ECRTOOLS 0.05
User's Guide

Analysis and Optimization of Electrical Conductivity Relaxation (ECR) Experiments

Francesco Ciucci, HKUST

7/21/2017

Table of Contents

Table of Contents 1

1. Introduction to ECRTOOLS 2

1.1. Where can one find ECRTOOLS? 2

How does one get support? 2

1.2. How does one cite ECRTOOLS? 2

1.3. How do install ECRTOOLS? 2

2. GUI interface explained 5

3. Demos 9

3.1. Demo 1.m – Generating Exact and Synthetic ECR responses 9

3.2. Demo 2.m – Computing the Sensitivity of ECR 10

3.3. Demo_3.m – Computing Asymptotic Confidence Regions and Other Parameters for Optimization 11

3.4. Demo_4.m – Fitting ECR data 13

3.5. Demo_5.m –Running Synthetic Experiments 15

3.6. Demo_6.m – Optimal Experimental Design for ECR 16

3.7. Demo_7.m – Robust Optimal Experimental Design 17

1. Introduction to ECRTOOLS

ECRTOOLS serves the Electrical Conductivity Relaxation (ECR) users. ECRTOOLS is a Matlab toolbox that you can use to:

1. Plot and analyze ECR data;
2. Extract from ECR data a fit of the surface exchange coefficients k and diffusion coefficients D;
3. Assess the quality of the fit by establishing posterior asymptotic confidence regions with respect to the two parameters;
4. Establish the sensitivity of ECR measurements;
5. Optimize, even in a robust manner, the ECR data.

The functionalities of the ECRTOOLS can be accessed from:

- GUI interface
- Matlab Scripts (the demos are a good starting point)

The GUI interface allows users to have a visual understanding of ECR, of its sensitivity, of the confidence region of the parameters as a function of various inputs such as parameters k and D, system size, number of measurement points, error input and measurement range. The GUI also allows one to fit ECR data, to visually check how good the fit is and to predict the quality of the estimated k and D. The demos are more advanced and they allow users to access the full functionalities of the toolbox, such as classical and robust Optimal Experimental Design (OED)

1.1. Where can one find ECRTOOLS?

The ECRTOOLS can be freely downloaded from the webpage

https://sites.google.com/site/ecrtools

How does one get support?

Just write to mefrank@ust.hk, I will try to respond as quickly as possible.

1.2. How does one cite ECRTOOLS?

Please cite:

- Francesco Ciucci. Electrical conductivity relaxation measurements: Statistical investigations using sensitivity analysis, optimal experimental design and ECRTOOLS. Solid State Ionics. Volume 239, 15 May 2013, Pages 28-40 https://doi.org/10.1016/j.ssi.2013.03.020
- Ting Hei Wan, Mattia Saccoccio, Chi Chen, and Francesco Ciucci. Assessing the identifiability of k and D in electrical conductivity relaxation via analytical results and nonlinearity estimates. Solid State Ionics. Volume 270, February 2015, Pages 18-32. https://doi.org/10.1016/j.ssi.2014.11.026

Other articles are forthcoming

1.3. How do install ECRTOOLS?

ECRTOOLS is now designed to work with Windows 10 and Matlab's versions > 2014a.

ECRTOOLS was developed to work with the OPTimization Interface (OPTI) Toolbox, a free and powerful

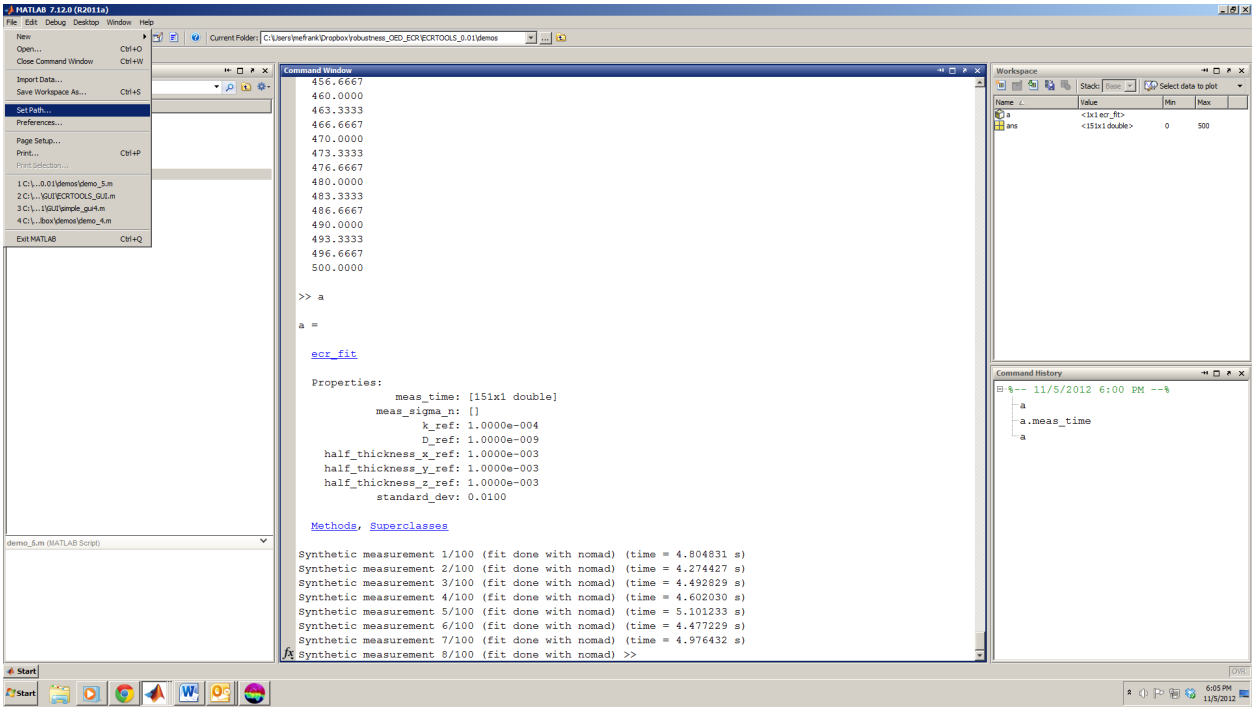
optimization toolbox

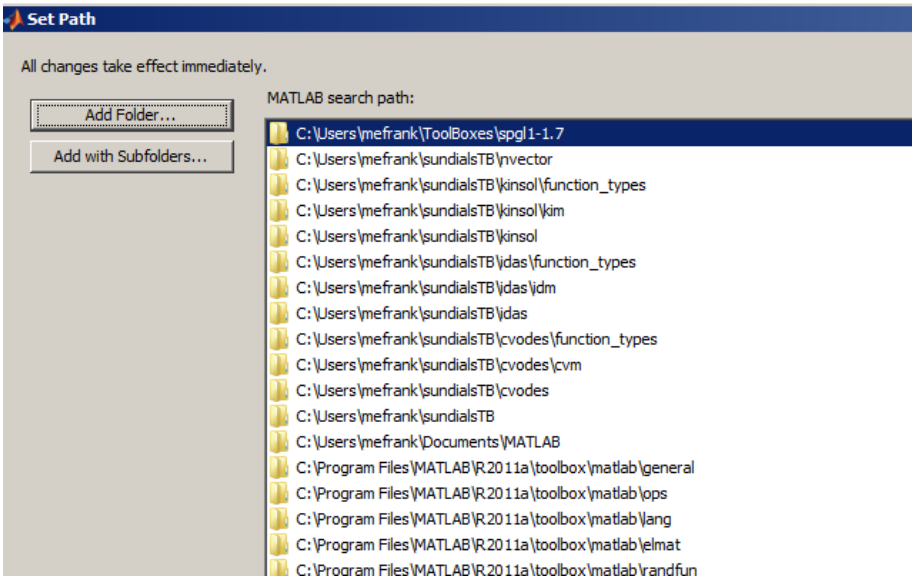
https://www.inverseproblem.co.nz/OPTI/ (currently at: https://github.com/jonathancurrie/OPTI)

The installation of OPTI Toolbox is simple and requires that one downloads it and then that one runs the script opti_Install.m from the Matlab command window starting from its installation directory.

