A Computer Based Kinetics Experiment using Monte Carlo Techniques

Michael Lake

30th July 2001

Contents

Acknowledgements

The author would like to thank the following persons who have helped in the development of this software;

Dr Louise Evans, whose initial suggestion for a computer based experiment led to this software. Dr Barbara Stuart who provided an example of a Monte Carlo simulation and provided some of the textual material used for the notes. Dr Philip Maynard for suggestions on the user interface. The Sydney Linux Users Group and its members for their assistance in developing this software.

This software was developed entirely with open source software such as Perl, gvim, LaTeX, latex2html, noweb and many others. It runs on a Linux/GNU system with the Apache Web Server. Particular thanks therefore go to all of the developers and contributors to the open source movement.

The author of this software can be contacted at:

Michael Lake
Department of Chemistry, Materials and Forensic Science
University of Technology, Sydney
PO Box 123, BROADWAY
NSW 2007
`Mike.Lake@uts.edu.au`

(or at my home email: `mikel@speleonics.com.au`)

Introduction

This is a computer based experiment to assist students in understanding chemical kinetics by studying what happens on a molecular scale using Monte Carlo techinques. Simulations of zero, first and second order reactions are done.

The experiment is most suitable for a pair of students however several pairs of students can run the experiment simultaneously.

The experiment itself is based on a paper entitled "Monte Carlo Experiments in Chemical Kinetics", author unknown [2].

Change History

Date	Version	Change
30-07-2001	1.08	* Removed all instances of the actual server name in code and documentation for security reasons
18-07-2001	1.07	* Replaced code in mc2 with "common preamble" and added $scriptName.
16-07-2001	1.06	* Added "Order [0—1—2] to title of graph, this makes it clear if the graph is not updated due to caching or unable to be written. * Changed name of subroutine `return_order` to `return_form` as it described better the purpose of the subroutine. * Added an option to create pdf in the Makefile.
14-07-2001	1.05	* Moved "Back" and "Analysis" links from bottom of form to to top of form as students sometimes don't see them. * Added section on latex2html configuration file * Changed location of `graph` and `plot` from `/usr/local/bin/` to `/usr/bin/` in scripts 'analysis', 'mc create graph' and test subroutine * Replaced `$server="martel"` with `` $server=`hostname` `` so I don't have to change it between home and the UTS. * latex2html needs ppmtogif to create gifs but Debian's netpbm utils do not have it. Created a short script which uses convert to turn pngs into gifs. * Changed URL from a web page to an email for myself as UTS no longer allow personal web pages.
27-06-2001	1.04	* Changed place where the HTML files get put too. Now files are put into a public_html direcroty under user "kinetics".
19-08-2000	1.03	* Added note at top of index.html to use Netscape as MS Explorer does not show the forms correctly. Added URL to enter.
10-08-2000	1.02	* print-random modified so a random number eg. 5 now prints as 05 as students didn't know whether a 5 meant 05 or 50
05-07-2000	1.01	* Fixed bug in viewtable so now previous and next do not extend beyond first and last data sets. * Regression now only extends to last non zero value rather than to last array value. * Fixed bug with labels on analysis plot * Replaced instances of `$tmp[$i]=$refA->[$i]` with `push(@tmp, $refA->[$i]);` otherwise under some conditions `$tmp[$i]` could contain unprocessed data.
04-07-2000	1.0	* Release 1.0 operational.

1 Using noweb or the Makefile

This documentation was written using `noweb`—a literate programming tool by Norman Ramsey [1]. You can extract the code and documentation using `noweb` commands directly or by using `make` and the supplied Makefile.

1.1 Creating Programs and Documentation using noweb Directly

Table 1 shows typical `noweb` commands used to extract code and documentation. There are more code and documentation options available than shown in this table; the `makefile` is the definative guide. The options supplied to `noweb` are listed in Table 2. The `Makefile` is a more convenient way to perform the same tasks and is covered on page 7.

