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)

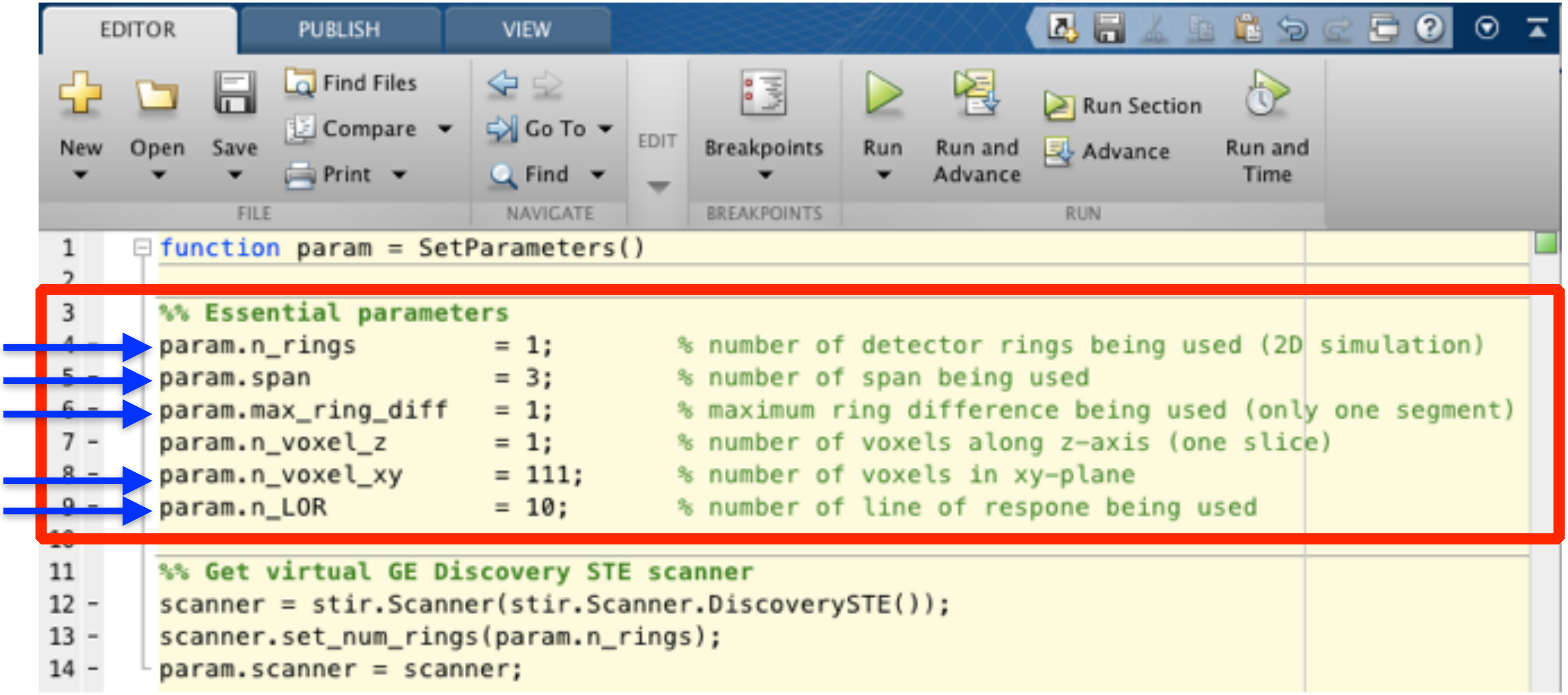
- A popular Quasi-Newton method
- Requires only the objective function and its gradient
- Used when the true inverse of Hessian is too complicated to handle or does not exist
- C code implementation with MATLAB mex wrapper is available at: https://github.com/stephenbeckr/L-BFGS-B-C

Framework : `RunSimulation()`

```matlab
%% 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);
```

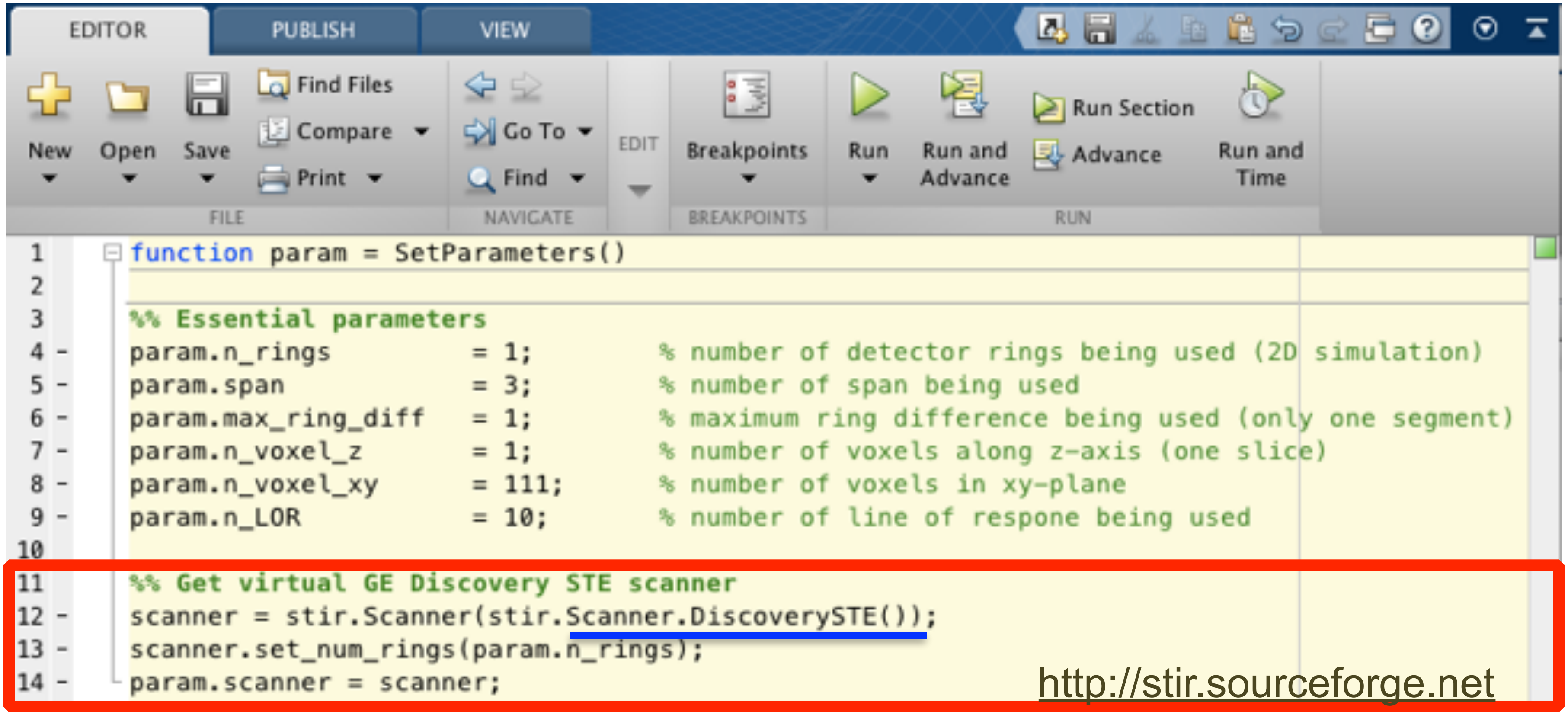
[function_value, gradient] = fun(x)