In order to install ECRTOOLS, one simply needs to add the ECRTOOLS folder and subfolders to path.

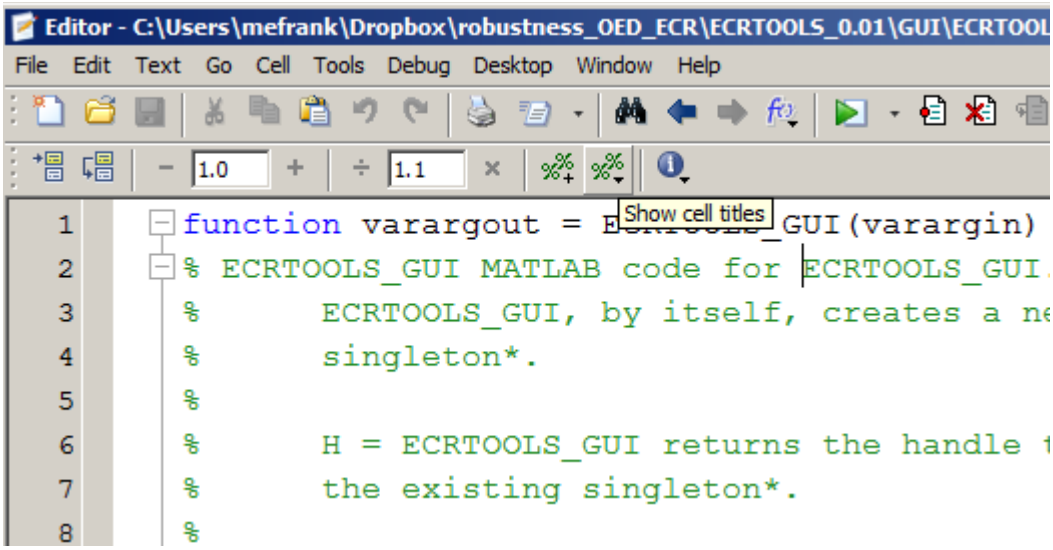
This can be done from by accessing the setpath option under the file folder and then adding all subfolders of the main ECRTOOLS distribution. The steps necessary for installation are shown using the screenshots below



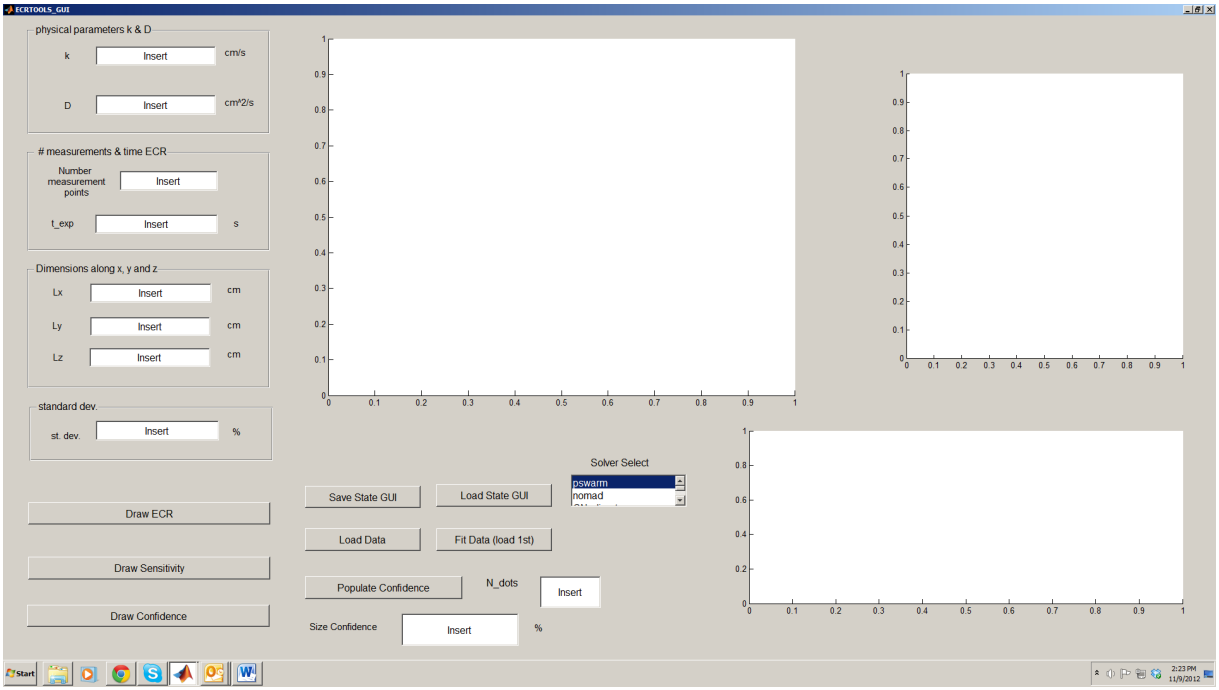

Set Path
All changes take effect immediately.
Add Folder...
Add with Subfolders...
MATLAB search path:
C:\Users\mefrank\ToolBoxes\spgl1-1.7
C:\Users\mefrank\sundialsTB\nvector
C:\Users\mefrank\sundialsTB\kinsol\function_types
C:\Users\mefrank\sundialsTB\kinsol\kim
C:\Users\mefrank\sundialsTB\kinsol
C:\Users\mefrank\sundialsTB\idas\function_types
C:\Users\mefrank\sundialsTB\idas\idm
C:\Users\mefrank\sundialsTB\idas
C:\Users\mefrank\sundialsTB\cvodes\function_types
C:\Users\mefrank\sundialsTB\cvodes\cvm
C:\Users\mefrank\sundialsTB\cvodes
C:\Users\mefrank\sundialsTB
C:\Users\mefrank\Documents\MATLAB
C:\Program Files\MATLAB\R2011a\toolbox\matlab\general
C:\Program Files\MATLAB\R2011a\toolbox\matlab\ops
C:\Program Files\MATLAB\R2011a\toolbox\matlab\lang
C:\Program Files\MATLAB\R2011a\toolbox\matlab\elmat
C:\Program Files\MATLAB\R2011a\toolbox\matlab\randfun

