGURE 10. Topology optimization of circular mask

10.3. *Optimization examples on discrete ε grid*

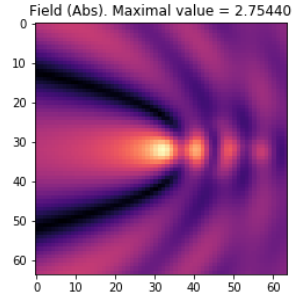
10.3.1. *Using Gradient solver*

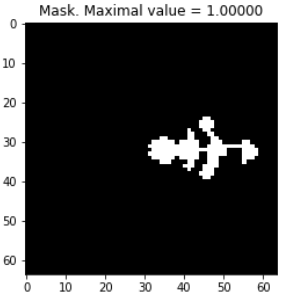
In this case the gradient-solver has been programmed. Initial point has been circle mask. In addition, only the principle of the building neighbors has been implemented in this topology optimization solver. The neighbors with the biggest gradient has been masked (mask on this neighbors with 0 mask changes and becomes to be equal to 1).



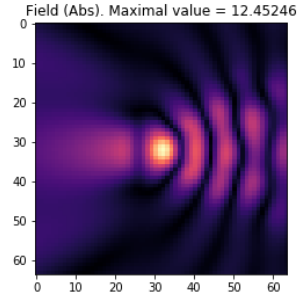


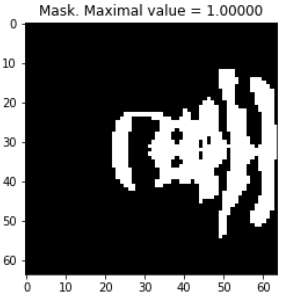
Absolute value of field vector Mask





Absolute value of field vector Mask





Absolute value of field vector Mask

FIGURE 11.

10.3.2. *Using non-gradient method*

In this case the non-gradient-solver has been programmed. Initial point has been the central point of the grid. In addition, only the principle of the building neighbors has been implemented in this topology optimization solver. But it has used broadcasting principle. Program tries to assign the 1 value in the cells of grid that are neighbors of the cells that are equal to 1. Next, program chooses the broadcasting of the biggest objective function.

