

Understanding Breakthrough Curves

v0.1

Generated by Doxygen 1.9.1

1 Todo List	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 AppCtx Struct Reference	7
4.1.1 Detailed Description	8
4.1.2 Field Documentation	8
4.1.2.1 b	8
4.1.2.2 beta	8
4.1.2.3 btc	8
4.1.2.4 compute_analytical	8
4.1.2.5 dtau	8
4.1.2.6 dtau_est	8
4.1.2.7 dx	9
4.1.2.8 eps	9
4.1.2.9 errnorm_all	9
4.1.2.10 f_jm1	9
4.1.2.11 isotherm	9
4.1.2.12 j	9
4.1.2.13 K	9
4.1.2.14 kappa	9
4.1.2.15 m	10
4.1.2.16 make_movie	10
4.1.2.17 movie_step_interval	10
4.1.2.18 omega	10
4.1.2.19 print_header	10
4.1.2.20 S_jm1	10
4.1.2.21 save_btc	10
4.1.2.22 tau_star	10
4.1.2.23 theta	11
4.1.2.24 viewer	11
4.2 EXP_ISO Struct Reference	11
4.2.1 Field Documentation	11
4.2.1.1 c_scale	11
4.2.1.2 c_star	11
4.2.1.3 f_scale	11
4.2.1.4 f_star	12
4.2.1.5 n	12

4.3 ISO Struct Reference	12
4.3.1 Field Documentation	12
4.3.1.1 data	12
4.3.1.2 dF	12
4.3.1.3 F	12
4.3.1.4 num_params	13
4.3.1.5 params	13
4.4 isotherm Struct Reference	13
4.5 isotherm Struct Reference	13
4.6 isotherms.Langmuir Class Reference	13
4.7 isotherms.Linear Class Reference	14
4.8 Variables_j Struct Reference	14
4.8.1 Detailed Description	14
4.8.2 Field Documentation	15
4.8.2.1 c_j	15
4.8.2.2 c_jm1	15
4.8.2.3 error	15
4.8.2.4 W_j	15
4.9 VecAndArray Struct Reference	15
4.9.1 Detailed Description	15
4.9.2 Field Documentation	16
4.9.2.1 arr	16
4.9.2.2 vec	16
5 File Documentation	17
5.1 src/isotherm.h File Reference	17
5.2 src/my_funcs.hpp File Reference	17
5.2.1 Detailed Description	17
5.2.2 Function Documentation	18
5.2.2.1 analytical_btc()	18
5.2.2.2 exp_l0()	18
5.2.2.3 integrate()	19
5.2.2.4 update_solution()	19
5.3 src/my_petsc.h File Reference	20
5.3.1 Detailed Description	21
5.3.2 Function Documentation	21
5.3.2.1 CalculateOmega()	21
5.3.2.2 DestroyVecAndArray()	21
5.3.2.3 eval_dS_i_j()	22
5.3.2.4 eval_S_i_j()	22
5.3.2.5 FormFunctionLocal()	23
5.3.2.6 FormJacobianLocal()	23

5.3.2.7 InitializeVecAndArray()	24
5.3.2.8 one_time_step()	24
5.3.2.9 run()	25
5.3.2.10 simulate()	25
5.3.2.11 theta_rule()	26
5.3.2.12 UpdateAnalytical()	26
5.3.2.13 UpdateIsotherm()	26
5.3.2.14 UpdateSolid()	27

Index	29
--------------	-----------

Chapter 1

Todo List

Class [Variables_j](#)

Change c_j and c_{jm1} to a_j and a_{jm1} to reflect notation change in paper

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

AppCtx	Useful parameters for solving, Application context	7
EXP_ISO	11
ISO	12
Isotherm	??
isotherm	13
isotherms.Langmuir	13
isotherms.Linear	14
Variables_j	Analytical and numerical variables at time j (and jm1 if analytical)	14
VecAndArray	15

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

src/isortherm.h	Functions and structs for adsorption isotherm	17
src/my_funcs.hpp	For calculating analytical solutions	17
src/my_petsc.h	Functions and structs for solving PDE	20

Chapter 4

Data Structure Documentation

4.1 AppCtx Struct Reference

Useful parameters for solving, Application context.

```
#include <my_petsc.h>
```