Set STIR Parameters : Parameters



```matlab
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 respone being used

%% Get virtual GE Discovery STE scanner
scanner = stir.Scanner(stir.Scanner.DiscoverySTE());
scanner.set_num_rings(param.n_rings);
param.scanner = scanner;
```

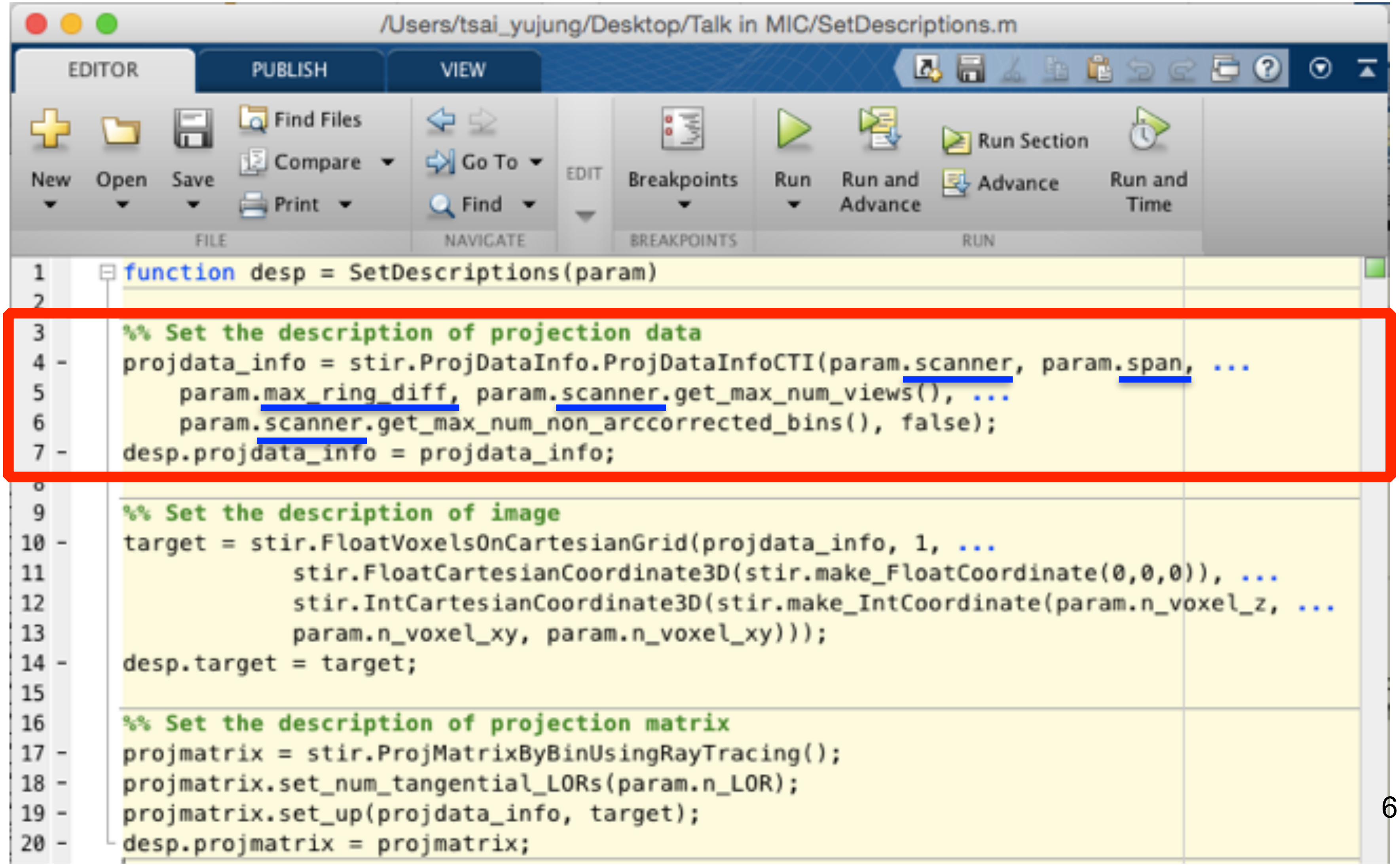
Set STIR Parameters : Scanner



```matlab
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 respone being used

%% Get virtual GE Discovery STE scanner
scanner = stir.Scanner(stir.Scanner.DiscoverySTE());
scanner.set_num_rings(param.n_rings);
param.scanner = scanner;
```