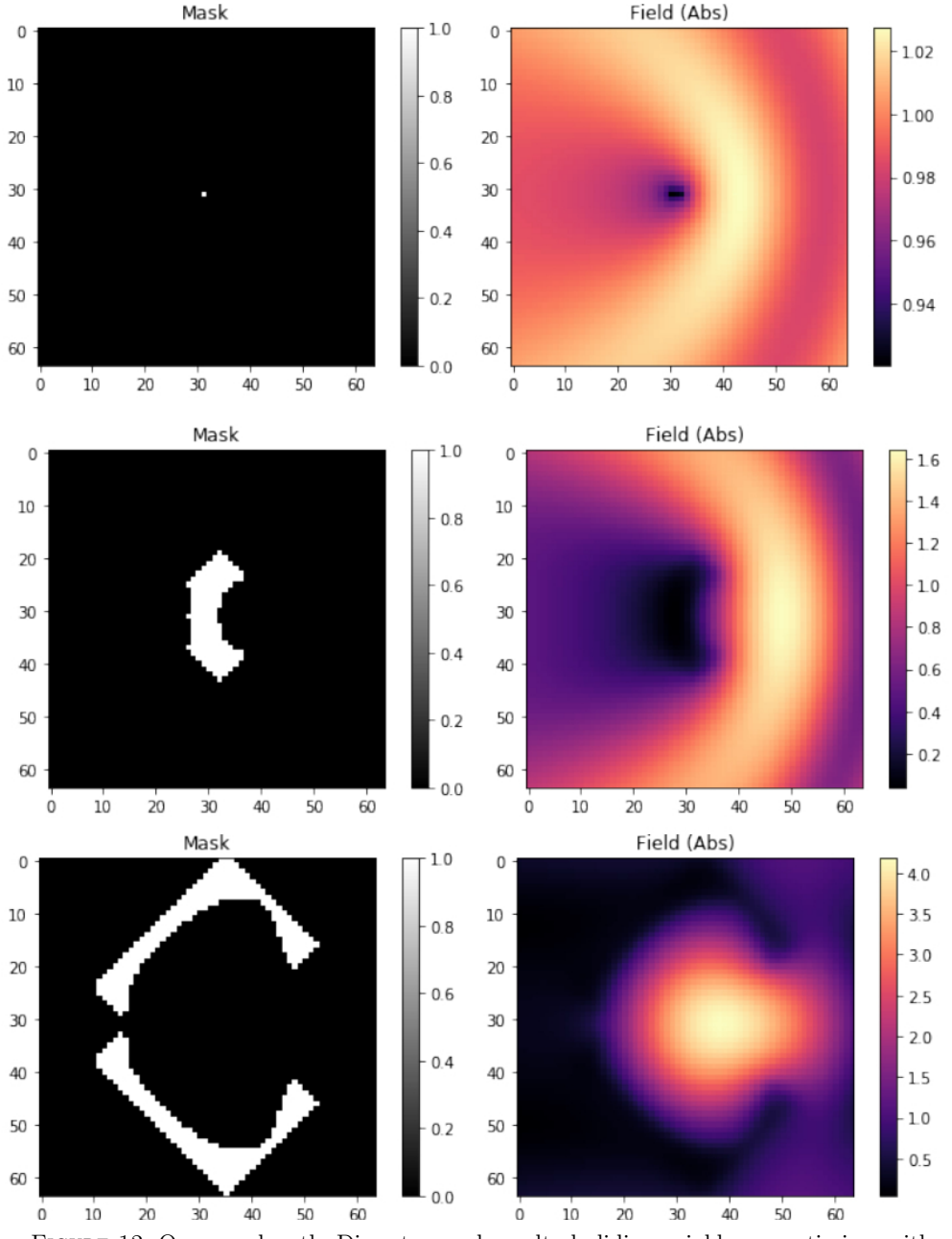


FIGURE 12. One wavelength. Discrete greedy multy-buliding neighbours optimizer with gradient extended by subtraction of mask

