

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)

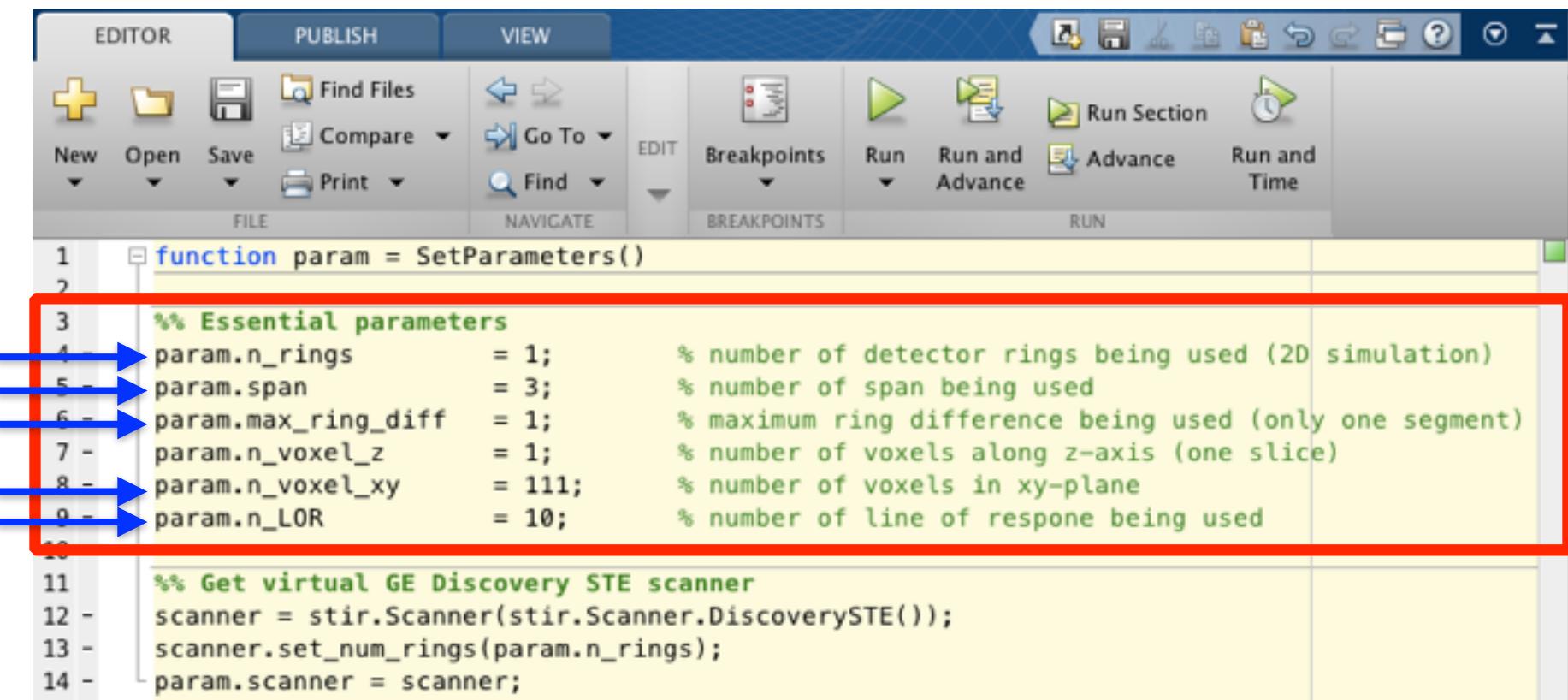
- » A popular Quasi-Newton method
- » Requires only the objective function and its gradient
- » Used when the true inverse of Hessian is too complicated to handle or does not exist
- » C code implementation with MATLAB mex wrapper is available at: <https://github.com/stephenbeckr/L-BFGS-B-C>

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



The screenshot shows a MATLAB IDE interface with a menu bar (EDITOR, PUBLISH, VIEW) and a toolbar (New, Open, Save, Find Files, Compare, Print, Go To, Find, Breakpoints, Run, Run and Advance, Run and Time). The central workspace displays a MATLAB script named 'SetParameters.m'.

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

A red box highlights the parameter assignment section (lines 3-9), and blue arrows point to each assignment line from the left margin.

