dPETSTEP

- Run your first simulation -

Introduction

dPETSTEP (https://github.com/CRossSchmidtlein/dPETSTEP) is a fast dynamic PET simulator designed for high throughput simulation of dynamic PET images. It allows full simulation of a user defined parametric image, kinetic model, input function, time sampling and more into realistic (noisy) dynamic PET-like images. Furthermore, the dynamic PET data can then be model fitted to produce parameter/parametric image estimates.

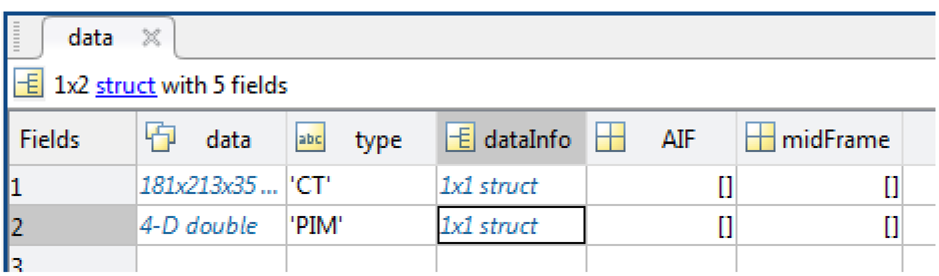
This application is written in MATLAB and designed as an extension of PETSTEP (https://github.com/CRossSchmidtlein/PETSTEP). Currently, MATLAB functions from PETSTEP are used by dPETSTEP, but there is no requirement of installing CERR (Computational Environment for Radiological Research, https://github.com/adityaapte/CERR).

Data structure

The data used by dPETSTEP should have a certain format, as will be specified below. It should contain three fields: data, type and dataInfo. It should contain two rows per field:

1. The first row should be the CT or mu-map of the object. Data should contain the 3D matrix of the object. Type should be "CT" or "mumap".
2. The second row should be the parametric image of the object. Data should be the 4D (3 spatial + 1 parametric) matrix with the parametric object. Type should be "PIM".

These data should be stored in a 1x2 structure named "data":


data

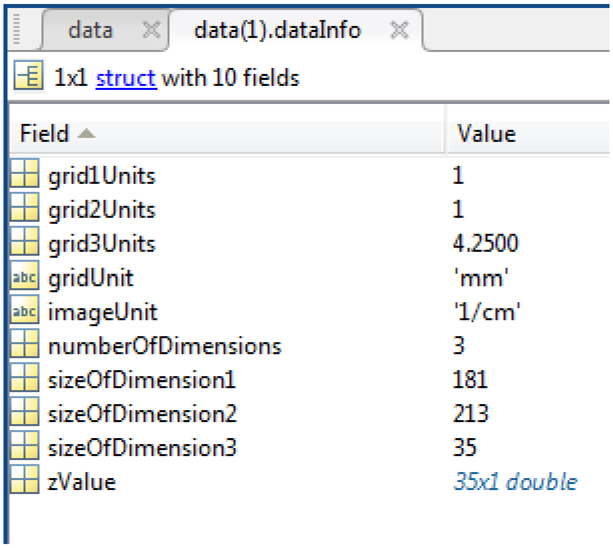
1x2 struct with 5 fields

Fields	data	type	dataInfo	AIF	midFrame
1	181x213x35 ...	'CT'	1x1 struct	[]	[]
2	4-D double	'PIM'	1x1 struct	[]	[]
3					

In the example folder in the github repository, there is a *.mat-file with a sample data structure file. You can use that data as a starting point and just change the data and relevant information for your own simulation.

CT or mu-map

The dataInfo for the CT or mu-map should contain the fields as in the figure below. Currently, the grid units have to be in millimeters. For mu-maps, the unit should be 1/cm. The zValue is a vector with the mid slice positions in millimeters.