To create the:	Run (all on one line):
Makefile	`notangle -t4 -RMakefile kinetics.nw | cpif Makefile`
mc program	`notangle -t4 -L -Rmc kinetics.nw | cpif mc`
LaTeX doc	`noweave -delay -index kinetics.nw > kinetics.tex`
dvi	`latex kinetics.tex`
PDF	`pdflatex kinetics.tex`
Postscript	`dvips kinetics.dvi -o kinetics.ps`
HTML docs	`noweave -html -filter l2h -x kinetics.nw > kinetics.html`
All the files:	`noweb kinetics.nw`

Table 1: Quick summary of noweb usage

notangle/noweave options	
-t4	Copy tabs untouched from input to output, and use tabs for indentation. Tabs get set to 8 by default in noweb.
-R	extracts root chunk from noweb file
-L	insert line numbers into the output code as comments

Table 2: Options in `notangle/noweave` usage

1.2 Creating Programs and Documentation using make

The following Makefile provides a convenient way to create the code or documentation after modifications to the `noweb` source file rather than typing all the `notangle` or `noweave` commands. Code or documentation changes are done changing the `noweb` source file and running the appropriate `make` command. One generally never changes the output files directly (except for quick hacks).

To extract the Makefile see Table 1 above. Run "`make`" with no options or "`make help`" to see what options there are.

For instance, after making changes to the HTML pages via the `noweb` source file run "`make dvi`" to update LaTeXdocumentation or "`make html`" to create the up-to-date web pages.

```
$ make all
$ ./allpngs2gifs
# make install
```

Note: you may have to run latex twice on each tex file that uses cross references such as equation cross references to resolve them before you create the HTML files from them.

