

# POPT - Installation and user guide.

## Table of contents

[Pre-Amble](#)

[Installation](#)

[The POPT Suite of Programs](#)

[To run POPT and evaluate sampling windows](#)

[Input files for running POPT and evaluate sampling windows](#)

[Interpreting Output](#)

[P2NM](#)

[Optimization Algorithms](#)

[Copyright Statement](#)

[Bibliography](#)

## Pre-amble

[BACK TO TOP](#)

POPT is a set of programs that have been written for MATLAB to help in the design of clinical studies in which PK or PKPD data is to be collected for the intention of developing models. POPT has mostly been written and developed in MATLAB ver 6.x, but is backwardly and upwardly compatible with MATLAB ver 5.x and ver 7.x. POPT does not require any specialised MATLAB toolboxes to be available. The provision of toolboxes will not improve the performance of POPT code as it is currently written, but it is likely that some amendments could be made to increase performance.

The programs will run on any operating system platform (e.g. Windows, Linux ...) under which MATLAB has been successfully installed. We have tried both Windows and Linux without problems.

POPT is available free of charge and may be downloaded at <http://www.uq.edu.au/pharmacy/sduffull/popt.htm>

POPT is subject to copyright. Please refer to section xx for details of the copyright license.

## Installation

[BACK TO TOP](#)

There are no special instructions for installation of POPT. POPT is made available in a zipped format which contains a directory of files "POPT Distribution copy" and a subdirectory "Example files". These directories can be unzipped to any location on your computer.

It is recommended that you then copy the contents of the directory "POPT Distribution copy" to a new directory in your current work space for MATLAB in order to ensure that you do not inadvertently change any of the download files. From your Current Directory in MATLAB you can highlight and run the appropriate programs.

## The POPT suite of programs

[BACK TO TOP](#)

There are 21 objects in the main directory of POPT.

### Programs for optimal design - DO NOT EDIT THESE

|                          |  |
|--------------------------|--|
| disp_2_scrn_sa.m         | graphical interim results of simulated annealing algorithm   |
| display_to_screen.m      | graphical interim results of exchange algorithm  |
| exch_setp.m              | initialization of the exchange algorithm   |
| exchange.m               | exchange algorithm   |
| exchange_with_v6_patch.m | this is the same as the exchange algorithm but is designed to work with version 6 of MATLAB (or any version that does not have access to the subroutine unidrnd()). If the usual version of the exchange algorithm does not work then rename exchange.m to exchange_v7.m and rename exchange_with_v6_patch.m to exchange.m |
| p_error.m                | checks for some user errors in POPT_INI.m  |
| p_expect.m               | computes expectation of det(information matrix) over upper and lower bound   |
| p_nars.m                 | searches for the max(det(information matrix)) by random sampling   |
| p_output0.m              | produces output to screen and file for single model optimizations  |
| p_output1.m              | produces output to screen and file for multiple model optimizations  |
| p_sa_a.m                 | simulated annealing algorithm for approximate design   |
| p_sa_dn.m                | simulated annealing algorithm for optimizing dose number for samples   |
| p_sa_m8.m                | general simulated annealing algorithm  |
| pop_run.m                | sets up matrices for computation of the information matrix   |
| pop_runs.m               | evaluates information matrix and computes its determinant  |
| popt.m                   | RUNS POPT (RUN THIS FILE)  |

### Programs for setting up optimal design - EDIT THESE AS REQUIRED

|            |  |
|------------|--|
| P_MODEL.m  | enter your structural model here                       |
| POPT_INI.m | enter your design specification and model details here |

### Programs for evaluating sampling windows - DO NOT EDIT THESE

|                 |   |
|-----------------|---|
| p_robustst_1.m  | RUNS sampling window estimation (RUN THIS FILE) |
| p_robust_popt.m | sets up sampling window estimation              |
| repeat_part_3.m | resamples over sampling windows                 |

### Programs for evaluating sampling windows - EDIT AS REQUIRED

|                   |  |
|-------------------|--|
| POPT_INI_robust.m | enter your sampling window and design specification and model details here |
|-------------------|--|

## Miscellaneous

variable\_names.txt  
P2NM

a list of variable names  
Creates a NONMEM data file from the current design in  
POPT\_INI.m

## To run POPT

[BACK TO TOP](#)

To run POPT change MATLAB to the current directory that the POPT programs are located and either type POPT at the command window prompt or open POPT and click on the run button.

POPT will then run and produce output to a file which is echoed to the screen. Outputs to files are appended - so no previous runs will be overwritten. This means that all runs will be appended one after the other and the output file can get very large. The output file is generated automatically and its naming is based on the optimization method, e.g. method0.res is the file name when no optimization method is selected (i.e. `method=0`).

To evaluate sampling windows run P\_ROBUST\_1.

## Input files for running POPT

[BACK TO TOP](#)

[Example of POPT\\_INI file](#)

[Example of POPT\\_INI\\_ROBUST file](#)

POPT does not have an advanced or particularly user friendly method for entering your models and designs. The structural model must be written in the file P\_MODEL.m and the design component must be written in the file POPT\_INI.m. Because of this inefficiency it is recommended that you create a new directory (copying all POPT files) for each new design project. The complete design is printed out with the output files and therefore you will always have a record of the design that you tried.

### Examples of P\_MODEL are provided for

- 2 compartment model with first order input
- 1 compartment model with zero order input
- 1 compartment model with first order input defined by ODES
- multiple models where a 1 compartment model and 2 compartment model are written
- multiple response (PK and PD)

Each P\_MODEL.m file contains a detailed description of how to construct the structural model for each of these occasions and further description is not provided here.

