

## My Project

Generated by Doxygen 1.9.1



<b>1 Todo List</b>	<b>1</b>
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 AppCtx Struct Reference	7
4.1.1 Detailed Description	8
4.1.2 Member Data 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 Member Data 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 Member Data 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 Member Data 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 Member Data Documentation	16
4.9.2.1 arr	16
4.9.2.2 vec	16
<b>5 File Documentation</b>	<b>17</b>
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_I0()	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

<b>Index</b>	<b>29</b>
--------------	-----------



# 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

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

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



## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">src/isortherm.h</a>	Functions and structs for adsorption isotherm . . . . .	17
<a href="#">src/my_funcs.hpp</a>	For calculating analytical solutions . . . . .	17
<a href="#">src/my_petsc.h</a>	Functions and structs for solving PDE . . . . .	20



## Chapter 4

# Class Documentation

### 4.1 AppCtx Struct Reference

Useful parameters for solving, Application context.

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

Collaboration diagram for AppCtx:

#### Public Attributes

- 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 Member Data 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  $\varepsilon$

#### 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  $\tau_*$



#### 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

### Public Attributes

- double \* [c\\_star](#)
- double \* [f\\_star](#)
- int [n](#)
- double [c\\_scale](#)
- double [f\\_scale](#)

### 4.2.1 Member Data 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/isotherm.h](#)

### 4.3 ISO Struct Reference

#### Public Attributes

- [Fun F](#)
- [dFun dF](#)
- int [num\\_params](#)
- double \* [params](#)
- double \* [data](#)

#### 4.3.1 Member Data 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)`

## Public Attributes

- **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)`

### Public Attributes

- **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:

### Public Attributes

- [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 Member Data 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>
```

### Public Attributes

- Vec `vec`
- PetscReal \* `arr`

### 4.9.1 Detailed Description

Used for converting between vector and array types

## 4.9.2 Member Data 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:
```

**Classes**

- 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  $\omega$*

### 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  $\omega$

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 $\varepsilon$
in	<i>tau_star</i>	total simulation time $\tau_*$
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	$W_{\leftarrow j}$	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