Collaboration diagram for AppCtx:

Data Fields

- PetscInt [j](#)
- PetscInt [m](#)
- PetscReal [errnorm_all](#)
- PetscReal [eps](#)
- [ISO isotherm](#)
- PetscReal [theta](#)
- PetscReal [dx](#)
- PetscReal [dtau](#)
- PetscReal * [btc](#)
- PetscReal [tau_star](#)
- [VecAndArray](#) [f_jm1](#)
- [VecAndArray](#) [S_jm1](#)
- bool [compute_analytical](#)
- bool [make_movie](#)
- bool [save_btc](#)
- bool [print_header](#)
- PetscViewer [viewer](#)
- PetscInt [movie_step_interval](#)
- PetscReal [omega](#)
- PetscReal [b](#)
- PetscReal [K](#)
- PetscReal [beta](#)
- PetscReal [kappa](#)
- PetscReal [dtau_est](#)

4.1.1 Detailed Description

Useful parameters for solving, Application context.

4.1.2 Field Documentation

4.1.2.1 **b**

```
PetscReal AppCtx::b
```

scale factor for beta

4.1.2.2 **beta**

```
PetscReal AppCtx::beta
```

constant calculated

4.1.2.3 **btc**

```
PetscReal* AppCtx::btc
```

array for break-through curve

4.1.2.4 **compute_analytical**

```
bool AppCtx::compute_analytical
```

Whether or not to compute analytical solution

4.1.2.5 **dtau**

```
PetscReal AppCtx::dtau
```

time step

4.1.2.6 **dtau_est**

```
PetscReal AppCtx::dtau_est
```

estimated dtau for setting it when tau_star unknown

4.1.2.7 dx

PetscReal AppCtx::dx

grid spacing

4.1.2.8 eps

PetscReal AppCtx::eps

value of ε

4.1.2.9 errnorm_all

PetscReal AppCtx::errnorm_all

infinity norm

4.1.2.10 f_jm1

VecAndArray AppCtx::f_jm1

Isotherm function at previous time step

4.1.2.11 isotherm

ISO AppCtx::isotherm

structure for adsorption isotherm

4.1.2.12 j

PetscInt AppCtx::j

time step

4.1.2.13 K

PetscReal AppCtx::K

scale factor for kappa

4.1.2.14 kappa

PetscReal AppCtx::kappa

constant calculated

4.1.2.15 m

`PetscInt AppCtx::m`

total number of steps

4.1.2.16 make_movie

`bool AppCtx::make_movie`

whether or not to make movie

4.1.2.17 movie_step_interval

`PetscInt AppCtx::movie_step_interval`

step interval for saving movie frames

4.1.2.18 omega

`PetscReal AppCtx::omega`

closeness to steady state at end of simulation

4.1.2.19 print_header

`bool AppCtx::print_header`

whether to print header when displaying results of simulation

4.1.2.20 S_jm1

`VecAndArray AppCtx::S_jm1`

Solid concentration at previous time step

4.1.2.21 save_btc

`bool AppCtx::save_btc`

flag to save btc

4.1.2.22 tau_star

`PetscReal AppCtx::tau_star`

total simulation time τ_*

4.1.2.23 theta

```
PetscReal AppCtx::theta
```

fraction of BFD/FFD

4.1.2.24 viewer

```
PetscViewer AppCtx::viewer
```

petsc viewer

The documentation for this struct was generated from the following file:

- [src/my_petsc.h](#)

4.2 EXP_ISO Struct Reference

Data Fields

- double * [c_star](#)
- double * [f_star](#)
- int [n](#)
- double [c_scale](#)
- double [f_scale](#)

4.2.1 Field Documentation

4.2.1.1 c_scale

```
double EXP_ISO::c_scale
```

scale factor for concentration, multiplying [c_scale](#) by [c_star](#) gives units of mol/m3

4.2.1.2 c_star

```
double* EXP_ISO::c_star
```

scaled fluid concentration

4.2.1.3 f_scale

```
double EXP_ISO::f_scale
```

scale factor for loading, multiplying [f_scale](#) by [f_star](#) gives units of mol/m3