### Examples of POPT\_INI are provided for

- multiple models where a 1 compartment model and 2 compartment model are written

- multiple response (PK and PD)

POPT\_INI is a relatively detailed (and therefore complex) input file. A description of each item is provided.

**MULTIPLE MODELS:** This situation arises when you want to optimize a design for more than 1 model at the same time. This may be because

1. you have 2 competing models and are not sure which is true,
2. you have 2 drugs which you want to optimize the design for at the same time,
3. you have 1 drug but 2 different sets of parameter values (e.g. for a paediatric and adult population)

**MULTIPLE RESPONSES:** This situation arises when you have more than 1 response from a model. Consider the framework that  $A \rightarrow B \rightarrow C$ , then C is dependent on B which is dependent on A. The information matrix for C will therefore also have information about parameters that describe the formation of B from A as well as those that describe  $B \rightarrow C$ . Situations where multiple response models occur can include:

1. parent, metabolite PK models
2. plasma, urine PK models
3. PKPD models

[Example POPT\\_INI.m file.](#)

[BACK TO TOP](#)

There are 5 parts to this file.

[Part 1 deals with model features](#)

[Part 2 design features](#)

[Part 3 optimization features](#)

[Part 4 optimization methods](#)

[Part 5 prior information](#)

The description of the POPT\_INI file will be divided into these sections.

**MODEL FEATURES**

[Back to this example](#)

```
weighting=[];
```

Use this function only if you have either multiple models or multiple responses. In this case you need to enter a weighting value for each candidate model. The weightings need not add to 1, but this is convenient. e.g. use:

```
weighting = [0.5 0.5];
```

```
mr = 0;
```

POPT cannot determine the difference between a multiple response model and multiple competing models. Use this flag variable to indicate to POPT which it is. If `mr = 0` then this is a multiple model and if `mr = 1` then it is a multiple response.

```
BETAI=[ ];
```

These are your prior values of the model parameters. If you have more than one set of parameters (multiple models or multiple response models) then enter a row for each model. You cannot have more than 2 rows for a multiple response model. You can have any number of rows for a multiple model. BETAI must be symmetric. If there are an uneven number of parameters between model 1 and model 2 then you need to insert place holders, e.g. notation:

```
BETAI = [1 4 20 1 1  
         1 2 20 4 5];
```

in this example the first row has two 1's as placeholders. Do not use 0's as placeholders.

```
POP_PAR(1,:)=( );
```

Enter the parameter names. This variable is free text. Example use

```
POP_PAR(1,:)=( '          Ka CL V' );
```

If you have 2 models then you will need two rows of names. The total length of the string vector must be the same for each, example use

```
POP_PAR(1,:)=( '          Ka CL V          ' );  
POP_PAR(2,:)=( '          Ka CL V Emax EC50' );
```

here the spaces in POP\_PAR(1,:) are placeholders to make the POP\_PAR matrix square.

```
FXFI=[ ];
```

Use this vector or matrix to fix none, one or more fixed effects parameters (i.e. to remove them from consideration for optimization). Enter the position number of the parameter in BETAI that you wish to fix. For a single model an example use is

```
FXFI=[ 2 ];
```

this will fix parameter 2. If BETAI has more than 1 row then FXFI should also have more than 1 row. You can use 0's as placeholders, example use

```
FXFI=[ 2  
       0 ];
```

in this syntax position 2 of the first row of BETAI is fixed but no positions are fixed in the second row.

```
OI=[ ];
```

OI are the diagonal elements of the variance-covariance matrix of random effects. Only the diagonal elements can be entered, example use

```
OI=[0.25 0.1 0.1];
```

There needs to be an element in OI for every element in BETAI. The value 0.00001 should be used as placeholders for values that are not intended to be estimated - and also FXI should indicate a fixed position.

```
FXI=[ ];
```

Use this vector or matrix to fix none, one or more variance of the between subject random effects parameters (i.e. to remove them from consideration for optimization). Enter the position number of the parameter in OI that you wish to fix. For a single model an example use is

```
FXI=[ 1 ];
```

This fixes position 1 in the OI matrix.

```
BSVMOD=[ ];
```

This vector (or matrix if it is a multiple model) indicates the structural form of the between subject variance-covariance matrix. A 0 indicates an additive (normal) model and a 1 an exponential (lognormal) model. Example use

```
BSVMODI=[1 1 1];
```

The dimensions of BSVMOD must equal the dimensions of OI which must equal the dimensions of BETAI. You can use either a 0 or 1 for a placeholder for BSVMOD - they are simply ignored if not needed (which is determined by FXI).

```
sd_prior=[ ];
```

```
sd_addi=[ ];
```

These two terms are used to provide the prior values for the standard deviation of the residual variance. Either or both additive and proportional error terms can be added. It is always recommended to include an additive term, even without prior evidence, as this ensures that the optimization routine does not equate smaller values of the response variable with greater precision in its estimation, which can lead to the erroneous situation where an infinitely small concentration is without error. An example of use is

```
sd_propi=[0.15];
```

```
sd_addi=[0.1];
```

If you have multiple models then an example of use is

```
sd_propi=[0.15  
          0.2];
```

```
sd_addi=[0.1  
         0.1];
```

These variables need to be the same number of rows as BETAI.

```
FXRI=[ ];
```

Use this vector or matrix to fix none, one or more variance of the residual random effects parameters (i.e. to remove them from consideration for optimization). Enter the position number of the parameter that you wish to fix.