8 ⟨*Makefile* 8⟩≡

```
# Makefile for creating kinetics scripts

# Version 1.1 July 2001

# The source for all code and documentation.
NW_SOURCE = kinetics.nw

# If the user just types 'make' with no args then some help, being the
# first routine will be invoked.
help:
    @echo 'Usage: make [mc html dvi ps htmldocs test all clean install'
    @echo '       all       - code, html, html2 (make all, make install will \
    upgrade)'
    @echo '       code      - the kinetics programs '
    @echo '       html      - Manually created HTML pages'
    @echo '       html2     - LaTeX generated HTML pages'
    @echo '       dvi       - dvi documentation'
    @echo '       ps        - PostScript documentation'
    @echo '       htmldocs  - HTML documentation'
    @echo '       tests     - HTML test pages and cgi scripts for Martel ONLY'
    @echo '       clean     - removes LaTeX, HTML (*.tex, *.html)'
    @echo '       install   - copies html files and cgi scripts to public \
    directories'
    @echo '                   (You must be root to install.'

# Create everything except test routines
all: code html html2

# Create code
code: $(NW_SOURCE)
    notangle -t4 -Rmc0 $(NW_SOURCE) | cpif cgi-bin/mc0
    notangle -t4 -Rmc1 $(NW_SOURCE) | cpif cgi-bin/mc1
    notangle -t4 -Rmc2 $(NW_SOURCE) | cpif cgi-bin/mc2
    notangle -t4 -Rviewdata $(NW_SOURCE) | cpif cgi-bin/viewdata
    notangle -t4 -Ranalysis $(NW_SOURCE) | cpif cgi-bin/analysis
    notangle -t4 -Rnph-mc $(NW_SOURCE) | cpif cgi-bin/nph-mc
    notangle -t4 -Rprint-random $(NW_SOURCE) | cpif print-random
    notangle -t4 -Rallpngs2gifs $(NW_SOURCE) | cpif allpngs2gifs

    @echo 'Setting executable status...'
    chmod 775 $(CGI_SCRIPTS)
    chmod 775 print-random allpngs2gifs

# List what files above will be copied to the installation directory.
CGI_SCRIPTS = cgi-bin/mc0 cgi-bin/mc1 cgi-bin/mc2 cgi-bin/viewdata cgi-bin/analysis

# Create HTML files
html: $(NW_SOURCE)
    @echo 'Extracting root HTML files... '
    notangle -t4 -Rstyles.css $(NW_SOURCE) | cpif styles.css
    notangle -t4 -Rindex.html $(NW_SOURCE) | cpif index.html
    notangle -t4 -Rmanual.html $(NW_SOURCE) | cpif manual/manual.html
    notangle -t4 -Rcomputer.html $(NW_SOURCE) | cpif computer/computer.html
    notangle -t4 -Rabout.html $(NW_SOURCE) | cpif about.html
    notangle -t4 -Rhelp.html $(NW_SOURCE) | cpif help.html
    notangle -t4 -Rhelp2.html $(NW_SOURCE) | cpif help2.html

# List all the above HTML files for copying to root installation directory.
# Note: these files will never have mathematical equations in them and so
# will never be created using latex2html
HTML_ROOT = styles.css index.html about.html help.html help2.html COPYING

# Create latex2html generated HTML files
```

```
html2: $(NW_SOURCE)
	@echo 'Extracting LaTeX files and converting to HTML ... '
	notangle -t4 -Rintro.tex $(NW_SOURCE) | cpif intro.tex
	notangle -t4 -Rtheory.tex $(NW_SOURCE) | cpif theory.tex
	notangle -t4 -Requations.tex $(NW_SOURCE) | cpif equations.tex
	latex2html -T Kinetics -split 0 -no_navigation -info 0 intro.tex > /dev/null
	latex2html -T Kinetics -split 0 -no_navigation -info 0 theory.tex > /dev/null
	latex2html -T Kinetics -split 0 -no_navigation -info 0 equations.tex > /dev/null

	@echo 'Override styles in latex2html generated files.'
	cat styles.css >> intro/intro.css
	cat styles.css >> theory/theory.css
	@echo 'If equations.tex is changed the images in the equations directory \