4.2.1.4 f_star

```
double* EXP_ISO::f_star
```

scaled solid concentration

4.2.1.5 n

```
int EXP_ISO::n
```

number of points

The documentation for this struct was generated from the following file:

- [src/isorthem.h](#)

4.3 ISO Struct Reference

Data Fields

- [Fun F](#)
- [dFun dF](#)
- int [num_params](#)
- double * [params](#)
- double * [data](#)

4.3.1 Field Documentation

4.3.1.1 data

```
double* ISO::data
```

data used for splines

4.3.1.2 dF

```
dFun ISO::dF
```

derivative of adsorption isotherm

4.3.1.3 F

```
Fun ISO::F
```

adsorption isotherm

4.3.1.4 num_params

```
int ISO::num_params
```

number of parameters used

4.3.1.5 params

```
double* ISO::params
```

parameters fit

The documentation for this struct was generated from the following file:

- [src/isotherm.h](#)

4.4 isotherm Struct Reference

The documentation for this struct was generated from the following file:

- [src/isotherm.h](#)

4.5 isotherm Struct Reference

The documentation for this struct was generated from the following file:

- [src/isotherm.h](#)

4.6 isotherms.Langmuir Class Reference

Public Member Functions

- `def __init__ (self, k)`
- `def f (self, c)`
- `def df_dc (self, c)`
- `def s (self, c)`
- `def get_c_rarefaction (self, x, t)`
- `def get_c_shock (self, x, t)`
- `def is_rarefaction_wave (self)`
- `def is_shock_wave (self)`
- `def shock_lambda (self)`
- `def get_c_local_equilibrium (self, x, t)`

Data Fields

- **k**
- **name**

The documentation for this class was generated from the following file:

- `src/isotherms.py`

4.7 isotherms.Linear Class Reference

Public Member Functions

- `def __init__(self)`
- `def f(self, c)`
- `def df_dc(self, c)`

Data Fields

- **name**

The documentation for this class was generated from the following file:

- `src/isotherms.py`

4.8 Variables_j Struct Reference

Analytical and numerical variables at time j (and $jm1$ if analytical)

```
#include <my_petsc.h>
```

Collaboration diagram for Variables_j:

Data Fields

- [VecAndArray W_j](#)
- [VecAndArray c_j](#)
- [VecAndArray c_jm1](#)
- Vec [error](#)

4.8.1 Detailed Description

Analytical and numerical variables at time j (and $jm1$ if analytical)

Todo Change `c_j` and `c_jm1` to `a_j` and `a_jm1` to reflect notation change in paper

4.8.2 Field Documentation

4.8.2.1 `c_j`

`VecAndArray Variables_j::c_j`

Analytical values of fluid concentration at time j

4.8.2.2 `c_jm1`

`VecAndArray Variables_j::c_jm1`

Analytical values of fluid concentration at time $j - 1$

4.8.2.3 `error`

`Vec Variables_j::error`

vector comprising errors (presumably b/t analytical and numerical)

4.8.2.4 `W_j`

`VecAndArray Variables_j::W_j`

Numerical values of fluid concentration at time j

The documentation for this struct was generated from the following file:

- [src/my_petsc.h](#)

4.9 VecAndArray Struct Reference

```
#include <my_petsc.h>
```

Data Fields

- Vec [vec](#)
- PetscReal * [arr](#)

4.9.1 Detailed Description

Used for converting between vector and array types

4.9.2 Field Documentation

4.9.2.1 arr

```
PetscReal* VecAndArray::arr
```

array part

4.9.2.2 vec

```
Vec VecAndArray::vec
```

vector part

The documentation for this struct was generated from the following file:

- [src/my_petsc.h](#)

Chapter 5

File Documentation

5.1 src/isotherm.h File Reference

Functions and structs for adsorption isotherm.

```
#include <petsc.h>  
Include dependency graph for isotherm.h:
```

5.2 src/my_funcs.hpp File Reference

For calculating analytical solutions.

Functions

- double [exp_I0](#) (double T, double z)
Calculate exponential term involving bessel function.
- double [integrate](#) (double T_jm1, double T_j, double z)
Integrate expI0 from T_jm1 to T_j.
- double [update_solution](#) (double *c_j, double *c_jm1, double T_jm1, double T_j, double *X, int n)
Updates solution from previous time step.
- void [analytical_btc](#) (double *, double *, double X, int)
Provides analytical break-through curve.