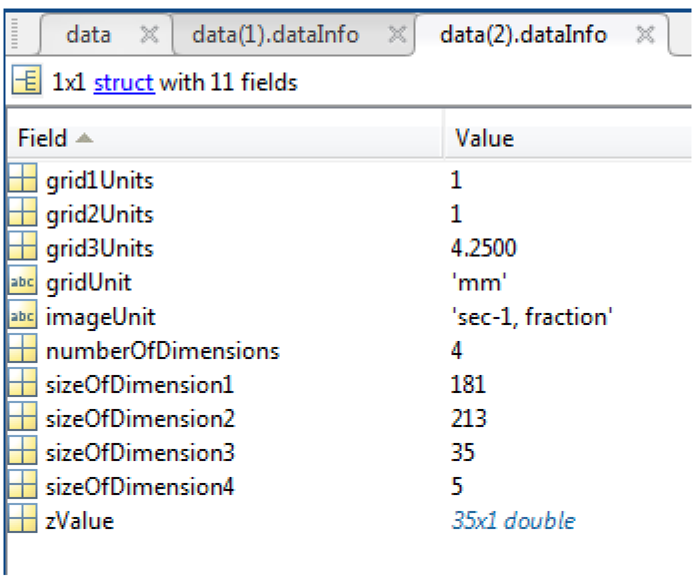
data | data(1).dataInfo

1x1 struct with 10 fields

Field	Value
grid1Units	1
grid2Units	1
grid3Units	4.2500
gridUnit	'mm'
imageUnit	'1/cm'
numberOfDimensions	3
sizeOfDimension1	181
sizeOfDimension2	213
sizeOfDimension3	35
zValue	*35x1 double*

Parametric image

The dataInfo for the parametric image should look like below. The grid units should be in millimeters, and the rate constants should be in 1/sec. The zValue is a vector with the mid slice positions in millimeters.



data | data(1).dataInfo | data(2).dataInfo

1x1 struct with 11 fields

Field	Value
grid1Units	1
grid2Units	1
grid3Units	4.2500
gridUnit	'mm'
imageUnit	'sec-1, fraction'
numberOfDimensions	4
sizeOfDimension1	181
sizeOfDimension2	213
sizeOfDimension3	35
sizeOfDimension4	5
zValue	*35x1 double*

Run a simulation

Once you have the input data structure set up properly according to above, you can start your simulation process. First however, there are a few things you need to do.

If you want to use the Graphical User Interfaces (GUI), make sure you've downloaded the GUI files from https://github.com/CRossSchmidtlein/dPETSTEP/tree/master/dPETSTEP_graphical_user_interfaces.

Type

```
>> dPETSTEP
```

to open the main menu

Kinetic modeling

You can generate the dynamic PET uptake image, followed by PET acquisition simulation, using either the graphical user interface (GUI), or directly using the Matlab codes.

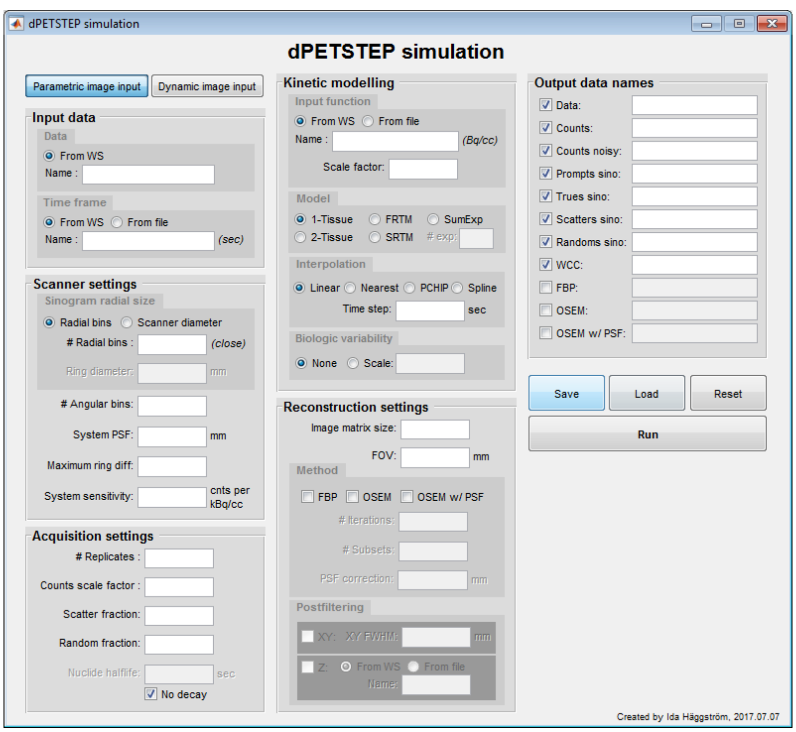
Using GUI

Make sure you've downloaded the GUI files from GitHub.