Set STIR Parameters : Scanner

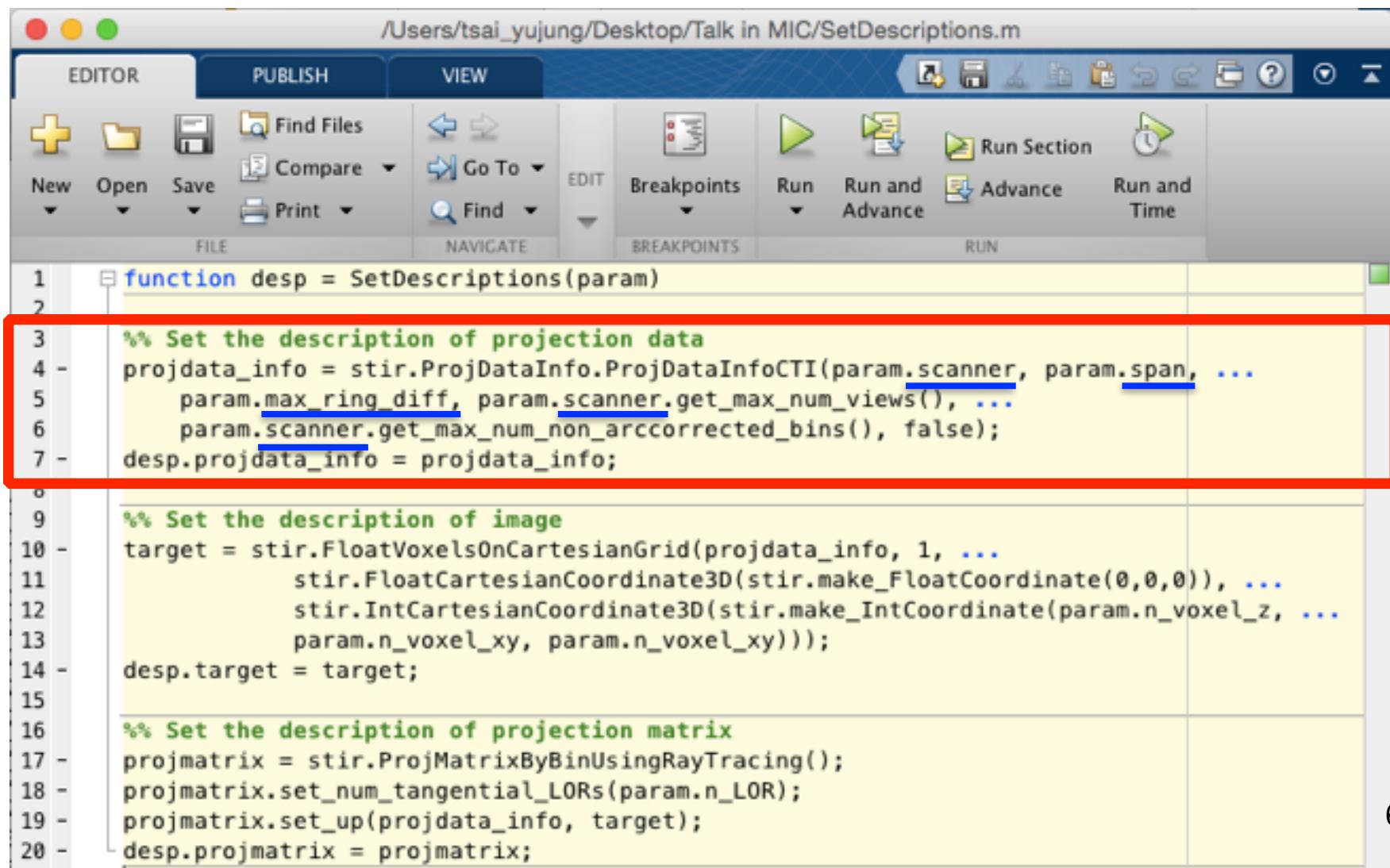
The screenshot shows a software interface with a menu bar (EDITOR, PUBLISH, VIEW) and a toolbar (New, Open, Save, Print, Find Files, Compare, Go To, Find, Breakpoints, Run, Run and Advance, Run and Time). The main area displays a MATLAB script named 'SetParameters.m'.

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

A red box highlights the line of code: `scanner = stir.Scanner(stir.Scanner.DiscoverySTE());`

<http://stir.sourceforge.net>

Set STIR Descriptions : Data



The screenshot shows the MATLAB IDE interface with the file `/Users/tsai_yujung/Desktop/Talk in MIC/SetDescriptions.m` open. The code defines a function `desp = SetDescriptions(param)` which sets descriptions for projection data, image, and matrix.

