

# Using an External Optimization Algorithm for PET Reconstruction with STIR in MATLAB

Yu-Jung Tsai, Matthias J. Ehrhardt and Kris Thielemans



## Example : LBFGS-B (limited-memory BFGS with boundary constraints)

- » A popular Quasi-Newton method
- » Requires only the objective function and its gradient
- » Used when the true inverse of Hessian is too complicated to handle or does not exist
- » C code implementation with MATLAB mex wrapper is available at: <https://github.com/stephenbecker/L-BFGS-B-C>

## Framework : RunSimulation ( )

```
%% Set STIR parameters
```

```
param = SetParameters();
```

```
%% Set STIR Image/Data descriptions
```

```
desp = SetDescriptions(param);
```

```
%% Create data
```

```
data = CreateData(param, desp);
```

```
%% Problem to solve (minimize)
```

```
fun = @(x)Problem2Minimize(x, param, desp, data);
```

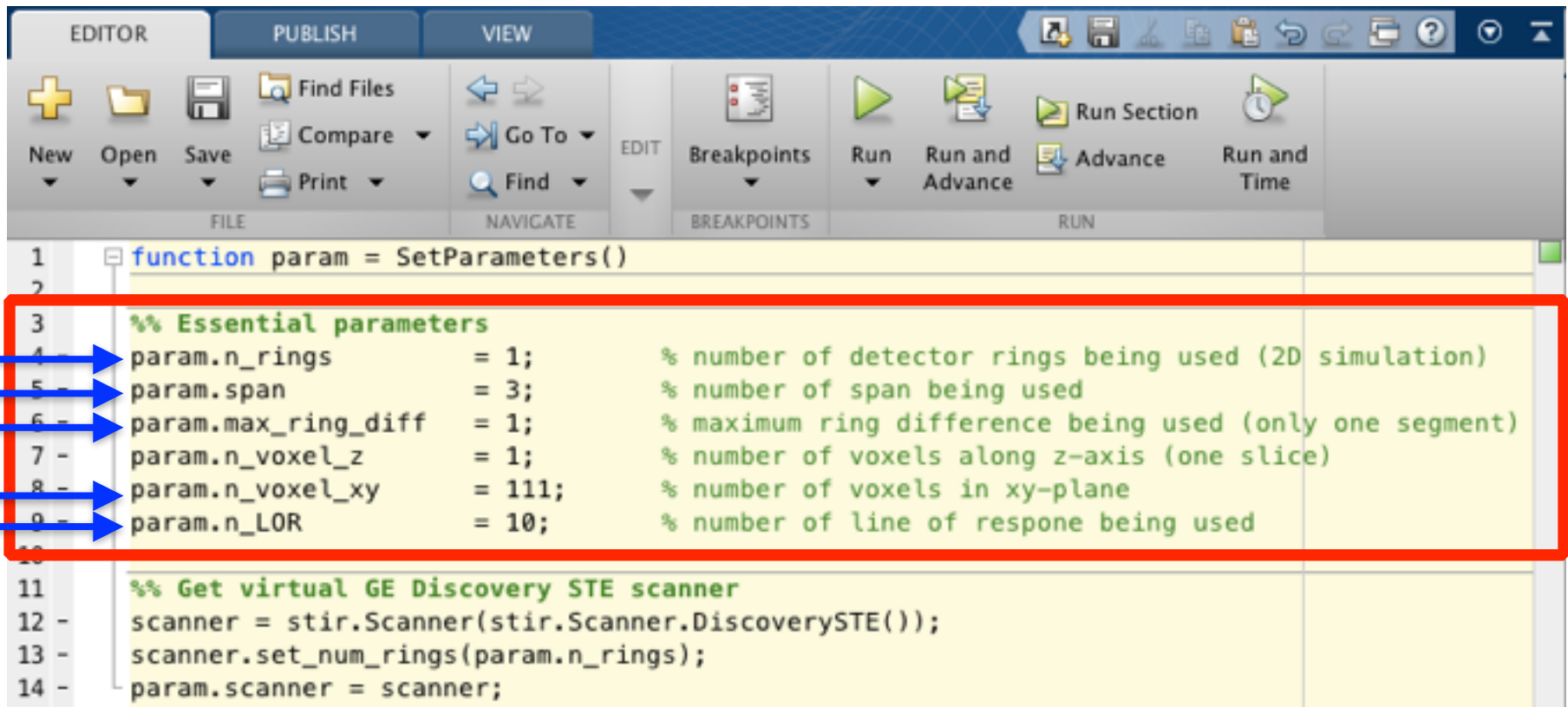
```
%% Set LBFGS-B options and execute LBFGS-B
```

```
[x, info] = RunLBFGS_B(param, fun);
```



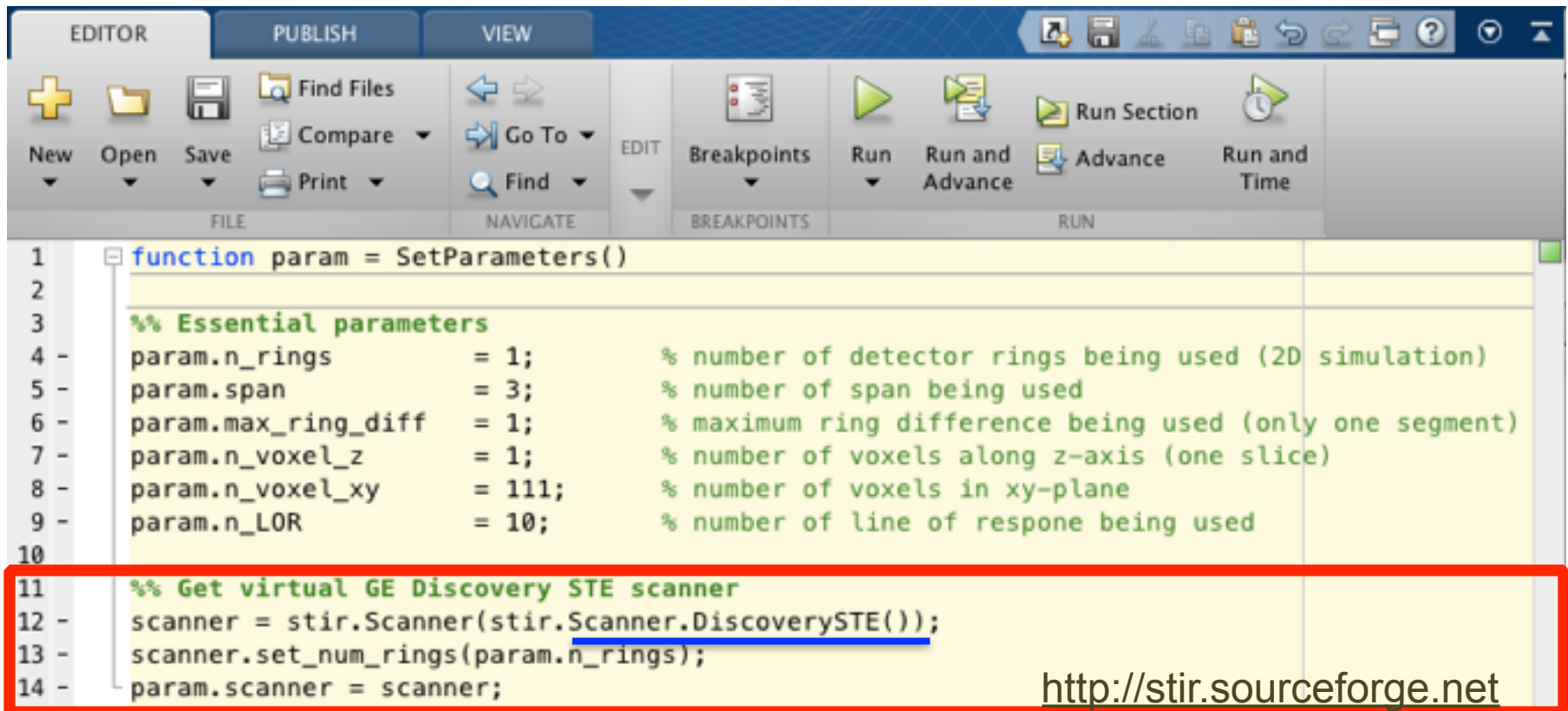
```
[function_value, gradient] = fun(x)
```

