FORSCHUNGSZENTRUM JÜLICH GmbH
Institut für Chemie und Dynamik der Geosphäre
D-52425 Jülich, Tel. (02461) 61-6646

User's Manual

Easy AtmoSpheric chemistrY (Ver.2)

Theo Brauers, Franz Rohrer

Version 2.9

(Last Change: July 22, 1999)

This document is under development.

Contents

List of Examples

IMPORTANT MESSAGES

Please see the Release notes Chapter 7 Section 7.1 for important changes in the facsimile code generation. Please check model runs before Version **2.9a** to make sure if

1. the settings are correct
2. times in the output files are correct.

You will find the version number in the header of your output data files and in the file containing the header (usually `*.hd`).

Chapter 1

Introduction

This document describes the application of the EASY compiler which provides a convenient interface for the modeling of chemical systems in zero and one dimensions. This program is based on IDL® [1] and Facsimile® [2]

This document is available as postscript file and pdf file and hypertext mark-up language file. It describes the program after a major change in Dec 1998. Please refer to the release notes (Chapter 7) for the last changes.

The authors can be reached via email at

- Theo Brauers `mailto:Th.Brauers@fz-juelich.de`
- Franz Rohrer `mailto:F.Rohrer@fz-juelich.de`

A EASY www page is maintained at ich313.ich.kfa-juelich.de.

1.1 Defining the problem

This program was designed to create an interface for the modeling of zero-dimensional (box) and one-dimensional (1-d) chemical reactions and transport systems.

The set of reaction can be given as the following

$$\mathrm{R}_j: \quad \sum_{i=0}^{N} \alpha_{ij} C_i \quad \rightarrow \quad \sum_{i=0}^{N} \beta_{ij} C_i$$

where α_{ij} and β_{ij} are the stoichiometric factors for specie i in reaction j for educts and products, respectively. Within this syntax sinks (reactions to Nirvana) the can be written as

$$\mathrm{R}_j: \quad C_i \quad \rightarrow$$

where no products occur. Sources are given by

$$\mathrm{R}_j: Q \quad \rightarrow \quad C_i$$

where Q stands for the source strength.

[1] IDL (RSI), the Interactive Data Language is a powerful, 4GL application development software package for data analysis and visualization. For further info see IDL home page. The program is available for UNIX and Windows systems.

[2] Facsimile (AEA Technology) is an efficient, robust and versatile computer program for modelling complex steady state and time dependent processes. lt is especially suitable for solving chemical reactions with diffusion and/or advection. For further info see Facsimile home page. The program is available for UNIX and Windows systems.

In addition to chemical transformation (including sinks and sources) the transport term is parameterized by a diffusion like transport term

$$T(C_i(x,t), k_D, p, T) = k_D \times \frac{M_i(x+h,t) - M_i(x,t)}{h}$$

with $M_i(x,t)$ is the mixing ratio of compound i calculated from $C_i(x,t)$ and the respective of temperature T and pressure p.

From the reactions and the transport equation the following differential equation can be defined:

$$\frac{\partial \mathbf{C}}{\partial t} = T(\mathbf{C}, k_D, p, T) + P(\mathbf{C}, \mathbf{k}(p, T, \ldots)) - D(\mathbf{C}, \mathbf{k}(p, T, \ldots)) \tag{1.1}$$

where $\mathbf{C} = (C_0(x,t), \ldots, C_N(x,t))$ denoting the array of concentrations of (trace) gases and $\mathbf{k} = (k_0, \ldots, k_M)$ denoting for the rate constant of the respective reaction.

1.2 What EASY does

In general Equation 1.1 cannot be solved analytically thus numerical integration is necessary to solve it with respect to initial and boundary conditions. Facsimile is one of the most abundant and well-tested programs to solve differential equations of the above type. However, it is not always easy and instructive to write Facsimile code in order to solve a given problem.

EASY transforms the simple user written code (see Chapter 3 for a detailed description of the user code) into a program code of Facsimile. Depending on the user input code — typically found in a `*.easy` file — the different species are handled as settings or as variables. All information about the run is also available through a `*.html` file giving a fast debug tool. Facsimile then handles the time dependent or steady state integration of the set of variables. After the integration EASY takes control again an provides a number of tools to plot output of the Facsimile run. It is planned to provide more tools over the time; sensitivity analysis has been implemented recently.

Chapter 2

Running EASY

This chapter explains how to get EASY to work. This includes the system requirements, the setup, and first steps to get EASY running. For a detailed set of examples please refer to Chapter 4.