2. GUI interface explained

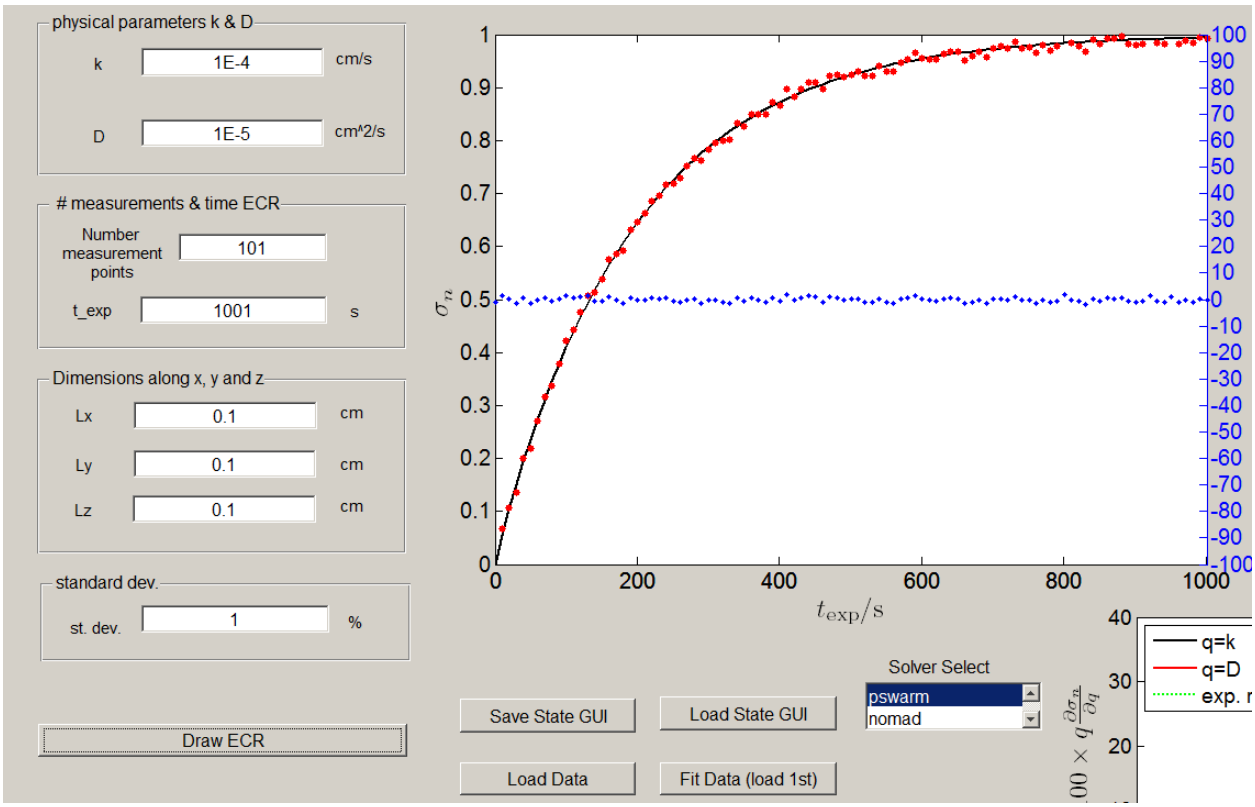
The GUI interface implementation is located in subfolder GUI of the main distribution. Inputting ECRTOOLS_GUI into the command window opens up the GUI. One may also run the GUI by clicking on the green button of the ECRTOOLS_GUI.m file or by simply pushing F5 when browsing the file



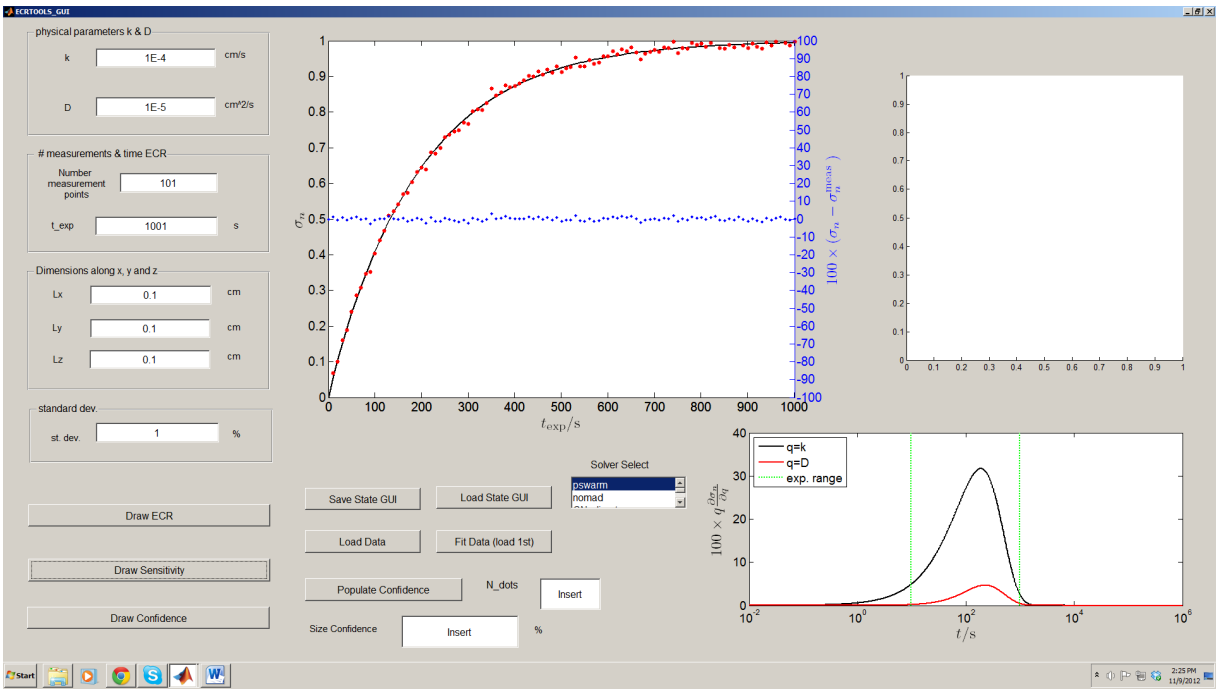
The GUI input data need to be filled up to exploit its functionalities.