5.2.1 Detailed Description

For calculating analytical solutions.

Used to calculate analytical solutions or break-through curves

Author

Robert F. DeJaco

5.2.2 Function Documentation

5.2.2.1 analytical_btc()

```
void analytical_btc (
    double * c,
    double * T,
    double X,
    int m )
```

Provides analytical break-through curve.

Parameters

c	fluid concentration
T	times
X	value for distance (1/epsilon)
m	number of times including 0

Returns

void

5.2.2.2 exp_I0()

```
double exp_I0 (
    double T,
    double z )
```

Calculate exponential term involving bessel function.

Evaluates $e^{-z-T} I_0(2\sqrt{zT})$, where I_0 is the modified Bessel function of zeroth order.

Parameters

T	Scaled value for time
z	Scaled value for distance

Returns

value of expression

5.2.2.3 integrate()

```
double integrate (
    double T_jm1,
    double T_j,
    double z )
```

Integrate `expI0` from `T_jm1` to `T_j`.

Evaluates

$$\int_{T_{j-1}}^{T_j} e^{-z-t} I_0 \left(2\sqrt{zt} \right) dt$$

See also

[exp_I0](#)

Parameters

<code>T_jm1</code>	scaled time at previous index j-1
<code>T_j</code>	scaled time at current/next index j
<code>z</code>	scaled distance

Returns

value of expression

5.2.2.4 update_solution()

```
double update_solution (
    double * c_j,
    double * c_jm1,
    double T_jm1,
    double T_j,
    double * X,
    int n )
```

Updates solution from previous time step.

Parameters

<code>c_j</code>	new solution at time j
<code>c_jm1</code>	old solution at time j-1
<code>T_j</code>	current/new value of time
<code>T_jm1</code>	previous value of time
<code>X</code>	distance along column
<code>n</code>	last index of distance array

Returns

void

Warning

{not tested, probably uses wrong value of n}

5.3 src/my_petsc.h File Reference

Functions and structs for solving PDE.

```
#include <stdbool.h>
#include <petsc.h>
#include "isotherm.h"
Include dependency graph for my_petsc.h:
```

Data Structures

- struct [VecAndArray](#)
- struct [AppCtx](#)
Useful parameters for solving, Application context.
- struct [Variables_j](#)
Analytical and numerical variables at time j (and jm1 if analytical)

Functions

- void [print_user_params](#) ([AppCtx](#) u)
print user params associated with application
- PetscErrorCode [InitializeVecAndArray](#) (DM da, [VecAndArray](#) *x)
Initialize Struct containing both vector and array.
- PetscErrorCode [DestroyVecAndArray](#) (DM da, [VecAndArray](#) *x)
Destroy Struct containing both vector and array.
- PetscErrorCode [one_time_step](#) (SNES snes, DMDALocalInfo *p_info, [AppCtx](#) *p_user, [Variables_j](#) *p_vars)
Perform one time step.
- PetscErrorCode [simulate](#) (DM da, SNES snes, DMDALocalInfo *p_info, [AppCtx](#) *p_user)
Perform calculations at all time steps.
- PetscErrorCode [run](#) (int argc, char **args, int m, double eps, double tau_star, double theta, [ISO](#) *isotherm, bool compute_analytical, double *btc, bool make_movie, bool print_header)
Main function called.
- PetscErrorCode [UpdateAnalytical](#) (DMDALocalInfo *info, PetscReal *c_j, PetscReal *c_jm1, [AppCtx](#) *user)
Updates analytical solution.
- PetscErrorCode [UpdateSolid](#) (DMDALocalInfo *info, PetscReal *W_j, [AppCtx](#) *user)
Update solid concentration.
- PetscErrorCode [UpdateIsotherm](#) (DMDALocalInfo *info, PetscReal *W, [AppCtx](#) *user)
- PetscReal [eval_S_i_j](#) (PetscReal f_j, PetscReal f_jm1, PetscReal exp_mdT, PetscReal S_jm1, PetscReal dT)
evaluate value of solid concentration
- PetscReal [eval_dS_i_j](#) (PetscReal df_i_j, PetscReal dT)
evaluates gradient in solid concentration wrt fluid