# Set STIR Parameters : Parameters



```
1 function param = SetParameters()
2
3 %% Essential parameters
4 param.n_rings = 1; % number of detector rings being used (2D simulation)
5 param.span = 3; % number of span being used
6 param.max_ring_diff = 1; % maximum ring difference being used (only one segment)
7 param.n_voxel_z = 1; % number of voxels along z-axis (one slice)
8 param.n_voxel_xy = 111; % number of voxels in xy-plane
9 param.n_LOR = 10; % number of line of response being used
10
11 %% Get virtual GE Discovery STE scanner
12 scanner = stir.Scanner(stir.Scanner.DiscoverySTE());
13 scanner.set_num_rings(param.n_rings);
14 param.scanner = scanner;
```

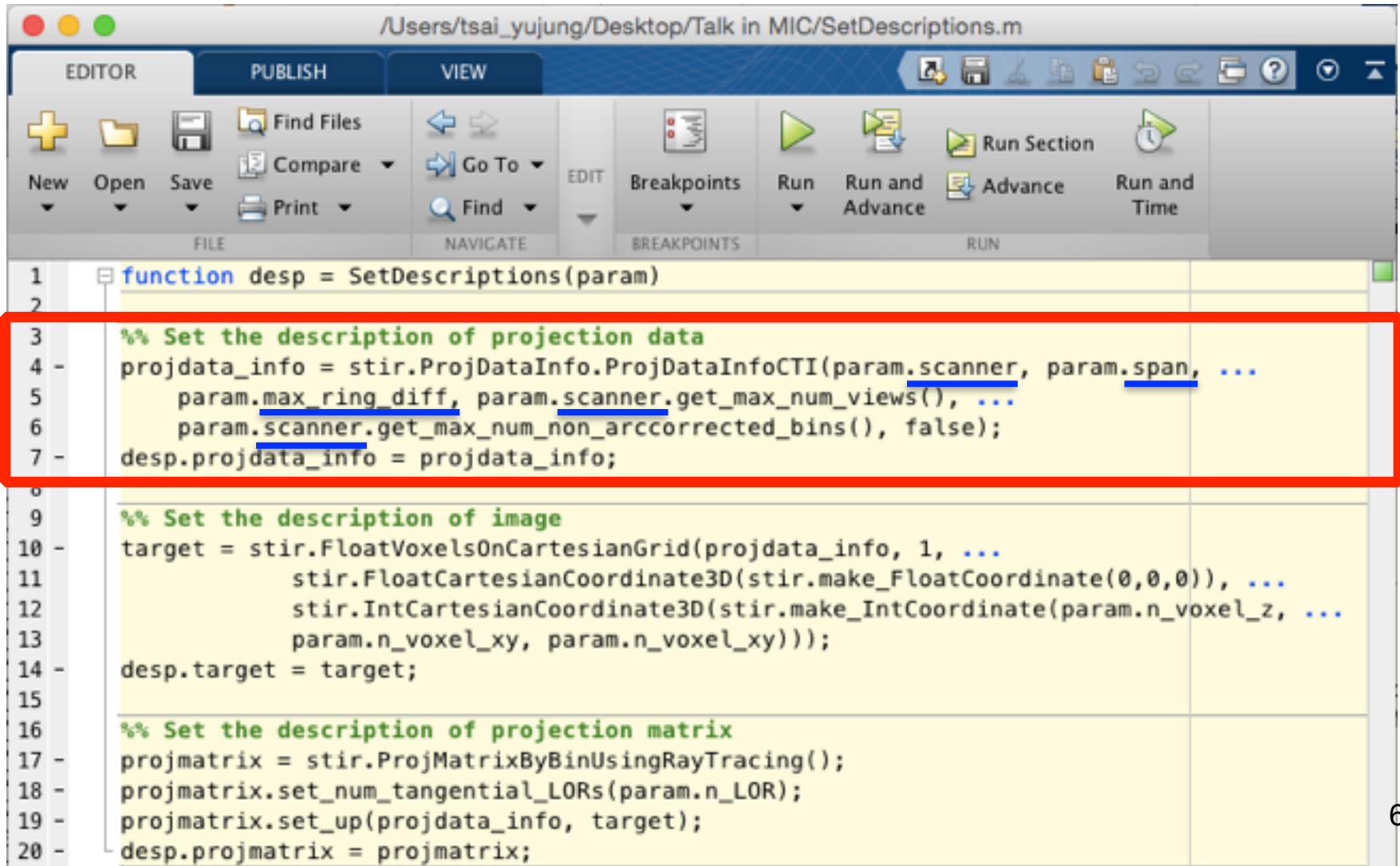
# Set STIR Parameters : Scanner



```
1 function param = SetParameters()
2
3 %% Essential parameters
4 - param.n_rings      = 1;      % number of detector rings being used (2D simulation)
5 - param.span        = 3;      % number of span being used
6 - param.max_ring_diff = 1;    % maximum ring difference being used (only one segment)
7 - param.n_voxel_z   = 1;      % number of voxels along z-axis (one slice)
8 - param.n_voxel_xy  = 111;    % number of voxels in xy-plane
9 - param.n_LOR       = 10;     % number of line of response being used
10
11 %% Get virtual GE Discovery STE scanner
12 - scanner = stir.Scanner(stir.Scanner.DiscoverySTE());
13 - scanner.set_num_rings(param.n_rings);
14 - param.scanner = scanner;
```