2.1 Before setup

EASY requires IDL which should be installed and tested. A copy of the Facsimile executable will be distributed with the setup.

2.1.1 IDL

The current version of EASY works with IDL versions 5.1 and 5.2 on WindowsNT® version 4.0 (ServicePack3) and on IBM Workstation using AIX. Even if you recompile the IDL code the programs will not work with versions of IDL before 5.00 since we use '`[...]`' for array subscribting.

In order to run EASY it is not necessary and not advised to compile the program yourself. For the different versions we provide a set of `easy_*.sav` files which include all routines which are required to run the program. However, you should have access to the `easy_startup.pro` routine which loads the `easy_*.sav` file into your IDL session.

2.1.2 FACSIMILE

A Facsimile executable is found within this distribution. It was compiled using the Lahey Fortran 90 compiler V4.5 for 32-bit Windows systems and the f77 compiler for AIX using the `usubrx.f`-file containing the i-o routines for large input and output files. This version of EASY does not run with the standard executable of Facsimile since the `usubr*` routines are required for proper file i-o.

Forschungszentrum Jülich is the owner of a Facsimile site license. The program must not be distributed outside Forschungszentrum Jülich without a permission of AEA Technology to do so.

2.2 Setup

The EASY files are located on
`ich227.ich.kfa-juelich.de/usr/local/icg/icg/idl_source/easy`

2.2.1 directories and files

All files are within the directory easy with a number of subdirs:

easy

program containing program code

idl idl input files

`*.pro`
`easy.sav`

facsimile facsimile executable

`facsim_b.exe`

doc documentation and info

`easy_doc.html`
`node*.html`
`easy_doc.ps`

data input data and examples

defaults usefull definitions

`*.dat`
`defaults.txt`
`alias.txt`

alpha test userscripts during the alpha testing phase

`alpha001.easy`
`alpha002.easy`
`alpha003.easy`

test_v2 tests for the version 2

`*.easy`

incoming directory for exchange with test users

install directory for the local installation containing

`easy_install.cmd` for a guided installation for WinNT4.0
`easy_remove.cmd` for removal of easy on WinNT4.0

The most convenient way to use easy is via the `../idl_source/easy/....` directory which can be used from both Windows and Unix systems. This directory will always contain the latest available version.

If you want a local copy of easy it is advised to copy the whole structure onto your workstation by the using the `easy_install.cmd` procedure.

2.2.2 Environment variables

Be sure to have a number of environment variables set to the respective values. The value may be different for different workstations. Please see your system administrator if you are not sure about these settings.

EDITOR editor program

WinNT: `C:\programme\pfe\pfe32.exe`
Unix: `export EDITOR=em`

BROWSER html browser program

WinNT: `C:\programme\netscape\navigator\program\netscape.exe`
Unix: `export BROWSER=netscpe`

GSVIEW viewer program for postscript files

WinNT: `C:\Programme\gstools\gsview\gsview32.exe`

Unix: `export GSVIEW=gv`

FACSIMILE facsimile executable
WinNT: `C:\easy\program\facsimile\facsimb.exe`
Unix: `export FACSIMILE=/.../facs_aix`

EASY_DIR directory of the easy program for search purposes of files
WinNT: `C:\easy`
Unix: `/usr/.../easy`

2.3 Getting started

The EASY environment is started within IDL.

1. start IDL

2. load program sav file
IDL> `easy_startup`

3. run EASY in the interactive mode
IDL> `easy`

You will be asked for a file name `*.easy` and you can

The following keyword can be used when calling EASY

/TEST_ENVIRONMENT test the setting of the environment variables

/NEW_EASY_FILE open the EASY user manual (this file in ghostview), do not run

/HELP_VIEW open the EASY user manual (this file in ghostview), do not run

/SHORT_HELP display a short help information message window, no run

/NO_MESSAGE quiet mode, no messages in the scroll IDL area

/FILE_MESSAGE write messages into file

/NO_TIMING no time information in the message file

/TEST_ENVIRONMENT check environment variables only, do not run compilation

/HALT_ON_ERROR halt on error and display trace back info (for developing purposes)

/VIEW_HTMLFILE start html browser to

/EDIT_USERFILE edit the user input file before starting the compile process

/GSVIEW_PSFILES view the post script output files after the run

/NO_CREATE_MODEL do not create the model sav file

/NO_FACS_FILE do not create the facsimile input file

/NO_HTML_FILE do not create the html file

/NO_RUN_FACS do not run facsimile from as a spawned process

/SENSITIVITY run facsimile for the specified sensitivity studies

/ANALYSIS_TOOLS load analysis tools for output (UNDER DEVELOPMENT)