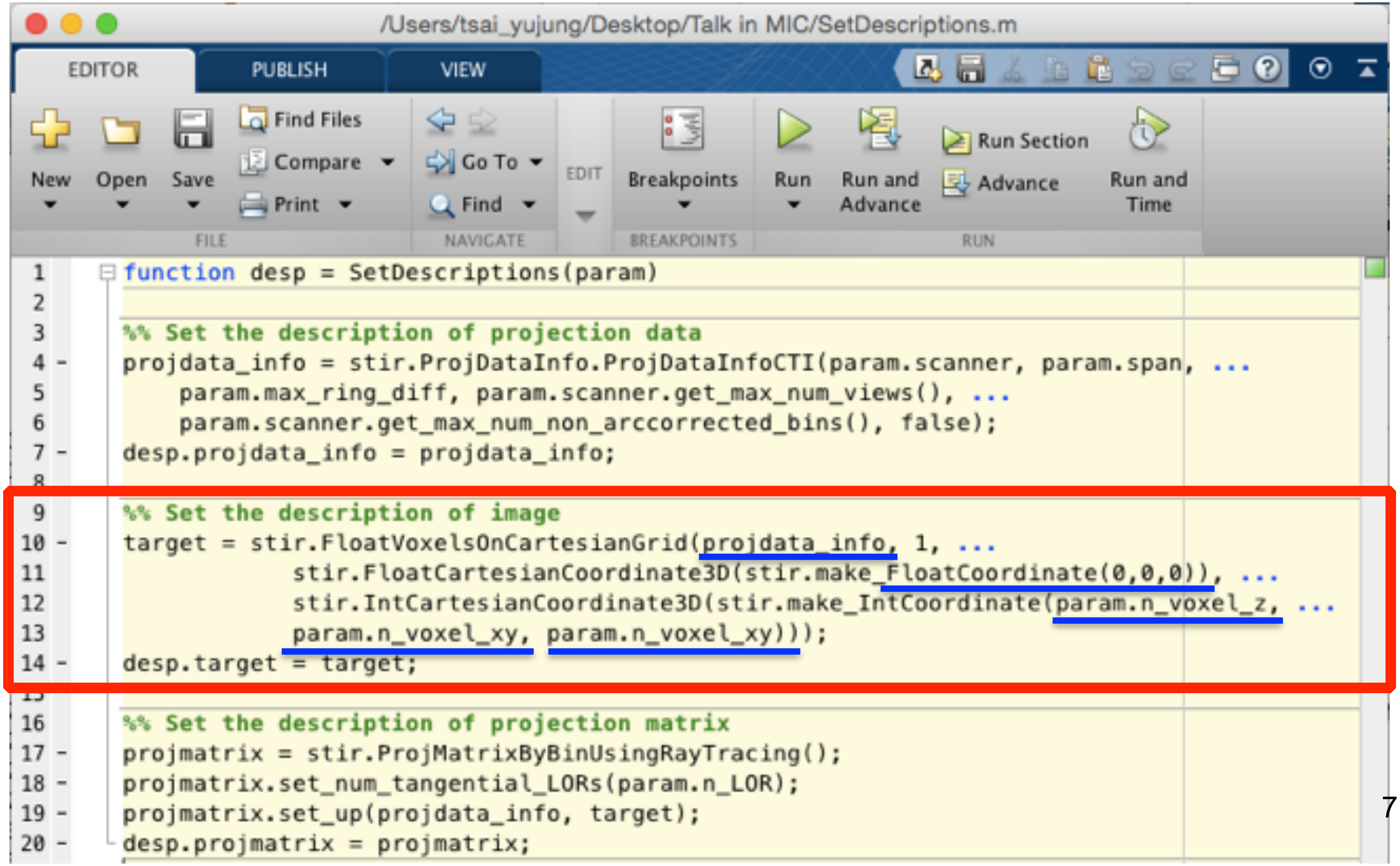
http://stir.sourceforge.net

Set STIR Descriptions : Data



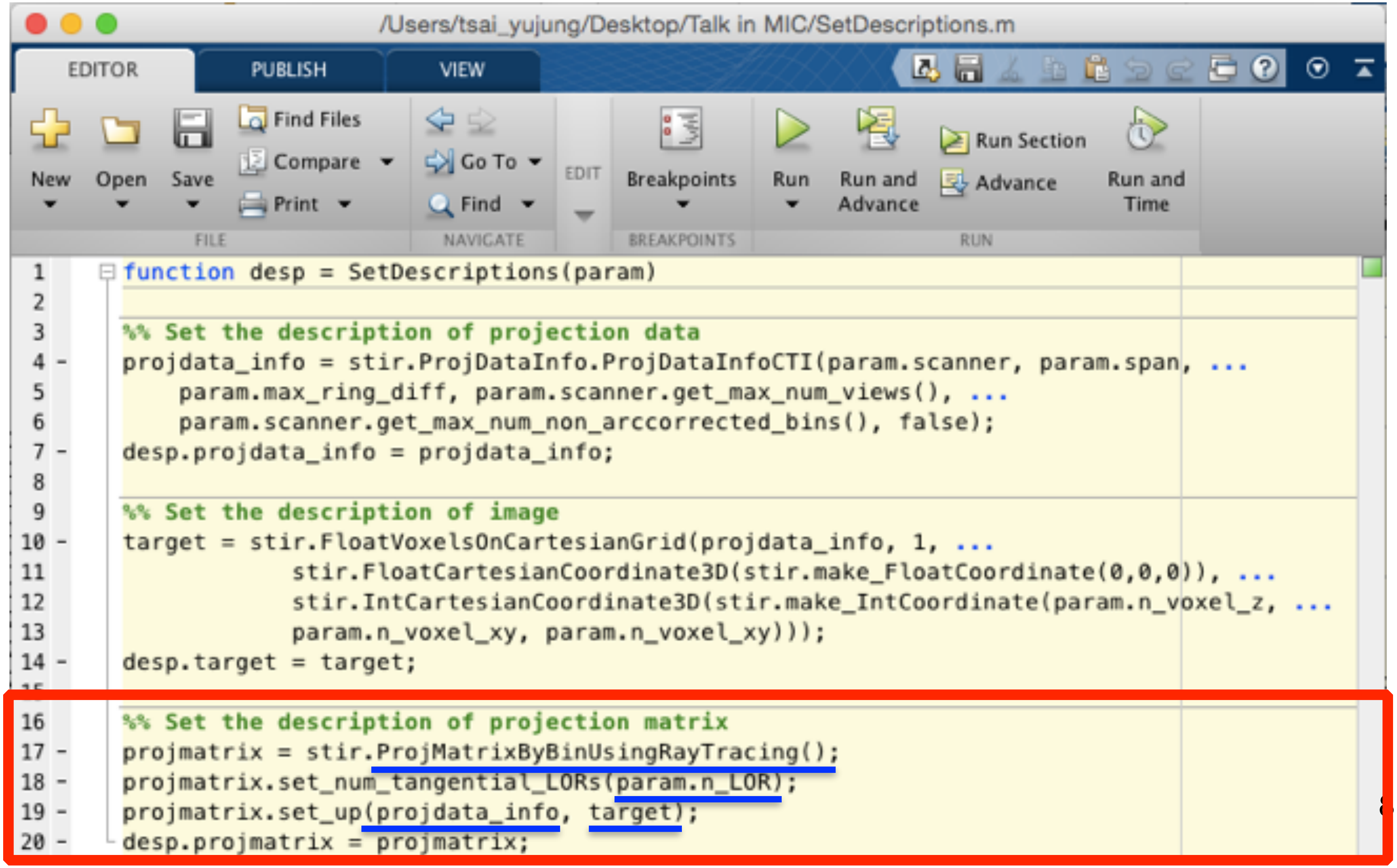
```
function desp = SetDescriptions(param)

%% 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;
```

Set STIR Descriptions : Image



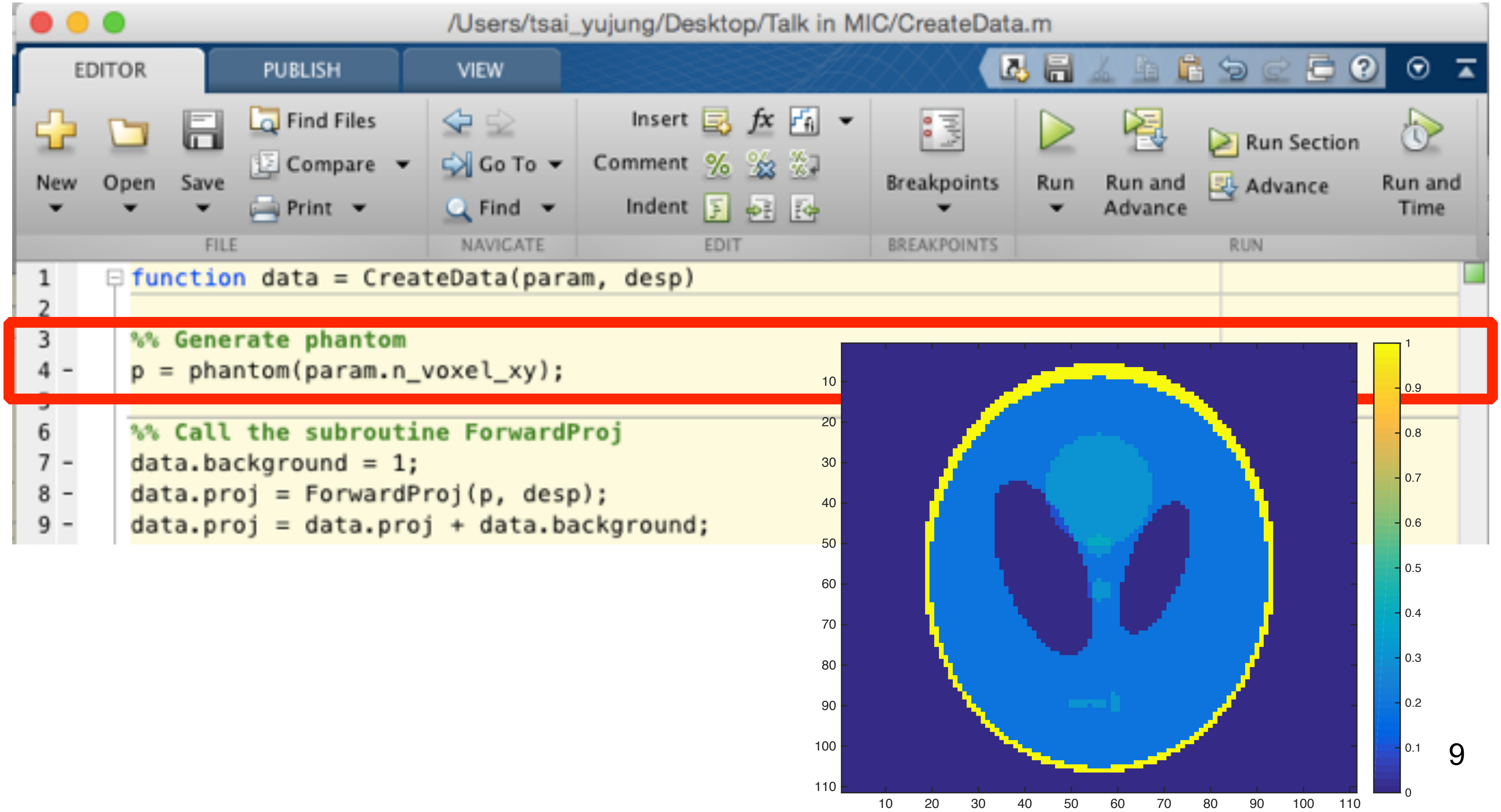
```
function desp = SetDescriptions(param)

%% 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;
```