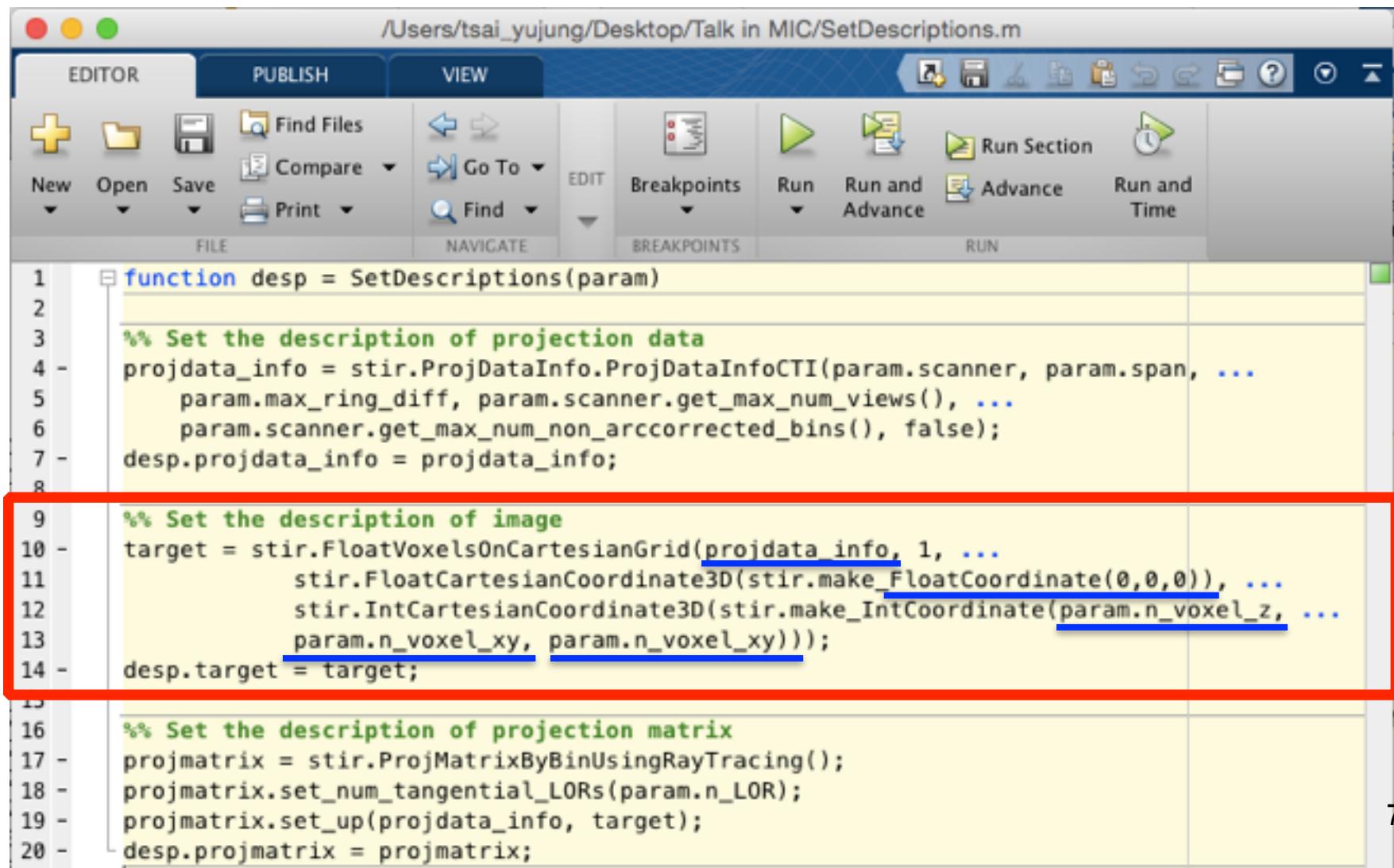
```
function desp = SetDescriptions(param)
% Set the description of projection data
projdata_info = stir.ProjDataInfo.ProjDataInfoCTI(param.scanner, param.span, ...
    param.max_ring_diff, param.scanner.get_max_num_views(), ...
    param.scanner.get_max_num_non_arccorrected_bins(), false);
desp.projdata_info = projdata_info;

% Set the description of image
target = stir.FloatVoxelsOnCartesianGrid(projdata_info, 1, ...
    stir.FloatCartesianCoordinate3D(stir.make_FloatCoordinate(0,0,0)), ...
    stir.IntCartesianCoordinate3D(stir.make_IntCoordinate(param.n_voxel_z, ...
        param.n_voxel_xy, param.n_voxel_xy)));
desp.target = target;

% Set the description of projection matrix
projmatrix = stir.ProjMatrixByBinUsingRayTracing();
projmatrix.set_num_tangential_LORs(param.n_LOR);
projmatrix.set_up(projdata_info, target);
desp.projmatrix = projmatrix;
```

A red box highlights the code for setting the projection data description, specifically the assignment of `projdata_info` to `desp.projdata_info`. The code uses the `stir` library to handle projection data, image, and matrix descriptions.