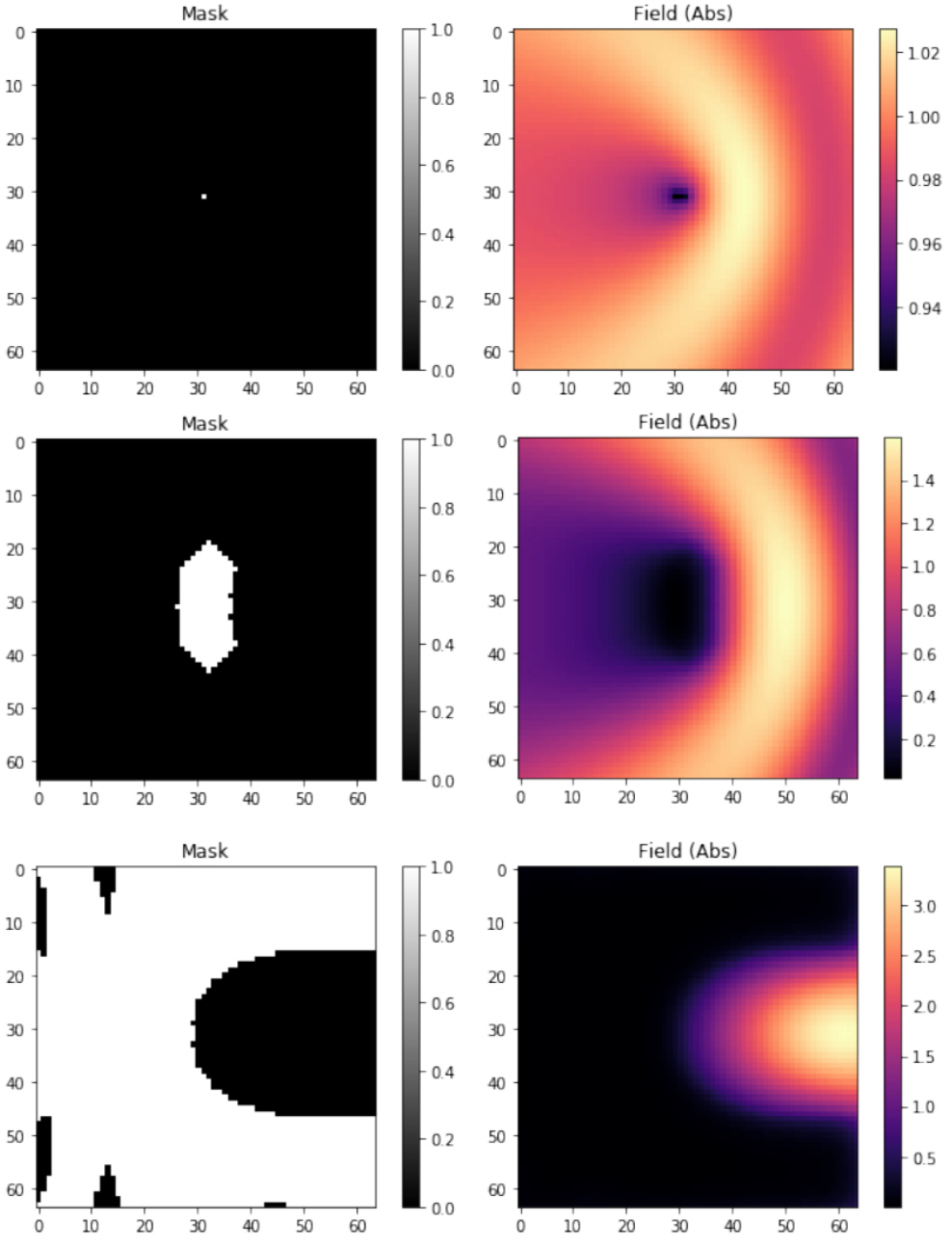


FIGURE 13. One wavelength. Discrete greedy multy buliding neighbours optimizer with gradient