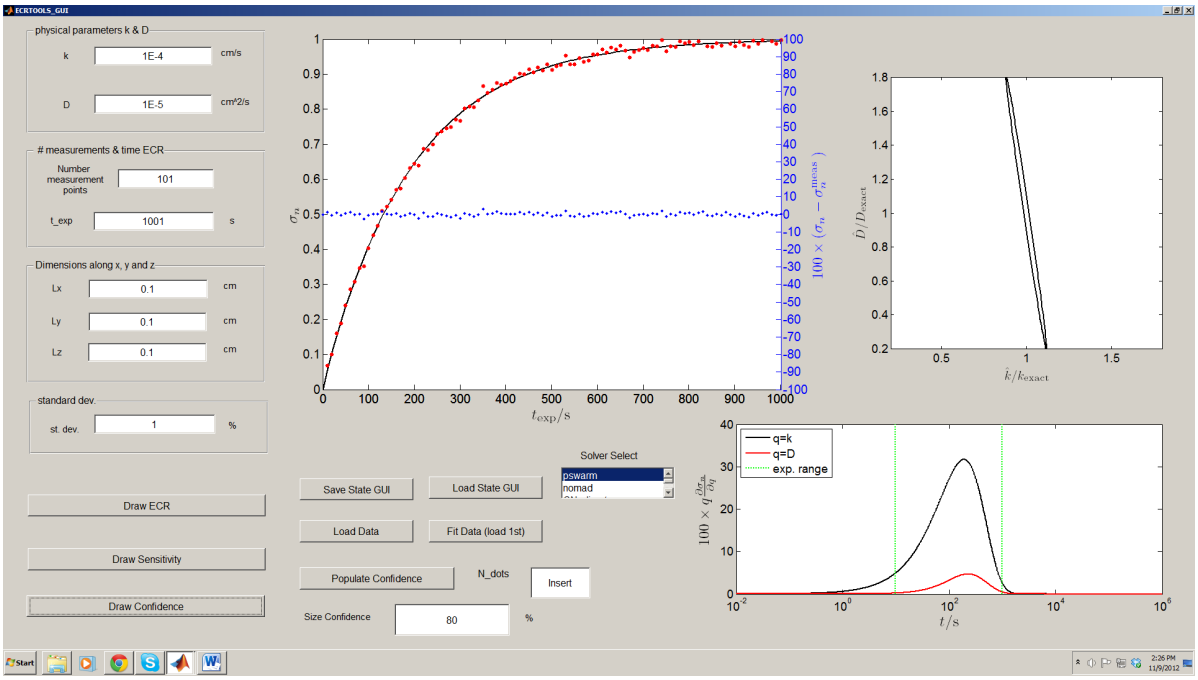
For example one can first input the physical parameters and push “Draw ECR” button, which outputs the exact ECR response (black), synthetic experimental points (red dots) and residual (underlying error).



One can also plot the sensitivity for the input parameters by pushing on the “Draw Sensitivity” button.

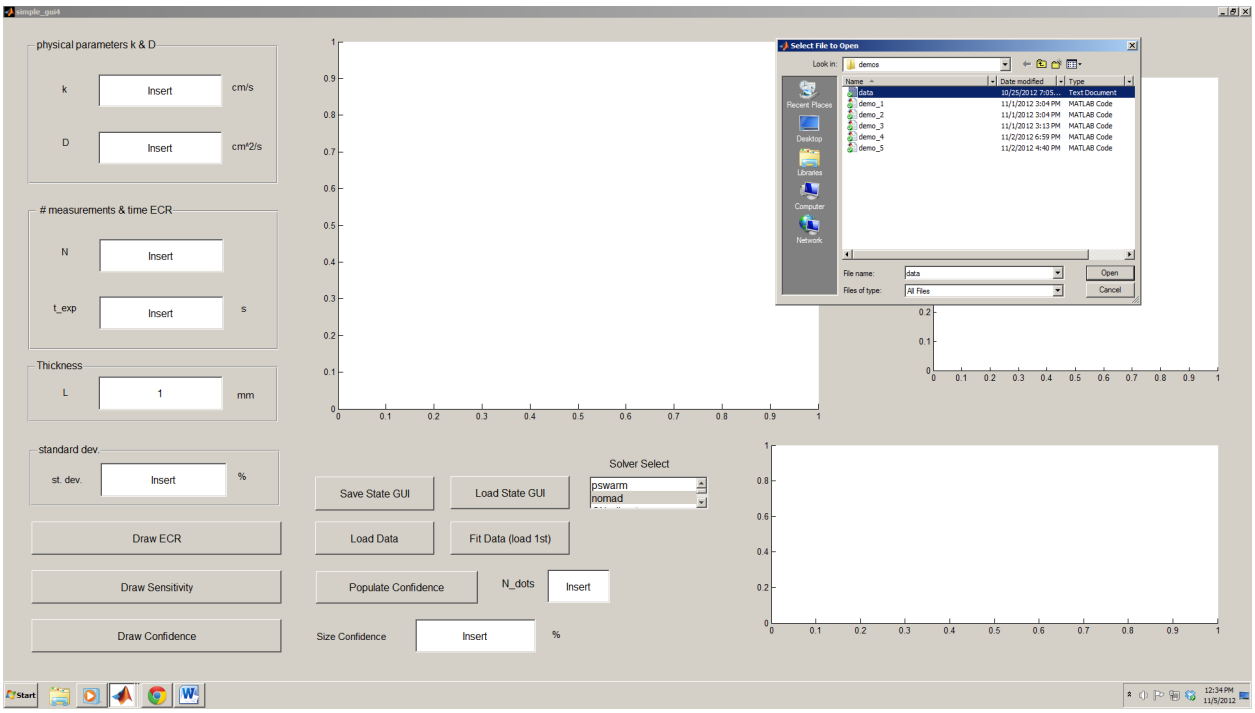


One can also plot the asymptotic ellipsoidal confidence region by pushing on the “Draw Confidence” button (one can tweak the size of the region by modifying the size confidence input)

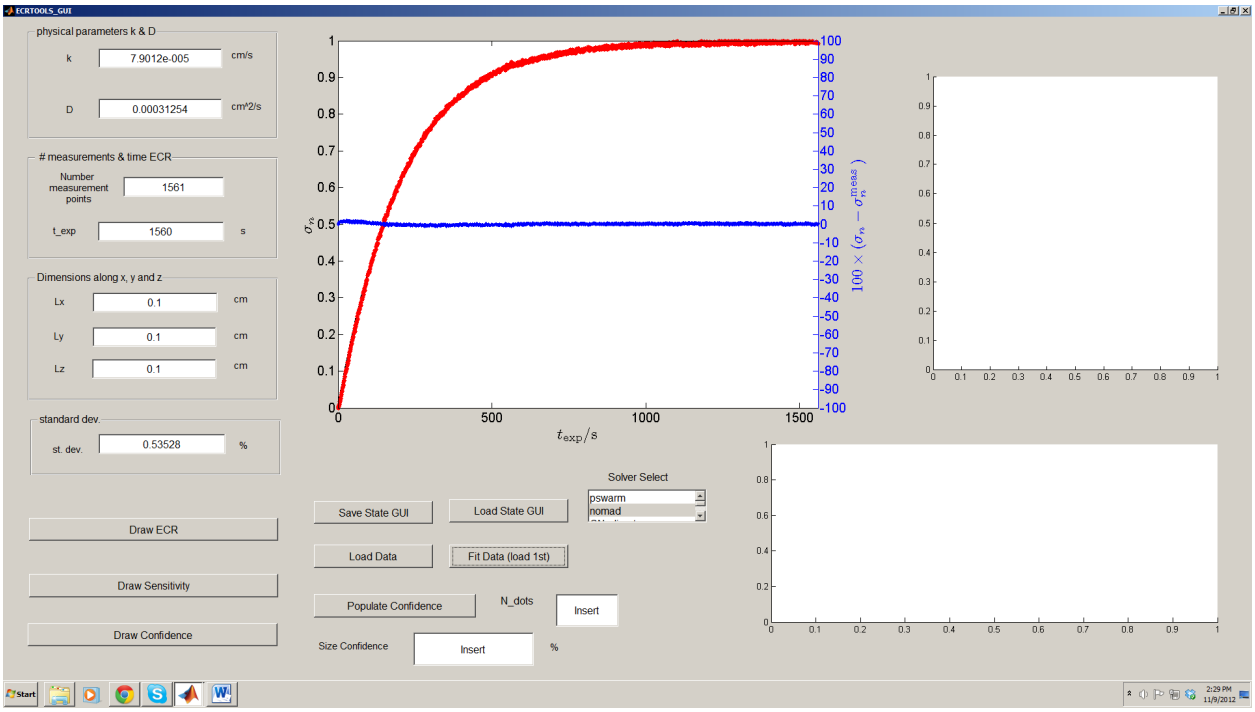


On can save the state of the GUI by pressing the “Save State GUI” button and upload the latest saved state by pressing the “Load State GUI” button.