	must be manually copied to the images directory prior to "make install".'

	@echo 'You may have to run the script ./allpngs2gifs before "make install"'

# Create documentation for this software.
dvi: $(NW_SOURCE)
	noweave -t4 -delay -index $(NW_SOURCE) >| kinetics.tex
	latex kinetics.tex
	@echo
	@echo 'You may have to run latex again (not - make dvi)'
	@echo
	@echo 'latex kinetics.tex'
	@echo

pdf: $(NW_SOURCE)
	noweave -t4 -delay -index $(NW_SOURCE) >| kinetics.tex
	pdflatex kinetics.tex
	@echo
	@echo 'You may have to run pdflatex again (not `make pdf')'
	@echo
	@echo 'pdflatex kinetics.tex'
	@echo

ps: dvi
	dvips kinetics.dvi -o kinetics.ps


htmldocs: $(NW_SOURCE)
	noweave -html -filter l2h -index $(NW_SOURCE) | htmltoc >| kinetics.html


# Create test routines; "make install" to install them.
tests: $(NW_SOURCE)
	@echo 'Extracting tests...'
	notangle -t4 -Rtests.html  $(NW_SOURCE) | cpif tests.html

	notangle -t4 -RtestApache  $(NW_SOURCE) | cpif cgi-bin/testApache
	notangle -t4 -Rtest-locations $(NW_SOURCE) | cpif cgi-bin/test-locations
	notangle -t4 -Rnph-testCGI $(NW_SOURCE) | cpif cgi-bin/nph-testCGI
	notangle -t4 -RtestCGI-1   $(NW_SOURCE) | cpif cgi-bin/testCGI-1
	notangle -t4 -RtestCGI-2   $(NW_SOURCE) | cpif cgi-bin/testCGI-2

# List what files above will be copied to the installation directory.
TEST_SCRIPTS = cgi-bin/testApache cgi-bin/test-locations \
cgi-bin/nph-testCGI cgi-bin/testCGI-1 cgi-bin/testCGI-2


# These HTML files either have, or in the future may have, mathematical
# equations in them. Hence they may be created using latex2html and so
# they are in directories created by latex2html.
INTRO  = intro/intro.html intro/*.gif intro/*.css
```

```
THEORY = theory/theory.html theory/*.gif theory/*.css
COMPUTER = computer/computer.html
MANUAL = manual/manual.html

install:
#    if [ ! -d /home/kinetics/public_html/kinetics/monte_carlo/intro ] ; then
#        mkdir /home/kinetics/public_html/kinetics/monte_carlo/intro
#    fi
    @echo 'Copying HTML files to /home/kinetics/monte_carlo/ directory...'
    cp $(HTML_ROOT) /home/kinetics/public_html/monte_carlo/
    cp $(INTRO)     /home/kinetics/public_html/monte_carlo/intro/
    cp $(THEORY)    /home/kinetics/public_html/monte_carlo/theory/
    cp $(MANUAL)    /home/kinetics/public_html/monte_carlo/manual/
    cp $(COMPUTER)  /home/kinetics/public_html/monte_carlo/computer/
    cp images/*.gif /home/kinetics/public_html/monte_carlo/images/

    @echo 'Copying CGI files to cgi-bin directory...'
    cp $(CGI_SCRIPTS)  /home/kinetics/cgi-bin/
    cp $(TEST_SCRIPTS) /home/kinetics/cgi-bin/
    cp tests.html /home/kinetics/public_html/monte_carlo/

    cp COPYING /home/kinetics/public_html/

    @echo 'Setting executable and owner status...'
    chown kinetics:www-data /home/kinetics/cgi-bin/*
    chmod 754 /home/kinetics/cgi-bin/*
    chown -R kinetics:www-data /home/kinetics/public_html/
    chown -R www-data:www-data /home/kinetics/public_html/monte_carlo/tmp/*

dist:
    @echo 'Creating distribution tarballs...'
    #tar cvf cgi-scripts.tar cgi-bin/*

clean:
    rm -f *.tex
    rm -f *.html
    rm -f print-random
    rm -f allpngs2gifs
    rm -f styles.css
    rm -f `find . -name *.old`

#   lintex  (without lintex we have to use lines below)
    rm -f *.aux
    rm -f *.dvi
    rm -f *.log
    rm -f *.toc

    @echo 'If you want to remove the Makefile do it manually.'
```