/KEEP_SENS_FILES keep sensitivity files after run

/NO_DIAGNOSTICS do not run the diagnostics, i.e. do not create postscript plots

/NO_INTERACTIVE do not run the widget interface

To start easy in the non-interactive mode, with messages written to file and stop on error the following call should be used:
IDL> `easy, /NO_INTER, /FILE_MESSAGE, /HALT_ON_ERROR`

Normally you start EASY using the interactive widget environment which allows you to specify all options by checking the appropirate boxes in the widget window. To start easy in the interactive

mode just use:
IDL> `easy` or specify a file name
IDL> `easy /home/ich399/data/test/test1234`

Chapter 3

Command Syntax

This chapter describes the syntax for the user input file (called userfile; `*.easy`) for EASY. The purpose of this syntax is to provide a convenient interface for the creation of Facsimile input files. Thus this syntax is intended to be closely related to commonly used syntax for chemical reactions. Each of the following sections describes one of the possible commands.

The user file is a plain ASCII file. It must not contain **Umlaute** or other special characters. Multiple white-space characters like SPACE and TAB are not required and are removed before processing starts.

There are three types of commands used:

pre-processor this commands will be executed before writing the log file
include
Comments
Continuation lines

reactions reactions are identified by a special syntax
`...-->...`
`.../->...`

assignments definitions of variables, names, functions, etc. such as:
INPUT, FUNCTION, K, CITE, GEOMETRY, FILES, INITIAL, . . .

multi-box-assignments special syntax for definitions of variables, names, functions, etc.

These types will be explained in the following sections.

3.1 Pre-processor control

A preprocessor will be executed before the user input file is interpreted. This preprocessor will include requested information from other files, will remove all comments, and will resolve all continuation lines. The output of the pre-processor will be written to the `elog` file. This file is an interpreted copy of the userfile which can be used as control and as an inputfile to be included in new userscripts.

3.1.1 Included files

The compiler can read one or more other userfiles which are included via the '`#include`' directive:

```
; read version1
#include version1.defaults
; overwrite with version 2
#includeversion2.defaults
; user defined include
#include my_last_userfile.txt
```

Only one level of include files is processed. If you want to include more levels of include files, be sure to process your files before. Multi level includes can be done using the `get_includes` function of IDL. The files to be included are searched in the path specified in the `EASY_DIR` environment variable expanding the tree of this directory through all levels if you have a `'+'` sign in front of the path.

3.1.2 Comments and Continuation lines

Comments and continuation lines are handled as in IDL `.pro` files. Comments are separated from code using a '`;`' character at any location of the input file. Any text behind '`;`' until the end of line will be ignored. A EASY input line is not (really) limited in length ($<$ 32767 chars.). However, if you want to break the lines for better readability you must use the '`$`' at the end of each line which is continued in the following line.

```
; this is a comment
A3=FUNCTION(%1*3.) ; this is also a comment
A = FUNCTION(       $
           %1 * %2 / %3 )
```

3.2 Reactions

3.2.1 Defining a reaction

The reactions of the chemical system are defined using the following type of command

```
f1 * name1 + f2 * name2 --> f3 * name3 + f4 * name4
```

The sides of the reaction are separated by a '`-->`' string while the separator between the reactants '`name`' on each side is '`+`'. An optional stoichometric '`factor`' is separated from the name of the reactant by '`*`'.

As a first step EASY is running a program which simplifies the respective sides of the chemical reaction in order to identify equal reactions

```
2*NO + O2 --> NO2 + NO2
NO+O2+NO  --> 2.00 * NO2
```

These reaction would be identified as one entry, since both sides of the reaction would give the same set of names of the reactants with the same set of stoichometric factors.

However, reactions

```
OH + CO --> CO2 + H
HO + CO --> CO2 + H
```

would be identified as two different entries to the database of reactions, since `OH` is different from `HO`. You can use the alias command to identify `OH` and `HO` as the same compound:

```
ALIAS[OH]=HO
```

All occurrences of `HO` will then be converted to `OH` before processing. Please keep in mind that the names of the substances are case-sensitive.