For a single model an example use is

```
FXRI=[ 2 ];
```

This fixes sd\_addi. Whereas:

```
FXRI=[ 1 ];
```

fixes sd\_propi

For a multiple model the syntax is not as obvious but to fix the additive component from model 1 and the proportional component from model 2 you would write:

```
FXRI=[ 2  
       1 ];
```

## DESIGN FEATURES

[Back to this example](#)

`ns =`

`maxs =`

These variables are used with (method=6) and allow for an approximate design to be constructed. `ns` is the total number of subjects in the study and `maxs` is the total number of samples allowed. POPT will then determine the best arrangement of dividing the samples up amongst the patients. Example syntax:

```
ns = 100;  
maxs = 500;
```

The arrangement will be based on the allocation of samples per group provided by NT. This method optimizes the number of patients per group and the sampling times.

`NUM_SUB=[ ];`

This variable is used with all other methods (i.e. `method <> 6`) and replaces `ns` and `maxs`. `NUM_SUB` is used to define the number of subjects in each group. Example use:

```
NUM_SUB=[100 100];
```

Where there are 100 patients allocated to group 1 and 100 patients allocated to group 2. Each group has a different design.

`NT=[ ]`

NT defines the number of sampling times per group. Example use:

```
NT=[3 3];
```

Indicates that group 1 has 3 sampling times as does group 2.

`D=[ ];`

This signifies the dose to be used for each group. Example use:

```
D=[100 100];
```

If you have multiple models (or a multiple response model) then dose needs to be entered for each model. This is especially important for multiple models where the models may represent different drugs. Example use:

```
D=[100 100  
100 100];
```

`dose_zero_fix=;`

This is a special feature that allows the user to set the information matrix to a matrix of zeros when no dose is to be given. This is of importance when model responses are present in the absence of dosing (e.g. a baseline value). If the value is 1 then a dose of zero yields an information matrix of zeros. If the value is 0 then a dose of zero yields an information matrix according to whatever components of the model can be estimated for this dose level.

Example use

```
Dose_zero_fix=0;
```

```
DI=[ ];
```

DI is the dose interval. There must be a dose interval for every element in the dose matrix (D). Example use

```
DI=[24 24];
```

Example use for multiple models:

```
DI=[ 24 24  
    24 24];
```

```
DPM=[ ];
```

DPM are the initial estimates of the design points. In this case design points are sampling times. This should be written as a row vector, which can be expressed as a matrix (although still read as a column vector) by the use of the "...", see example, the syntax

```
DPM=[ 0.1 1 4 ...  
      2 6 12];
```

Is equivalent to

```
DPM=[ 0.1 1 4 2 4 12];
```

DPM must contain the same number of elements as the sum of NT. All elements of DPM must fall between the lower (LBA) and upper bounds (UBA) provided below.

```
DPE=[ ];
```

DPE are discrete potential values of the design variables that may be chosen for optimization. These are only used with `method=9` and is optional.

Example use

```
DPE=[ 0.01 0.5 1 2 3 4 5 6 7 8 9 10 11 12 18 24 ...  
      0.01 0.5 1 2 3 4 5 6 7 8 9 10 11 12];
```

In this setting the exchange algorithm would select possible best sampling times from only those provided in the list above. The row vector may be any length. Do not use 0 (zero) as it is a reserved variable in the exchange algorithm. It is usual not to enter these values, but rather let the exchange algorithm select discrete values from within the sampling space itself. Note in this example that the second sampling time cannot exceed 12.

```
DPE_index=[ ];
```

This row vector provides the break points for DPE so that the exchange algorithm can determine which set of discrete sampling times should be used for each group. Example use (based on DPE example above):

```
DPE_index=[ 16 14];
```

This indicates that group 1 has a total of 16 times from which to choose but group 2 has only 14. The number of elements in DPE\_index must equal the number of elements in NT.

```
FX_DPM=[ ];
```

FX\_DPM provides the index for any sampling times in DPM that are fixed and are not permitted to be optimized. The default is not to fix any sampling times, see example

```
FX_DPM=[ ];
```

Leaving the row vector blank is read by POPT as entering a row vector of 0's the same length as `DPM`. To fix a single sampling time (e.g. the 12 hour sampling time from group 2 only) then a 1 is used in the same position in the row vector for `FX_DPM` as the position of 12 is in `DPM`. The other non-fixed elements in `DPM` are given a 0. Example use

```
FX_DPM=[ 0 0 0 ...
         0 0 1];
```

```
DN=[ ];
```

The dose number for which the samples are to be taken is provided by `DN`. For instance a `DN` value of 5 would indicate that the sample should be taken on the 5<sup>th</sup> dose, i.e. on the dose following 4 previous doses given at the specified dose interval. `DN` is a row vector that should be of equal length as `DPM`. The default value of `DN` is a row vector of 1's, indicating that all samples are taken from the first dose. Example use

```
DN=[ ];
```

If samples are to be taken from anything other than the first dose then the dose number for each sample must be provided. The length of `DN` must be the same as the length of `DPM`. Example use

```
DN=[ 4 4 4 ...
     4 4 1];
```

In this example all samples are taken from the 4<sup>th</sup> dose, except the 3<sup>rd</sup> sampling time from group 2 which is taken on the 1<sup>st</sup> dose.

```
MX_DN=[ ];
```

The dose number that any sample is taken can be optimized using `method=7`. In this setting `MX_DN` provides the upper bound for the dose number that doses can be taken from. If it is allowed that samples are taken from any dose up to steady state then it is recommended to set the `MX_DN` to be the dose where approximately 5 half-lives have elapsed, otherwise the optimization algorithm will have difficulty distinguishing the difference between a dose after 10 half-lives vs a dose after 20 half-lives. Example use

```
MX_DN= [ 5 5 5 ...
        1 1 1];
```

In this example, the first group of subjects can have their samples taken on any dose up to and including the 5<sup>th</sup> dose, while group 2 subjects will have all of their samples taken on the first dose.

```
UBA=[ ];
```

This is the upper bound on the optimized sampling times. The default is the upper bound equals the dose interval. Example use

```
UBA=[ ];
```

Any elements in `UBA` cannot be greater than the dose interval. If some further sampling constraints remain, e.g. a patient must leave a clinic at 6 hours after their dose then individual upper bounds can be used. Example use

```
UBA=[ 24 24 24 ...
     12 12 12];
```

Here group 1 patients have an upper bound equal to the dose interval (`DI`), but group 1 patients are constrained to have their samples fall within the first 12 hours post-dose only. `UBA` cannot be smaller than `LBA`.