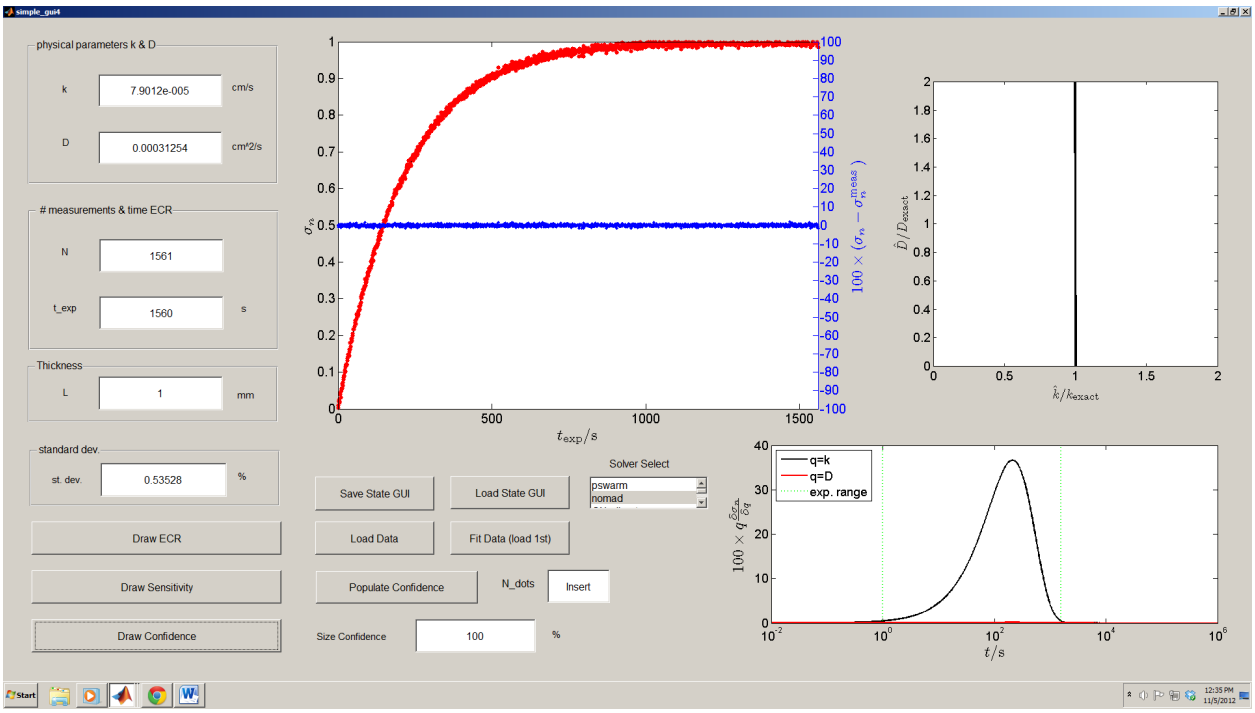
ECRTOOLS_GUI can also be used to fit data. First, one needs to press “Load Data” in order to select a suitable data file; it is necessary that one defines the sample thickness and solver used (in the screenshot nomad was ch osen)



Pushing the “Fit Data” button provides the results of the fitting exercise and outputs to screen the estimated k and D along with the estimated standard deviation. This is also plotted to screen.



The loaded data can also be used to generate synthetic experiments (this overrides the previous ECR plot), sensitivity analysis and confidence ellipsoidal regions. From the data provided one can see that the experiment is sensitive to k but not to D and this is reflected in the asymptotic confidence region which is narrow along the k direction and wide along the D direction.



3. Demos

3.1.Demo 1.m – Generating Exact and Synthetic ECR responses

Demo_1 shows you how to use the ECR_base class and how to output an exact, σ_n, and synthetic, σ_n^{meas}, ECR response. The exact ECR response depends on the parameters k (surface exchange coefficient) and D (diffusion coefficient), and it is plotted within the experimental timespan. The synthetic ECR response is a random (stochastic) process which depends on two additional parameters, the number of measurement points N and the error made at each measurement point. We take the error to be Gaussian and with zero mean (no systematic errors), this means that the standard deviation is sufficient to describe the measurement error.

First you need to define an object of the class ecr_base

```
% define class
a = ecr_base;
```

then you load into the class the parameters of the ECR

```
% input into class the value of k and D
a.k_ref = 1.5E-4; % unit of k is m/s
a.D_ref = 2.5E-9; % unit of D m^2/s

% input into class the half size of the sample
a.half_thickness_x_ref = 1E-3; % unit of x-thickness is m
a.half_thickness_y_ref = 1E-3; % unit of y-thickness is m
a.half_thickness_z_ref = 1E-3; % unit of z-thickness is m

% upload the error of made in the measurement:
% standard deviation = 0.01
a.standard_dev = 1E-2;
```

Subsequently you define the timespan at which the ECR occurs

```
% Define Timespan
N = 101;
time = linspace(0, 1E2, N)'; % Maximum time is
100 s
% note that you will have one measurement every
second
```

Then you produce the exact response σ_n

```
% compute the exact normalized conductivity
\sigma_n
sigma_n = a.sigma_n_det(time);
```

Note that up to this point you have not used the standard deviation/error structure.

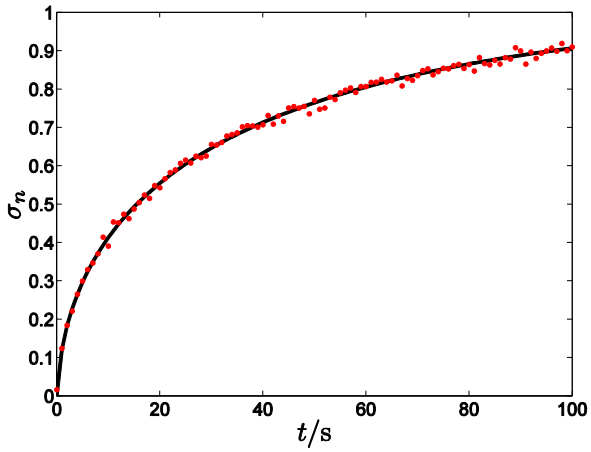


Figure 1 – Output of Demo1.m: exact (solid black line) and synthetic (dots) ECR response.