Set STIR Descriptions : Projection Matrix



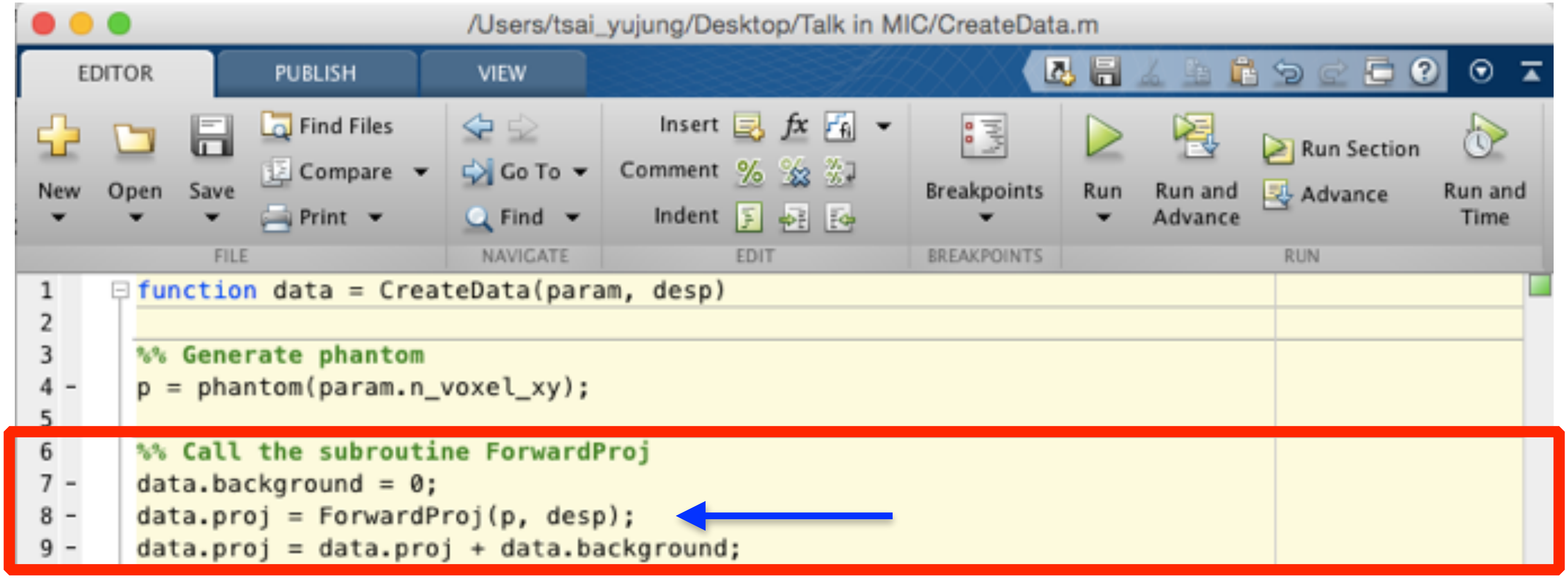
```matlab
function desp = SetDescriptions(param)

%% 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;
```

Create Data : Phantom



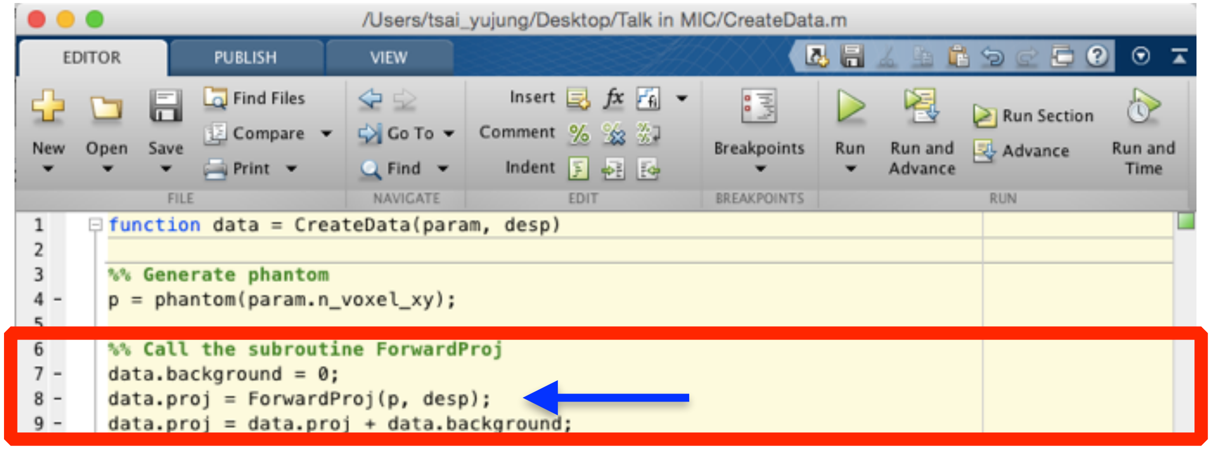
```
function data = CreateData(param, desp)

%% Generate phantom
p = phantom(param.n_voxel_xy);

%% Call the subroutine ForwardProj
data.background = 1;
data.proj = ForwardProj(p, desp);
data.proj = data.proj + data.background;
```

