

# DOT SIMULATOR – INSTALLATION AND SIMULATION INSTRUCTIONS

## Contents

1.	Installation .....	3
a.	Gmsh .....	4
b.	Toast++ .....	5
c.	Integrating Gmsh with Toast++ .....	6
2.	Simulation .....	7
a.	Files Overview .....	8
b.	Parameters Overview .....	9
c.	Simulation Example .....	10

## 1. Installation

## DOT SIMULATOR – INSTALLATION AND SIMULATION INSTRUCTIONS

### a. Gmsh

- Navigate to <https://gmsh.info/>
- Download the latest stable release compatible with your operating system.

#### Download






Gmsh is distributed under the terms of the [GNU General Public License \(GPL\)](#):

- Current stable release (version 4.12.2, 21 January 2024):
  - Download Gmsh for [Windows, Linux, macOS \(x86\) or macOS \(ARM\)](#) \*
  - Download the [source code](#)
  - Download the Software Development Kit (SDK) for [Windows, Linux, macOS \(x86\) or macOS \(ARM\)](#) \*
  - Download both Gmsh and the SDK with pip: `'pip install --upgrade gmsh'`

- Unzip the downloaded zip file.
- Run the executable file (“gmsh.exe”), which is the Gmsh user application.

## b. Toast++

- Navigate to <https://github.com/toastpp/toastpp/releases>
- Download the source code and bin files compatible with your current operating system.

▼ Assets 5		
 toast_bin_darwin64.zip	1.22 MB	Feb 10, 2017
 toast_bin_linux64.zip	33.4 MB	Feb 10, 2017
 toast_bin_win64.zip	3.95 MB	Feb 10, 2017
 Source code (zip)		Feb 10, 2017
 Source code (tar.gz)		Feb 10, 2017

- Unzip the downloaded zip files.
- Copy the extracted content of the bin zip to the source code extracted folder. Assuming you are running windows, copy win\x64 directory, into the toastpp-2.0.2\win directory.
- To ensure compatibility with MATLAB and verify the presence of a suitable mex compiler, execute the command "mex -setup" in your MATLAB command window.

```

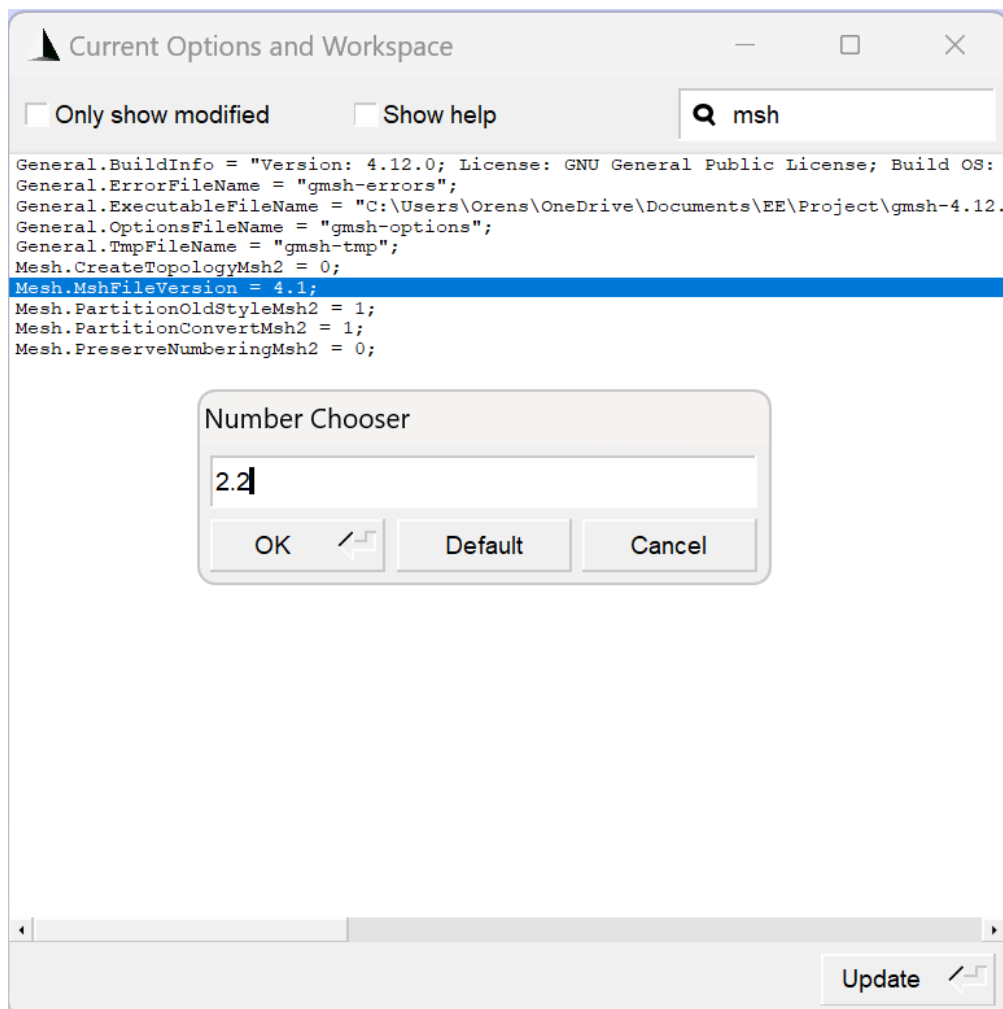
Command Window
>> mex -setup
MEX configured to use 'Microsoft Visual C++ 2015 Professional (C)' for C language compilation.
Warning: The MATLAB C and Fortran API has changed to support MATLAB
variables with more than 2^32-1 elements. You will be required
to update your code to utilize the new API.
You can find more information about this at:
http://www.mathworks.com/help/matlab/matlab\_external/upgrading-mex-files-to-use-64-bit-api.html.

fx To choose a different language, select one from the following:
  