Either open the main dPETSTEP menu as above, click Simulation, or in the Matlab prompt type

```
>> dPETSTEPgui_sim
```

to open window below. Choose "Parametric image input". For further instructions on all fields, look at tooltips and the User's guide (Users_guide_dPETSTEP_simulation_GUI.pdf).



Using Matlab files

1. You need to adjust all settings of the simulation according to your preference. Do this by opening "Dynamic_setSimParameters.m" and adjust the fields as you like.

   ```
   >> open Dynamic_setSimParameters.m
   ```

```
function simSet = Dynamic_setSimParameters(frame,Cif)
%**************************************************************************************************
% Sets dPETSTEP simulation parameters.
%
% USAGE  : simSet = Dynamic_setSimParameters(frame,Cif)
%
% INPUT  : frame    Vector with start and end frame times in sec,
%                   [frameStart1; frameStart2=frameEnd1; frameStart3=frameEnd2;...].
%          Cif      Vector with input function to model (AIF or reference tissue TAC).
%
% OUTPUT : simSet  Structure with all simulation settings
%**************************************************************************************************
% IH, 19/04/2016
%
% Copyright 2016, C. Ross Schmidtlein, on behalf of the dPETSTEP development team.
%
% This file is part of the Dynamic PET Simulator for Tracers via Emission Projection (dPETSTEP) software.
%
% dPETSTEP development has been led by:  Ida Häggström, Bradley J. Beattie and C. Ross Schmidtlein.
%
% dPETSTEP has been financially supported by the Cancer Research Foundation in Northern Sweden,
% the US National Institutes of Health and the National Cancer Institute under multiple grants.
%
% dPETSTEP is distributed under the terms of the Lesser GNU Public License.
%
%     This version of dPETSTEP is free software: you can redistribute it and/or modify
%     it under the terms of the GNU General Public License as published by
%     the Free Software Foundation, either version 3 of the License, or
%     (at your option) any later version.
%
% dPETSTEP is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
% without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
% See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with dPETSTEP.  If not, see <http://www.gnu.org/licenses/>.
%**************************************************************************************************

%% Simulation parameters
simSet = struct;

% input parameters
simSet.CTscanNum  = 1;                 % CT  scan's ID : should be automated
simSet.PIMscanNum = 2;                 % Parametric image scan's ID : should be automated
simSet.PTscanNum  = 3;                 % Dynamic PET image scan's ID : should be automated

% count data
simSet.countSens      = (265/324)*6.44; % 3D sensitivity (counts/kBq/s) (GE DLS)
simSet.SF             = 0.289;          % scatter fraction S/(T+S)
simSet.RF             = 0.02;           % randoms fraction R/(T+S+R)
```

```
% Dynamic settings
simSet.frame           = frame;          % Vector with frame times in sec.
simSet.dwellTime       = diff(frame);    % Frame lengths.
simSet.Cif             = Cif;            % Vector with input function in Bq/cc. Either arterial input function.
                                         % or reference tissue TAC, depending on what model you use.
simSet.CifScaleFactor = 1;               % Scale factor to multiply the supplied input function with.
simSet.halflife        = 'none';         % Halflife of nuclide in sec, or 'none' for no decay.
simSet.timeStep        = 0.5;            % Convolution time step in sec.
simSet.interpMethod    = 'linear';       % Interpolation method.
simSet.kineticModel    = '2-Tissue';     % Desired kinetic model, '1-Tissue', '2-Tissue', 'FRTM', 'SRTM' or 'sumExp'.

% scanner charaterisitics
simSet.RingData    = 880;        % the diameter of the scanner ring (GE DLS)
simSet.tanBin      = 336;        % Sets inital projetion data size (GE DLS)
simSet.maxRingDiff= 11;          % Maximum allowed ring difference
simSet.psf         = 5.1;        % Assumes a PSF for the system, uses same for correction. FWHM.
simSet.blurT       = 5.1;        % DLS

% image reconstruction definitions
simSet.fovSize     = 128;        % size of dynamic image FOV in mm. Voxel size = fovSize/simSize.
simSet.simSize     = 128;        % matrix size of reconstructed image
simSet.zFilter     = [1 4 1];    % post recon Z-axis filter 3-point smoothing
                                 % Heavy[ 1 2 1]/4, Standard[1 4 1]/6, Light[1 6 1]/8, None[0 1 0]
simSet.postFilter = 6;           % FWHM in (mm) of post reconstruction filter. FWHM = 2*sqrt(2*log(2))*sigma.
simSet.iterNUM     = 5;          % number of iterations
simSet.subNUM      = 16;         % number of subsets

% Biologic variability
simSet.addVariability    = false;  % Flag to add biologic variability or not
simSet.variabilityScale = 10;      % Scale factor of variability (variance of gaussian noise = image/scale)

% Reconstructions
simSet.FBP_OUT       = false;
simSet.OS_OUT        = false;
simSet.OSpsf_OUT     = false;

% number of replicate data sets
simSet.nREP        = 1;

end
```