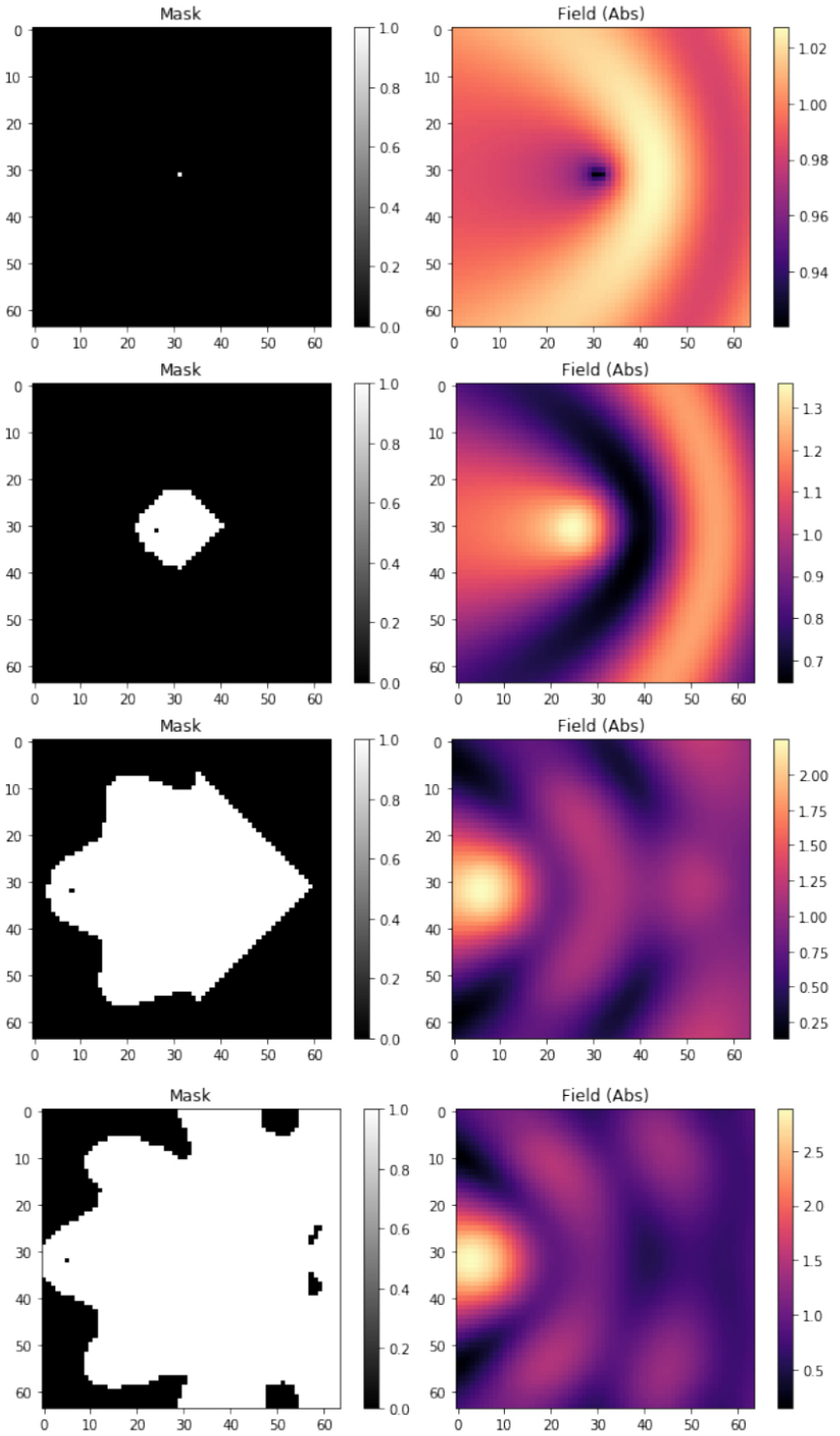


FIGURE 14. One wavelength. Discrete greedy multy buliding neighbours optimizer with gradient

[h]