3.2.2 Un-defining a reaction

There is also an un-defining command which has the same syntax as the previous command but it removes an entry from the set of reactions used:

```
  f1 * name1 + f2 * name2  -/-> f3 * name3 + f4 * name4
```

Undefining a reaction is especially useful if you are using a pre-defined reaction database to discard a few reactions.

```
OH+CH4 --> 1.5*CO + H2O
OH+1.5*CO --> CO2 + 1.0*H
; define NO2/NO3/N2O5 system
NO2 + O3 --> NO3 + O2
NO2 + NO3 --> N2O5
NO2+NO3-->NO+OH+O3
; now undefine 2 previous reactions
NO2 + O3 -/-> NO3 + O2
NO2 + O3 -/-> N2O5
; define a new combined one
2*NO2 + O3 --> N2O5+ O2
```

Example 3.1: Defining and undefining reactions. Please keep in mind mind that always the last command in a input file matters. I.e. information read from an included file is inserted exactly at the location where the `#include` command appears.

3.3 Assignments

The command lines in this section have the general appearance :

```
  name[identifier]{optional} = functionname(arguments)
```

The following sections describe the different assignment. In general you can assign any name to any function or variable as long as this name is different from any of the commands. It is also advised to use `T` for temperature in K since this parameter is used for the calculation of rate constants with Arrhenius expressions.

3.3.1 Parameters

Parameters are defined by the user either to a return value of a function or an element of a time series to be read from a data input file:

```
  parameter_name = functionname(arguments)
  parameter_name = INPUT(COL=nnn)
```

The '`arguments`' of the function can be any other parameter or const number. If '`parameter_name`' is equal to the name of a species which has been defined in one of the reactions then this species is considered a parameter not a variable, i.e. it does not appear on the left side of the differential equation. For the options of the '`INPUT`' command see section 3.3.4. '`parameter_name`' should not be one of the reserved names which are used for commands:

```
 FILES      INPUT      K      INITIAL      ALIAS
 FUNCTION      UNIT      CITE      FACS      PLOT
 GEOMETRY
```

The parameters can be read from an input file or be defined as a function of other parameters. All species can be either variables (appear on the left side of an differential equation) or parameters.

Those parameters which are read from an input file must be columns of the input file which is specified by the '`FILES[dat]`' or '`FILES[enz]`' command.

```
T = CONST(295.)
P   = CONST(1013.)
CH4 = CONST(2.5E19*1.9E-6)
CH4 = INPUT(COL=4)
OH  = CONST(5.e6)
XT = CONST(1./T)
```

Example 3.2: Defining parameters as functions and from data input files.

```
  FILES[DAT]=filename
  FILES[ENZ]=filename
```

3.3.2 Functions

The function command is used to assign a name to a function which can be used to assign data to a parameter.

```
  function_name = FUNCTION[code_of_function]
```

`code_of_function` can be a any mathematical operation which can be expressed by the operators available in the Facsimile command language. (**binary operators:** `+ - * / @`, **unary functions:** `sin(..)`, `cos(..)`, `exp(..)`, `log(..)`, `log10(..)`, `artan(..)`, `tanh(..)`, `randm(..)`, `abs(..)`, `sign(..)` and **binary functions:** `amax(..)`, `amin(..)`, `amod(..)`) Positional arguments which are used during the function call are numbered from `%1,%2, ...`

```
CONST=FUNCTION[%1]
DENSITY=FUNCTION[%1/(%2*1.379E-19)]
ARRH=FUNCTION[%1*exp(%2/T)]
TROE=FUNCTION[M*%1*(300./T)@%2 $
    /(1.+M*%1*(300./T)@%2/(%3*(300./T)@%4)) $
    *0.6@(1./(1.+(log10(M*%1*(300./T)@%2    $
    /(%3*(300./T)@%4)))**2))   ]
```

Example 3.3: Function definitions from the file defaults.txt. It is advised to include defaults.txt in all user scripts in order to have a proper definition of Arrhenius and Troe functions. The function `CONST` is necessary for assigning any value. It is not valid to write `ABC=5.` you must use `ABC=CONST(5.)`.

3.3.3 Initialize

The initial command is used to assign a start value to a variable

```
  INITIAL[variable_name]=function_name(parameters)
```

where `function_name` can be a any function which has been defined in this userscript or any of included files. `variable_name` can be any name of a species in a reaction. This command is

```
INITIAL[*]=CONST(1) ; initializes all parameters to 1
INITIAL[OH]=CONST(1E6)
INITIAL[O3]=CONST(xO3*M*1E-9)
```

Example 3.4: Initializing variables

only valid for variables. If a species is a parameter then this command is disregarded.