2. You need to specify the frames of your simulation as input to Dynamic_setSimParameters; a vector called “frame”, unit (sec). f = number of frames.
   frame = [frameStart_1 ; frameEnd_1=frameStart_2 ; frameEnd_2=frameStart_3 ; … frameEnd_f].
3. For the second input, you specify the input function to your model, a vector called “Cif”. Should be the arterial input function for the 1-, and 2-Tissue models and sum of exponentials, and a reference tissue TAC for the FRTM and SRTM models, in unit (Bq/cc).
   Cif = [Cif_1 ; Cif_2; … ; Cif_f-1].
4. Finally, you need to specify a scale factor to scale number of counts in the sinogram. This factor is a scalar in unit (Bq), e.g. 1e6, and corresponds to the average activity in a frame. This factor determines the level of noise in the sinograms.

To run a simulation, simply run:

>>[data,simSet,FBP4D,OS4D,OSpsf4D,counts,countsNoise,nFWprompts,FWtrues,FWscatters,FWrandoms,wcc]=Dynamic_main_kineticModelling(data, frame, Cif, scaleFactor);

(See section Matlab functions below). This adds a pristine (noiseless) dynamic image based on your parametric image and settings to your structure “data”. It then simulates the dynamic PET from that dynamic pristine input. The data that is created is:

data	Same data structure as input, but with the pristine 4D image added to it.
simSet	Structure with simulation settings.
FBP4D	4D matrix with reconstructed dynamic FBP image.
OS4D	4D matrix with reconstructed dynamic OSEM image.
OSpsf4D	4D matrix with reconstructed dynamic OSEM image with PSF correction.
counts	Structure with the pristine sinogram counts.
countsNoise	Structure with the noisy sinogram counts.
nFWprompts	Noisy prompts sinogram.
FWtrue	Noiseless true sinogram.
FWscatters	Noiseless scatters sinogram.
FWrandoms	Noiseless randoms sinogram.
wcc	Well-counter-calibration factor to get unit Bq/cc.

Dynamic image

If you do not want to start with a parametric image, but instead only simulate a PET acquisition of your own input dynamic image, there is an option for that too.

You can generate the dynamic PET uptake image, followed by PET acquisition simulation, using either the graphical user interface (GUI), or directly using the Matlab codes.

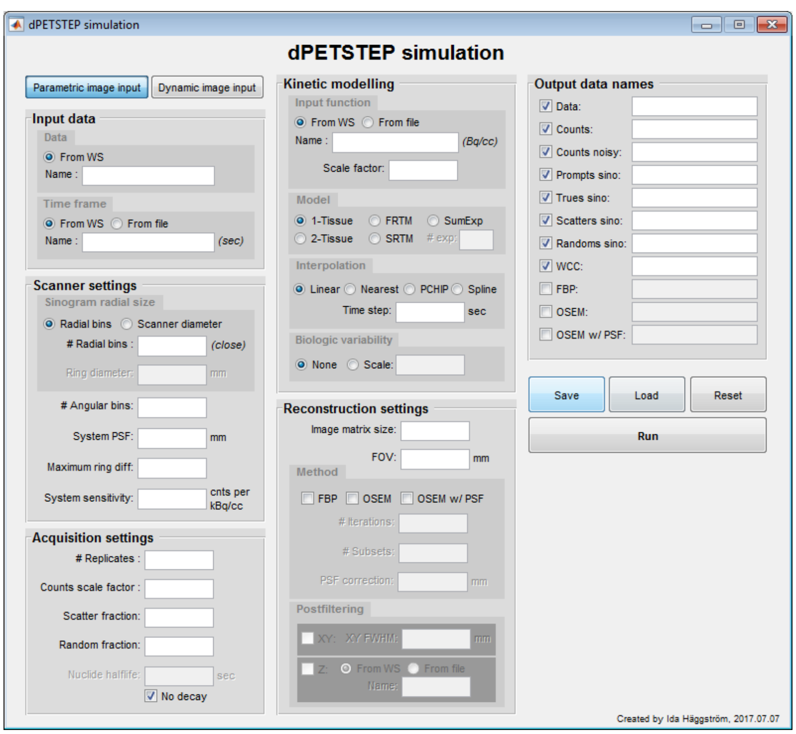
Using GUI

Make sure you've downloaded the GUI files from GitHub.