```

### c. Integrating Gmsh with Toast++

To integrate Toast++ 2.0.2 MATLAB toolbox with the latest Gmsh and ensure compatibility with older revisions (2.x) of MSH files, follow these steps:

- Click on "Help" and then select "Current Options and Workspace."
- Within the options and workspace window, change the version from 4.1 to 2.2.
- Click on the "Update" button to apply the changes.
- When saving a mesh file, it will now be formatted correctly to be compatible with Toast++.

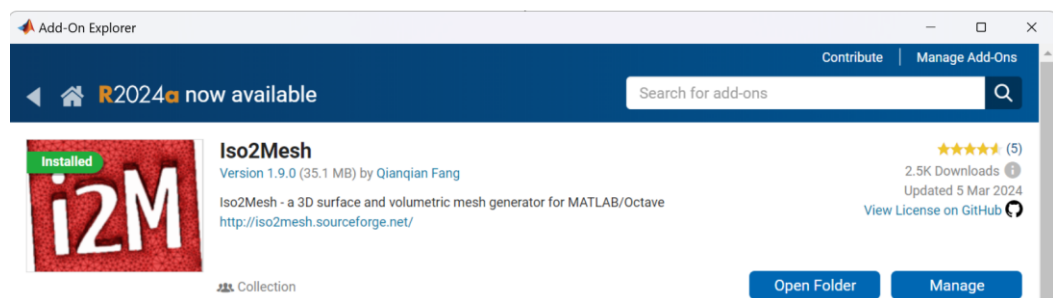


## 2. Simulation

To create a realistic simulation resembling a genuine Diffuse Optical Tomography (DOT) scan, we acquired a breast mesh from Digi breast. This mesh serves as the foundation for our simulation, accurately replicating the shapes of real breasts, including the presence of a tumour inside.

Before beginning:

- Clone (or download) <https://github.com/SchorenTK/ML-DOT-Cancer-Scan-Reconstruct>
- Copy toast++ directory to the specified directory in the git ('Toast\_app').
- Install iso2mesh toolbox (via MATLAB's add-ons)



- Run 'updatepath.m'

### a. Files Overview

The following files are essential for the simulation and located at the auxiliary folders:

- ‘Base\_Mesh\_struct.mat’: This file contains an array comprising elements and nodes extracted from the Digi Breast dataset, it will be used as the basis of any simulation.
- ‘conv\_nodes\_elements\_arrays\_to\_msh.m’: This function converts node and element arrays to a Gmsh .msh file format.
- ‘stretch\_mesh.m’: This script defines a function to stretch a mesh along specified scaling factors.
- ‘project\_meshrefine.m’: This script defines a function to refine a mesh within a spherical region and calculate element centroids, it will be used to set a tumour area.
- ‘test\_3d\_scan.m’: This script initializes the simulation, loads a base mesh, selects a random node, refines the mesh around the selected node, and simulates a DOT scan.