LBA=[ ] ;

This is equivalent to UBA but for the lower bound. The default lower bound is immediately post-dose, provided by a row vector of zeros.

## OPTIMIZATION FEATURES

[Back to this example](#)

output\_on=;

this option toggles GUI output. If output\_on=1 then the interim GUI is produced however if output\_on=0 then the GUI screen output is set to off. Optimization runs slightly slower with the GUI interim output feature set to on.

Example syntax

output\_on=1;

prnt=;

prnt is the number of iterations between the refresh of the graphical and tabulated results. The more frequently interim results are echoed to the user the slower the algorithm will run. However, if the model is extremely complex then up to an hour may elapse before 500 iterations have passed. In general, prnt can be set to 500 for simulated annealing (method=6, 7, 8) and 50 (method=9) for the exchange algorithm.

re\_scale=;

When optimizing the user has the opportunity to re\_scale the design variables so that they all have a value of 1.0. This may be of value if one design variable has an initial value of 0.01 and another has a value of 1000, which may result in numerical difficulties when attempting to change the step-size. Example use to request re-scaling

re\_scale=1;

If no re-scaling is required then re\_scale=0 can be used. It is not needed for the exchange algorithm (method=9), and generally simulated annealing seems robust to scale issues.

constr=;

To force the optimization algorithm to arrange the sampling times per group in a monotonic increasing manner such that sampling time n+1 > sampling time n then set constr=1. This reduces the search space and avoids samples being exchanged. This is not needed for simulated annealing. Setting constr=0 avoids this requirement.

dpm\_constrain=

Further constraints to sampling times are available. Setting

dpm\_constrain=0 no further constraints.

dpm\_constrain=1 all sampling times are constrained to be the same for all groups

dpm\_constrain=2 sampling times are the same within a group for each occasion – user must define how to divide samples between occasions (using first\_occ). There must be the same number of samples per occasion.

`dpm_constrain=3` constrains all sampling times within a time period to be the same for all groups [only with `method=8`; user must specify `dpm_3_time`]. When using this constraint the user MUST ensure that there is the same number of sampling times for each group within the time period specified by `dpm_3_time`.

`first_occ=;` used in conjunction with `dpm_constrain=2`. Defines how many samples constitute an occasion in each group. For example if there are 6 samples per group and samples 4, 5 and 6 are required to be identical to samples 1, 2 & 3 then set `first_occ=3`. This does not constrain samples 1, 2 and 3 from group 1 to equal samples 1, 2 and 3 from group 2.

`dpm_3_time=;` defines the time period that all samples from all groups should be the same. WARNING: you must ensure that ALL patients have the same number of samples that will occur within the `dpm_3_time` period. Do this by fixing boundaries with UBA. WARNING: all samples within this period should be exchangeable (as they are sorted) therefore DN must be the same for every time - this cannot be tested for!

`lockout=;`  
 To stop sampling times from occurring at the same time, or within a defined time frame, set `lockout` to the minimal difference allowable. For example, if a pharmacodynamic response takes 20 minutes to observe then two samples within a 20 minute period would not be of use. In this case set `lockout` to be, for example, `lockout=0.5` (assuming time is in hours this gives 10 minutes extra).

## OPTIMIZATION METHOD

[Back to this example](#)

There are currently 9 optimization methods available.

|                       |   |
|-----------------------|---|
| <code>method=0</code> | none  |
| <code>method=1</code> | Simplex (not available with current version)  |
| <code>method=2</code> | non-adaptive random search  |
| <code>method=3</code> | simulated annealing superseded by [ <code>method=8</code> ]                             |
| <code>method=4</code> | grid (partitions=100 2 dimensions only)   |
| <code>method=5</code> | compute expectation over sampling times   |
| <code>method=6</code> | simulated annealing with approximate design over treatment allocation                   |
| <code>method=7</code> | simulated annealing with DN optimized too [ <code>method=7</code> ]                     |
| <code>method=8</code> | simulated annealing with fixed times - this method supersedes [ <code>method=3</code> ] |
| <code>method=9</code> | exchange algorithm  |

`Rwt=` **For use with non-adaptive random search**  
 number of candidate sets of parameter values

`n_exp=` **For use with method=5**  
 Provide the number of evaluations of the

information matrix from which to work out the expectation of the determinant and standard errors, usually 1000.