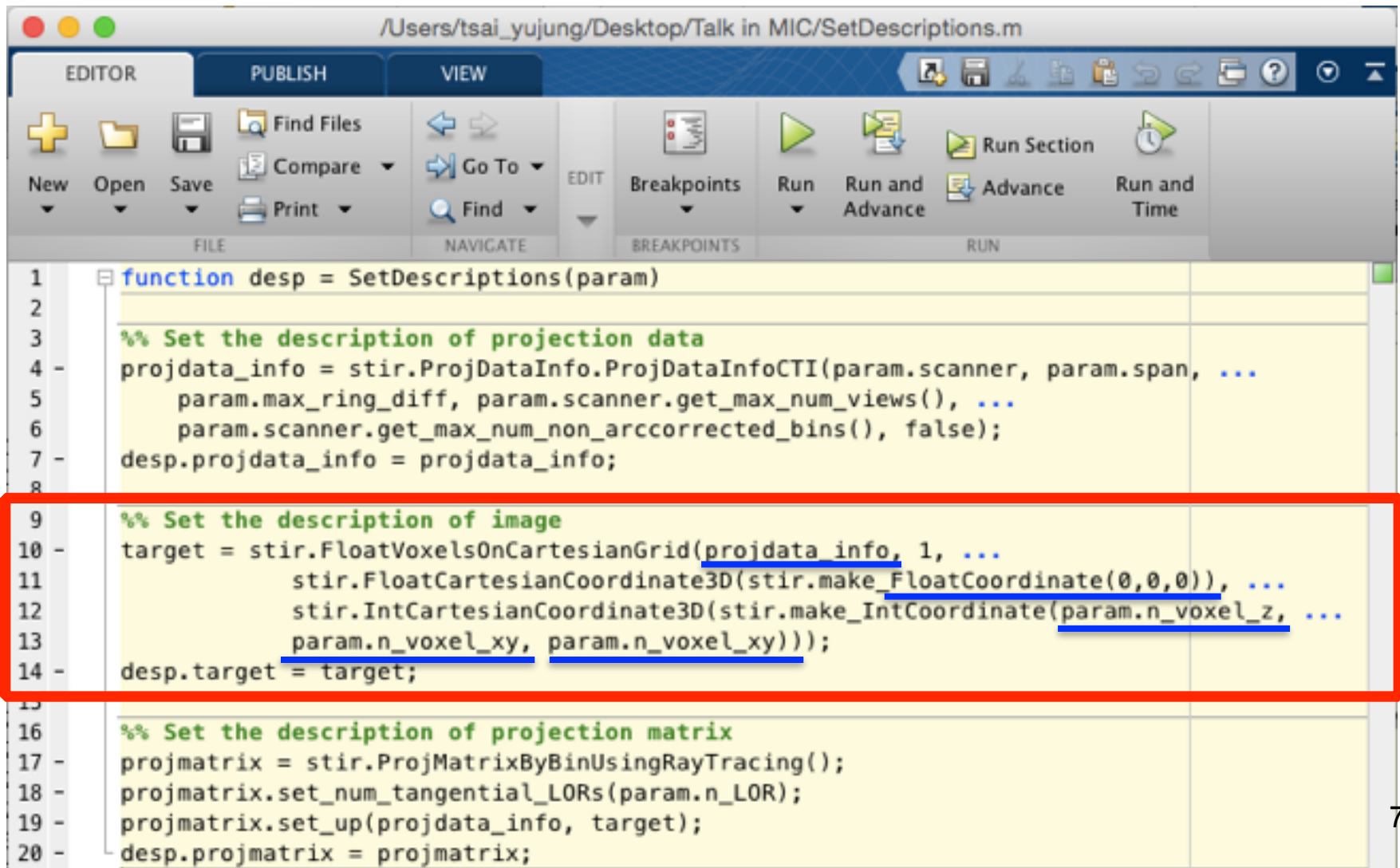
<http://stir.sourceforge.net>

# Set STIR Descriptions : Data



```
1 function desp = SetDescriptions(param)
2
3 %% Set the description of projection data
4 - projdata_info = stir.ProjDataInfo.ProjDataInfoCTI(param.scanner, param.span, ...
5   param.max_ring_diff, param.scanner.get_max_num_views(), ...
6   param.scanner.get_max_num_non_arccorrected_bins(), false);
7 - desp.projdata_info = projdata_info;
8
9 %% Set the description of image
10 - target = stir.FloatVoxelsOnCartesianGrid(projdata_info, 1, ...
11   stir.FloatCartesianCoordinate3D(stir.make_FloatCoordinate(0,0,0)), ...
12   stir.IntCartesianCoordinate3D(stir.make_IntCoordinate(param.n_voxel_z, ...
13   param.n_voxel_xy, param.n_voxel_xy)));
14 - desp.target = target;
15
16 %% Set the description of projection matrix
17 - projmatrix = stir.ProjMatrixByBinUsingRayTracing();
18 - projmatrix.set_num_tangential_LORs(param.n_LOR);
19 - projmatrix.set_up(projdata_info, target);
20 - desp.projmatrix = projmatrix;
```

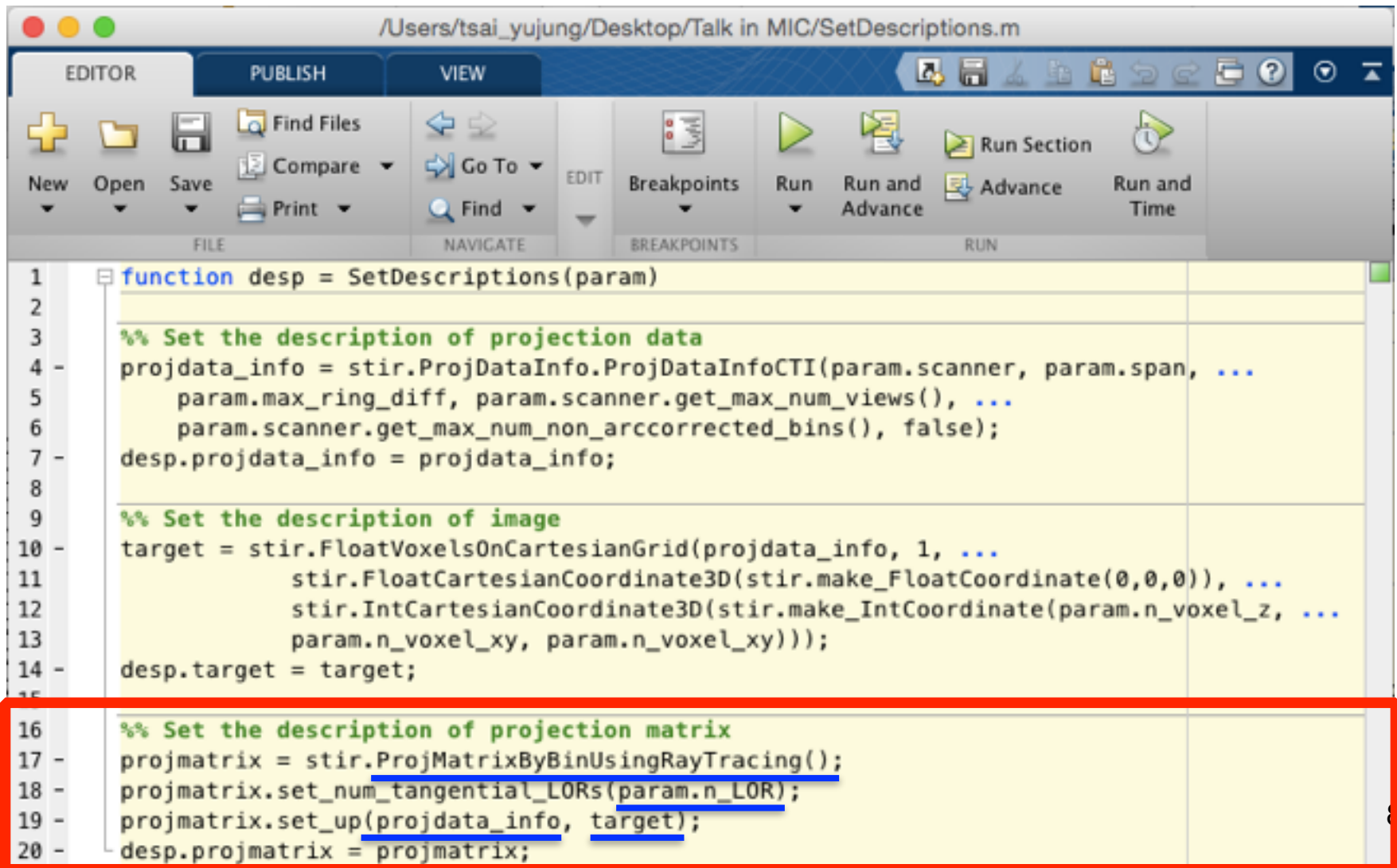
# Set STIR Descriptions : Image



```
1 function desp = SetDescriptions(param)
2
3 %% Set the description of projection data
4 - projdata_info = stir.ProjDataInfo.ProjDataInfoCTI(param.scanner, param.span, ...
5     param.max_ring_diff, param.scanner.get_max_num_views(), ...
6     param.scanner.get_max_num_non_arccorrected_bins(), false);
7 - desp.projdata_info = projdata_info;
8
9 %% Set the description of image
10 - target = stir.FloatVoxelsOnCartesianGrid(projdata_info, 1, ...
11     stir.FloatCartesianCoordinate3D(stir.make_FloatCoordinate(0,0,0)), ...
12     stir.IntCartesianCoordinate3D(stir.make_IntCoordinate(param.n_voxel_z, ...
13     param.n_voxel_xy, param.n_voxel_xy)));
14 - desp.target = target;
15
16 %% Set the description of projection matrix
17 - projmatrix = stir.ProjMatrixByBinUsingRayTracing();
18 - projmatrix.set_num_tangential_LORs(param.n_LOR);
19 - projmatrix.set_up(projdata_info, target);
20 - desp.projmatrix = projmatrix;
```