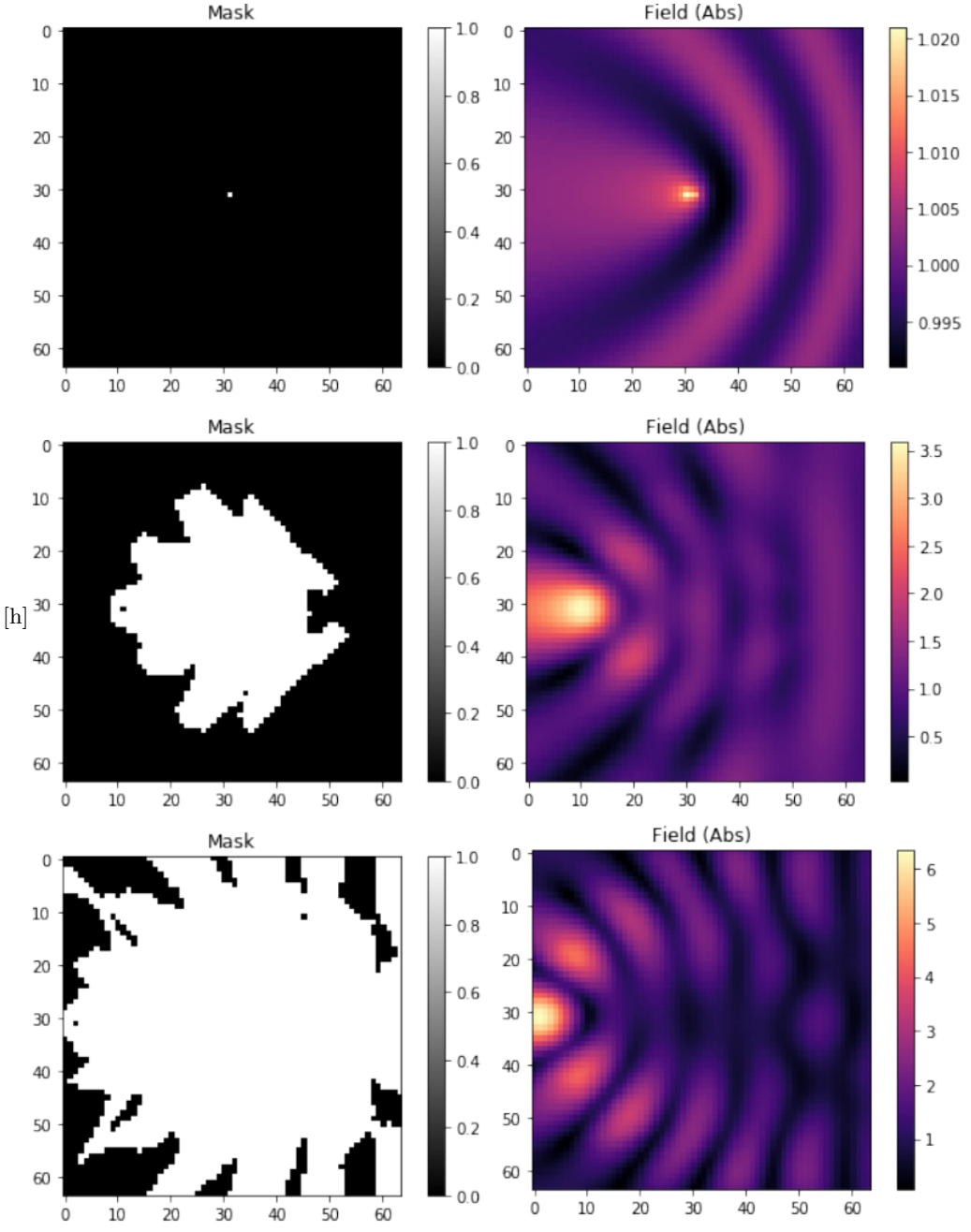


FIGURE 15. Two wavelengths. Discrete greedy multy-buliding neighbours optimizer with gradient

11. The derivation of formulas:

The topology optimization problem to be resolved is the following:

$$\begin{gathered}\underset{m\in D}{\text{objective\_function}} = \max_{i\in g}|x_i(m)|,\\ \textit{subject to}\ \begin{cases} m_i \in \{0,1\}, \forall i \in \overline{1,size^2}, \\ g = \underset{\forall i\in\overline{1,size^2}}{\arg}(m_i = 0)\end{cases}\end{gathered} \tag{11.1}$$

$$\begin{gathered}\widetilde{\underset{m\in D}{\text{objective\_function}}} = \max_{i\in g}(x_i(m))^2,\\ \textit{subject to}\ \begin{cases} m_i \in \{0,1\}, \forall i \in \overline{1,size^2}, \\ g = \underset{\forall i\in\overline{1,size^2}}{\arg}(m_i = 0)\end{cases}\end{gathered} \tag{11.2}$$

These two topology optimization problem formulations (11.1) and (11.2) are equiavalent with the respect to the searching arguments, because of property of the modulus of complex number:

$$\begin{aligned} i^* = \arg\max_{i\in g}|x_i(m)| = \operatorname*{argmax}_{i\in g}|function(i)| \equiv \operatorname*{argmax}_{i\in g} function^2(i) = \\ = \arg\max_{i\in g}(x_i(m))^2 = \arg\widetilde{\underset{m\in D}{\text{objective\_function}}}\end{aligned} \tag{11.3}$$

Let one consider the gradient of the equivalent objective function $\widetilde{\underset{m\in D}{\text{objective\_function}}}$ for maximal element of field vector solution $x(m)$. According to (11.2) and (11.3):