3.3.4 Input

The input command is used to assign a name to the data set in the data input file. The name can be any species in a chemical reaction or a parameter to be defined. A column of the data file is called

by using a `data_set_name` exactly like the header of one of the data columns in the file or via a column number (the first column corresponds to `nnn = 0`).

```
parameter_name = INPUT(COL=nnn)
parameter_name = INPUT(data_set_name)
```

The column with the name '`TIME ...`' is always assigned to the time of the run. This column is necessary for the operation of the model. You must not assign the parameter time. However you can assign the column with time to any other parameter.

```
O3 = INPUT(COL=3)
TC = INPUT(TEMP)
xx = INPUT(TIME)
```

Example 3.5: Getting input from the data file

3.3.5 Alias

Alias names are used to identify species which are identified by different names:

```
ALIAS[name0]=name1 name2
```

If `name1` or `name2` are found in the list of reactants their names will be replaced by `name0`:

```
HO + CO --> CO2 + H
ALIAS[OH]=HO hydroxyl
```

would be converted to

```
CO + OH --> CO2 + H
```

common examples are These and other common alias names are defined in the file alias001.txt

```
ALIAS[OH]=HO hydroxyl
ALIAS[HONO]=HNO2
```

Example 3.6: Alias names demonstrated

which is part of this distribution.

3.3.6 Units

not yet implemented / under construction

All calculations in the Facsimile are in units s, cm-3, K etc. To convert units for input and output of data use:

```
UNIT[speciename]=unitname
```

examples are:

```
UNIT[NO3]=ppt
UNIT[O3]=ppb
```

3.3.7 File names

EASY filenames can be set through the files command:

```
FILES[default_extension]=filename
```

where `default_extension` is used to identify the file type. Valid types are

DAT data input file for Facsimile special format [1]

[1] dat files use one line of header with ',' separated entries. all other lines of the files are input data records which contain numerical input only.

ENZ data input file for Facsimile using the enz [2] file style

ELOG log file after preprocessing the user script

HTML html document describing all variables used for writing the facsimile file

FAC facsimile input file

OUT facsimile output log file

CS facsimile output data file for concentrations and parameters

PD facsimile output data file for production and destruction rates

KV facsimile output data file for the actual values of the rate constants and reaction rates

TR facsimile output data file for the actual values of the transport matrix and mixing ratios

Only `dat`/`enz` and `easy` files are input, all other are produced by the program. In addition postscript graphic files are produced from the `cs`, `pd`, `tr`, and `kv` files. These files have '`.ps`' appended to the respective file names of the output data file.

```
FILES[ENZ]=master1.enz
FILES[ELOG]=my_log
FILES[HTML]=xxxxx
FILES[FAC]=input1
FILES[OUT]=input1
FILES[PD]=pd123.dat
FILES[CS]=cs.dat
FILES[KV]=abc
FILES[TR]=abc
```

Example 3.7: File definitions. In normal operation it is not necesarry to use these command since EASY will choose the projectname with the respective default extension to create a filename.

If `filename` includes an extension the extension in `filename` is used. If no extension is provided the `default_extension` will be attached to the filename. If no filename is provided for a certain filetype the name of the userscript will be used. Please keep in mind that Facsimile requires DOS type (8.3) filenames.

3.3.8 Citations

Short tags for literature references can be defined by the following :

```
  cite_short=CITE[long citation text]
```

where `short_cite` is a string (no spaces) defining a tag which can be used to identify certain reaction rates belonging to a data set. An example follows:

```
JPL94=CITE[DeMore et al., JPL Reference Publication]
k[OH+CH4-->CH3+H2O]{JPL94}=ARRH[2.7E-12,1800]
k[OH+CH2O-->HCO+H2O]{JPL94}=CONST[1.E-11]
```

Example 3.8: Usage of citations: The parameter JPL94 is used as a shortcut for the full reference.

3.3.9 Facsimile parameters

Facsimile can be initialized with some special parameters:

[2] enz files use a line &&&&& as separator between header and data. The header must at least contain COLUMNx=text for each data column and NUMBER OF COLUMNS=n. Further and detailed information can be accessed from ICH313 home page or at http://ich313.ich.kfa-juelich.de/dataformat.txt However, the data must not contain any non numerical input which was allowed in earlier versions of the enz format.

```
FACS[HMAX]= value
FACS[OUTSTEP]= value
FACS[STEADY_STATE_TIME]= number of seconds
FACS[USUBR_MSG]= 0 / 1
```

None of these parameters is required for a standard run. The time frame of the run is provided by the input file and output is written at each new record from this file.