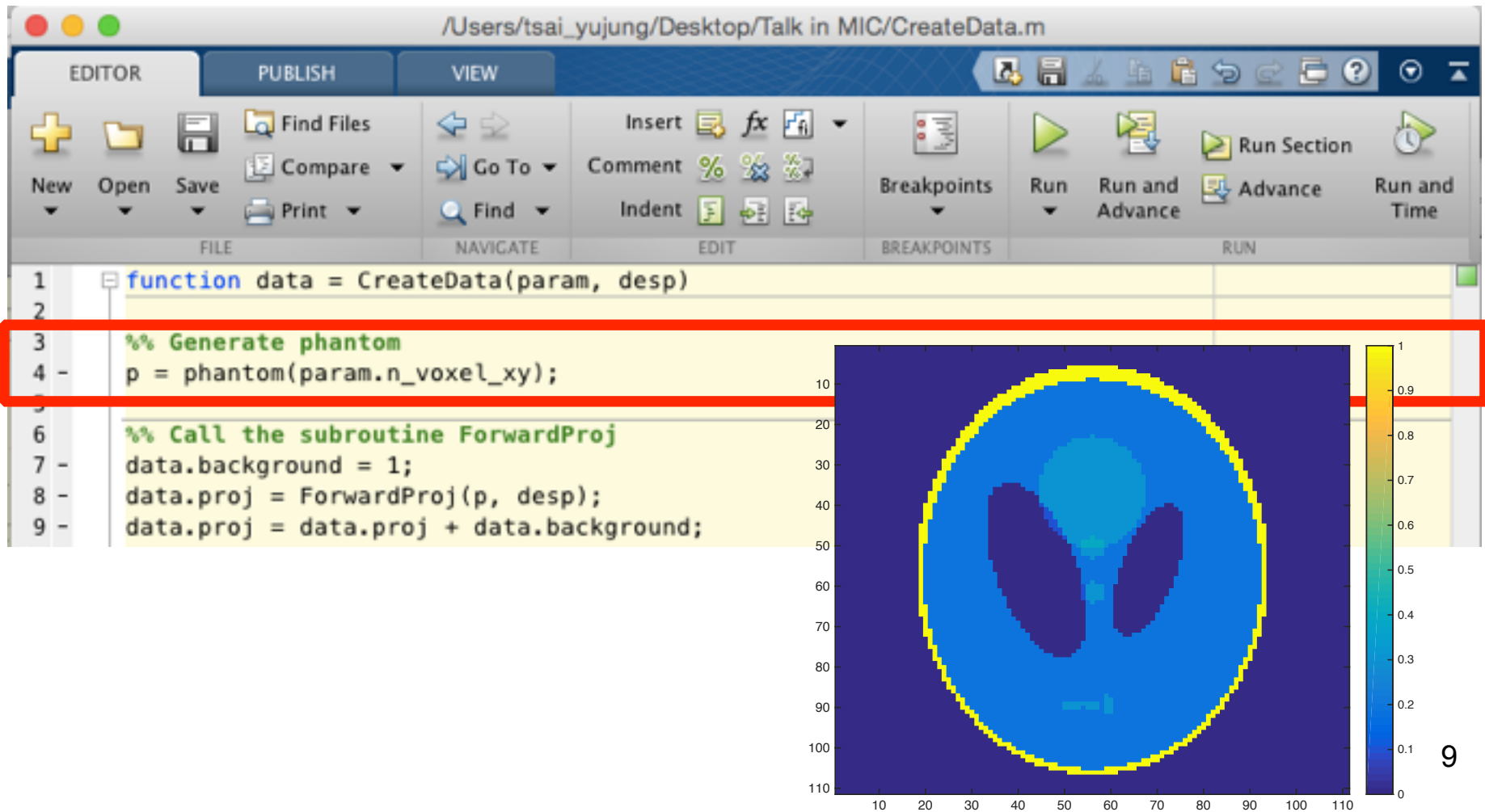
# Set STIR Descriptions : Projection Matrix



```
1 function desp = SetDescriptions(param)
2
3 %% Set the description of projection data
4 - projdata_info = stir.ProjDataInfo.ProjDataInfoCTI(param.scanner, param.span, ...
5     param.max_ring_diff, param.scanner.get_max_num_views(), ...
6     param.scanner.get_max_num_non_arccorrected_bins(), false);
7 - desp.projdata_info = projdata_info;
8
9 %% Set the description of image
10 - target = stir.FloatVoxelsOnCartesianGrid(projdata_info, 1, ...
11     stir.FloatCartesianCoordinate3D(stir.make_FloatCoordinate(0,0,0)), ...
12     stir.IntCartesianCoordinate3D(stir.make_IntCoordinate(param.n_voxel_z, ...
13     param.n_voxel_xy, param.n_voxel_xy)));
14 - desp.target = target;
15
16 %% Set the description of projection matrix
17 - projmatrix = stir.ProjMatrixByBinUsingRayTracing();
18 - projmatrix.set_num_tangential_LORs(param.n_LOR);
19 - projmatrix.set_up(projdata_info, target);
20 - desp.projmatrix = projmatrix;
```