Then you produce a synthetic measurement σ_n^{meas}

```
% synthetic measurement
sigma_n_meas = a.sigma_n_meas(time);
```

You can plot both σ_n and σ_n^{meas} to figure to obtain Figure 1.

3.2.Demo 2.m – Computing the Sensitivity of ECR

Demo_2 shows how to use the ECR_sens class and how to output the sensitivity of the ECR response, which is the normalized derivative of the exact response with respect to the parameter of choice. Specifically the output is given by $k\frac{\partial\sigma_n(t,k,D)}{\partial k}$ and $D\frac{\partial\sigma_n(t,k,D)}{\partial D}$. For a fixed set of physical conditions, the sensitivity depends on the size of the system and on the timespan. In addition the sensitivity is linked to the identifiability as follows: the more sensitive is the measurement, the greater is the confidence on the experimental result.

In demo_2, first you need to seta to be in the class ecr_sens, linked to the parent class ecr_base

```
% define class
a = ecr_sens;
```

then you need to load into the class the parameters of the ECR

```
% input into class the value of k and D
a.k_ref = 1.5E-4; % unit of k is m/s
a.D_ref = 2.5E-9; % unit of D m^2/s

% input into class the half size of the sample
a.half_thickness_x_ref = 1E-3; % unit of x-thickness is m
a.half_thickness_y_ref = 1E-3; % unit of y-thickness is m
a.half_thickness_z_ref = 1E-3; % unit of z-thickness is m
```

Subsequently, wedefine the timespan at which we compute the gradient

```
% Define timespan for gradient computation

N = 1000;
t_min = 1E-2; % Minimum time is 1E-2 s
t_max = 1E4; % Maximum time is 1E4 s
time      =      logspace(log10(t_min),
log10(t_max), 1000)';
```

Then we produce the exact response σ_n

```
grad_out = a.grad_sigma_n(time);
```

and we can plot it in Figure 2. Note from Figure 2 that the ECR response is more sensitive to D than it is to k, note also that we have not used the standard deviation/error structure.

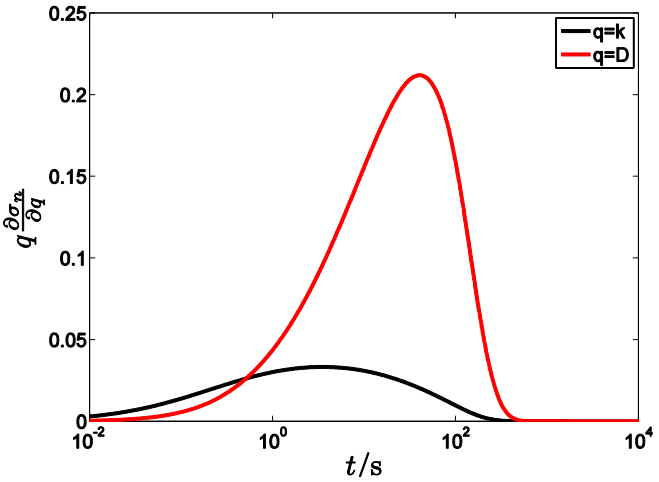


Figure 2 - Sensitivity of the exact ECR response with respect to the two parameters k and D.

3.3.Demo_3.m – Computing Asymptotic Confidence Regions and Other Parameters for Optimization

In this tutorial various functions of the asymptotic covariance matrix $\mathbf{V}$ are computed. Those functions are determinant $\det(\mathbf{V})$, trace $\mathrm{tr}(\mathbf{V})$, condition number $\kappa(\mathbf{V})$, and maximum eigenvalue $\max(\lambda(\mathbf{V}))$.

```
% define class
a = ecr_sens;
a.k_ref = 1.5E-4; % unit of k is m/s
a.D_ref = 2.5E-9; % unit of D m^2/s
a.standard_dev = 1E-2;
```

Define the experiment to go from 0 to 10,000s

```
t_min = 0.0;
t_max = 1E4;
```

and the time to be

```
time = linspace(t_min, t_max, 1E4+1)'; % Maximum time is 10,000 s
```

Define the size of the system to be

```
% vector of half-sizes
half_thickness_min = 1E-4; % 1E-4 m = 100 \mu m
half_thickness_max = 1E-2; % 1E-2 m = 1 cm
half_thickness_vec              =              logspace(log10(half_thickness_min),
log10(half_thickness_max), 100);
```

Define the size of the

```
det_cov_sigma_n_vec = zeros(size(half_thickness_vec));
trace_cov_sigma_n_vec = zeros(size(half_thickness_vec));
cond_cov_sigma_n_vec = zeros(size(half_thickness_vec));
max_eig_cov_sigma_n_vec = zeros(size(half_thickness_vec));

for iter_half_thickness = 1 : numel(half_thickness_vec)

    % input into class the size of the sample
    a.half_thickness_x_ref = half_thickness_vec(iter_half_thickness);
    a.half_thickness_y_ref = half_thickness_vec(iter_half_thickness);
    a.half_thickness_z_ref = half_thickness_vec(iter_half_thickness);

    det_cov_sigma_n_vec(iter_half_thickness) = a.det_cov_sigma_n(time);
    trace_cov_sigma_n_vec(iter_half_thickness) = a.trace_cov_sigma_n(time);
    cond_cov_sigma_n_vec(iter_half_thickness) = a.cond_cov_sigma_n(time);
    max_eig_cov_sigma_n_vec(iter_half_thickness) =…
      a.max_eig_cov_sigma_n(time);

end
```