- PetscReal [theta_rule](#) (PetscReal val_i, PetscReal val_im1, PetscReal theta)
does theta rule for discretization in space
- PetscErrorCode [FormFunctionLocal](#) (DMDALocalInfo *info, PetscReal *W_j, PetscReal *FF, [AppCtx](#) *user)
residuals of nonlinear equations
- PetscErrorCode [FormJacobianLocal](#) (DMDALocalInfo *info, PetscReal *W_j, Mat J, Mat P, [AppCtx](#) *user)
jacobian of nonlinear equations
- PetscErrorCode [CalculateOmega](#) (DMDALocalInfo *info, PetscReal *W, [AppCtx](#) *user)
calculate ω

5.3.1 Detailed Description

Functions and structs for solving PDE.

Author

Robert F. DeJaco

5.3.2 Function Documentation

5.3.2.1 CalculateOmega()

```
PetscErrorCode CalculateOmega (
    DMDALocalInfo * info,
    PetscReal * W,
    AppCtx * user )
```

calculate ω

Parameter used to determine if simulation is at steady state

Returns

void

Parameters

	<i>info</i>	grid object
<i>in</i>	<i>W</i>	solution
<i>in, out</i>	<i>user</i>	stores attribute omega

5.3.2.2 DestroyVecAndArray()

```
PetscErrorCode DestroyVecAndArray (
```

```
DM da,
VecAndArray * x )
```

Destroy Struct containing both vector and array.

See also

[VecAndArray](#)

Parameters

in	<i>da</i>	manages an abstract grid object and its interactions with the algebraic solvers
	<i>x</i>	Vector and array to destroy

5.3.2.3 eval_dS_i_j()

```
PetscReal eval_dS_i_j (
    PetscReal df_i_j,
    PetscReal dT )
```

evaluates gradient in solid concentration wrt fluid

returns gradient, $\Delta\tau F'/2/\varepsilon$

Parameters

in	$\frac{df}{di_j}$	gradient in isotherm
in	<i>dT</i>	scaled time step, likely $\Delta\tau/\varepsilon$

5.3.2.4 eval_S_i_j()

```
PetscReal eval_S_i_j (
    PetscReal f_j,
    PetscReal f_jm1,
    PetscReal exp_mdT,
    PetscReal S_jm1,
    PetscReal dT )
```

evaluate value of solid concentration

Returns

value of solid concentration S_i^j

Parameters

f_j	value of adsorption isotherm at i and j
f_{jm1}	value of adsorption isotherm at i but j - 1
exp_mdT	value for $e^{-\Delta\tau/\varepsilon}$
S_{jm1}	Value for solid concentration at i but j - 1
dT	value for $\Delta\tau/\varepsilon$

5.3.2.5 FormFunctionLocal()

```
PetscErrorCode FormFunctionLocal (
    DMDALocalInfo * info,
    PetscReal *  $\bar{W}_j$ ,
    PetscReal * FF,
    AppCtx * user )
```

residuals of nonlinear equations

Returns

void

Parameters

<i>info</i>	Abstract object with grid info
\bar{W}_j	Solution, used to evaluate residuals
<i>FF</i>	residuals or form function
<i>user</i>	application context

5.3.2.6 FormJacobianLocal()

```
PetscErrorCode FormJacobianLocal (
    DMDALocalInfo * info,
    PetscReal *  $\bar{W}_j$ ,
    Mat J,
    Mat P,
    AppCtx * user )
```

jacobian of nonlinear equations

Returns

void

Parameters

<i>info</i>	Abstract object with grid info
$W_{\leftarrow j}$	Solution, used to evaluate residuals
<i>J</i>	not used?
<i>P</i>	jacobian
<i>user</i>	application context

5.3.2.7 InitializeVecAndArray()

```
PetscErrorCode InitializeVecAndArray (
    DM da,
    VecAndArray * x )
```

Initialize Struct containing both vector and array.