# Create Data : Phantom

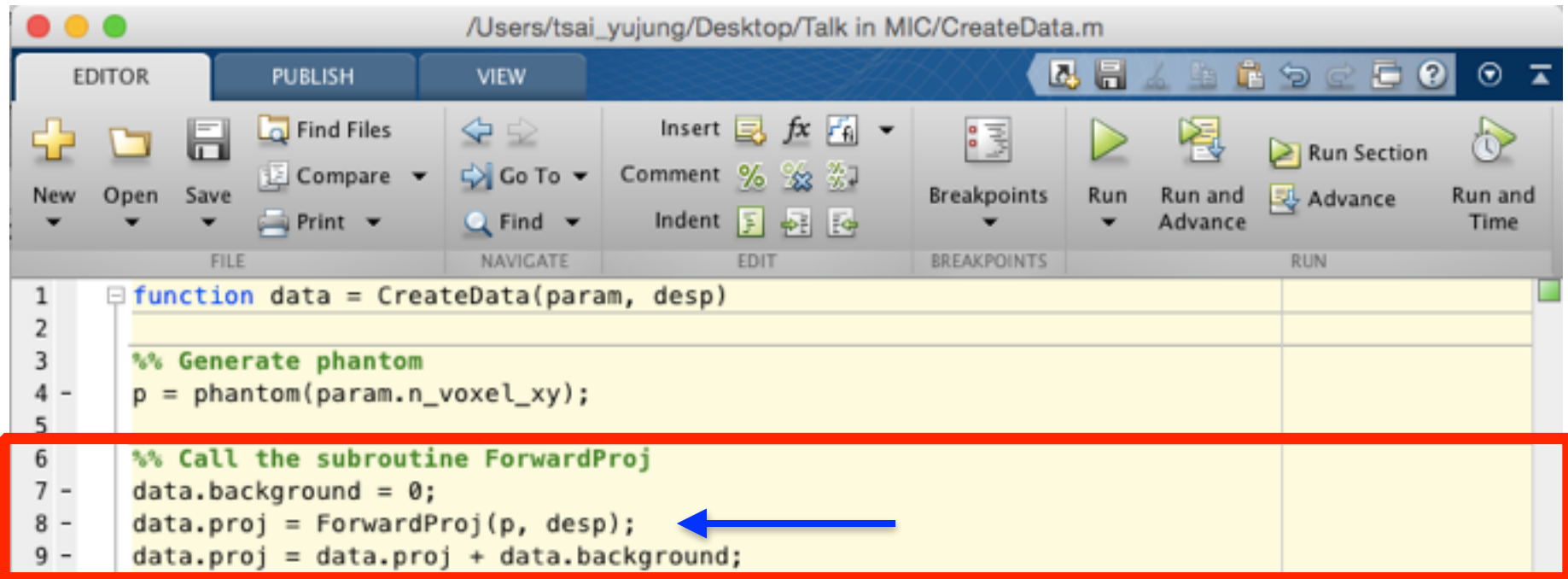


The screenshot displays the MATLAB environment with a script titled `CreateData.m`. The script defines a function `data = CreateData(param, desp)` that generates a phantom image. The code is as follows:

```
1 function data = CreateData(param, desp)
2
3 %% Generate phantom
4 - p = phantom(param.n_voxel_xy);
5
6 %% Call the subroutine ForwardProj
7 - data.background = 1;
8 - data.proj = ForwardProj(p, desp);
9 - data.proj = data.proj + data.background;
```

The visualization on the right is a heatmap of the phantom image. The x and y axes both range from 0 to 110. A color bar on the right indicates intensity values from 0 (dark blue) to 1 (yellow). The phantom image shows a central circular region with a complex internal structure, surrounded by a bright yellow ring.

# Create Data : Projection Data



The screenshot shows a MATLAB editor window with the following code:

```
1 function data = CreateData(param, desp)
2
3 %% Generate phantom
4 - p = phantom(param.n_voxel_xy);
5
6 %% Call the subroutine ForwardProj
7 - data.background = 0;
8 - data.proj = ForwardProj(p, desp);
9 - data.proj = data.proj + data.background;
```

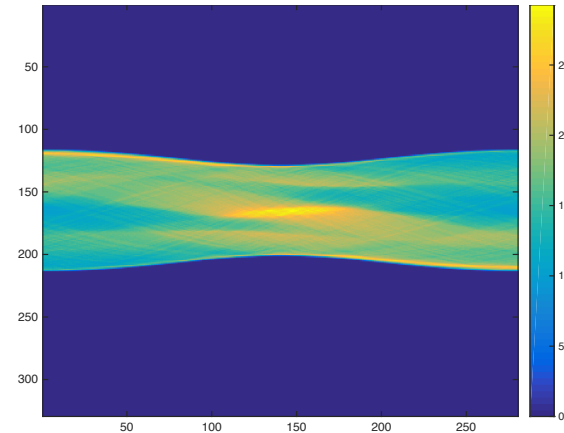
A red rectangular box highlights lines 6 through 9. A blue arrow points to the right-hand side of line 8.

# Create Data : Forward Projector

```

/Users/tsai_yujung/Desktop/Talk in MIC/CreateData.m
EDITOR PUBLISH VIEW
New Open Save Find Files Compare Go To Comment % fx Insert Breakpoints Run Run and Advance Run Section Run and Time
FILE NAVIGATE EDIT BREAKPOINTS RUN
1 function data = CreateData(param, desp)
2
3 %% Generate phantom
4 p = phantom(param.n_voxel_xy);
5
6 %% Call the subroutine ForwardProj
7 data.background = 0;
8 data.proj = ForwardProj(p, desp);
9 data.proj = data.proj + data.background;

```



```

11 %% Subroutine : create forward projector and project image
12 function x_forward = ForwardProj(x, desp)
13
14 forwardprojector = stir.ForwardProjectorByBinUsingProjMatrixByBin(desp.projmatrix);
15 projdata = stir.ProjDataInMemory(stir.ExamInfo(), desp.projdata_info);
16 desp.target.fill(x);
17 forwardprojector.forward_project(projdata, desp.target);
18 x_forward = to_matlab(projdata);
19 end

```

# Problem to Solve : Objective Function

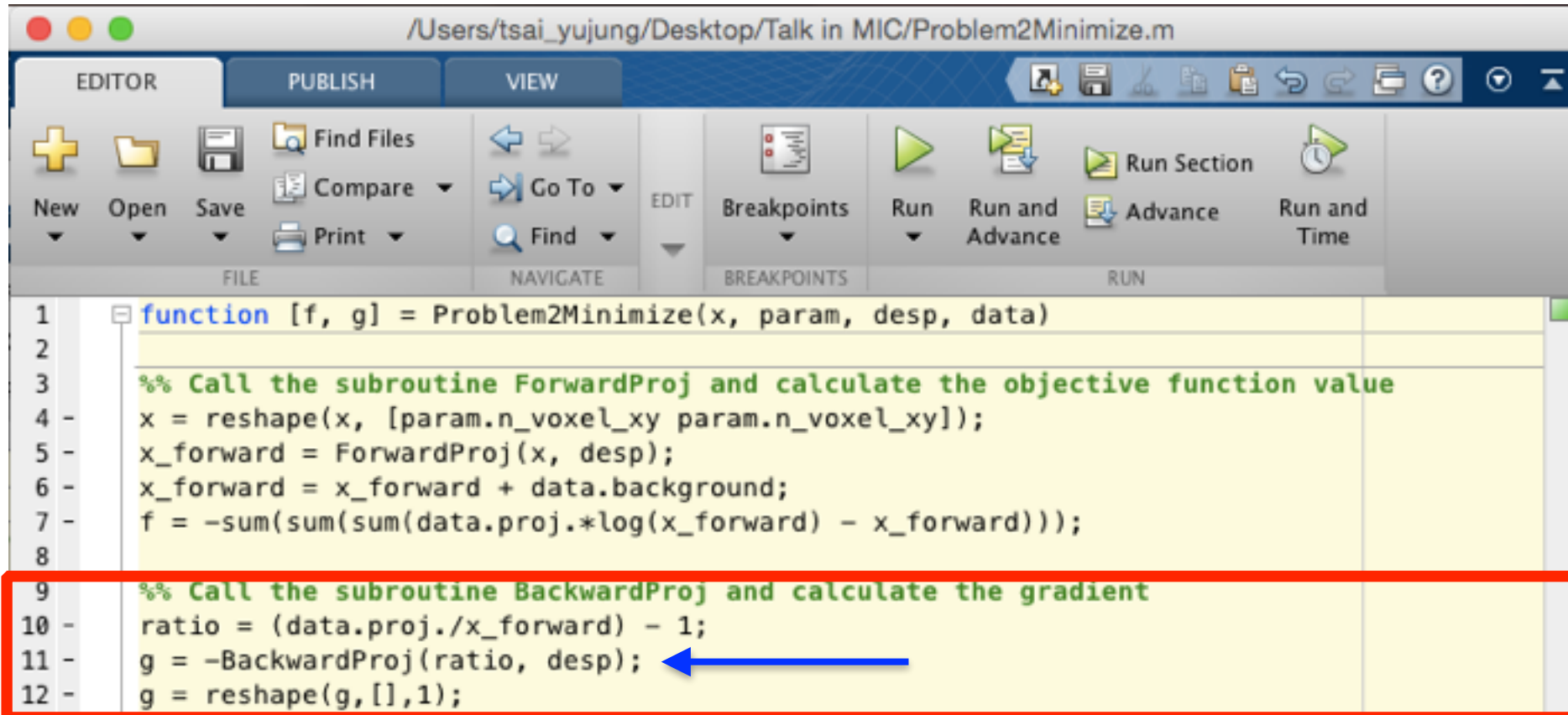
```

1 function [f, g] = Problem2Minimize(x, param, desp, data)
2
3 %% Call the subroutine ForwardProj and calculate the objective function value
4 x = reshape(x, [param.n_voxel_xy param.n_voxel_xy]);
5 x_forward = ForwardProj(x, desp);
6 x_forward = x_forward + data.background;
7 f = -sum(sum(sum(data.proj.*log(x_forward) - x_forward)));
8
9 %% Call the subroutine BackwardProj and calculate the gradient
10 ratio = (data.proj./x_forward) - 1;
11 g = -BackwardProj(ratio, desp);
12 g = reshape(g, [], 1);