You might use the following data file for a simulation of 1day:

```
time
0
43200
86400
-1
```

Example 3.9: Data file for a one day run in the DAT format with output at 0 h, 12 h, 24 h

If you want to have additional output at other times you might either add lines to the data file or you might specify the `OUTSTEP` keyword like in the following example:

```
FACS[OUTSTEP]= 300
```

Example 3.10: Using the outstep parameter to specify output every 5 min in addition to the output defined by the input file.

If you want to run a steady state run you must specify the time after which you think the steady stae is reached. Pleas specify the `STEADY_STATE_TIME` keyword like in the following example:

```
FACS[STEADY_STATE_TIME]= 864000
```

Example 3.11: Using the steady state time parameter to specify a 10-day approximation of the staedy state for each each input data set.

3.3.10 Plot Parameters

The plot parameters are used to setup the style for the diagnostic plots

```
PLOT[rows]=number_of_rows
PLOT[cols]=number_of_columns
PLOT[merge]=0 or 1
PLOT[ref_date]=dd.mm.yyyy hh:mm:ss
PLOT[ref_date]=use
PLOT[landscape]=0 or 1
PLOT[pagetitle]=text
PLOT[stamp]=-1 or 0 or 1
PLOT[skip]=
PLOT[missv]=number
```

The parameter `PLOT[ref_date]` can either provide a reference date corresponding to `time=0` or if `PLOT[ref_date]=use` is set the reference time `TIME/ SEC SINCE dd.mm.yyyy hh:mm:ss` provided by the input data file is used for plotting the time axis. If this parameter is not set the axis will be in seconds as given by the input file.

3.4 Multi box assignments

The command lines for multi box problem differ slightly from their previously defined commands. In general, there is an optional argument `box_index` which is set in `<...>` brackets.

```
name[identifier]<box_index> = $
functionname(<arguments, .... >)
```

Nearly all commands of section 3.3 can be modified in order to assign parameters to arrays of boxes. Since EASY can work on 1-dimensional problems only there is only one additional index for the box number.

3.4.1 Geometry

Facsimile can be initialized with some special parameters:

```
GEOMETRY[NBOX]= number_of boxes
GEOMETRY[BODY]= type_of_body
GEOMETRY[DELTA_H] =<thickness_of_layer>
GEOMETRY[AREA]= <area_between_layers>
GEOMETRY[VOLUME]= <volume_of_the_layer>
GEOMETRY[K_D]=<exchange_parameter>
GEOMETRY[LENGTH]= <length>
```

These parameters are not required for a box model, since no volume depended terms are calculated. For a 1-d model you must at least specify the number of boxes `GEOMETRY[NBOX]` and the exchange parameter `GEOMETRY[K_D]`. All parameters are assumed to be in the same units. If you specify the layer thickness in meters the exchange parameter is given in meters^2/sec.

The `GEOMETRY[BODY]` command has several options. It can be set to ONE of the following values: `SPHERE` `TUBE` `BARREL` `CUBE` and `COLUMN`. The latter is the default setting. From these body parameters and the respective thickness spacing the volume and surface area of a layer is calculated and used for the calculation of the transport from one box to one or neighboring boxes.

3.4.2 Other commands

All parameters and initial values can be set different for each box of the model. However, the rate constants of reactions and the set of the reactions themselves are are the same for every box. However, you can introduce parameters into reactions or rate constants in order to run different chemistry models in the different boxes of the model.

```
GEOMETRY[NBOX]=3
GEOMETRY[K_D]=<.5, .5, 1.>
A --> B
B --> C
k[A --> B]=CONST(beta*alpha)
k[B --> C]=CONST(beta*alpha)
alpha<*>=0.01
alpha<1>=0
beta=ARRH(1E-12, 200)
.....
```

Example 3.12: 1d model

3.5 Sensitivity Studies

Sensitivity analysis allows the re-execution of the a model with one input parameter changed by a factor. With syntax for sensitivity you can specify an arbitrary number of factors in order to change the input parameter `identifier` over a wide range of values:

```
SENSITIVITY[identifier1]= factor11, factor12, ...
SENSITIVITY[identifier2]= factor21, factor22, ...
SENSITIVITY2[identifier2]= delta
```

Please keep in mind that for every factor a full model run will be executed which might need much computer time. Even if the SENSITIVITY command is present in the EASY script it must be started via the /SENSITIVITY keyword or via the sensitivity check box of the widget.

The sensitivity2

3.6 Analysis tools

The output of the EASY model runs is converted into time series plots as first level tool for the analysis of the output. However, often it is requested to have correlations between different input and output values of the model or to get an overview over the budgets or ...