|                                   |   |
|-----------------------------------|---|
| <code>method_sa=0</code>          | <b>For use with simulated annealing algorithm</b><br>normal temperature decline (slowest – but more robust)   |
| <code>method_sa=1</code>          | faster temperature decline (geometric cooling)  |
| <code>method_sa=2</code>          | fastest temperature decline   |
| <code>minv=</code>                | Convergence criteria. This corresponds to the smallest stepsize which provides the number of significant digits required. Usually set <code>minv=0.0001</code>  |
| <code>tempe=</code>               | Initial value for temperature. This value needs to be adjusted. Generally <code>tempe=1000</code> is fine for most PK and PKPD problems. Suggest monitoring <code>Fraction Wrong Moves</code> on the interim output. It should start at zero and then within 1000 to 2000 iterations start accepting wrong moves. If <code>Fraction Wrong Moves</code> is initially non-zero then increase the temperature and start again. Decreasing the temperature is not usually required.   |
| <code>num_rand_arrangments</code> | <b>For use with exchange algorithm</b><br>This specifies the number of times that the discrete variables are randomly rearranged between each set of exchanges. This is used to help the algorithm out of local minima. Generally a value of 5 is sufficient.   |
| <code>dpe_num</code>              | This variable is used with <code>DPE</code> is not provided. If discrete values of potential sampling times are not provided by the user then the exchange algorithm will provide its own discrete values. It does so by dividing the design space ( <code>UBA – LBA</code> ) into a geometric series of discrete candidate sampling times (that includes the upper ( <code>UBA</code> ) and lower bound ( <code>LBA</code> ) values). The number of discrete sampling times is provided by <code>dpe_num</code> . Generally <code>dpe_num=25</code> to 50 is fine. |

## PRIOR INFORMATION

[Back to this example](#)

If prior information is available then this can be incorporated as a fixed component of the information matrix which is specified by

```
MFD_prior=[ ];
```

This information matrix must have the same number of rows and columns as the information matrix estimated in the current design. `MFD_prior` is then added to the current information and the determinant of the sum of these matrices is computed. The current sampling times are therefore optimized for value added information.

The user has the opportunity to not use prior information by setting the variable `use_prior`. If no prior information is available then `use_prior` should be set to zero. If prior information exists then the user can decide whether to use it or not.

### Example POPT\_INI\_ROBUST

[BACK TO TOP](#)

There are 4 parts to this file

- 1) [Model features](#)
- 2) [Design features](#)
- 3) [Setup features for sampling windows](#)
- 4) [Prior information](#)

### MODEL FEATURES FOR SAMPLING WINDOWS

These are identical to those set out in the POPT\_INI file  
<[GO TO POPT\\_INI Model features](#)>.

### DESIGN FEATURES FOR WINDOWS

[Back to this example](#)

These are the same as those set out in the POPT\_INI file  
<[GO TO POPT\\_INI Design features](#)> except for specification of the lower and upper bounds.

In this setting DPM should be set to the optimized sampling times from a previous run. UBA should be the upper boundary of the sampling window and LBA the lower boundary.

Lower and upper bounds must be chosen in order to represent the design space from which to evaluate sampling windows. The following suggested rules apply:

- 1) all design points must have a specified upper and lower bound
- 2) no two bounds for sampling times should overlap for samples in the same group (unless specified in rule 3).
- 3) if 2 optimal sampling times have the same value then it is recommended to set the upper and lower bounds to be the same

Some examples

For a 2 group design with 3 samples per group. Recall that the first 3 samples are for group 1 and the second 3 are for group 2. It is OK to have sampling boundaries that overlap for samples that arise from different groups.

It is NOT OK to have sampling boundaries that overlap for samples that arise from the same group (except under suggested rule 3).

The following example satisfy all of the requirements

```
UBA = [ 0.5    2    8    3    6    12 ]
DPM = [ 0.1    1    5    2    4    8 ]
LBA = [ 0      0.5  2    0    3    6 ]
```

The following example does not satisfy all requirements and should not be used

```
UBA = [ 1      2    8    3    6    12 ]
DPM = [ 0.1    1    5    2    4    8 ]
LBA = [ 0      0.5  2    0    3    6 ]
```

The following example shows how to deal with a replicated optimal sample

```
UBA = [ 2      2    8    3    6    12 ]
DPM = [ 1      1    5    2    4    8 ]
LBA = [ 0      0    2    0    3    6 ]
```

### Specification of the values of LBA and UBA.

There are no right or wrong values to choose for LBA and UBA (as long as the suggested rules are followed). If UBA and LBA are set to extreme values (i.e. a long way apart) then it will take a long time to find the sampling windows. If they are too close together then your boundaries may be smaller than the windows and they may force unnecessary constraints. It is usually necessary to try a few runs first to find out how you should set your boundary conditions. A suggested recipe:

- 1) Try a set of LBA and UBA values, set the range to be smaller for smaller values of the sampling time (sampling time windows are generally proportional to the value of the sampling time)
- 2) Run `p_robust_1`
- 3) The 4th column in the output displays the fraction of samples accepted. You should aim for a value between 2 and 5%. If the value is too small then constrict some of your sampling windows. If the value is too high then you have set your sampling windows to be too constrictive.

### SETUP FEATURES FOR SAMPLING WINDOWS [Back to this example](#)

There is no current analytical solution to defining sampling windows for nonlinear mixed effects models. In POPT sampling windows are generated based on a 3 stage process.

- 1) Generate random samples from a uniform distribution with limits of UBA & LBA and accept those samples that have efficiencies greater than a predefined percent.
- 2) Re-sample from those samples that are accepted in stage 1 to generate many different combinations of sampling times. Accept those that have an efficiency greater than a pre-defined level. Continue this to inflate the number of samples accepted to some predefined level.

- 3) Re-sample a pre-defined number of times from those samples accepted in stage 2. Accept any that have an efficiency greater than some predefined value. Discard those that don't. Repeat stage 3 until the fraction of those that are discarded is lower than 10%.

Finally a joint sampling window is constructed as the percentile range (defined by the user) of the sampling distributions from stage 3.