2 Setting up the Apache Web Server

Enter the following URL into your web browser, replacing SERVER with the name of the host machine (obtained from the hostname command):

http://SERVER/monte/

The Kinetics Experiment Main Page should appear because the default page to load when there is a trailing / is index.html - which is the name of the main page.

To test the Apache web server or test the forms enter `tests.html` after the URL above.

The Apache version used for this development was apache-1.3.9-13.2 and its configuration files are in: `/etc/apache/`.

2.1 Apache Configuration File - `httpd.conf`

The file `httpd.conf` is where all configuration is now. The files `access.conf` and `srm.conf` are no longer used. All of my additions and changes in are indicated by an "MRL" string.

```
# MRL June2001: added Alias for kinetics experiement.
Alias /monte/ /home/kinetics/public_html/monte_carlo/

# ScriptAlias: This controls which directories contain server
# scripts. ScriptAliases are essentially the same as Aliases,
# except that documents in the realname directory are treated
# as applications and run by the server when requested
# rather than as documents sent to the client. The same rules
# about trailing "/" apply to ScriptAlias directives as to
# Alias.
#
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
# MRL: added line below
ScriptAlias /kinetics/cgi-bin/ /home/kinetics/cgi-bin/


<DirectoryMatch ^/home/.*/public_html>
Options Indexes SymLinksIfOwnerMatch
AllowOverride None
# MRL added "order deny,allow" and "deny from all"
order deny,allow
deny from all
#allow from 192.168.1.      # allow from all machines in subnet
#allow from 127.0.0.1       # why doesn't 127.0.0.1 work?
#allow from localhost       # why does localhost work?
allow from 138.25.82.16     # allow from my own IP addr
allow from 138.25.82.56     # allow from mol
</DirectoryMatch>


#
# New Kinetics Experiment - under /home/kinetics
# MRL June 2001:
#
<Directory /home/kinetics/public_html>
    AllowOverride FileInfo AuthConfig Limit
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    Order deny,allow
    deny from all
    allow from 138.25.82.56     # allow from mol
```

```
    # MRL added following to stop caching of gifs
    ExpiresActive On
    # set to 12 days in the past to be sure.
    ExpiresByType image/gif A-999999
</Directory>

<Directory /home/kinetics/cgi-bin>
    AllowOverride None
    Options ExecCGI
    Order deny,allow
    deny from all
    allow from 138.25.82.56     # allow from mol
</Directory>
```

Note regarding the `ScriptAlias` directive: I call the script in my html files with the syntax: `<FORM METHOD=GET ACTION="/kinetics/cgi-bin/testApache">` so if you change the ScriptAlias you must change how the scripts are called.

2.2 Permissions on Directories

The correct permissions on `/home/kinetics/` are:

```
drwxr-xr-x   5 root     root     1024 Jun 2 21:25 kinetics
```

The correct permissions on directories under `/home/kinetics/` are:

```
drwxrwxr-x   3 kinetics   kinetics   1024 Jul 23 23:58 cgi-bin
TODO add public_html

/home/kinetics/public_html/monte_carlo
Files:
-rw-r--r--    1 kinetics www-data    17992 Jul 15 17:27 COPYING
-rw-r--r--    1 kinetics www-data     2232 Jul 15 17:27 about.html
-rw-r--r--    1 kinetics www-data      436 Jul 15 17:27 help.html
-rw-r--r--    1 kinetics www-data      832 Jul 15 17:27 index.html
-rw-r--r--    1 kinetics www-data      180 Jul 15 17:27 styles.css

Directories:
drwxr-xr-x    2 kinetics www-data     4096 Jul 17  2000 computer
drwxr-xr-x    2 kinetics www-data     4096 Jul 14 22:23 images
drwxr-xr-x    3 kinetics www-data     4096 Jul 14 21:45 intro
drwxr-xr-x    2 kinetics www-data     4096 Jul 25  2000 manual
drwxr-xr-x    2 kinetics www-data     4096 Jul 14 22:23 theory
drwxrwxrwx    2 kinetics www-data     4096 Jul 15 18:09 tmp
```

Note that the last directory, `tmp`, must have write permission for the data files and gifs to be wriiten to it.

```
    martel:/tmp$
    plot -T gif --bitmap-size 400x400 Kinetics.192.168.1.4.meta > \
        /home/kinetics/public_html/monte_carlo/tmp/Kinetics.192.168.1.4.gif

    sh: /home/kinetics/public_html/monte_carlo/tmp/Kinetics.192.168.1.4.gif:
    Permission denied


martel:/home/kinetics/public_html/monte_carlo/tmp# ls -l
total 0
-rw-r--r--    1 kinetics www-data        0 Jul 15 18:31 Kinetics..gif
```

```
-rw-r--r--    1 kinetics www-data        0 Jul 15 18:31 Kinetics.192.168.1.4.gif
-rw-r--r--    1 kinetics www-data        0 Jul 15 18:31 Kinetics.192.168.1.4.processed.gif

martel:/home/kinetics/public_html/monte_carlo/tmp# chmod g+w *
martel:/home/kinetics/public_html/monte_carlo/tmp# ls -l

total 0
-rw-rw-r--    1 kinetics www-data        0 Jul 15 18:31 Kinetics..gif
-rw-rw-r--    1 kinetics www-data        0 Jul 15 18:31 Kinetics.192.168.1.4.gif
-rw-rw-r--    1 kinetics www-data        0 Jul 15 18:31 Kinetics.192.168.1.4.processed.gif

martel:/home/kinetics/public_html/monte_carlo/tmp#
```

2.3 Trouble Shooting Apache Problems

Tailing the Apache access and error logs will give some information on errors:

```
tail -f /var/log/apache/access_log
tail -f /var/log/apache/error_log
```

Common errors are listed below.