Create Data : Projection Data

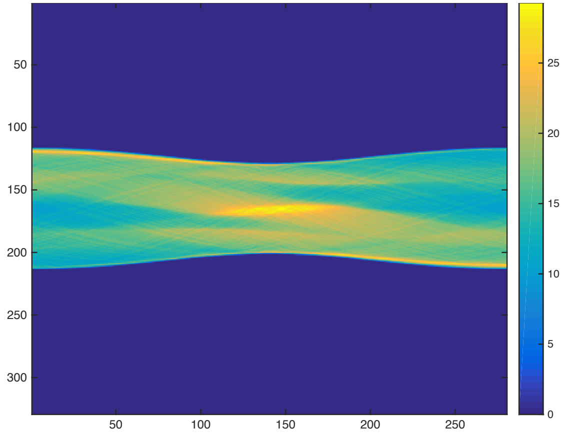


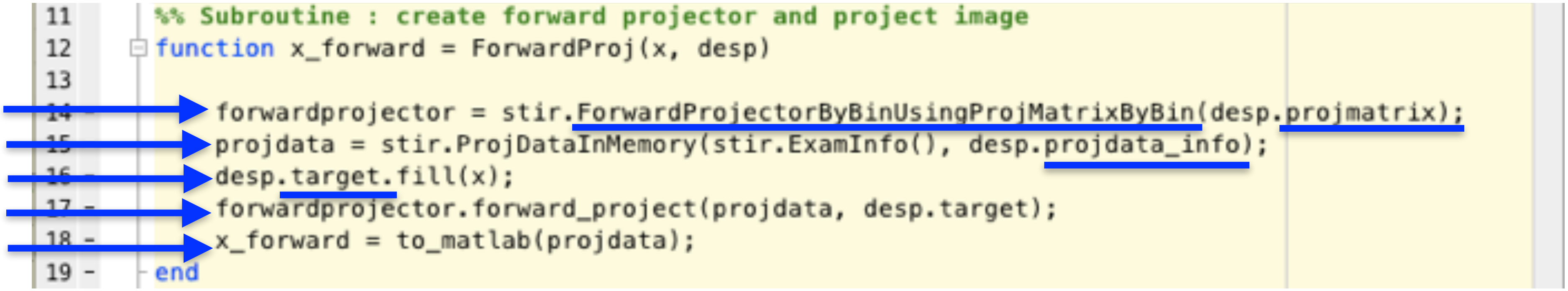
```
function data = CreateData(param, desp)

%% 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;
```

Create Data : Forward Projector



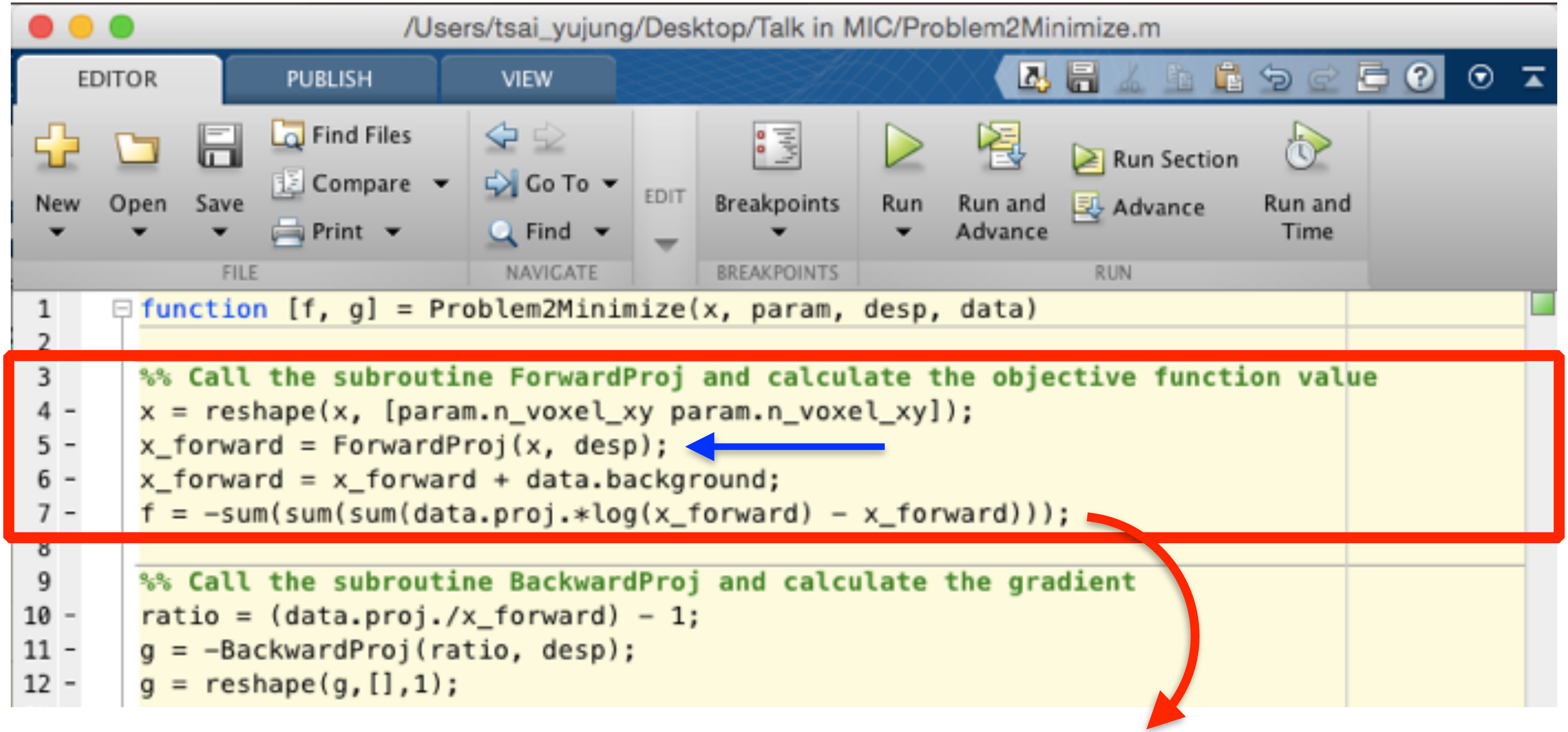
```
function data = CreateData(param, desp)

%% 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;
```





```
%% 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
```