The selection menu

Proposed structure of a analysis menu:

- selection of analysis
- selection of columns / files to be requested
- output of the selected columns in single file !!
- plot of selected columns versus one other column
- correlation analysis
- ....

Chapter 4

Examples

In this chapter a few simple examples will show the application of EASY. All examples will be distributed with the EASY program for test purposes.

4.1 Two coupled first order reactions in one box

In the following we will simulate the reactions

$$A \rightarrow B \rightarrow C$$

```
; write down the reactions
A --> B
B -->C
; write down the reaction constants
k[A --> B]{ex1}=CONST(LAMBDA/200.)
k[B --> C]{ex1}=CONST(GAMMA/400.)
; define the function used
CONST=FUNCTION(%1)
; initialize the variables
INITIAL[*]=CONST(1.)
INITIAL[A]=CONST(1000.)
; define the scaling factor
LAMBDA=CONST(1.)
GAMMA=CONST(1.)
; set facsimile parameters
FACS[HMAX]=2
FACS[OUTSTEP]=20
; input data file
FILES[DAT]=1000sec.dat
; cite
ex1=CITE[example1]
```

Example 4.1: EASY code for a simple set of 2 coupled reactions.

If you run this example you start IDL in the `easy\program\idl` directory

IDL> `easy_startup`

IDL> `easy, '..\..\data\examples\exampl1'`

the following output will be produced:

- log file including all commands of the userfile and the included files
- hypertext mark-up language file

- facsimile input file, the program code to run Facsimile: exampl1.fac
- facsimile output file, the status report of the facsimile run
- data output file for concentrations and settings
- data output file for production and destruction rates
- data output file for rate constants
- postscript plot for concentrations and settings
- postscript plot for production and destruction rates exampl1.pd.ps
- postscript plot for rate constants exampl1.kv.ps

4.2 Diffusion of one species

column shaped diffusion, one space coordinate z

$$\frac{\partial C}{\partial t} = k_D \frac{\partial}{\partial z}(\frac{\partial C}{\partial z}) - \frac{C}{\tau} + Q(z) \quad \forall z > 0 \tag{4.1}$$

$$Q(z) = \{ \begin{array}{ll} \neq 0 & \forall z < Z_0 \\ = 0 & \forall z > Z_0 \end{array} \tag{4.2}$$

```
FILES[DAT]=1000sec.dat
M=CONST(2.5E19)
Q=CONST(0.)
W=CONST(0.)
Q<0>=CONST(1E10)
W<8>=CONST(1E10)
Q--> A
k[Q-->A]{}=CONST(1.)
A-->
k[A-->]=CONST(.01)
A + W -->
k[A + W-->]{}=CONST(.01)
GEOMETRY[NBOX]=9
GEOMETRY[BODY]=COLUMN
GEOMETRY[K_D]=<.4>
GEOMETRY[DELTA_H]=<4.19,29.3,79.6,155,256,381,532,708,909>
INITIAL[A<0>]=CONST(2.5E10)
INITIAL[A<1:8>]=CONST(1.0)
FACS[HMAX]=1
FACS[OUTSTEP]=5
```

Example 4.2: EASY code for diffusion of one tracer in a column geometry.

Sphere shaped diffusion, one space coordinate r

$$\frac{\partial C}{\partial t} = k_D \frac{1}{r^2} \frac{\partial}{\partial r}(r^2 \frac{\partial C}{\partial r}) - \frac{C}{\tau} + Q(r) \quad \forall r > 0 \tag{4.3}$$

For $t \to \infty$ the stationary solution with a rectangular source distribution

$$Q(r) = \{ \begin{array}{ll} \neq 0 & \forall r < R_0 \\ = 0 & \forall r > R_0 \end{array} \tag{4.4}$$

and the boundary condition $C(\infty) = 0$ is given as:

$$C(r) = \alpha \frac{\exp(-r/\beta)}{r} \quad \text{with} \quad \beta = \sqrt{k_D \cdot \tau} \tag{4.5}$$

The integration constant α can be obtained by integrating the profile $C(r)$:

$$4\pi \int_0^\infty C(r) r^2 dr = 4\pi\alpha \int_0^\infty \exp(-\frac{r}{\beta}) r dr = 4\pi \cdot \alpha \cdot k_D \cdot \tau \tag{4.6}$$