Set STIR Descriptions : Image

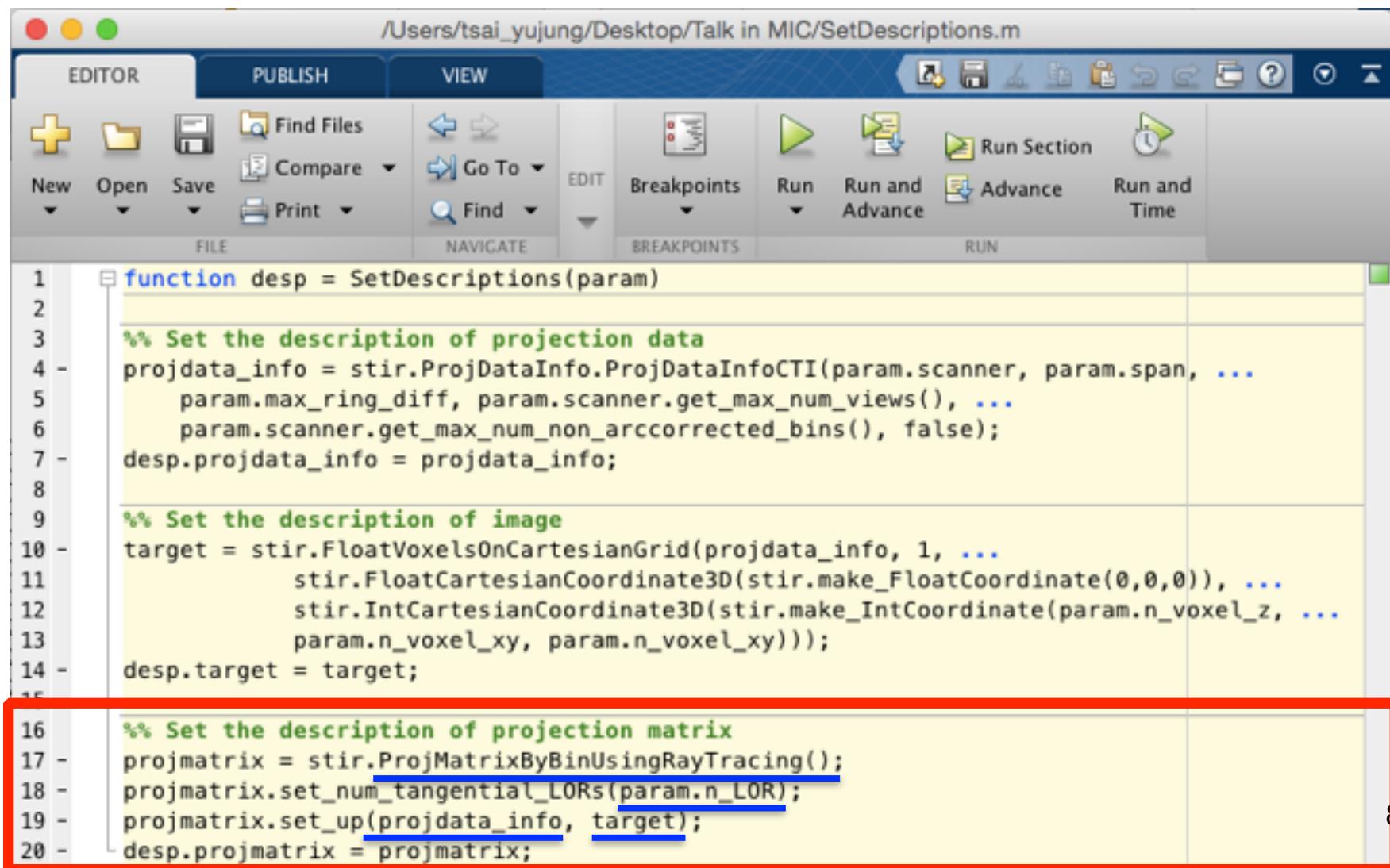


The screenshot shows the MATLAB IDE interface with the following details:

- Title Bar:** /Users/tsai_yujung/Desktop/Talk in MIC/SetDescriptions.m
- Toolbar:** Includes buttons for New, Open, Save, Find Files, Compare, Print, Go To, Find, Breakpoints, Run, Run and Advance, Run and Time, and Help.
- Editor Area:** Displays the MATLAB code for setting STIR descriptions. A red box highlights the section from line 9 to 14.
- Code Content:**

```
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



The screenshot shows a MATLAB IDE interface with the following details:

- Title Bar:** /Users/tsai_yujung/Desktop/Talk in MIC/SetDescriptions.m
- Toolbar:** Includes buttons for New, Open, Save, Find Files, Compare, Print, Go To, Find, Breakpoints, Run, Run and Advance, and Run and Time.
- Menu Bar:** EDITOR, PUBLISH, and VIEW.
- Code Editor:** Displays the following MATLAB script:

```
function desp = SetDescriptions(param)
    % Set the description of projection data
    projdata_info = stir.ProjDataInfo.ProjDataInfoCTI(param.scanner, param.span, ...
        param.max_ring_diff, param.scanner.get_max_num_views(), ...
        param.scanner.get_max_num_non_arccorrected_bins(), false);
    desp.projdata_info = projdata_info;

    % Set the description of image
    target = stir.FloatVoxelsOnCartesianGrid(projdata_info, 1, ...
        stir.FloatCartesianCoordinate3D(stir.make_FloatCoordinate(0,0,0)), ...
        stir.IntCartesianCoordinate3D(stir.make_IntCoordinate(param.n_voxel_z, ...
            param.n_voxel_xy, param.n_voxel_xy)));
    desp.target = target;

    % Set the description of projection matrix
    projmatrix = stir.ProjMatrixByBinUsingRayTracing();
    projmatrix.set_num_tangential_LORs(param.n_LOR);
    projmatrix.set_up(projdata_info, target);
    desp.projmatrix = projmatrix;
```

The lines from 16 to 20 are highlighted with a red box.

Create Data : Phantom

/Users/tsai_yujung/Desktop/Talk in MIC/CreateData.m

EDITOR PUBLISH VIEW

New Open Save Find Files Compare Print Find Go To Comment Indent Insert Breakpoints Run Run and Advance Run Section Run and Time

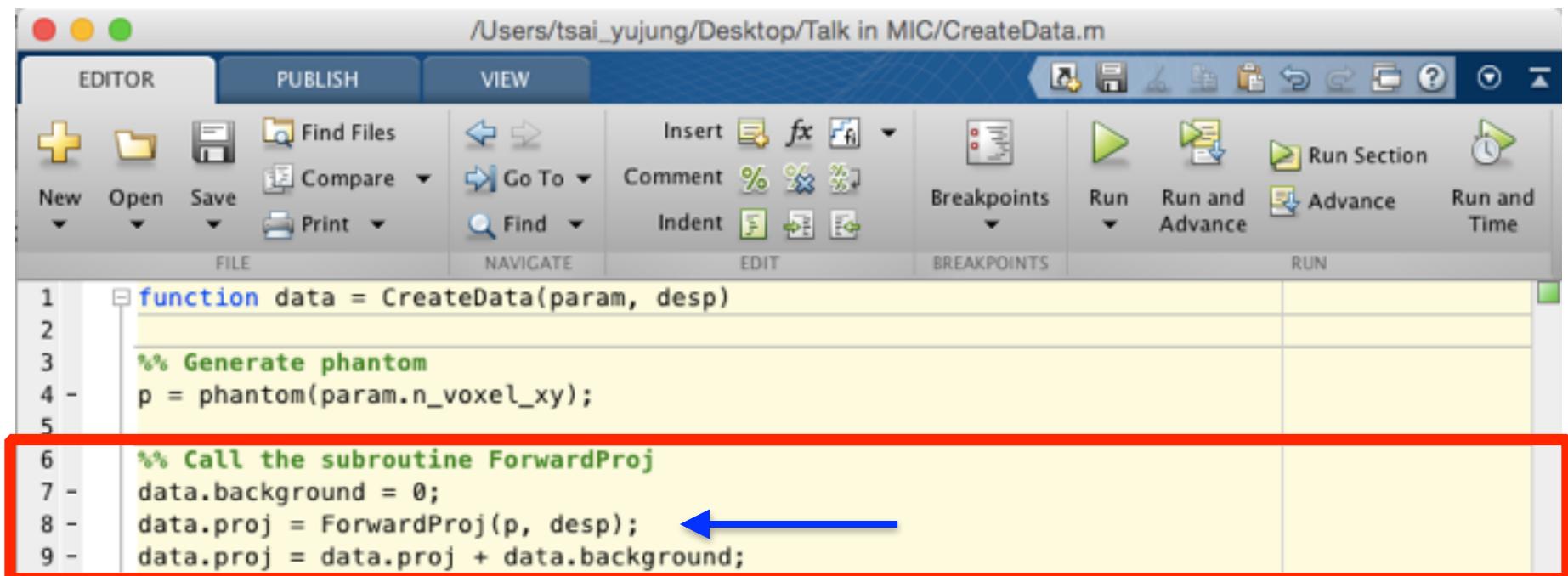
FILE NAVIGATE EDIT BREAKPOINTS RUN