## b. Parameters Overview

Users can customize various parameters to simulate different DOT scans with distinct characteristics:

- **Source and Detector Locations:** Adjust the number of sources and detectors above the breast mesh by modifying the variable 'nqm'.
- **Breast Mesh Scales:** In the 'stretch\_mesh' function, users can change the scaling factors ('scaleX', 'scaleY', 'scaleZ') to stretch or compress the breast mesh along different axes, controlling its size.
- **Tumor Mesh Size and Location:** When calling the 'project\_meshrefine' function, users can specify parameters for refining the mesh within a spherical region, such as 'centroid', 'radius', and 'maxvol', determining the size and location of the simulated tumor within the breast mesh.
- **Optical Parameters for the Breast and Tumor:** Optical properties of the breast and tumor tissues are defined in variables like mua\_bkg, mus\_bkg, and ref\_bkg. Users can adjust these values to simulate different optical characteristics for the breast. Additionally, similar optical properties are assigned to tumor nodes, allowing users to model variations in the tumor's optical properties.

By adjusting these parameters within the provided MATLAB code, users can create diverse scenarios for DOT scans, exploring different configurations of source-detector pairs, breast mesh scales, tumor characteristics, and optical properties.

### c. Simulation Example

Currently, our 'toast\_3d\_scan' functions use the 'Base\_Mesh\_struct.mat' as the structure without any additional changes (such as using the 'stretch\_mesh.m' function). By changing the highlighted parameters in the provided code below, you can bypass any randomness of the code (tumour location and size) or adjust the scan environment (number of sources\detectors and optical parameters).

```
clear all
close all

fullpath = mfilename('fullpath');
[AuxiliaryDirectory, ~, ~] = fileparts(fullpath);
[ProjectFolderPath, ~, ~] = fileparts(AuxiliaryDirectory);
mesh = open(fullfile(ProjectFolderPath, 'Auxiliary
Variables\Base_Mesh_struct.mat'));
randRowIndex = randi(size(mesh.ForwardMesh.node, 1)); % Generate a
random row index
centroid = mesh.ForwardMesh.node(randRowIndex, :); % Select the random
row
radius = randi([8,15]);
[mesh_refined, tumor_nodes_idx] =
project_meshrefine(mesh.ForwardMesh, centroid, radius, 0.1);

current_datetime = now;
formattedDateTime = datestr(current_datetime, 'yyyy-mm-dd_HH-MM-SS');
formattedDateTime = strrep(formattedDateTime, '-', '_');
filename = strcat(formattedDateTime, '.msh');
msh_path = fullfile(ProjectFolderPath, 'MESH');
msh_path = fullfile(msh_path, filename);

conv_nodes_elements_arrays_to_msh(mesh_refined.node, mesh_refined.elem, msh
h_path, 4);

% Calculate the min and max of X, Y, Z in the mesh nodes
minX = min(mesh_refined.node(:,1));
maxX = max(mesh_refined.node(:,1));
minY = min(mesh_refined.node(:,2));
maxY = max(mesh_refined.node(:,2));
minZ = min(mesh_refined.node(:,3));
maxZ = max(mesh_refined.node(:,3));

% % Open mesh
mesh = toastMesh(msh_path, 'gmsh');

% Create the background parameters
mua_bkg = 0.01;
mus_bkg = 1.0;
ref_bkg = 1.4;
nnd = mesh.NodeCount;
mua = ones(nnd,1)*mua_bkg;
mus = ones(nnd,1)*mus_bkg;
ref = ones(nnd,1) * ref_bkg;

mua(tumor_nodes_idx) = 0.05;
mus(tumor_nodes_idx) = 2;
```