Either open the main dPETSTEP menu as above, click Simulation, or in the Matlab prompt type

```
>> dPETSTEPgui_sim
```

to open window below. Choose "Dynamic image input". For further instructions on all fields, look at tooltips and the User's guide (Users_guide_dPETSTEP_simulation_GUI.pdf).



Using Matlab files

1. Same as step 1 above under kinetic modeling, using Matlab files. Set the aquisition parameters. Some settings will not be used (the ones related to parametric imaging).
2. Same as step 2 above. Specify what time sampling you used.
3. Same as step 4 above. Specify a scale factor to scale number of counts in the sinogram. This factor is a scalar in unit (Bq), e.g. 1e6, and corresponds to the average activity in a frame. This factor determines the level of noise in the sinograms.

To run a simulation, simply run:

```
>>[data,simSet,FBP4D,OS4D,OSpsf4D,counts,countsNoise,nFWprompts,FWtrues,FWscatters,FWrandoms,wcc]=Dynamic_main_dynamicImage(data, frame, scaleFactor);
```

Kinetic model fitting

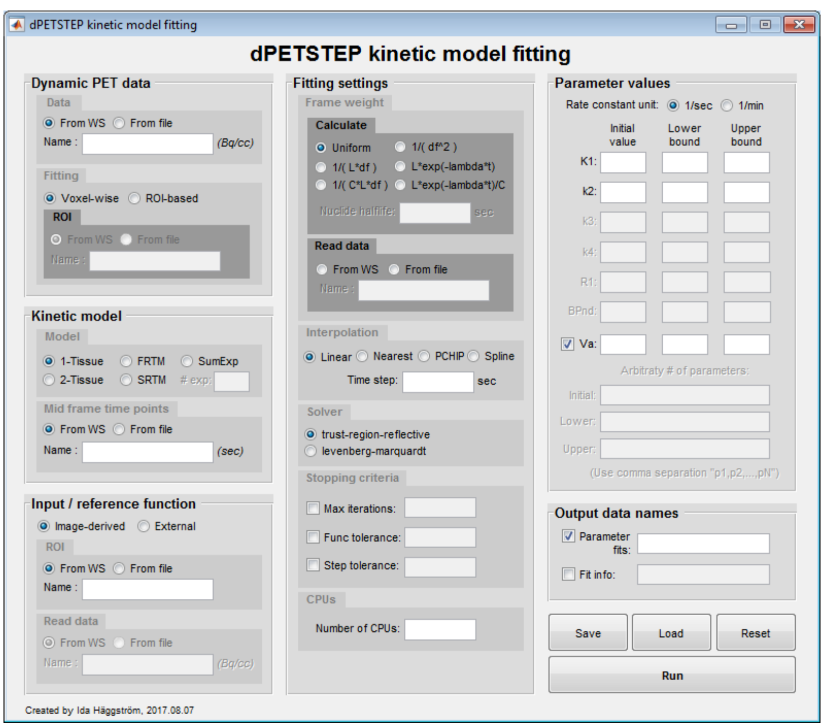
Using GUI

If you want to use the Graphical User Interfaces (GUI), make sure you've downloaded the GUI files from https://github.com/CRossSchmidtlein/dPETSTEP/tree/master/dPETSTEP_graphical_user_interfaces.

Either open the main dPETSTEP menu as above, click Model fitting, or in the Matlab prompt type

```
>> dPETSTEPgui_fit
```

to open window below. For further instructions on all fields, look at tooltips and the User's guide (Users_guide_dPETSTEP_modelFitting_GUI.pdf).



Matlab functions

To run dPETSTEP, you will need to download the MATLAB functions from both dPETSTEP and PETSTEP (found in the PETSTEP repository on github).