Error message:

```
Global symbol "$scriptName" requires explicit package name at \
    /home/kinetics/cgi-bin/mc2 line 263.
Execution of /home/kinetics/cgi-bin/mc2 aborted due to compilation errors.
[Thu Jul 19 02:14:08 2001] [error] [client 138.25.82.56] \
    Premature end of script headers: /home/kinetics/cgi-bin/mc2
```

Solution:

Declare the variable by prepending "my": `my $scriptName = "mc2";`

Error message:

```
Cannot open file data for writing! at /home/kinetics/cgi-bin/mc
line 37.  [Sun May 21 13:18:43 2000] [error] [client 192.168.1.1]
Premature end of script headers: /home/kinetics/cgi-bin/mc
```

Solution:

TODO

Error message:

```
[Sun May 21 18:31:16 2000] [error] [client 192.168.1.1]
file permissions deny server execution:
    /home/kinetics/cgi-bin/testCGIPush-nph
```

Solution:

Using: `chmod +x testCGIPush-nph`
Change: `-rw-rw-r--  kinetics  kinetics  May 21 18:31 testCGIPush-nph`
To: `-rwxrwxr-x  kinetics  kinetics  May 21 18:31 testCGIPush-nph`

Error message:

```
You don't have permission to access /cgi-bin/testApache on
this server.
```

Solution:

You probably don't have the script executable. Check in cgi-bin that its mode is executable.

Also is the executable bit set on the parent directory so that Apache has access? Test this by su'ing to the same user as Apache runs as and try to run the script as that user, ie. if Apache runs as www-data (check with ps -aux or look in `httpd.conf` for the directive "User www-data")

```
$ su root   (have to su root as only root can su to a non-login account)
# su www-data
cgi-bin\$ ./testApache
```

and testApache should run.

Don't forget to exit from user 'www-data'.

Error message:

```
The requested URL /kinetics/cgi-bin/testApache was not found on
this server.
```

Solution:

TODO

Error message:

```
The server encountered an internal error or misconfiguration and was unable
to complete your request.
Please contact the server administrator, root@localhost and inform them of
the time the error occurred, and anything you might have done that may
have caused the error.
More information about this error may be available in the server error log.
```

Solution:

Running `./mc` from the command line gives the following error:
`Cannot open file data for writing! at ./mc line 20.`

We can see from...

```
public_html:$ ls -l
drwxrwxrwx   2 kinetics    kinetics        1024 May 21 22:45 tmp

TODO check if it is kinetics or www-data

public_html:$ ls -l tmp/
-rw-r--r--   1 kinetics   kinetics      41358 May 23 22:40 data
-rw-r--r--   1 kinetics   kinetics      10608 May 23 22:40 order.gif
-rw-r--r--   1 kinetics   kinetics      38095 May 23 22:40 temp.meta
```

That as `mikel` you can't write to `public_html/tmp` files as they are owned by another person.

Error message:

When you access `http://server_name/monte` you get a directory listing of `/home/kinetics/public_html/monte_carlo/` instead of the web page ie. `index.html`. You can see the file `index.html` there yet when you click on it in the browser the error message "The requested URL /monte/index.html was not found on this server." appears.