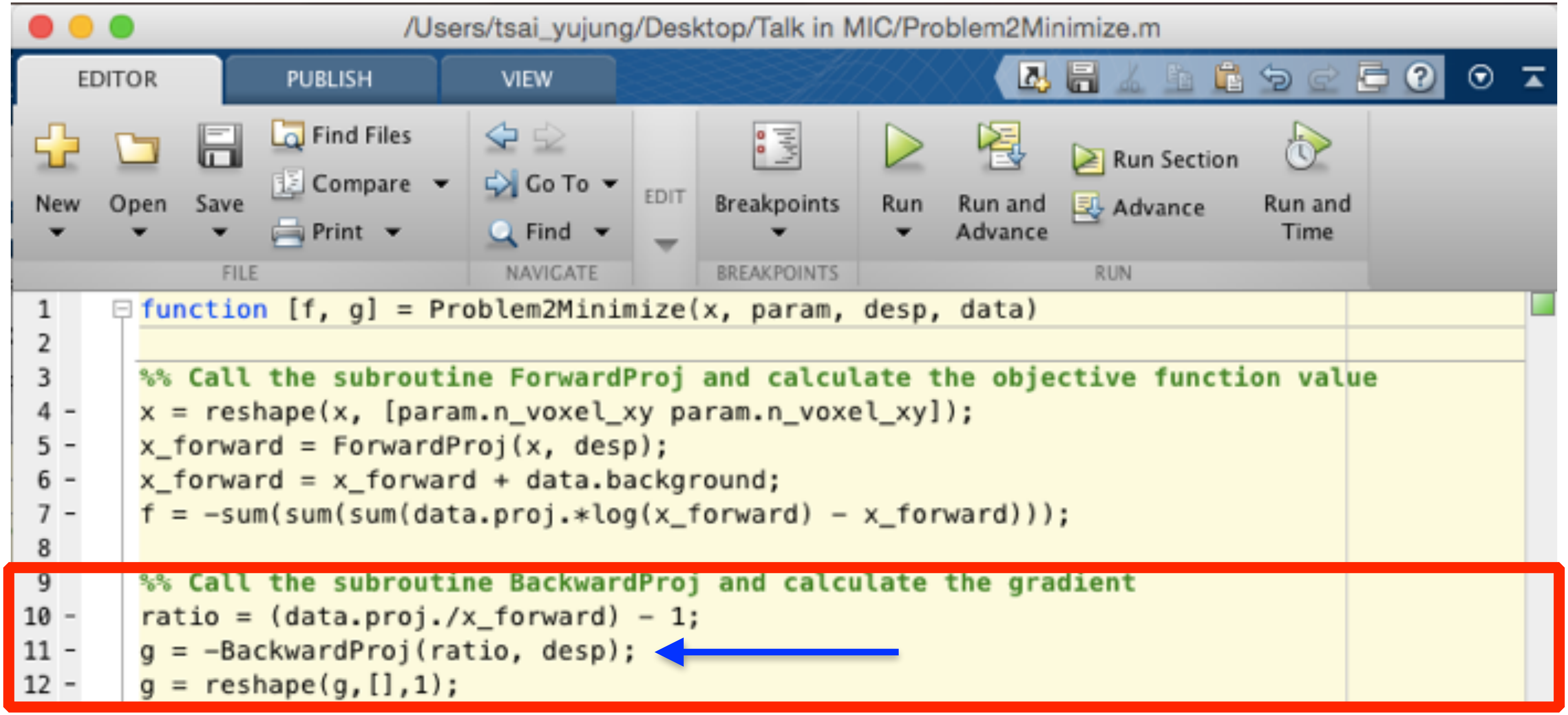
Problem to Solve : Objective Function



```
function [f, g] = Problem2Minimize(x, param, desp, data)

%% Call the subroutine ForwardProj and calculate the objective function value
x = reshape(x, [param.n_voxel_xy param.n_voxel_xy]);
x_forward = ForwardProj(x, desp);
x_forward = x_forward + data.background;
f = -sum(sum(sum(data.proj.*log(x_forward) - x_forward)));

%% Call the subroutine BackwardProj and calculate the gradient
ratio = (data.proj./x_forward) - 1;
g = -BackwardProj(ratio, desp);
g = reshape(g,[],1);
```

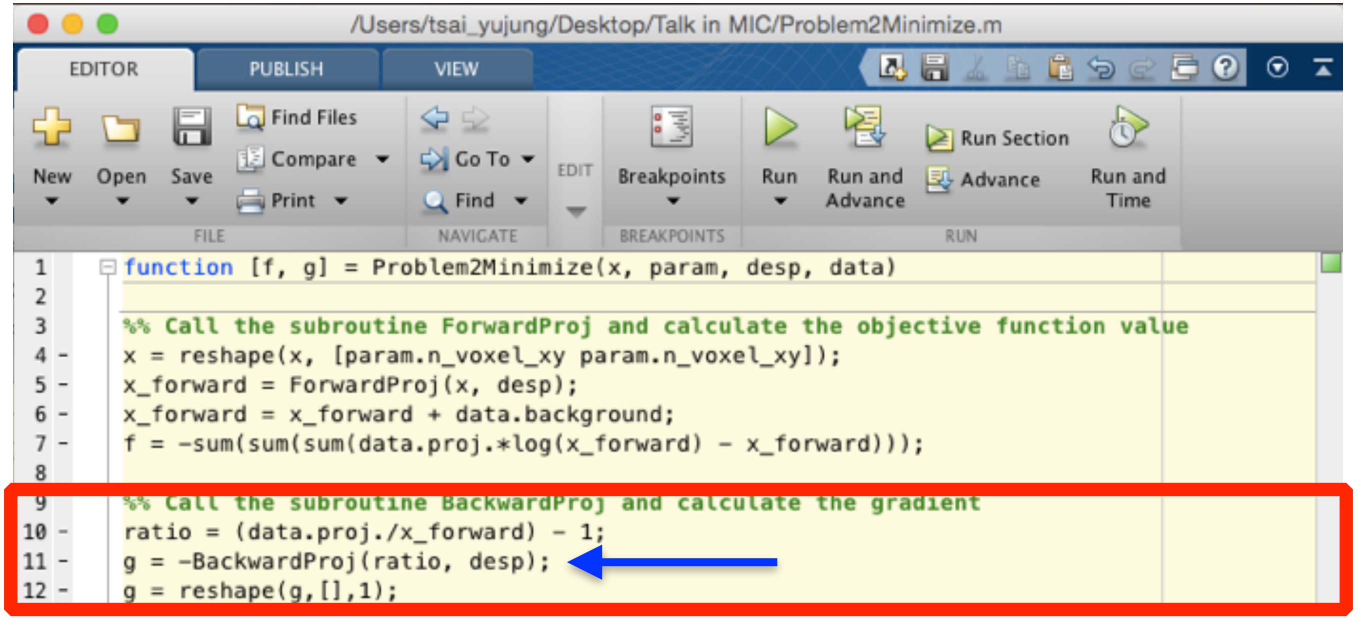
$$-data\cdot \log\{forward\_prj(x)+\mathrm{background}\}+\{forward\_prj(x)+\mathrm{background}\}$$

Problem to Solve : Gradient

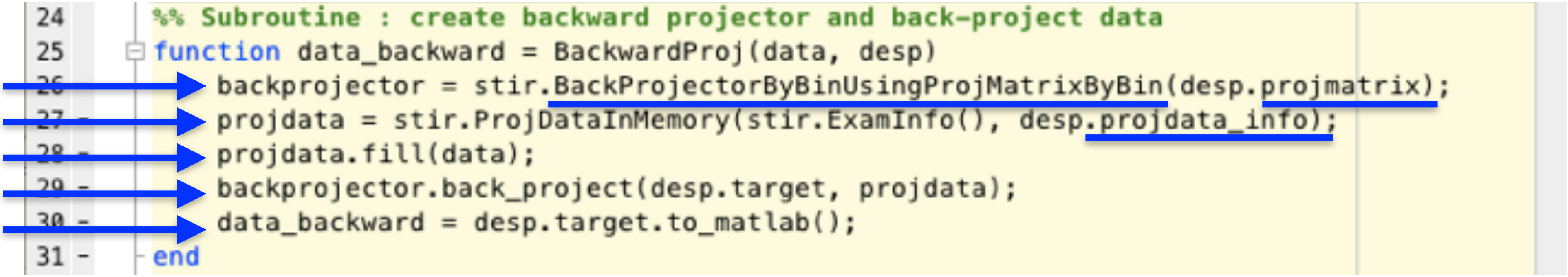


```matlab
function [f, g] = Problem2Minimize(x, param, desp, data)

%% Call the subroutine ForwardProj and calculate the objective function value
x = reshape(x, [param.n_voxel_xy param.n_voxel_xy]);
x_forward = ForwardProj(x, desp);
x_forward = x_forward + data.background;
f = -sum(sum(sum(data.proj.*log(x_forward) - x_forward)));

%% Call the subroutine BackwardProj and calculate the gradient
ratio = (data.proj./x_forward) - 1;
g = -BackwardProj(ratio, desp);
g = reshape(g,[],1);
```