```

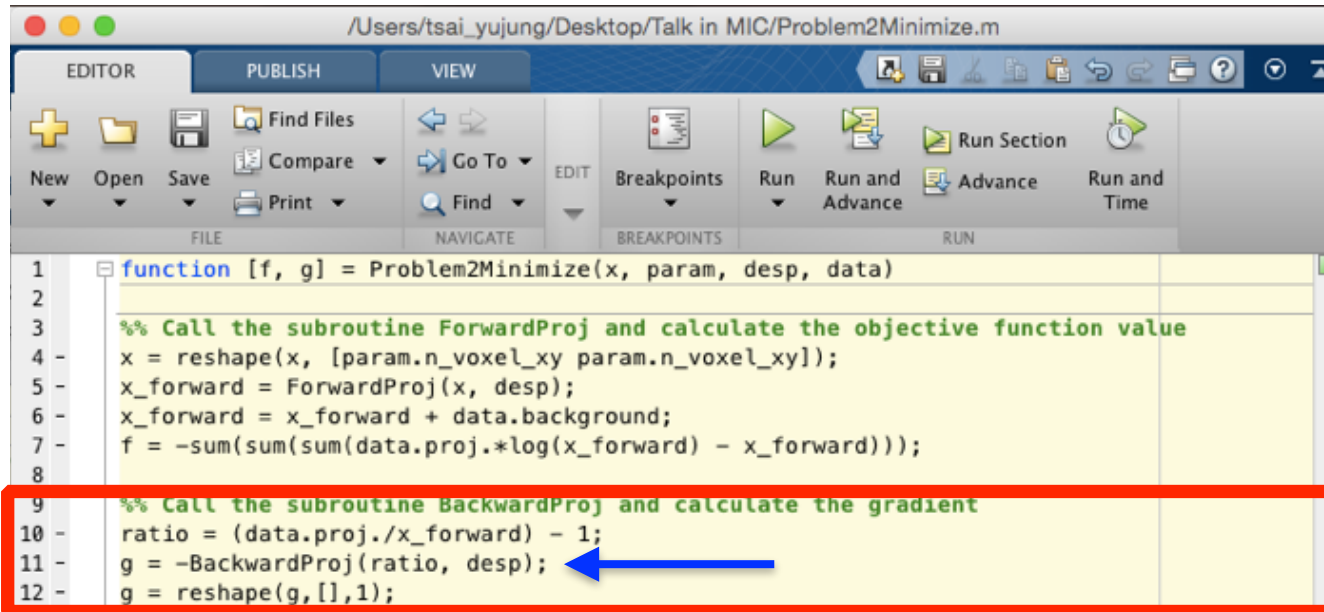
$$-data \cdot \log\{ \text{forward\_prj}(x) + \text{background} \} + \{ \text{forward\_prj}(x) + \text{background} \}$$

# Problem to Solve : Gradient

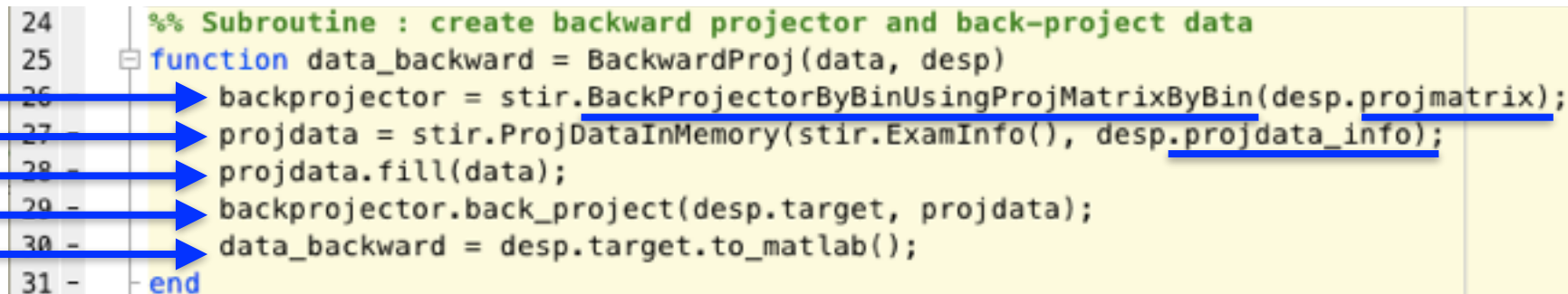


```
1 function [f, g] = Problem2Minimize(x, param, desp, data)
2
3 %% Call the subroutine ForwardProj and calculate the objective function value
4 x = reshape(x, [param.n_voxel_xy param.n_voxel_xy]);
5 x_forward = ForwardProj(x, desp);
6 x_forward = x_forward + data.background;
7 f = -sum(sum(sum(data.proj.*log(x_forward) - x_forward)));
8
9 %% Call the subroutine BackwardProj and calculate the gradient
10 ratio = (data.proj./x_forward) - 1;
11 g = -BackwardProj(ratio, desp);
12 g = reshape(g, [], 1);
```