$$\begin{aligned} \frac{\partial\left(\widetilde{\underset{m\in D}{\text{objective\_function}}}\right)}{\partial m} = \frac{\partial|x_{max}|^2}{\partial m} = \frac{\partial\left[\overline{x_{max}(m)}x_{max}(m)\right]}{\partial m} = \\ = \frac{\partial\overline{x_{max}(m)}}{\partial m}x_{max}(m) + \frac{\partial x_{max}(m)}{\partial m}\overline{x_{max}(m)} = \\ = \overline{\frac{\partial x_{max}(m)}{\partial m}}x_{max}(m) + \frac{\partial x_{max}(m)}{\partial m}\overline{x_{max}(m)} = \\ = \overline{\frac{\partial x_{max}(m)}{\partial m}\overline{x_{max}(m)}} + \frac{\partial x_{max}(m)}{\partial m}\overline{x_{max}(m)} = \\ = 2\,\mathbb{R}\mathrm{e}\left\{\frac{\partial x_{max}(m)}{\partial m}\overline{x_{max}(m)}\right\} = 2\,\mathbb{R}\mathrm{e}\left\{[J(x(m))]_{i^*,*}\,\overline{x_{max}(m)}\right\} \equiv \\ \equiv 2\,\mathbb{R}\mathrm{e}\left\{J(x_{max}(m))\overline{x_{max}(m)}\right\},\end{aligned} \tag{11.4}$$

where $i^* = \operatorname*{argmax}_{i\in g}|x_i(m)|$ according to (11.3) and $[J(x(m))]_{i^*,*}$ is the i^*-th **row** of Jacobi matrix $J(x(m)) = \frac{\partial x(m)}{\partial m}$ that corresponds to the optimal element $x_{max}(m)$ of field vector solution $x(m)$ for $size^2$-dimensional vector parameter of mask m.

According to (4.1), (5.2), (6.2), the searching field vector solution $x(m)$ is represented by linear equations system:

$$\begin{gathered} A(m)\cdot x(m) = f(m)\\ \text{where: } A(m) = I - G\cdot\operatorname{diag}[k^2\cdot(\varepsilon-1)\cdot m]\end{gathered} \tag{11.5}$$

The partial derivative of (11.5):

$$\begin{gathered}\frac{\partial}{\partial m}\left[A(m)\cdot x(m)\right]=\frac{\partial f(m)}{\partial m}\\ \frac{\partial A(m)}{\partial m}\cdot x(m)+A(m)\cdot\frac{\partial x(m)}{\partial m}=\frac{\partial f(m)}{\partial m}=0_{size^2\times size^2}\\ \frac{\partial\left[I-G\cdot\operatorname{diag}[k^2\cdot(\varepsilon-1)\cdot m]\right]}{\partial m}x(m)=-A(m)\frac{\partial x(m)}{\partial m}\\ A(m)\frac{\partial x(m)}{\partial m}=G\cdot k^2\cdot(\varepsilon-1)\cdot\frac{\partial\operatorname{diag}(m)}{\partial m}\cdot x(m)=G\cdot k^2\cdot(\varepsilon-1)\cdot I^{3D}\cdot x(m),\end{gathered}\tag{11.6}$$

where I^{3D} is the sparse 3D cubic $size^2\times size^2\times size^2$-dimensional tensor with $size^2$ non-zero elements that are equal to 1 and located on the main diagonal of the I^{3D}: $I^{3D}_{iii}=1,\forall i\in\overline{1,size^2}$. Since the 3D cubic $size^2\times size^2\times size^2$-dimensional I^{3D} tensor by $size^2$-dimensional $x(m)$ vector multiplication is the $\operatorname{diag}(x(m))$:

$$\begin{gathered}A(m)\frac{\partial x(m)}{\partial m}=G\cdot k^2\cdot(\varepsilon-1)\cdot\operatorname{diag}(x(m))\\ \textbf{Thus, } A(m)J(x(m))=B(m),\\ \text{where: }\begin{cases}A(m)=I-k^2\cdot(\varepsilon-1)\cdot G\cdot\operatorname{diag}(m)\\ B(m)=k^2\cdot(\varepsilon-1)\cdot G\cdot\operatorname{diag}(x(m))\\ J(x(m))=\frac{\partial x(m)}{\partial m}\end{cases}\end{gathered}\tag{11.7}$$