Problem to Solve : Backward Projector

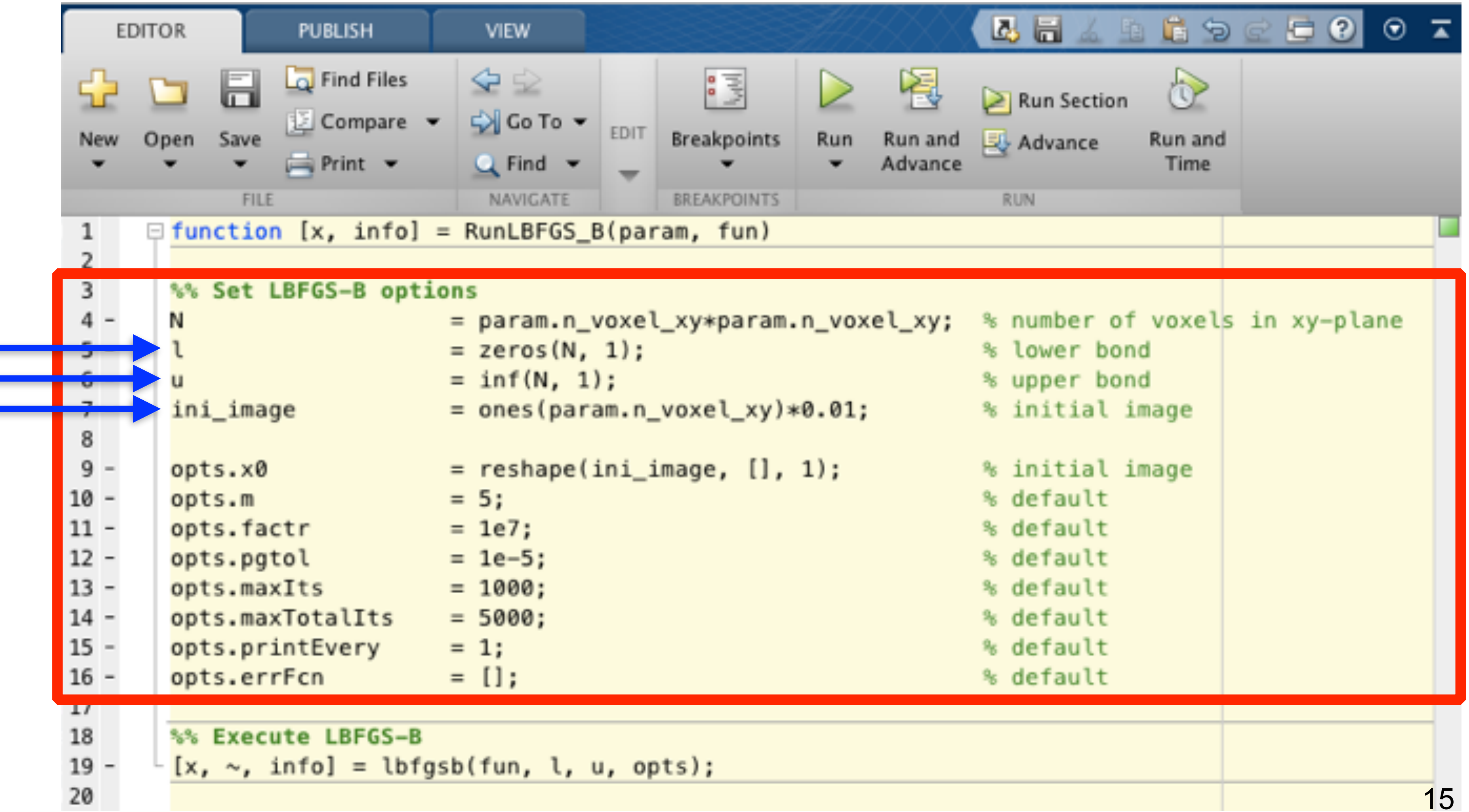


```matlab
function [f, g] = Problem2Minimize(x, param, desp, data)

%% Call the subroutine ForwardProj and calculate the objective function value
x = reshape(x, [param.n_voxel_xy param.n_voxel_xy]);
x_forward = ForwardProj(x, desp);
x_forward = x_forward + data.background;
f = -sum(sum(sum(data.proj.*log(x_forward) - x_forward)));

%% Call the subroutine BackwardProj and calculate the gradient
ratio = (data.proj./x_forward) - 1;
g = -BackwardProj(ratio, desp);
g = reshape(g,[],1);
```



```matlab
%% Subroutine : create backward projector and back-project data
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
```

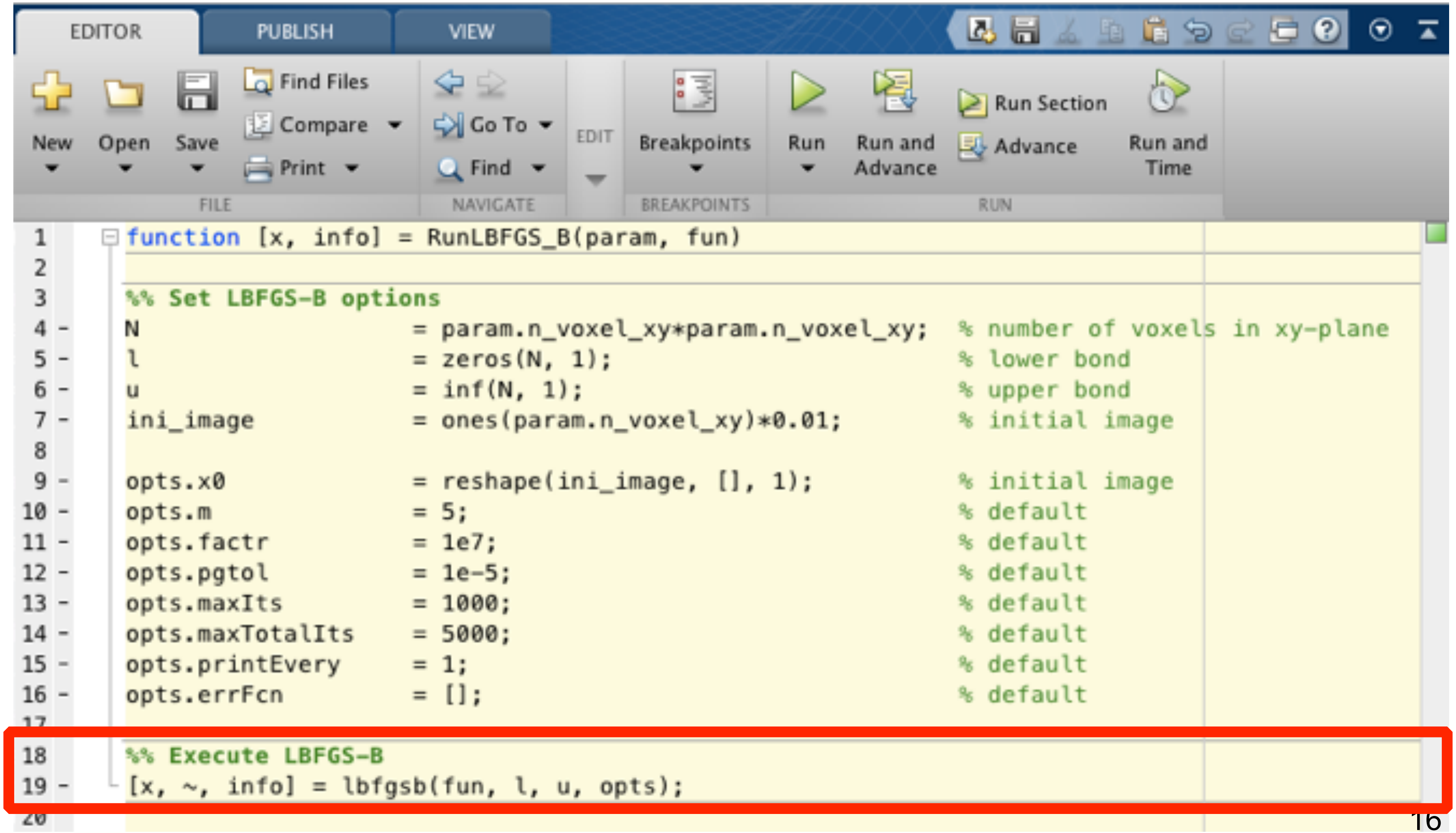
LBFGS-B : Set Options



```matlab
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 bond
u                    = inf(N, 1);                          % upper bond
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          = [];                                 % default

%% Execute LBFGS-B
[x, ~, info] = lbfgsb(fun, l, u, opts);