## DOT SIMULATOR – INSTALLATION AND SIMULATION INSTRUCTIONS

```
figure
mesh.Display(mua);
title('mua display')
figure
mesh.Display(mus);
title('mus display')
figure
mesh.Display;
title('mesh and source-detector display')

% Create the source and detector positions
z_delta = 0;
Q_z_pos = maxZ + z_delta; % Z position above the mesh
M_z_pos = minZ - z_delta; % Z position below the mesh
nqm = 5; % Number of sources and detectors in one dimension

Q = zeros(nqm^2, 3); % Initialize source positions
M = zeros(nqm^2, 3); % Initialize detector positions

index = 1; % Initialize a separate index for filling Q and M
for i = 1:nqm+1
    for j = 1:nqm+1
        x = minX + (maxX - minX)*(j-1)/nqm;
        y = minY + (maxY - minY)*(i-1)/nqm;

        % Assign calculated positions to Q and M using the separate
index
        Q(index,:) = [x, y, Q_z_pos];
        M(index,:) = [x, y, M_z_pos];

        % Increment the index after each assignment
        index = index + 1;
    end
end

mesh.SetQM(Q, M);

hold on; % Ensure that new plots are added to the existing figure

% Plot the source positions in red with marker 'o'
plot3(Q(:,1), Q(:,2), Q(:,3), 'ro', 'MarkerFaceColor', 'r');

% Label each source position with its index number
for idx = 1:size(Q, 1)
    text(Q(idx,1), Q(idx,2), Q(idx,3), num2str(idx),
'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right', 'Color',
'red');
end

% Plot the detector positions in blue with marker 's'
plot3(M(:,1), M(:,2), M(:,3), 'bs', 'MarkerFaceColor', 'b');

% Label each detector position with its index number
for idx = 1:size(M, 1)
    text(M(idx,1), M(idx,2), M(idx,3), num2str(idx),
'VerticalAlignment', 'top', 'HorizontalAlignment', 'left', 'Color',
'blue');
end
```