# Problem to Solve : Backward Projector



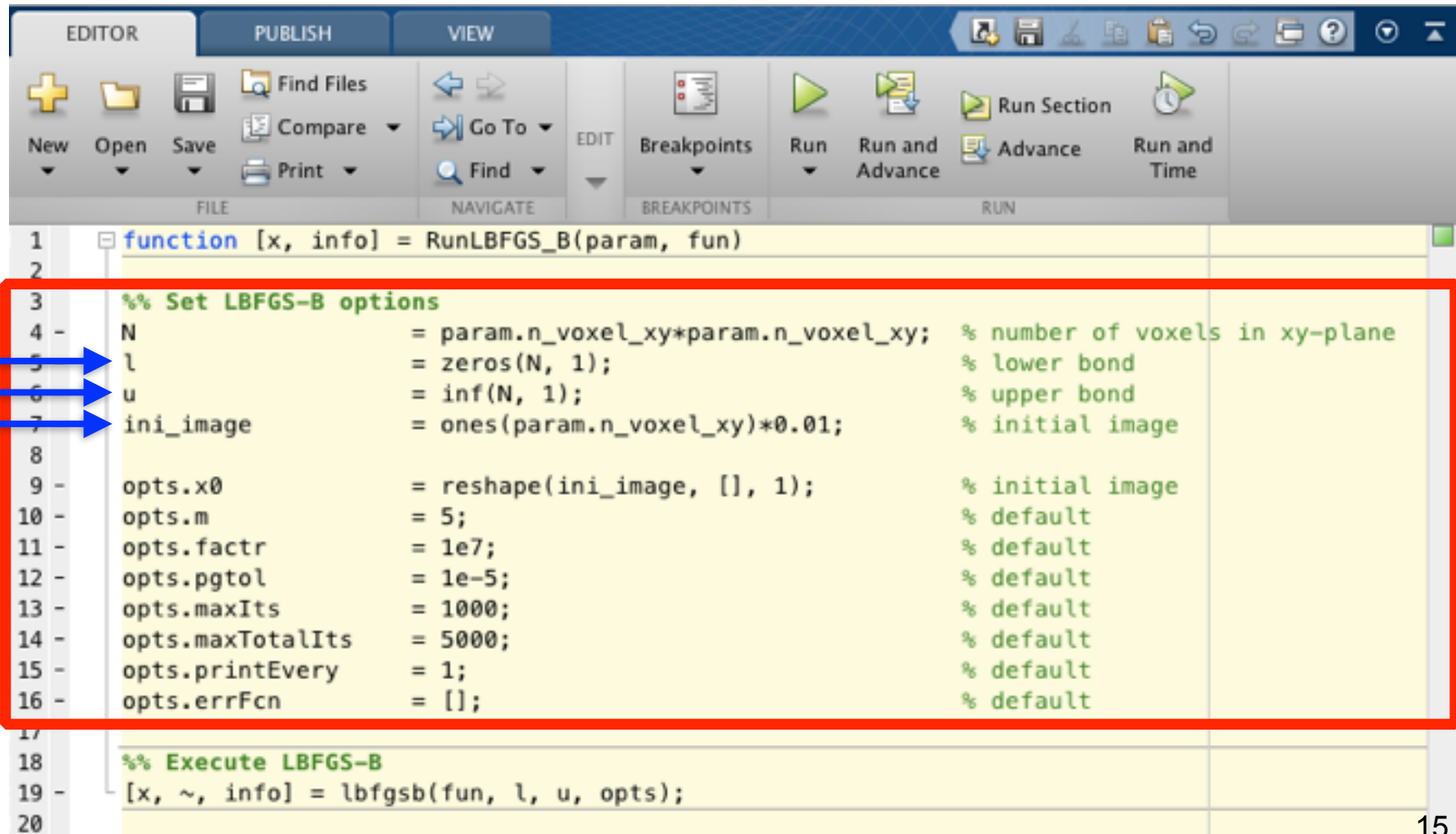
```
1 function [f, g] = Problem2Minimize(x, param, desp, data)
2
3 %% Call the subroutine ForwardProj and calculate the objective function value
4 x = reshape(x, [param.n_voxel_xy param.n_voxel_xy]);
5 x_forward = ForwardProj(x, desp);
6 x_forward = x_forward + data.background;
7 f = -sum(sum(sum(data.proj.*log(x_forward) - x_forward)));
8
9 %% Call the subroutine BackwardProj and calculate the gradient
10 ratio = (data.proj./x_forward) - 1;
11 g = -BackwardProj(ratio, desp);
12 g = reshape(g,[],1);
```



```
24 %% Subroutine : create backward projector and back-project data
25 function data_backward = BackwardProj(data, desp)
26 backprojector = stir.BackProjectorByBinUsingProjMatrixByBin(desp.projmatrix);
27 projdata = stir.ProjDataInMemory(stir.ExamInfo(), desp.projdata_info);
28 projdata.fill(data);
29 backprojector.back_project(desp.target, projdata);
30 data_backward = desp.target.to_matlab();
31 end
```



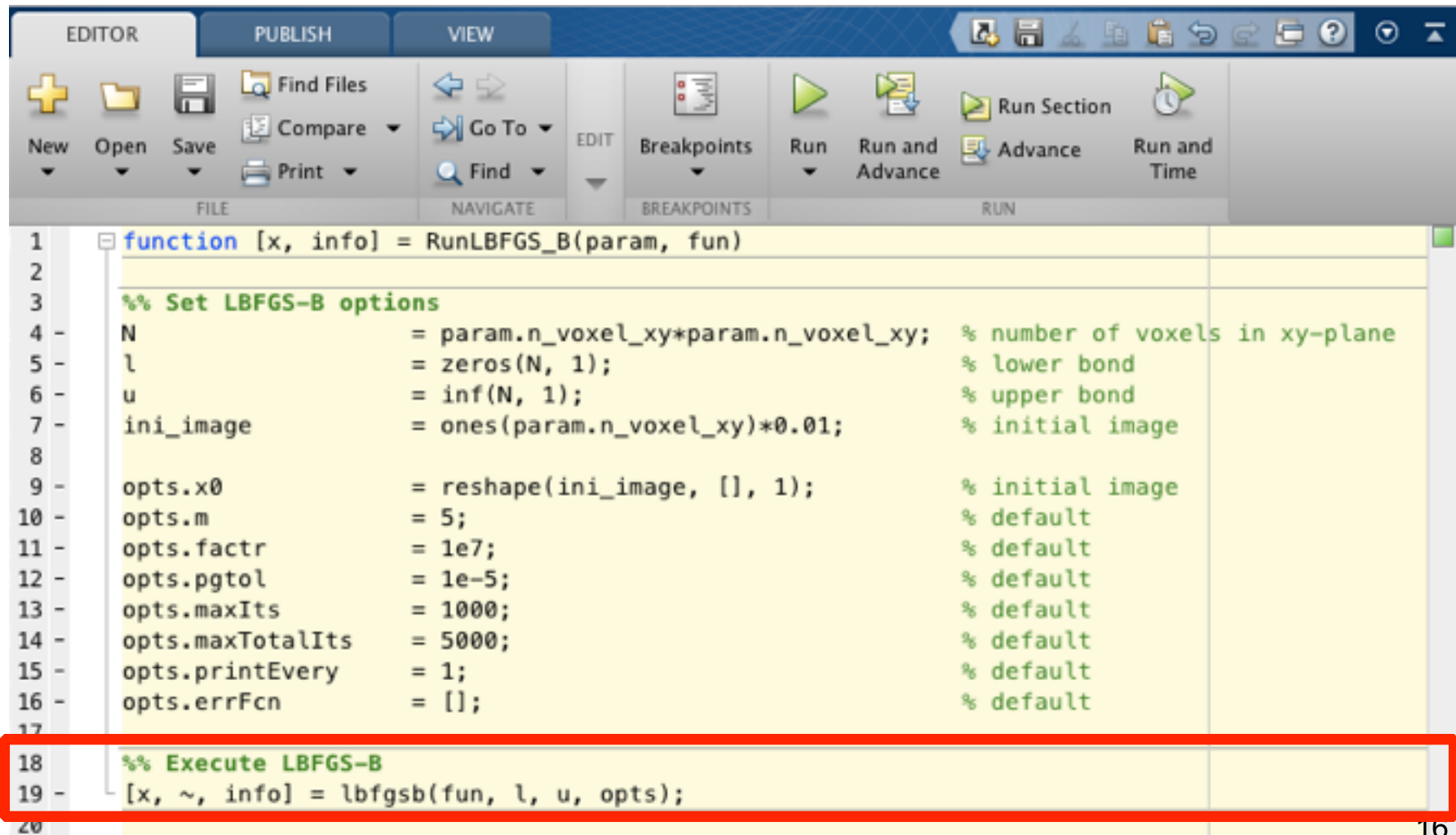
# LBFGS-B : Set Options



```
1 function [x, info] = RunLBFGS_B(param, fun)
2
3 %% Set LBFGS-B options
4 - N = param.n_voxel_xy*param.n_voxel_xy; % number of voxels in xy-plane
5 - l = zeros(N, 1); % lower bond
6 - u = inf(N, 1); % upper bond
7 - ini_image = ones(param.n_voxel_xy)*0.01; % initial image
8
9 - opts.x0 = reshape(ini_image, [], 1); % initial image
10 - opts.m = 5; % default
11 - opts.factr = 1e7; % default
12 - opts.pgtol = 1e-5; % default
13 - opts.maxIts = 1000; % default
14 - opts.maxTotalIts = 5000; % default
15 - opts.printEvery = 1; % default
16 - opts.errFcn = []; % default
17
18 %% Execute LBFGS-B
19 - [x, ~, info] = lbfgsb(fun, l, u, opts);
20
```