The results of the computation are shown in the figure below in Figure 3

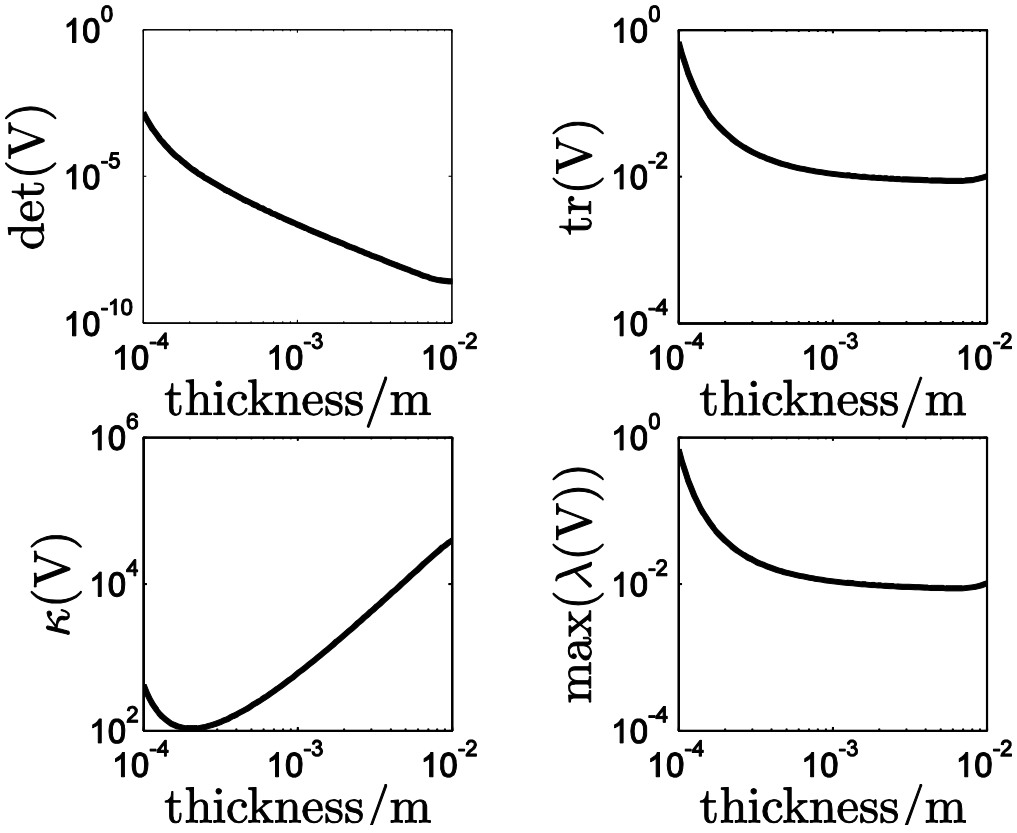


Figure 3 - Plot of various functions of the asymptotic covariance matrix as a function of thickness between 0.01 cm and 1 cm

3.4.Demo_4.m – Fitting ECR data

ECRTOOLS can also be used to

1. fit experimental data

2. plot the outcome of the fit by showing simultaneously experimental data, fit and residual

3. plot the asymptotic confidence region

The steps described in detail below.

First, we load the ECR experimental data

```
ECR_data = load('data.txt');
```

Second. we load the experimental time array into a Matlab vector, the initial time is set to 0

```
time_exp = ECR_data(:,1)-ECR_data(1,1); % measurement_time
```

save the sigma_n_ext

```
sigma_n_exp = ECR_data(:,2); % measured \sigma_n
```

Third, we load the ECR all data into an object in the ecr_fit class (guess the value of k_ref and D_ref)

```
a = ecr_fit;

a.k_ref = 1E-4; % unit of k is m/s
a.D_ref = 1E-6; % unit of D m^2/s

% input into class the half size of the sample
a.half_thickness_x_ref = 1E-3; % unit of half x-thickness is m
a.half_thickness_y_ref = 1E-3; % unit of half y-thickness is m
a.half_thickness_z_ref = 1E-3; % unit of half z-thickness is m

% upload the error of made in the measurement:
% standard deviation = 0.01
a.standard_dev = 1E-2; % this will be recomputed later
```

We also load experimental data (this is different from previous cases)

```
a.meas_time = time_exp;
a.meas_sigma_n = sigma_n_exp;
```

We then perform estimation over a large set

```
% first run fit within a large span
theta_out = a.log10_fit();
```

We then update the value of the parameters

```
% update the reference value
a.k_ref = theta_out(1)*a.k_ref;
a.D_ref = theta_out(2)*a.D_ref;
```

and subsequently we refine the estimation over a smaller set & update the value of the parameters

```
theta_out = a.fit('pswarm');
a.k_ref = theta_out(1)*a.k_ref;
```

```
a.D_ref = theta_out(2)*a.D_ref;
```

We store the fitted sigma_n into the workspace

```
sigma_computed = a.output_sigma_n_det(theta_out);
```

We then compute the residual (for plotting purposes)

```
error_meas = (sigma_computed-sigma_n_exp);
```

We compute the standard deviation of the measurement and load it to the a variable

```
std_computed = a.compute_std_meas();
a.standard_dev = std_computed;
```

We plot the experimental data against fit and residual along with the 3σ confidence bands (output in Figure 4)

```
[AX,H1,H2] = plotyy(time_exp, sigma_computed, time_exp, 100*error_meas,'plot','plot');
hold on
plot(time_exp, sigma_n_exp, '+r', 'MarkerSize', 4.5)
plot([0, max(time_exp)], [3*std_computed, 3*std_computed]+0.5, '--g', 'Linewidth', 3);
plot([0, max(time_exp)], [-3*std_computed, -3*std_computed]+0.5, '--g', 'Linewidth', 3);
```

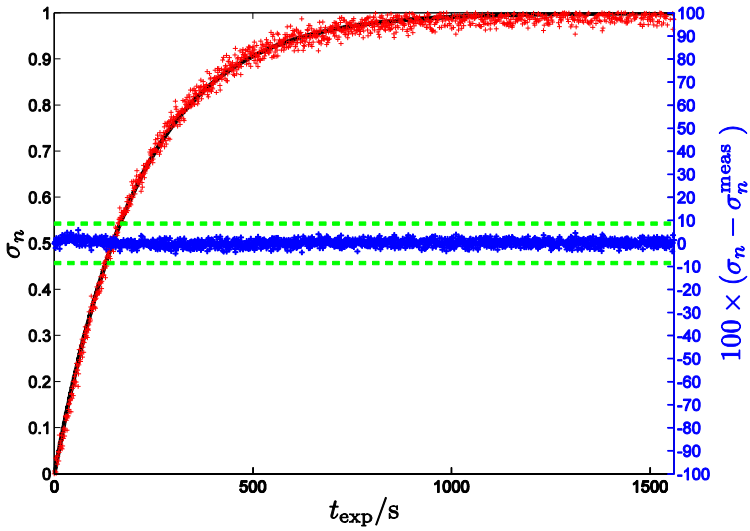


Figure 4 – Corrupted ECR experimental data (red line + symbols) plotted against the fitted ECR (black solid line) and concurrently with the residual (blue line) and its 3 σ band.

Based on this data, an approximation of the confidence region of the parameters can also be shown. This provides a pictorial view of the asymptotic confidence region of the two parameters k and D. If the experiments are repeated, the confidence region is expected to contain most of the values of those two parameters. In reference to the values estimated, one can see that k is identifiable while D is not.

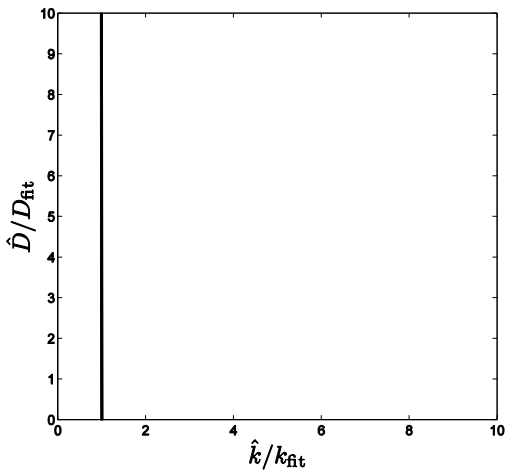


Figure 5 - Confidence region for the ECR data

3.5.Demo_5.m –Running Synthetic Experiments

The demo describes how to:

1. Perform synthetic experiments

2. Fit the synthetic experiments

3. Plot the outcome of the synthetic experiment fits and output that to figure along with the asymptotic covariance matrix