`n_samples=;`

This variable indicates how many samples that are generated at random have to be accepted before stage 1 is completed. Typically `n_samples = 1000;`

`win_eff_k =;`

The minimum efficiency of any set of samples randomly generated in stage 1. Typically, `win_eff_k = 0.90;`

`max_it=;`

The maximum number of iterations. Typically, `max_it = 1e+6;`

`inflation =;`

The number of re-sampled sets of sampling times to be generated in stage 2 from the joint samples generated from stage 1. Typically, `inflation = 5000;`

`win_eff_r=;`

The minimum relative efficiency of resampled sets of sampling times generated in stage 3 at which level are accepted.

`win_eff_r = 0.95;`

`lb_win=;`

`ub_win=;`

`P_ROBUST_1` reports the lower bound (`lb_win`) and upper bound (`ub_win`) of the sampling windows as the percentiles of the distribution for each sampling window. The percentiles are typically, `lb_win=5;` & `ub_win=95;` indicating that `P_ROBUST_1` will report the 5<sup>th</sup> and 95<sup>th</sup> percentile of the distribution of the accepted samples from stage 3.

### **PRIOR INFORMATION FOR SAMPLING WINDOWS**

These are identical to those set out in the `POPT_INI` file  
<[GO TO POPT\\_INI Prior Information](#)>.

[Back to this example](#)

### **Interpreting Output**

[BACK TO TOP](#)

[Screen output for POPT](#)

[Output file for POPT](#)

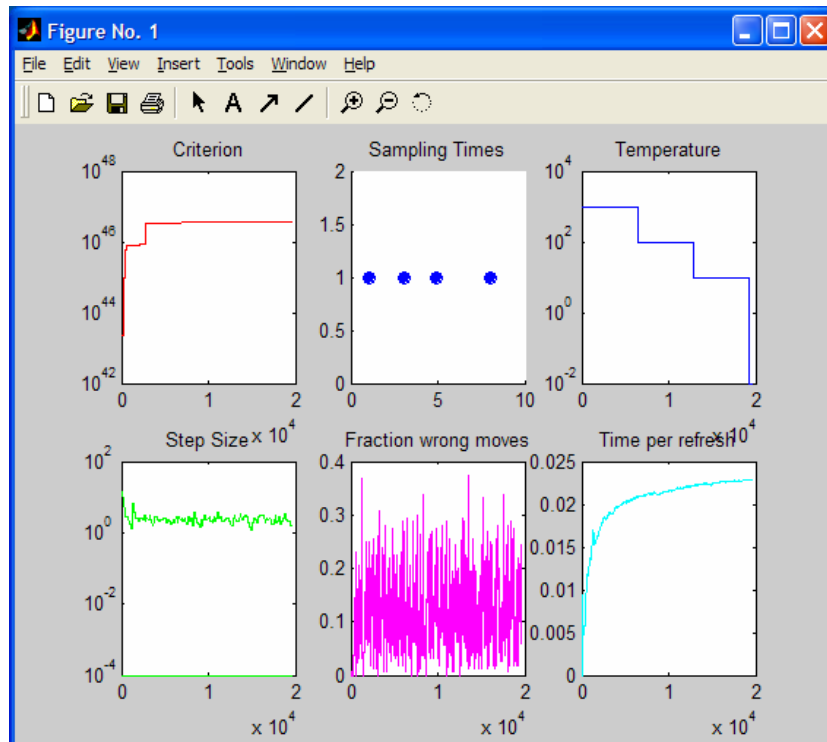
[Screen output for P\\_ROBUST\\_1](#)

[Output file for P\\_ROBUST\\_1](#)

## Screen output for POPT

When POPT is running with either `method=8` or `method=9` it will produce a screen window like the following example. This screen can be turned off by setting `output_on = 0;`

For `method=8` (simulated annealing)



- The upper left panel represents the log of the determinant of the information matrix.
- The upper middle panel is the current sampling times (the y-axis is meaningless) presented across the x-axis
- The upper right panel is the current value of the temperature
- The lower left panel is the step size (the amount by which the average sampling time is allowed to deviate during any iteration of the algorithm. When this reaches  $10^{-4}$  the algorithm will have converged).
- The lower middle panel is the fraction of moves that have been accepted away from the optimum. This should generally lie between 0.1 and 0.4
- The lower right panel is the average period of time per iteration.

Output will also be numerically echoed to the screen as text. The column headings are the current values of:

Column 1: Iteration number

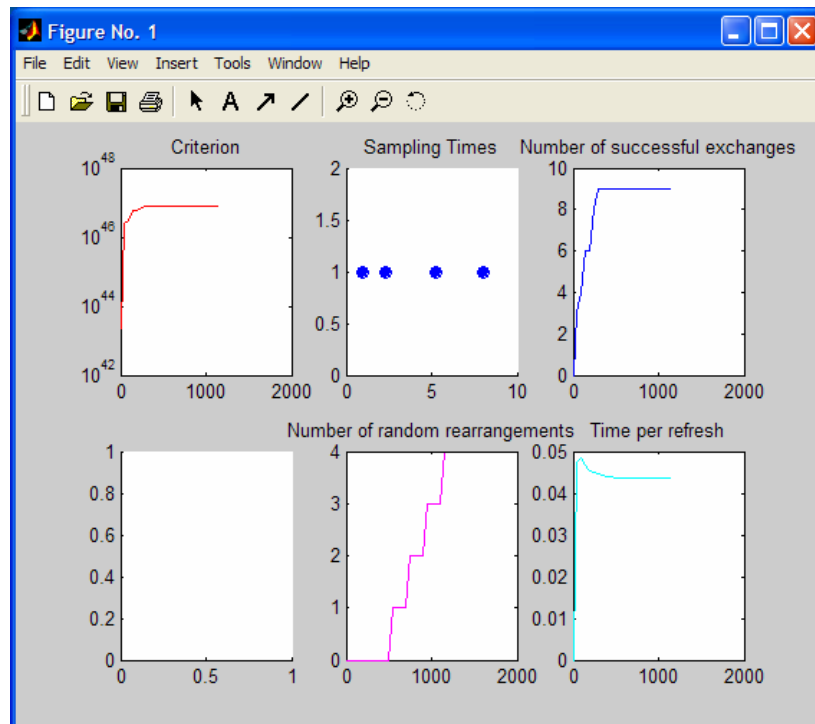
Column 2: Temperature

Column 3: Maximum stepsize (stepsize for worst estimated sampling time)