# LBFGS-B : Execute

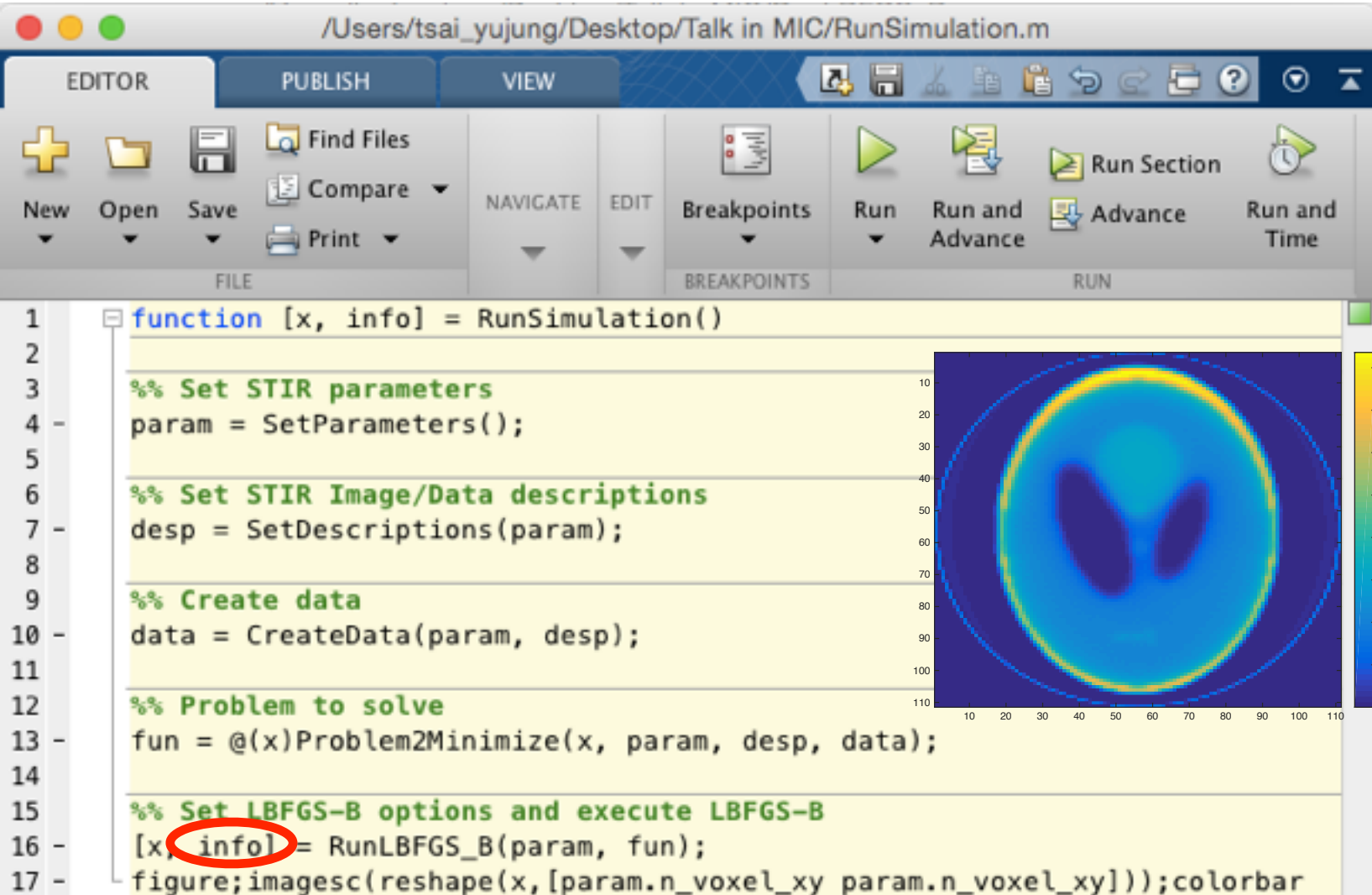


The screenshot shows the MATLAB editor interface with the following tabs: EDITOR, PUBLISH, and VIEW. The toolbar includes icons for New, Open, Save, Find Files, Compare, Print, Go To, Find, Breakpoints, Run, Run and Advance, Run Section, Advance, and Run and Time. The code in the editor is as follows:

```
1 function [x, info] = RunLBFGS_B(param, fun)
2
3 %% Set LBFGS-B options
4 - N = param.n_voxel_xy*param.n_voxel_xy; % number of voxels in xy-plane
5 - l = zeros(N, 1); % lower bond
6 - u = inf(N, 1); % upper bond
7 - ini_image = ones(param.n_voxel_xy)*0.01; % initial image
8
9 - opts.x0 = reshape(ini_image, [], 1); % initial image
10 - opts.m = 5; % default
11 - opts.factr = 1e7; % default
12 - opts.pgtol = 1e-5; % default
13 - opts.maxIts = 1000; % default
14 - opts.maxTotalIts = 5000; % default
15 - opts.printEvery = 1; % default
16 - opts.errFcn = []; % default
17
18 %% Execute LBFGS-B
19 - [x, ~, info] = lbfgsb(fun, l, u, opts);
20
```

The lines 18 and 19 are highlighted with a red box, indicating the execution of the LBFGS-B function.

# Run Simulation



The screenshot displays the MATLAB environment with the script `RunSimulation.m` open. The script defines a function `RunSimulation()` that performs the following steps:

- 1. `function [x, info] = RunSimulation()`
- 2. `%% Set STIR parameters`
- 3. `param = SetParameters();`
- 4. `%% Set STIR Image/Data descriptions`
- 5. `desp = SetDescriptions(param);`
- 6. `%% Create data`
- 7. `data = CreateData(param, desp);`
- 8. `%% Problem to solve`
- 9. `fun = @(x)Problem2Minimize(x, param, desp, data);`
- 10. `%% Set LBFGS-B options and execute LBFGS-B`
- 11. `[x, info] = RunLBFGS_B(param, fun);`
- 12. `figure; imagesc(reshape(x, [param.n_voxel_xy param.n_voxel_xy])); colorbar`

The visualization on the right is a 2D heatmap showing a cross-section of a brain. The color scale ranges from 0 (dark blue) to 0.8 (yellow). The brain structure is clearly visible, with the two hemispheres and the central region highlighted in yellow and orange.

**For more results ...**

## **Performance Evaluation of MAP Algorithms with Different Penalties, Object Geometries and Noise Levels**

<sup>1</sup>Yu-Jung Tsai, <sup>1</sup>Alexandre Bousse, <sup>2</sup>Matthias J. Ehrhardt, <sup>1\*</sup>Brian Hutton, <sup>2</sup>Simon Arridge, <sup>1</sup>Kris Thielemans

<sup>1</sup>Institute of Nuclear Medicine, University College London, UK  
<sup>2</sup>Department of Computer Science, University College London, UK

# Conclusion

- » Calling STIR from MATLAB makes the library more flexible
- » STIR in MATLAB is still in progress
  - » Need more work to prevent crash (swig, C++, MATLAB)