```
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 = 1;
8 data.proj = ForwardProj(p, desp);
9 data.proj = data.proj + data.background;
```

The figure shows a 2D grayscale phantom image. The image is circular and contains several internal structures, including two dark regions on the left and right sides and a central bright region. A color bar on the right side of the plot indicates intensity values ranging from 0 (dark blue) to 1 (yellow). The plot has axes labeled from 10 to 110.

9

Create Data : Projection Data



The screenshot shows the MATLAB IDE interface with the following details:

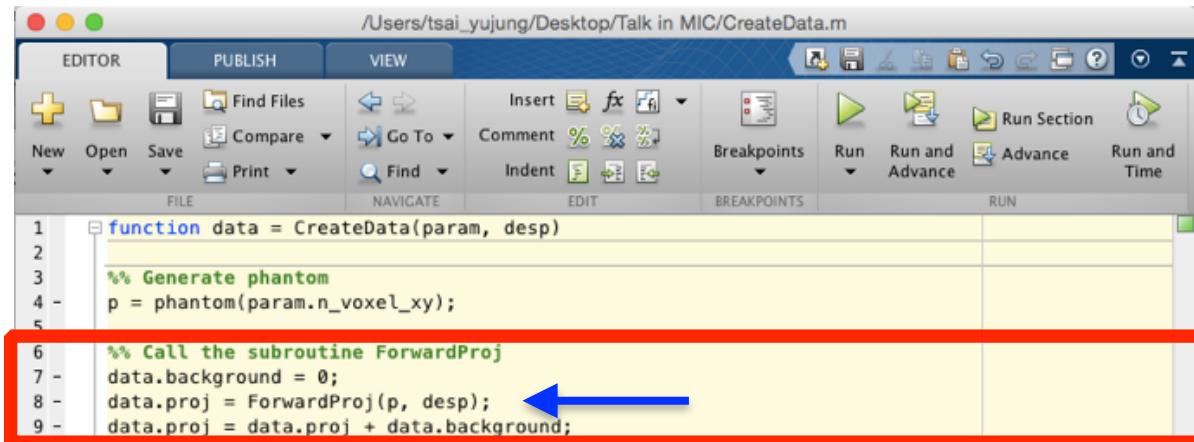
- Title Bar:** /Users/tsai_yujung/Desktop/Talk in MIC/CreateData.m
- Toolbar:** Includes buttons for New, Open, Save, Find Files, Compare, Print, Insert, Comment, Indent, Breakpoints, Run, Run and Advance, Run Section, Advance, and Run and Time.
- MenuBar:** EDITOR, PUBLISH, VIEW.
- Toolstrip:** FILE, NAVIGATE, EDIT, BREAKPOINTS, RUN.
- Code Editor:** Displays the MATLAB script CreateData.m. The code is as follows:

```
function data = CreateData(param, desp)
    %% Generate phantom
    p = phantom(param.n_voxel_xy);

    %% Call the subroutine ForwardProj
    data.background = 0;
    data.proj = ForwardProj(p, desp); ←
    data.proj = data.proj + data.background;
```

A red box highlights the line `data.proj = ForwardProj(p, desp);`, and a blue arrow points to the same line from the left.

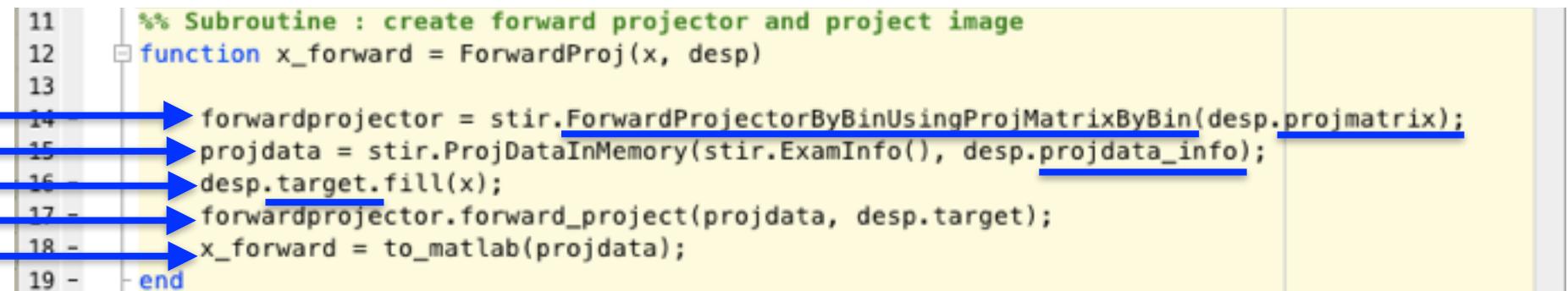
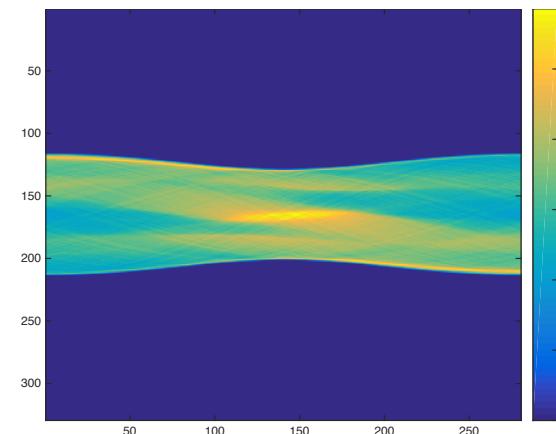
Create Data : Forward Projector



A screenshot of the MATLAB IDE showing the script `CreateData.m`. The code defines a function `CreateData` that generates a phantom and performs a forward projection. A red box highlights the line `data.proj = ForwardProj(p, desp);`, which is annotated with a blue arrow pointing to the corresponding subroutine definition in the bottom window.