However, according to the (11.4), for solving optimization problem (11.1) only the i^*-th **row** of Jacobi matrix $J(x(m))$ is needed. To accelerate computations, one will calulate only the **row** of Jacobi matrix $J(x(m))$ using the special formula that is deriving below from (11.7):

$$\begin{gathered}J^T(x(m))\cdot A^T(m)=B^T(m)\Rightarrow J^T(x(m))\cdot A^T(m)\cdot z(m)=B^T(m)\cdot z(m)\\ J^T(x(m))\cdot y=B^T(m)\cdot z(m)\end{gathered}\tag{11.8}$$

where system has been multiplied on the right by special vector $z(m)$ that depends on sparse vector y, such that: $y=A^T(m)\cdot z(m)$, $y_{i=i^*}=1$ and $y_{i\neq i^*}=0$, where $i^*=\underset{i\in g}{\operatorname{argmax}}|x_i(m)|$, according to (11.3). Thus, (11.8) will be the following:

$$J^T(x(m))\cdot y=[J(x(m))]^T_{i^*,*}\equiv J^T(x_{max}(m))=B^T(m)\cdot z(m)\tag{11.9}$$

Let one consider $B^T(m)\cdot z(m)$. According to (11.7):

$$\begin{gathered}B^T(m)\cdot z(m)=k^2\cdot(\varepsilon-1)\cdot\operatorname{diag}(x(m))\cdot G\cdot z(m)=\\ =k^2\cdot(\varepsilon-1)\cdot\operatorname{diag}(x(m))\cdot\text{FFT\_matvec}\{G,z(m)\},\end{gathered}\tag{11.10}$$

because G is symmetric in our problem statement and any diagonal matrix is symmetric. Ditto, $A^T(m)$ and $A^T(m)\cdot z(m)$, according to (11.5):

$$\begin{gathered}A^T(m)=\{I-k^2\cdot(\varepsilon-1)\cdot G\cdot\operatorname{diag}(m)\}^T=\\ =I-\operatorname{diag}(m)\cdot G^T=I-k^2\cdot(\varepsilon-1)\cdot\operatorname{diag}(m)\cdot G\\ A^T(m)\cdot z(m)=\left[I-k^2\cdot(\varepsilon-1)\cdot\operatorname{diag}(m)\cdot G\right]\cdot z(m)=\\ =z(m)-k^2\cdot(\varepsilon-1)\cdot\operatorname{diag}(m)\cdot\text{FFT\_matvec}\{G,z(m)\}\\ \widehat{\widehat{A(m)}}=\widehat{I}-k^2\cdot(\varepsilon-1)\cdot\widehat{\operatorname{diag}(m)}\cdot\widehat{\text{FFT\_matvec}}\{G,\cdot\},\end{gathered}\tag{11.11}$$

where $\widehat{\widetilde{A(m)}}-$is the linear operator that corresponds to the matrix A^T; $\widehat{I}-$ is the linear operator that corresponds to the identity matrix I; $\widehat{\text{diag}(m)}-$is the linear operator that corresponds to the diagonal matrix $\text{diag}(m)$; and $\widehat{\text{FFT\_matvec}}\{G,\cdot\}$ is the linear operator that executes the FFT_matvec for multiplication of matrix G by vector represented in the next to $\widehat{\widetilde{A(m)}}$ operator.

Thus, according to (11.8), (11.9), (11.10) and (11.11), one resolves the following system:

$$\begin{cases} i^* = \underset{i\in g}{\text{argmax}}\,|x_i(m)| \\ y_i = \begin{cases} 0, i \neq i^* \\ 1, i = i^* \end{cases} \\ \widetilde{\widetilde{A(m)}} \cdot z(m) = y \\ [J(x(m))]^T_{i^*,*} \equiv J^T(x_{max}(m)) = k^2 \cdot (\varepsilon - 1) \cdot \text{diag}(x(m)) \cdot \text{FFT\_matvec}\{G, z(m)\} \end{cases} \tag{11.12}$$

REFERENCES

Jensen, J. S. and Sigmund, O., "Topology optimization for nano-photonics," *Laser & Photon.*, vol. 5 (2011), pp. 308–321. doi:10.1002/lpor.201000014.

Alexander, Y., Piggott, et al, *Inverse design and implementation of a wavelength demultiplexing grating coupler*, (details).