First one defines the experimental time

```
% define experimental time
t_exp = linspace(0, 500, 151)';
```

then one declares the working instrument for the fit, that is the class ecr_fit

```
a = ecr_fit;
a.k_ref = 1E-4; % k - unit of k is m/s
a.D_ref = 1E-9; % D - unit of D m^2/s
a.half_thickness_x_ref = 1E-3; % half
thickness - unit is m
a.half_thickness_y_ref = 1E-3; % half
thickness - unit is m
a.half_thickness_z_ref = 1E-3; % half
thickness - unit is m
a.standard_dev = 1E-2; % 1% error
a.meas_time = t_exp; % experimental
time - unit is s
```

with this info one can compute the covariance matrix and the boundary of the confidence region

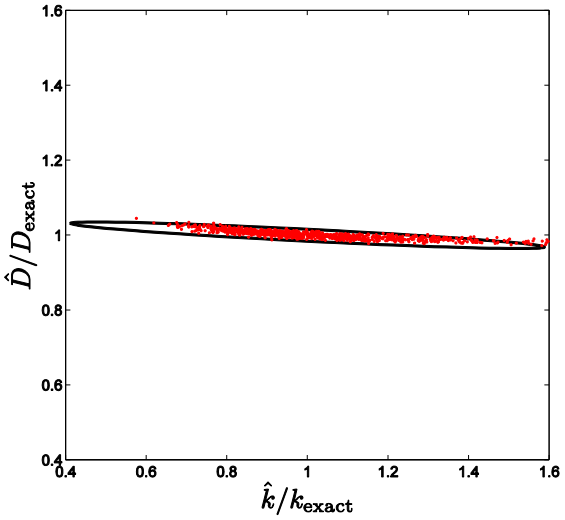


Figure 6 - Outcome of the fits of 1000 synthetic experiments.

```
boundary_cov =
a.boundary_cov_sigma_n(a.meas_time);
```

then one can run 1000 synthetic experiments and output the result to screen,

```
theta_vec = a.synthetic_exp('nlopt', 1);
```

the outcome of the fit of each synthetic experiment is shown in Figure 6 and compared to the asymptotic confidence region.

3.6.Demo_6.m – Optimal Experimental Design for ECR

The demo shows how to optimize ECR experiments, provided that:

1. measurement error is available
2. an estimate for the experimental parameters k and D are available

We shall first define the bounds for the experimental design

```
% what time span is acceptable?
time_exp_bound = [1E2, 1E4];
% which thicknesses are ok?
half_thickness_bound = [5E-4, 5E-2];
```

Then we fix the number of measurements per second

```
N_meas_per_second = 1;
```

and establish the criterion used for optimization

```
% in this case, select the determinant
criterion = 'det';
% other options are
% criterion = 'trace';
% criterion = 'max_eig';
% criterion = 'cond';
```

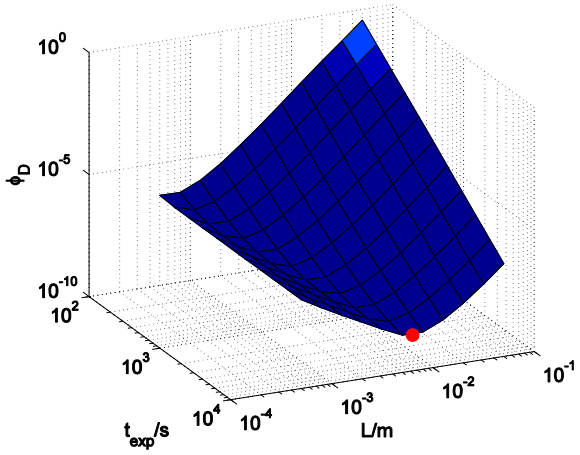


Figure 7 - Minimum for the number per second problem

Then we solve the problem

```
[time_det_opt, thickness_det_opt, det_opt] =
a.optimize_exp_Nps(time_exp_bound,
half_thickness_bound,      N_meas_per_second,
criterion, 3);
```

If instead the total number of measurements is fixed (rather than the number of measurements per second), then we declare N_tot (number of time points)

```
N_tot = 1E2;
% we define the criterion and optimize _Ntot
= number total points
criterion = 'det';
[time_det_opt2, thickness_det_opt2, det_opt2]
=         a.optimize_exp_Ntot(time_exp_bound,
half_thickness_bound, N_tot, criterion, 3);
```

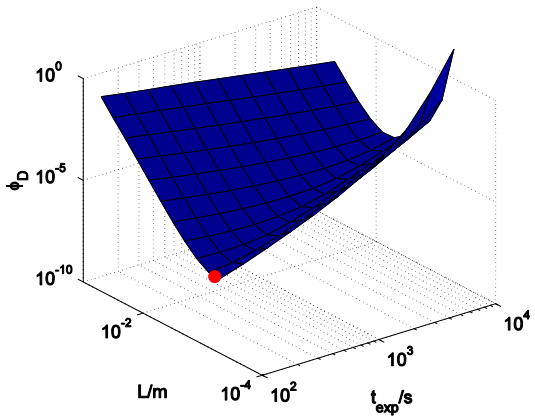


Figure 8 - Minimum for the total number of points.

The results are checked by plotting them against the admissible set as shown in Figure 7 (number of measurements per second) and Figure 8 (number of total points in the timespan assigned).

3.7. Demo_7.m – Robust Optimal Experimental Design

The demo shows how to optimize robustly ECR experiments, provided that:

1. measurement error is available
2. an estimate for the range of the experimental parameters k and D is available

First the bounds for the design need to be defined

```
% define the bounds for the experimental design
% what time span is acceptable?
time_exp_bound = [5E2, 5E4];

% which thicknesses are ok?
half_thickness_bound = [5E-4, 5E-2];
```

Second, the acceptable span for the ECR parameters needs to be defined

```
% expected span for the robust design
k_int = linspace(0.1*a.k_ref,a.k_ref, 11);
D_int = linspace(a.D_ref,10.*a.D_ref, 11);
```

Third, an optimization criterion needs to be selected

```
% in this case, select the determinant
criterion = 'det';
```

Fourth, we robustly optimize for _Ntot = number total points, both in the average sense

```
 [time_opt_avg_Ntot,        thickness_opt_avg_Ntot,        goal_opt]        =
a.optimize_avg_robust_Ntot(time_exp_bound,    half_thickness_bound,    N_tot,
criterion, k_int, D_int);

(note that the 2D integral is computed by quadrature using k_int and D_int) and
in the minmax sense

[time_opt_minmax_Ntot,      thickness_opt_minmax_Ntot,      goal_opt]       =
a.optimize_minmax_robust_Ntot(time_exp_bound,   half_thickness_bound,   N_tot,
criterion, k_int, D_int);
```

The same procedure can be repeated for fixed number of measurements per second

```
[time_opt_avg_Nps,         thickness_opt_avg_Nps,         goal_opt]         =
a.optimize_avg_robust_Nps(time_exp_bound,                half_thickness_bound,
N_meas_per_second, criterion, k_int, D_int);

[time_opt_minmax_Nps,      thickness_opt_minmax_Nps,      goal_opt]       =
a.optimize_minmax_robust_Nps(time_exp_bound,             half_thickness_bound,
N_meas_per_second, criterion, k_int, D_int);
```

Warning: demo_7.m is slow!