See also

[VecAndArray](#)

Parameters

in	<i>da</i>	manages an abstract grid object and its interactions with the algebraic solvers
	<i>x</i>	Vector and array to initialize

5.3.2.8 one_time_step()

```
PetscErrorCode one_time_step (
    SNES snes,
    DMDALocalInfo * p_info,
    AppCtx * p_user,
    Variables_j * p_vars )
```

Perform one time step.

Parameters

in	<i>snes</i>	Solver
in	<i>p_info</i>	Information about some parameters needed
in, out	<i>p_user</i>	Application context
out	<i>p_vars</i>	Variables at time <i>j</i> , updated for next time step

5.3.2.9 run()

```
PetscErrorCode run (
    int argc,
    char ** args,
    int m,
    double eps,
    double tau_star,
    double theta,
    ISO * isotherm,
    bool compute_analytical,
    double * btc,
    bool make_movie,
    bool print_header )
```

Main function called.

Parameters

in	<i>argc</i>	number of command arguments
in	<i>args</i>	command arguments
in	<i>m</i>	number of time steps
in	<i>eps</i>	value of epsilon ε
in	<i>tau_star</i>	total simulation time τ_*
in	<i>theta</i>	fraction of bfd/ffd
in	<i>isotherm</i>	adsorption isotherm struct. Needs to be allocated/initialized beforehand
in	<i>compute_analytical</i>	whether or not to compute analytical solution
in, out	<i>btc</i>	values of break-through curve
in	<i>make_movie</i>	whether or not to make movie
in	<i>print_header</i>	whether or not to print header

5.3.2.10 simulate()

```
PetscErrorCode simulate (
    DM da,
    SNES snes,
    DMDALocalInfo * p_info,
    AppCtx * p_user )
```

Perform calculations at all time steps.

Parameters

<i>da</i>	! Abstract grid object
<i>snes</i>	! solver
<i>p_info</i>	! local info on grid
<i>p_user</i>	User Application context

5.3.2.11 theta_rule()

```
PetscReal theta_rule (
    PetscReal val_i,
    PetscReal val_im1,
    PetscReal theta )
```

does theta rule for discretization in space

Returns

value of expression averaged wrt theta, $v_i(1 - \theta) + v_{i-1}\theta$ for some v_i, v_{i-1}

Parameters

<i>val_i</i>	value at spatial index i
<i>val_im1</i>	value at spatial index i - 1
<i>theta</i>	value for $0 \leq \theta \leq 1$

5.3.2.12 UpdateAnalytical()

```
PetscErrorCode UpdateAnalytical (
    DMDALocalInfo * info,
    PetscReal * c_j,
    PetscReal * c_jm1,
    AppCtx * user )
```

Updates analytical solution.

Updates analytical solution for time j based off of info for $j - 1$.

Parameters

in	<i>info</i>	information about local grid
in, out	<i>c_j</i>	new solution computed
in	<i>c_jm1</i>	old solution
in	<i>user</i>	Application context

5.3.2.13 UpdateIsotherm()

```
PetscErrorCode UpdateIsotherm (
    DMDALocalInfo * info,
    PetscReal * W,
    AppCtx * user )
```


Parameters

in	<i>info</i>	local information about grid
in	W	Solution of PDE at some value of time
in, out	<i>user</i>	application context. new values of isotherm stored here

5.3.2.14 UpdateSolid()

```
PetscErrorCode UpdateSolid (
    DMDALocalInfo * info,
    PetscReal *  $W_j$ ,
    AppCtx * user )
```

Update solid concentration.

Parameters

in	<i>info</i>	information about grid
in	$W_{\leftrightarrow j}$	Numerical solution of fluid concentration
in	<i>user</i>	application context