```
function data = CreateData(param, desp)
    %% Generate phantom
    p = phantom(param.n_voxel_xy);

    %% Call the subroutine ForwardProj
    data.background = 0;
    data.proj = ForwardProj(p, desp); ←
    data.proj = data.proj + data.background;
```



A screenshot of the MATLAB IDE showing the subroutine `ForwardProj`. The code uses the `stir` library to create a forward projector and project an image. Blue arrows highlight several lines of code, likely indicating the execution flow or specific function calls.

```
%> Subroutine : create forward projector and project image
function x_forward = ForwardProj(x, desp)

forwardprojector = stir.ForwardProjectorByBinUsingProjMatrixByBin(desp.projmatrix);
projdata = stir.ProjDataInMemory(stir.ExamInfo(), desp.projdata_info);
desp.target.fill(x);
forwardprojector.forward_project(projdata, desp.target);
x_forward = to_matlab(projdata);

end
```

Problem to Solve : Objective Function

The screenshot shows the MATLAB IDE interface with the following details:

- Toolbar:** Includes buttons for New, Open, Save, Find Files, Compare, Print, Go To, Breakpoints, Run, Run Section, Advance, and Run and Time.
- Code Editor:** Displays the script `/Users/tsai_yujung/Desktop/Talk in MIC/Problem2Minimize.m`. The code calculates the objective function value using the ForwardProjection subroutine.
- Red Box:** A red box highlights the following code block:

```
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(data.proj.*log(x_forward) - x_forward));
```

A blue arrow points from the text "Call the subroutine ForwardProj and calculate the objective function value" to the line `x_forward = ForwardProj(x, desp);`. A red arrow points from the text "Call the subroutine BackwardProj and calculate the gradient" to the line `f = -sum(sum(data.proj.*log(x_forward) - x_forward));`.

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

Problem to Solve : Gradient

The screenshot shows the MATLAB IDE interface with the following details:

- Title Bar:** /Users/tsai_yujung/Desktop/Talk in MIC/Problem2Minimize.m
- Toolbar:** Includes buttons for New, Open, Save, Find Files, Compare, Print, Go To, Breakpoints, Run, Run and Advance, and Run and Time.
- MenuBar:** EDITOR, PUBLISH, VIEW.
- Toolstrip:** FILE, NAVIGATE, BREAKPOINTS, RUN.
- Code Editor:** Displays the following MATLAB script:

```
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(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);
```

The line `g = -BackwardProj(ratio, desp);` is highlighted with a red rectangle and has a blue arrow pointing to it from the left.

Problem to Solve : Backward Projector

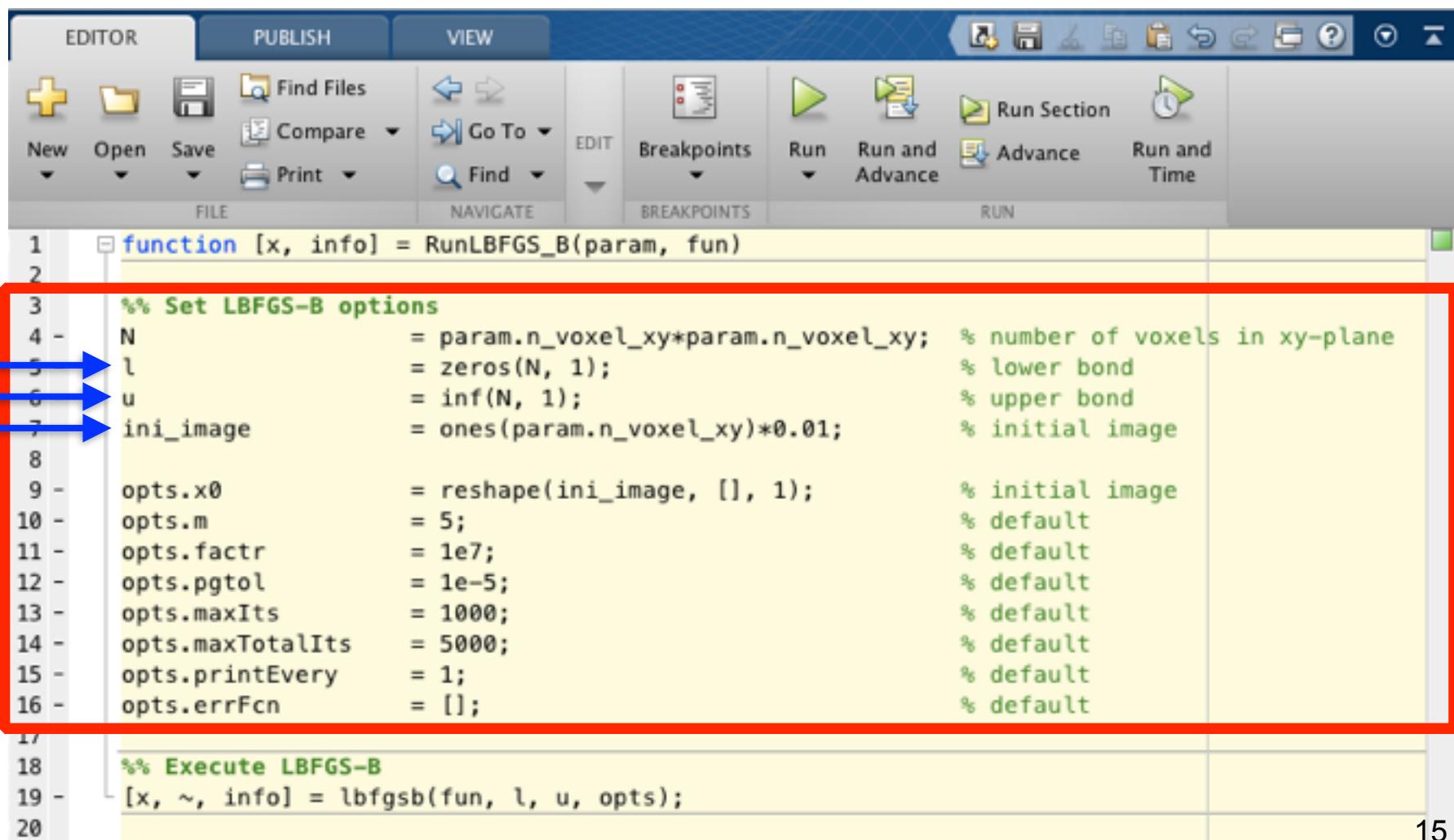
The screenshot shows the MATLAB IDE interface with the following details:

- Title Bar:** /Users/tsai_yujung/Desktop/Talk in MIC/Problem2Minimize.m
- Toolbar:** Includes buttons for New, Open, Save, Find Files, Compare, Go To, Find, Breakpoints, Run, Run and Advance, Run Section, Advance, and Run and Time.
- Code Editor:** Displays the MATLAB script `Problem2Minimize.m`. The code defines a function `Problem2Minimize` that performs forward projection and calculates the objective function value. It then calls the `BackwardProj` subroutine to calculate the gradient. The line `g = -BackwardProj(ratio, desp);` is highlighted with a red box and a blue arrow pointing to it.
- Subroutine Definition:** Below the main function, there is a subroutine definition:

```
function data_backward = BackwardProj(data, desp)
    backprojector = stir.BackProjectorByBinUsingProjMatrixByBin(desp.projmatrix);
    projdata = stir.ProjDataInMemory(stir.ExamInfo(), desp.projdata_info);
    projdata.fill(data);
    backprojector.back_project(desp.target, projdata);
    data_backward = desp.target.to_matlab();
end
```

Blue arrows point from the first four lines of this code block to the corresponding lines in the main function above.

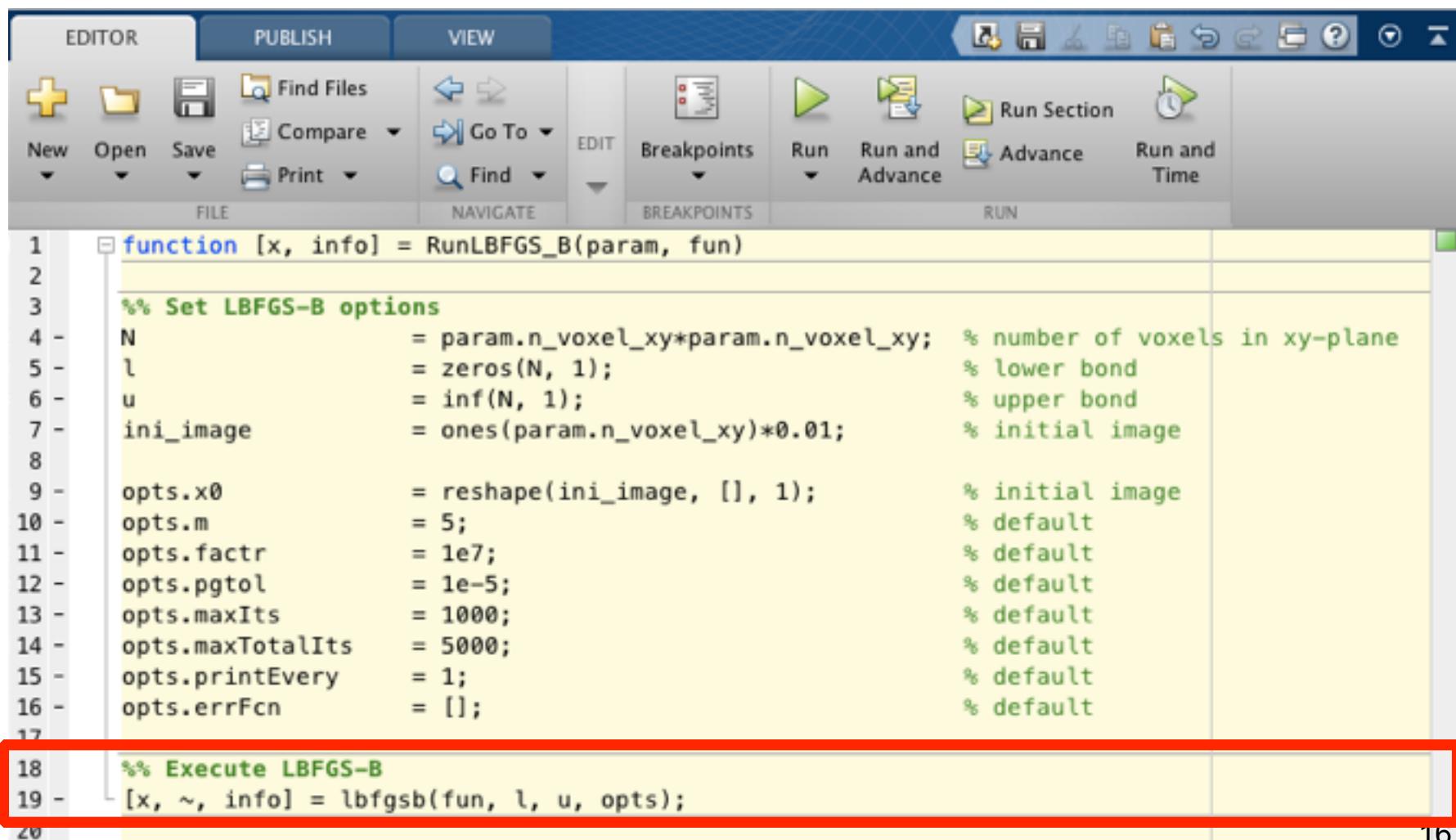
L-BFGS-B : Set Options



The screenshot shows the MATLAB interface with the code for setting L-BFGS-B options. A red box highlights the section from line 3 to line 16, and three blue arrows point to lines 5, 6, and 7, indicating specific parameters being set.