```

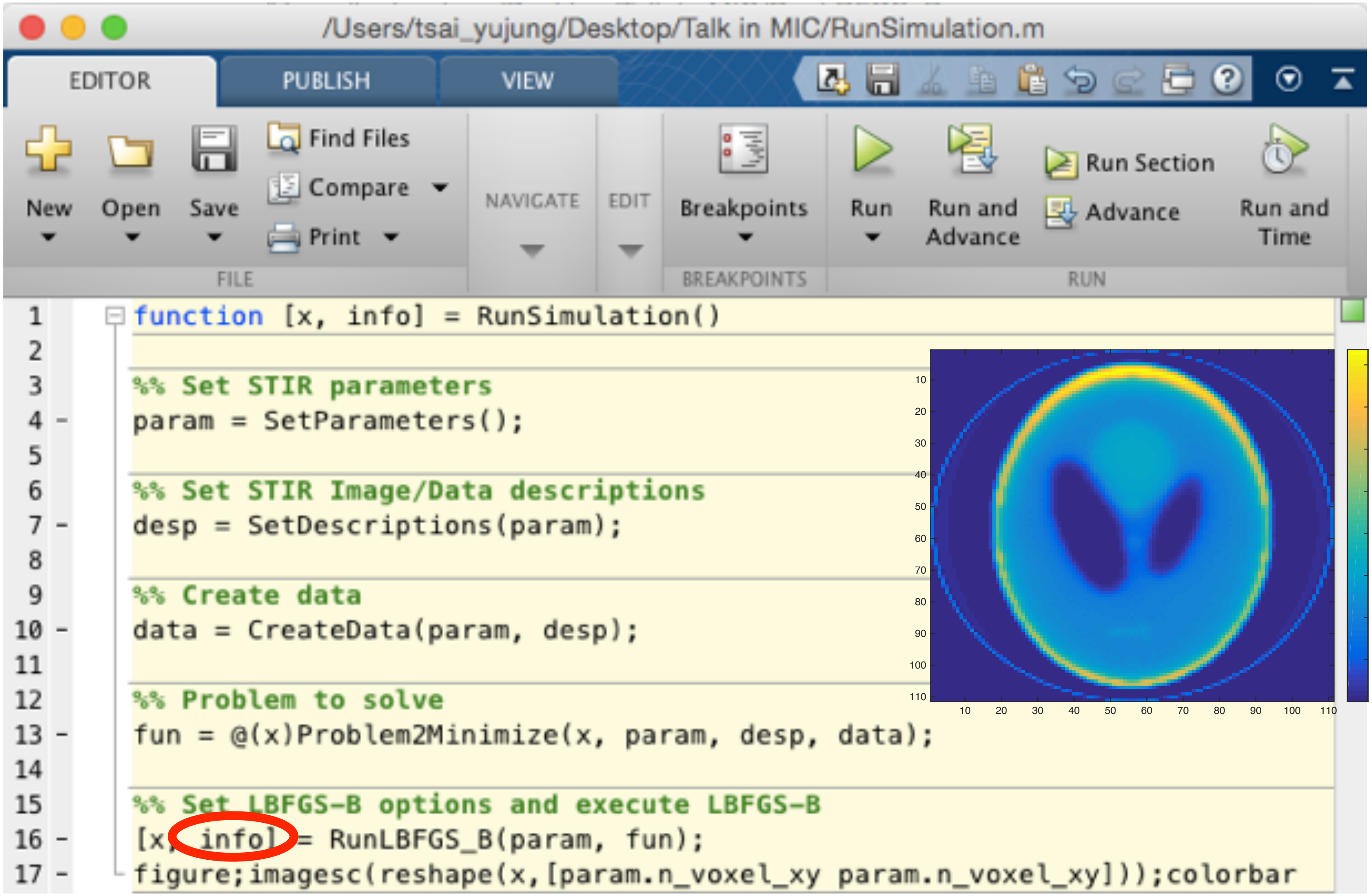
https://github.com/stephenbeckr/L-BFGS-B-C

LBFGS-B : Execute



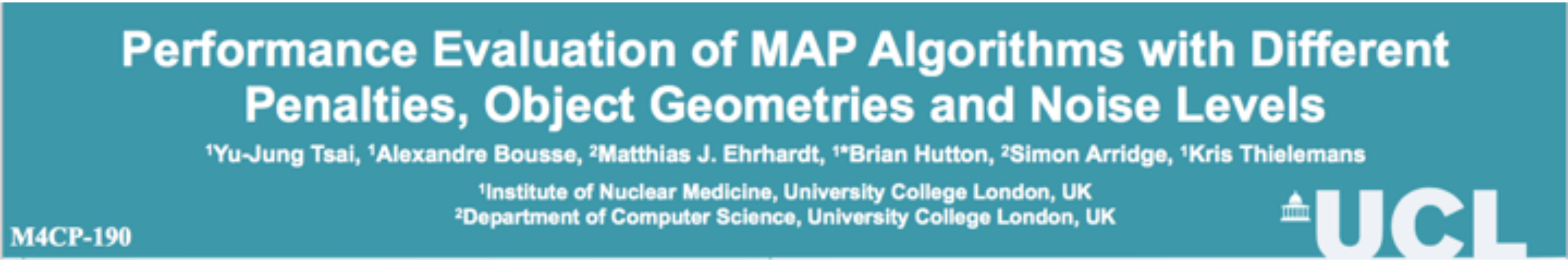
```
function [x, info] = RunLBFGS_B(param, fun)

%% Set LBFGS-B options
N                   = param.n_voxel_xy*param.n_voxel_xy;    % number of voxels in xy-plane
l                   = zeros(N, 1);                          % lower bond
u                   = inf(N, 1);                            % upper bond
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         = [];                                   % default

%% Execute LBFGS-B
[x, ~, info] = lbfgsb(fun, l, u, opts);
```

Run Simulation



```matlab
function [x, info] = RunSimulation()

%% Set STIR parameters
param = SetParameters();

%% Set STIR Image/Data descriptions
desp = SetDescriptions(param);

%% Create data
data = CreateData(param, desp);

%% Problem to solve
fun = @(x)Problem2Minimize(x, param, desp, data);

%% Set LBFGS-B options and execute LBFGS-B
[x, info] = RunLBFGS_B(param, fun);
figure;imagesc(reshape(x,[param.n_voxel_xy param.n_voxel_xy]));colorbar
```

For more results …



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

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

[1]Institute of Nuclear Medicine, University College London, UK
[2]Department 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)