Index

- analytical_btc
 - my_funcs.hpp, [18](#)
- AppCtx, [7](#)
 - b, [8](#)
 - beta, [8](#)
 - btc, [8](#)
 - compute_analytical, [8](#)
 - dtau, [8](#)
 - dtau_est, [8](#)
 - dx, [8](#)
 - eps, [9](#)
 - errnorm_all, [9](#)
 - f_jm1, [9](#)
 - isotherm, [9](#)
 - j, [9](#)
 - K, [9](#)
 - kappa, [9](#)
 - m, [9](#)
 - make_movie, [10](#)
 - movie_step_interval, [10](#)
 - omega, [10](#)
 - print_header, [10](#)
 - S_jm1, [10](#)
 - save_btc, [10](#)
 - tau_star, [10](#)
 - theta, [10](#)
 - viewer, [11](#)
- arr
 - VecAndArray, [16](#)
- b
 - AppCtx, [8](#)
- beta
 - AppCtx, [8](#)
- btc
 - AppCtx, [8](#)
- c_j
 - Variables_j, [15](#)
- c_jm1
 - Variables_j, [15](#)
- c_scale
 - EXP_ISO, [11](#)
- c_star
 - EXP_ISO, [11](#)
- CalculateOmega
 - my_petsc.h, [21](#)
- compute_analytical
 - AppCtx, [8](#)
- data
 - ISO, [12](#)
- DestroyVecAndArray
 - my_petsc.h, [21](#)
- dF
 - ISO, [12](#)
- dtau
 - AppCtx, [8](#)
- dtau_est
 - AppCtx, [8](#)
- dx
 - AppCtx, [8](#)
- eps
 - AppCtx, [9](#)
- errnorm_all
 - AppCtx, [9](#)
- error
 - Variables_j, [15](#)
- eval_dS_i_j
 - my_petsc.h, [22](#)
- eval_S_i_j
 - my_petsc.h, [22](#)
- exp_l0
 - my_funcs.hpp, [18](#)
- EXP_ISO, [11](#)
 - c_scale, [11](#)
 - c_star, [11](#)
 - f_scale, [11](#)
 - f_star, [11](#)
 - n, [12](#)
- F
 - ISO, [12](#)
- f_jm1
 - AppCtx, [9](#)
- f_scale
 - EXP_ISO, [11](#)
- f_star
 - EXP_ISO, [11](#)
- FormFunctionLocal
 - my_petsc.h, [23](#)
- FormJacobianLocal
 - my_petsc.h, [23](#)
- InitializeVecAndArray
 - my_petsc.h, [24](#)
- integrate
 - my_funcs.hpp, [18](#)
- ISO, [12](#)

- data, 12
- dF, 12
- F, 12
- num_params, 12
- params, 13
- isotherm, 13
 - AppCtx, 9
- isotherms.Langmuir, 13
- isotherms.Linear, 14
- j
 - AppCtx, 9
- K
 - AppCtx, 9
- kappa
 - AppCtx, 9
- m
 - AppCtx, 9
- make_movie
 - AppCtx, 10
- movie_step_interval
 - AppCtx, 10
- my_funcs.hpp
 - analytical_btc, 18
 - exp_I0, 18
 - integrate, 18
 - update_solution, 19
- my_petsc.h
 - CalculateOmega, 21
 - DestroyVecAndArray, 21
 - eval_dS_i_j, 22
 - eval_S_i_j, 22
 - FormFunctionLocal, 23
 - FormJacobianLocal, 23
 - InitializeVecAndArray, 24
 - one_time_step, 24
 - run, 24
 - simulate, 25
 - theta_rule, 25
 - UpdateAnalytical, 26
 - UpdateIsotherm, 26
 - UpdateSolid, 27
- n
 - EXP_ISO, 12
- num_params
 - ISO, 12
- omega
 - AppCtx, 10
- one_time_step
 - my_petsc.h, 24
- params
 - ISO, 13
- print_header
 - AppCtx, 10
- run
 - my_petsc.h, 24
- S_jm1
 - AppCtx, 10
- save_btc
 - AppCtx, 10
- simulate
 - my_petsc.h, 25
- src/isotherm.h, 17
- src/my_funcs.hpp, 17
- src/my_petsc.h, 20
- tau_star
 - AppCtx, 10
- theta
 - AppCtx, 10
- theta_rule
 - my_petsc.h, 25
- update_solution
 - my_funcs.hpp, 19
- UpdateAnalytical
 - my_petsc.h, 26
- UpdateIsotherm
 - my_petsc.h, 26
- UpdateSolid
 - my_petsc.h, 27
- Variables_j, 14
 - c_j, 15
 - c_jm1, 15
 - error, 15
 - W_j, 15
- vec
 - VecAndArray, 16
- VecAndArray, 15
 - arr, 16
 - vec, 16
- viewer
 - AppCtx, 11
- W_j
 - Variables_j, 15