Solution:

The Alias in `/etc/apache/httpd.conf` or `srm.conf` has probably got an incorrect trailing slash. The following will give this problem:

`Alias /monte/ /home/kinetics/public_html/monte_carlo`

Then when you try to add index.html onto the end of the URL a tail of the Apache error log will show:

```
[Mon Jul 16 21:50:53 2001] [error] [client 138.25.82.56] File does not
exist: /home/kinetics/public_html/monte_carloindex.html
```

Carefully note that no / appears between "carlo" and "index" in the error message. This is because `monte/` will be replaced by the full path - without a trailing / - and then the `index.html` will be immediately appended.

Make sure therefore that you use either:

`Alias /monte/ /home/kinetics/public_html/monte_carlo/` or

`Alias /monte /home/kinetics/public_html/monte_carlo`

Error message:

When you access the "Computer Experiment Pages" and select an experiment eg. "Run First Order Reaction" the page appears but there is no graph - a broken image symbol appears.

Solution:

The Apache error log will show:

```
[Mon Jul 16 22:09:25 2001] [error] [client 138.25.82.56] File does not exist: \
/home/kinetics/public_html/monte_carlo/tmp/Kinetics.56.gif
```

First does the directory monte_carlo/tmp exist for the gif to be written into?
Are the permissions set so that it can be written into by Apache?

This is not sufficient:
```
drwxr-sr-x    2 kinetics www-data     4096 Jul 16 22:16 tmp
```

These permissions are required:
```
drwxrwxr-x    2 kinetics www-data     4096 Jul 16 22:16 tmp
```

via `chmod 775 /home/kinetics/public_html/monte_carlo/tmp/`

Error message:

The experiment runs but the graph, ie the gif image, does not appear to change from run to run or even from one order to another.

Solution:

The gif for a particular data set or run may be there but it can't be written too by Apache so its not being updated. Check that the image can be written by running the script as the same user as Apache runs as, eg. "www-data", from the command line.

This is what is entered into the browser: `./mc0?Ao=40&Bo=0&TotalMols=2000&maxN=1000` To run this same script from the command line we do: `./mc0 Ao=40 Bo=0 TotalMols=2000 maxN=1000` ie. use spaces where there were ampersands.

You will probably find a line like this:
```
sh: /home/kinetics/public_html/monte_carlo/tmp/Kinetics..gif: Permission denied
```

Tips:

Netscape will cache pages so if you want to see changes to your web pages after making changes to the cgi scripts you may need to force Netscape to flush its cache. Hold down the control key as you click reload or alternatively set the following under `Edit/Preferences/Advanced`:

> Disk cache 0 kBytes
> Memory cache 0 kBytes
> Document in cache is compared to document on network - Never

This can also be useful as a means to tell if the graph is not being updated by Netscape caching it or by the script not being able to write a new one over the old one.

2.4 Using sudo

We can execute the cgi scripts as user *kinetics* by using `sudo`.
```
sudo -u kinetics ~/cgi-bin/mc2
```

First though user mikel must be entered into the `/etc/sudoers` file. Otherwise if you try to execute the above you will get the following error:

```
Sorry, user mikel is not allowed to execute ./mc as kinetics
on Martel.bluesky.com.au
```

File: `/etc/sudoers`

```
# sudoers file.
# This file MUST be edited with the 'visudo' command as
# root.  # See the man page for the details on how to
# write a sudoers file.

# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL) ALL
```

```
mikel   localhost=/bru/bru
mikel   localhost=/usr/local/bin/ppp-on
mikel   localhost=/usr/local/bin/ppp-off
mikel   martel=(kinetics) NOPASSWD: /home/mikel/cgi-bin/
```

2.5 Security

`perldoc -q int`

How do I make sure users can't enter values into a form that cause my CGI script to do bad things?

Read the CGI security FAQ, at

http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html,

and the Perl/CGI FAQ at

http://www.perl.com/CPAN/doc/FAQs/cgi/perl-cgi-faq.html.

In brief: use tainting (see perlsec), which makes sure that data from outside your script (eg, CGI parameters) are never used in C eval or C system calls. In addition to tainting, never use the single-argument form of system() or exec(). Instead, supply the command and arguments as a list, which prevents shell globbing.