Column 4: Determinant

Column 5: Number of false acceptances

For method=9 (exchange algorithm)



- The upper left panel represents the log of the determinant of the information matrix.
- The upper middle panel is the current sampling times (the y-axis is meaningless) presented across the x-axis
- The upper right panel is the current number of successful exchanges made
- The lower left panel is not used.
- The lower middle panel is the number of random rearrangements of the discrete sample space (usually the algorithm terminates after 5 rearrangements)
- The lower right panel is the average period of time per iteration.

Output will also be numerically echoed to the screen as text. The column headings are the current values of:

Column 1: Iteration Number  
Column 2: Number of re-randomizations  
Column 3: Number of successful exchanges  
Column 4: Number of echoes to the screen  
Column 5: Determinant

## Output file for POPT

[Back to this example](#)

Output files are created for all successful POPT runs. POPT creates an output file labelled methodx.res, where x is the method number used. For multiple models or multiple response models POPT creates the file Methodx\_comp.res. A break down of the output for a single model single response is provided. The output for a multiple model/multiple response is the same as the output for a single model/response design except that the information matrix and standard errors are computed for each model/response.

| <b>Variable</b>             | <b>Interpretation</b>  |
|-----------------------------|--|
| dose                        | Dose per group   |
| NT                          | Number of sampling times per group   |
| NS                          | Number of subjects per group   |
| DN                          | Dose number that the sampling was taken from   |
| method                      | Optimization method  |
| initial_starting_times      | Your initial estimates (DPM)   |
| sampling_times              | Optimized sampling times – if you have chosen <code>method=0</code> then these will equal your <code>initial_sampling_times</code> |
| information_matrix          | The information matrix – you can use this to get your prior information for subsequent runs.                                       |
| fixed_effect_parameters     | Values of the fixed effects parameters (BETA1)   |
| POP_PAR                     | The names of the parameters  |
| FXFI                        | Which parameters were fixed – the numbers represent the position of the parameter vector (BETA1)                                   |
| standard_errors_fe          | Standard errors of the fixed effects parameters  |
| standard_errors_percent_fe  | Standard errors of the fixed effects parameters expressed as a percent   |
| BSV_parameters              | Values of the between subject variance (OI)  |
| FXI                         | Which parameters were fixed – the numbers represent the position of the parameter vector (OI)                                      |
| standard_errors_bsv         | Standard errors of the BSV parameters  |
| standard_errors_percent_bsv | Standard errors of the BSV parameters  |

|                             |   |
|-----------------------------|---|
|                             | expressed as a percent  |
| BSV_model                   | The variance model used for BSV – 1 = exponential, 0 = additive   |
| residual_standard_deviation | Values of the residual standard deviation   |
| FXR                         | Which values of the residual standard deviation were fixed  |
| standard_errors_res         | Standard errors of the residual standard deviation  |
| standard_errors_percent_res | Standard errors of the residual standard deviation expressed as a percent   |
| determinant                 | The determinant of the information matrix   |
| criterion                   | The determinant to the power of the inverse of the number of parameters (fixed effects + BSV + RUV) in the model  |
| iterations                  | The number of iterations used by the optimization routine   |
| eigen_values                | The eigen values of the information matrix. The ratio of the highest to lowest numbers is often quoted as a measure of the  |
| time_taken_seconds          | Time taken to run the problem and produce the output. If you are using method=0 then echoing the output to the screen can take as long as evaluating the information matrix. In this case the screen output will include two times the first being the time to evaluate the information matrix and the second the overall time. |
| az                          | Page separator.   |

---

## SCREEN OUTPUT FOR P\_ROBUST\_1

[Back to this example](#)

When P\_ROBUST\_1 is running it will produce a text screen output. The screen output changes depending on which stage of the sampling design is being evaluated. For a description of the stages see section [<SET UP FEATURES FOR SAMPLING WINDOWS>](#).

Stage 1. Samples are generated randomly from  $\sim U(LBA, UBA)$ , until the total number of accepted samples is 1000 (column 3). A reasonable acceptance rate (column 4) is 0.02 to 0.1.

| Stage | Iteration number | Number of accepted sets of candidate samples | Fraction of sets of samples that are accepted |
|-------|------------------|--|---|
|-------|------------------|--|---|

For Stage 2 Samples are generated by resampling from the stage 1 samples, until the number of accepted sets of candidate samples = 5000 (column 3).

| Stage | Iteration number | Number of accepted sets of candidate samples |
|-------|------------------|--|
|-------|------------------|--|

For Stage 3 Samples are generated by resampling from the stage 2 samples until the proportion of rejected samples is  $< 0.1$ .

| Stage | Iteration number | Fraction of sets of samples that are accepted |
|-------|------------------|---|
|-------|------------------|---|

The stage number increments as stage 3 is repeated until the fraction reduces to less than 0.1.