```
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 bound
6 - u           = inf(N, 1);                          % upper bound
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 IDE interface with the following details:

- Toolbar:** Includes buttons for New, Open, Save, Find Files, Compare, Print, Go To, Find, Breakpoints, Run, Run and Advance, and Run and Time.
- Menu Bar:** Shows EDIT, PUBLISH, and VIEW tabs.
- Code Editor:** Displays the MATLAB script `RunLBFGS_B.m`. The code sets up options for the LBFGS-B solver and executes it.

```
function [x, info] = RunLBFGS_B(param, fun)
%% Set LBFGS-B options
N = param.n_voxel_xy*param.n_voxel_xy; % number of voxels in xy-plane
l = zeros(N, 1); % lower bound
u = inf(N, 1); % upper bound
ini_image = ones(param.n_voxel_xy)*0.01; % initial image
opts.x0 = reshape(ini_image, [], 1); % initial image
opts.m = 5; % default
opts.factr = 1e7; % default
opts.pgtol = 1e-5; % default
opts.maxIts = 1000; % default
opts.maxTotalIts = 5000; % default
opts.printEvery = 1; % default
opts.errFcn = [];
%% Execute LBFGS-B
[x, ~, info] = lbfgsb(fun, l, u, opts);

```

- Code Editor Selection:** Line 18, which starts the execution command, is highlighted with a red rectangular border.
- Code Editor Status:** The status bar at the bottom right shows the number 16.

Run Simulation

/Users/tsai_yujung/Desktop/Talk in MIC/RunSimulation.m

EDITOR PUBLISH VIEW

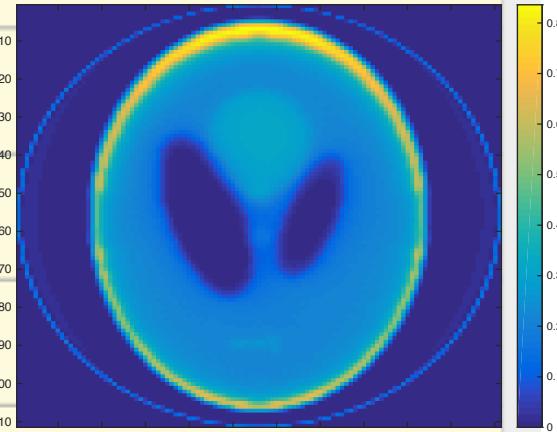
New Open Save Find Files Compare Print

NAVIGATE EDIT Breakpoints

Run Run and Advance Run Section Advance Run and Time

FILE BREAKPOINTS RUN

```
1 function [x, info] = RunSimulation()
2
3 %% Set STIR parameters
4 param = SetParameters();
5
6 %% Set STIR Image/Data descriptions
7 desp = SetDescriptions(param);
8
9 %% Create data
10 data = CreateData(param, desp);
11
12 %% Problem to solve
13 fun = @(x)Problem2Minimize(x, param, desp, data);
14
15 %% Set LBFGS-B options and execute LBFGS-B
16 [x, info] = RunLBFGS_B(param, fun);
17 figure;imagesc(reshape(x,[param.n_voxel_xy param.n_voxel_xy]));colorbar
```



17

For more results ...

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

¹Yu-Jung Tsai, ¹Alexandre Bousse, ²Matthias J. Ehrhardt, ^{1*}Brian Hutton, ²Simon Arridge, ¹Kris Thielemans

¹Institute of Nuclear Medicine, University College London, UK

²Department of Computer Science, University College London, UK

M4CP-190

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)