## DOT SIMULATOR – INSTALLATION AND SIMULATION INSTRUCTIONS

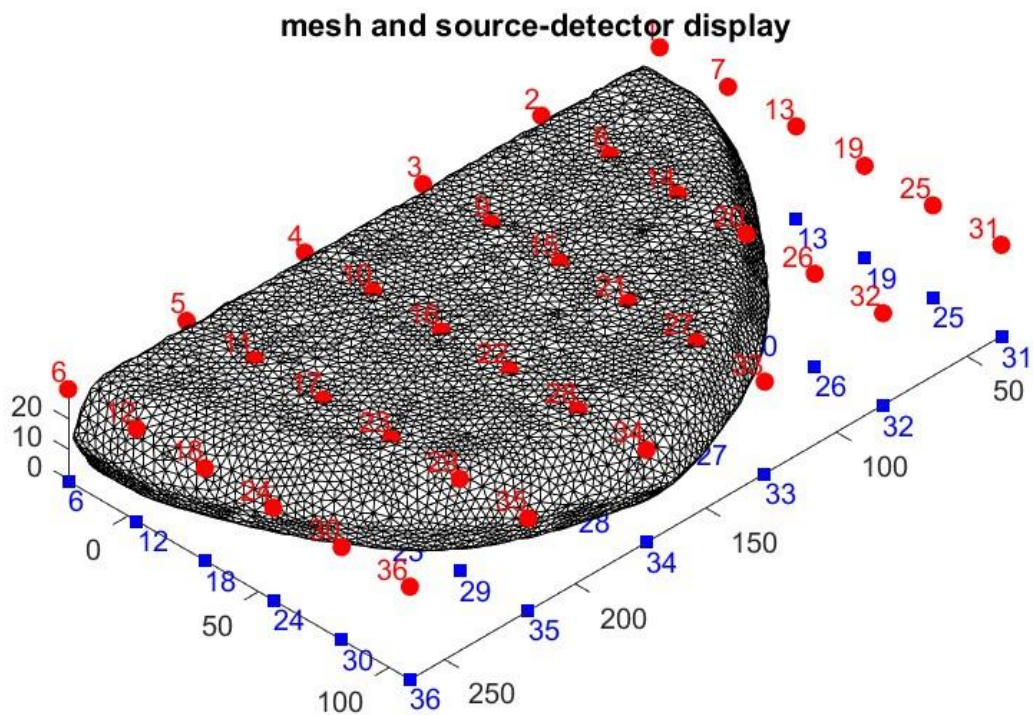
```
% Create the source and boundary projection vectors
qvec = real(mesh.Qvec('Neumann', 'Gaussian', 5));
mvec = real(mesh.Mvec('Gaussian', 2, ref_bkg));

% Solve the FEM linear system (Simulate DOT Scan)
K = dotSysmat(mesh,mua,mus,ref);
Phi = K\qvec;
Y = mvec.' * Phi;

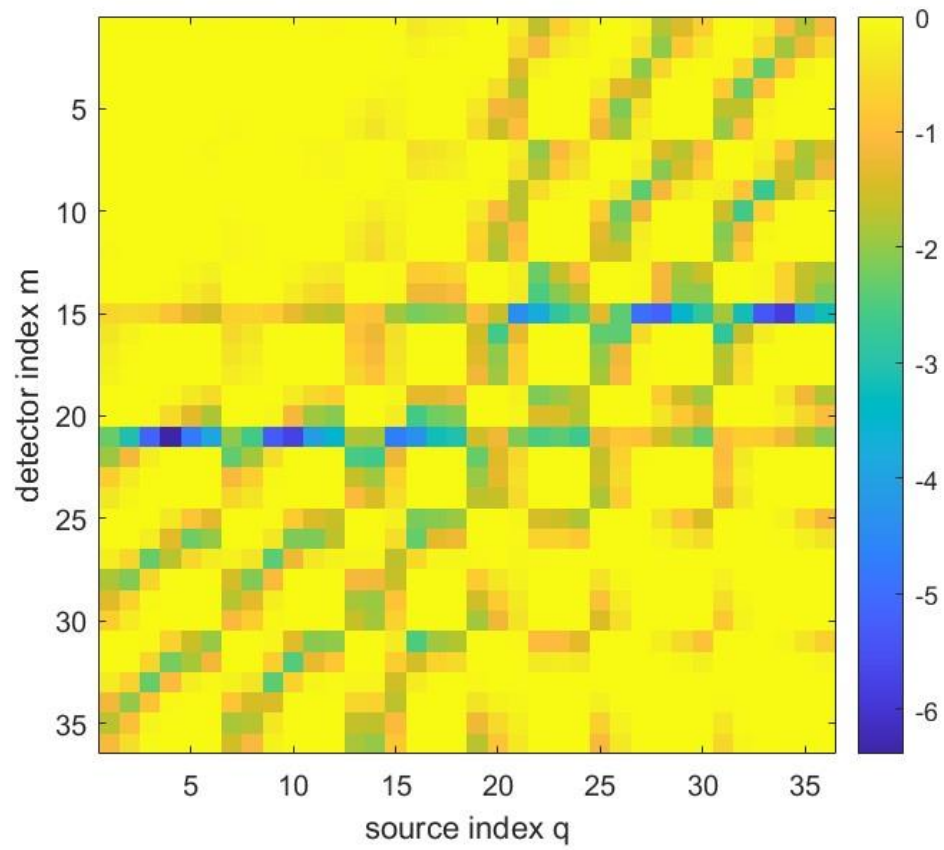
mua(tumor_nodes_idx) = mua_bkg;
mus(tumor_nodes_idx) = mus_bkg;
K = dotSysmat(mesh,mua,mus,ref);
Phi = K\qvec;
Y_homogeneous = mvec.' * Phi;

% Display sinogram
figure;
imagesc(log(Y) - log(Y_homogeneous));
xlabel('source index q');
ylabel('detector index m');
axis equal tight;
colorbar;
```

after running the function, figures of generated structure and scan results are shown. Additionally, the generated structure is saved as an 'msh' in the 'MESH' directory.



## DOT SIMULATOR – INSTALLATION AND SIMULATION INSTRUCTIONS



## DOT SIMULATOR – INSTALLATION AND SIMULATION INSTRUCTIONS