Under steady state conditions the value is equal to the product of source strength and lifetime $Q \cdot \tau$ yielding

$$\alpha = \frac{1}{4\pi} \frac{Q}{k_D} \tag{4.7}$$

The time dependent solution for $Q(r) = 0$ and $C(r, t = 0) = \delta(r)$ is given by

$$C(r,t) = (4\pi k_D t)^{-3/2} \times \exp(-\frac{r^2}{4k_D t}) \times \exp(-\frac{t}{\tau}) \tag{4.8}$$

```
FILES[DAT]=1000sec.dat
M=CONST(2.5E19)
Q=CONST(0.)
Q<0>=CONST(1E10)
Q--> A
k[Q-->A]{}=CONST(1.)
A-->
k[A-->]=CONST(.02)
GEOMETRY[NBOX]=12
GEOMETRY[BODY]=SPHERE
GEOMETRY[K_D]=<.05>
GEOMETRY[DELTA_H]=<1.,1.,1.,1.,1.,1.,1.,1.,1.,10.,100.>
INITIAL[A]=CONST(0.0)
FACS[HMAX]=0.1
FACS[OUTSTEP]=1
```

Example 4.3: EASY code for diffusion of one tracer in a sphere geometry. (`diff_t1.easy`)

Chapter 5

Frequently Asked Questions

5.1 relevant news groups

Q: Which are relevant news groups and information on the web
A: kfa.icg3.forum.idl
kfa.icg3.forum.modell
kfa.forum.idl

5.2 FACSIMILE

Q: Is there a size limit for arrays in Facsimile **?**
A:

5.3 IDL

Q: Which versions of IDL can be used?
A: IDL5.0, 5.1, 5.2, ...

5.4 Easy

Q: What is the maximum number of reactions, species, etc. ?
A: We never tested that so far. However, there is a limitation reading the output files with IDL in order to produce postscript files. A single line of data must not exceed 65534 char of input with is a maximum of about 4600 columns in one of the output files.

Q: When are species in reactions parameters and variables?
A: If a specie name occurs on the left side of an assignment it is always set to a parameter and it occurs in the settings section of html file. All other species are assumed to be variables.

Chapter 6

Things to do

analysis tools budget
sensitivity
scatter plots necessary.
The analysis tools are still in a preliminary status

Chapter 7

Release notes

7.1 notes: 1998-july-20

VERSION Version number now 2.9a

facsimile code IMPORTANT CHANGE IN FACSIMILE CODE GENERATION
during a change in the facsimile code generation the calculation of settings were corrupted. An error in the facsimile code could occur if a setting was function of an other setting. Also an error with time handling in the facsimile code was corrected. Please see the output of your model runs or ask Theo Brauers or Franz Rohrer for further info.

7.2 notes: 1998-july-14

analysis tools plot environment connected to analysis

facsimile code works now with one specie, one reaction, and/or one setting. However, one entry is necessary for each field.

7.3 notes: 1998-may-4

analysis tools new section with analysis tools UNDER DEVELOPMENT

sensitivity analysis new section with sensitivity tools UNDER DEVELOPMENT

7.4 notes: 1998-may-12

warnings extended warnings handling (functions, plot parameters, ..)

7.5 notes: 1998-apr-27

sensitivity new sensitivity section

7.6 notes: 1998-jan-31

reaction handling substance names in reaction are case sensitive

test of multi box column, sphere, tube and barrel geometry

search path for included and data files via environment variable EASY_DIR

7.7 notes: 1998-dec-22

new plot parameters some of the plot options can be set in the user file

new widget interface most of the options can be called through widget. /NOINTERACTIVE keyword for classic style

input of enz file new FILES[ENZ]=.... directive

facsimile new version with largely enhanced array sizes and a interface to enz files via the usubr0 ... usubr5 interface

7.8 notes: 1998-feb-25

rename to easy.pro etc. the new version is now called easy and the program will be called by easy, ...

7.9 notes: 1998-feb-24

new version of XXX.sav etc.

7.10 notes: 1998-feb-20

FILES command files command default ext .DAT for data file to be read from Facsimile (see easy13tb.log)

UNITS command units command not yet implemented

7.11 notes: 1998-feb-19

dos names dos names are required for all files used by Facsimile

model2facsimile changed nset to nvar in output loop (see easy13tb.log)

Chapter 8

Acknowledgements

We gratefully acknowledge the help of Jörg-Peter Kohlmann with the USUBRxx routines for the Facsimile program and Helga London for help with Fortran compiler. The EASY program also includes a number of IDL routines which were written by Reimar Bauer, Ray Sterner, and other contributors to the IDL libaries.

Index