## OUTPUT FILE FOR P\_ROBUST\_1

[Back to this example](#)

## P2NM

[BACK TO TOP](#)

P2NM = POPT too NONMEM

This program creates a NONMEM data file template suitable for simulation-estimation runs in NONMEM. This program calls POPT\_INI.m and uses the design provided in this file to create a NONMEM data file that has the appropriate number of patients, doses, sampling times etc. The file is saved to NM\_dat.csv. If the file NM\_dat.csv already exists then the file is overwritten. The csv file is written from row 2, where the first row is left blank in order for the user to manually input a header row if desired. P2NM uses "0" as a "." Placeholder. The columns created are ID, TIME, AMT, DV, MDV.

The NM data file is automatically started at ID=1. If a different starting ID value is required then edit line 25

```
init_id_value=;
```

The value of `init_id_value` is taken as the first value of ID in the data file and additional patients are given consecutive values. Example syntax `init_id_value=2000` will start the data file numbering at ID=2000.

## Optimization algorithms

[BACK TO TOP](#)

### Exchange

The exchange algorithm implemented in POPT is a simple  $k=1$  algorithm, where on each iteration a single sampling time is exchanged with one from a list of possible sampling times that provides the same or a better value of the determinant. The list of possible sampling times is either provided by the user or if left blank is generated automatically by POPT. If generated automatically POPT divides the dose interval into 50 (or some other value as determined by the user) geometrically spaced samples and searches over this discrete space. The search for exchanges occurs by starting from the lower bound of the interval and progressing through the list until an exchange occurs. If an exchange occurs then the search re-starts at the lower bound of the search space again. When the search progresses through the list one complete time without an exchange the algorithm randomizes the search space and tries again. This process is repeated until 5 re-randomizations occur (or as defined by the user). On some occasions this can result in renewed exchanges.

### Simulated Annealing

In this implementation of simulated annealing the convergence criterion refers to the accuracy of the design points (this is set by the user). A value of 0.01 means that all design points are estimated to a value of less than 0.01 significant digits of time units. Note that other implementations may set the convergence to be a function of the current temperature. We have found this latter method to be unreliable for searching over the surface of the determinant of a population Fisher information matrix. We have found the starting temperature of 1000 to be suitable for most PK and PKPD designs to date. The initial temperature selected should be one that provides few false acceptances (last column of interim output). The choice of fast, faster and slow is dependent on how sure you want to be that you have found a global minimum. For most purposes fast or faster are fine. The slower the process the better the statistical quality of the convergence – but it can become very slow. Don't forget it is the utility of the design that is located rather than finding the design with the absolute highest possible determinant. Unlike estimation methods, lack of convergence (or if the algorithm is terminated during its run) does NOT indicate that the estimates of the sampling times are in anyway biased.

In principle simulated annealing works by search across the design space for regions of high probability. It determines these regions by improvements in the objective function (the determinant in this case). It explores the design region by moving away from the region of highest probability and accepting "false" candidates (i.e. ones that are not as good as the current best region) at

their probability level of occurring using a Metropolis step. The acceptance probability of these false samples becomes more difficult as the temperature declines.

## Copyright Statement

[BACK TO TOP](#)

This program 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 2 of the License, or (at your option) any later version.

This program 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.

POPT (C) copyright Stephen Duffull ver 3.0; Oct, 5th 2003  
PFIM\_M (C) copyright Stephen Duffull ver 1.2(i) May; 28th 2003

TO MAINTAIN DESIRED FUNCTIONALITY, DO NOT EDIT THIS FILE

## Bibliography

[BACK TO TOP](#)

### General Design

Atkinson AV. Donev AN. Ooptimum Experimental Designs. Oxford UK; Oxford Scientific Publications. 1992.

Atkinson AC. Cox DR. Planning experiments for discriminating between models. JRSS 1974;36:321-348

Box GEP. Lucas HL. Design of experiments in non-linear situations. Biometrika 1959;46:77-90/

Cramér H. Mathematical Methods of Statistics. Princeton, NJ: Princeton University Press, 1946

Draper NR. Hunter WG. Design of experiments for parameter estimation in multiple response situations. Biometrika 1966;53:525-533

### Population Fisher Information Matrix

Mentre F. Mallet A. Baccar D. Optimal design in random effects regression models. Biometrika. 1997;84:429-442

Retout S, Duffull S, Mentre F. Development and implementation of the population Fisher information matrix for the evaluation of population pharmacokinetic designs. *Comput Meth Prog Biomed* 2001;65:141-151.

S. Retout and F. Mentre. Further developments of the Fisher information matrix in nonlinear mixed effects models with evaluation in population pharmacokinetics. *J. Biopharm. Stat.* 13:209-227 (2003).

## **POPT**

Duffull S, Waterhouse T, Eccleston J. Some considerations on the design of population pharmacokinetic studies. *J Pharmacokinet Pharmacodyn* 2005;32:441-457

Waterhouse TH, Redmann S, Duffull SB, Eccleston JA. Optimal design for model discrimination and parameter estimation for itraconazole population pharmacokinetics in cystic fibrosis patients. *J Pharmacokinet Pharmacodyn* 2005;32:521-545

Green B, Duffull SB. Prospective evaluation of a D-optimal designed population pharmacokinetic study. *J Pharmacokinet Pharmacodyn* 2003; 30:145-161

Duffull SB, Retout S, Mentré F. The use of simulated annealing for finding optimal population designs. *Comp. Methods Programs Biomed.* 2002;69:25-35

Duffull SB, Mentré F, Aarons L. Optimal design of a population pharmacodynamic experiment for ivabradine. *Pharm Res* 2001;18:83-9

## **Simulated Annealing**

Duffull SB, Retout S, and Mentre F. The use of simulated annealing for finding optimal population designs. *Comput. Meth. Prog. Biomed.*, 2002; 69:25-35

Goffe WL, Ferrier GD, Rogers J. Global optimisation of statistical functions with simulated annealing. *J Econom* 1994;60:65-99