Also see `perldoc -q cgi` for Security and performance issues.

3 Required Packages

3.1 Required plotutils Package

The GNU Plotting Utilities provides the programs `graph` and `plot`. See the Appendix for installation details.

3.2 Required Perl Modules

The following Perl modules are required:

- CGI
- Statistics::OLS

The `CGI` module is used to provide a simple and clean interface for the programs. You will need to import this module with: `use CGI qw/:standard/;`

The Perl module `Statistics::OLS` is used for calculating ordinary least squares regression. Details for its installation can be found in Appendix 1.

Documentation on thes modules can be found in the usual way with Perl ie. `perldoc CGI` and `perldoc Statistics::OLS`. Both of these modules have a good example at the start.

3.3 Required LaTeXPackages

The `html` package is used within the Introduction and Theory sections.

3.4 Required latex2html Package

The `latex2html` program is used to convert LaTeXdocuments to HTML documents, equations being converted to png or gifs.

Specifying `-split +0` puts the entire document into a single HTML file.

`-t TITLE` specifies the title.

`-info 0` suppresses the creation of an "About this document..." which otherwise will be generated at the botttom of the document.

The `/etc/latex2html.config` file can be modified so that gifs are created by default instead of pngs. But, if you do not have the utility `ppmtogif` to convert the ppm files you will not get any output. In this case leave the setting as png and use the `convert` utility from the ImageMagick suite to convert them to gifs manually afterwards. There is a script to do this for all pngs in the Appendices.

This is what latex2html does behind the scenes, use the `-debug` flag to see this:

```
Running "/usr/bin/ppmquant -floyd 256 < /tmp/l2h3754/p3909.pnm | /usr/bin/ppmtogif \
-interlace -trans '#ffffff'  > img11.gif"

# MRL July 2001: commented out the line below and added a new line with
#                order changed to (gif png)
#@IMAGE_TYPES = qw(png gif);
@IMAGE_TYPES = qw(gif png);
$IMAGE_TYPE  = $IMAGE_TYPES[0];
```

The `./latex2html-init` file in your home directory probably doesn't need to be modified as all required parameters are passed to latex2html via the command line in the Makefile. Nevertheless below are the changes that acomplish the same.

```
#LaTeX2HTML Version 96.1 : dot.latex2html-init
```

```
# MRL: search for this string for my changes

# If this is set to 0 then any inlined images generated from "figure"
# environments will NOT be transparent.
$TRANSPARENT_FIGURES = 1;

# MRL June2001: changed MAX_SPLIT_DEPTH from default of 8 to zero.
$MAX_SPLIT_DEPTH = 8;   # Stop making separate files at this depth
$MAX_LINK_DEPTH = 4;    # Stop showing child nodes at this depth
$NOLATEX = 0;           # 1 = do not pass unknown environments to Latex
$EXTERNAL_IMAGES = 0;   # 1 = leave the images outside the document
$ASCII_MODE = 0;        # 1 = do not use any icons or internal images

$default_title = "Kinetics";  # MRL July 2001: added a title
$TITLE = $default_title;      # The default is "No Title"

$NO_NAVIGATION = 1;     # 1 = do not put a navigation panel at the top
                        #     of each page
$AUTO_NAVIGATION = 0;
$NEXT_PAGE_IN_NAVIGATION = 0;
$INFO = 0;              # 0 = do not make a "About this document..." section
$NETSCAPE_HTML = 1;
```

4 Monte Carlo Simulation Scripts

There are three simulation programs `mc0`, `mc1` and `mc2` for zeroth, first and second order reactions respectively.

4.1 Overview of the Scripts

4.1.1 Zeroth Order

20 ⟨*mc0* 20⟩≡

```
⟨common preamble 23a⟩ # use modules, global file locations and time.
⟨mc preamble 23b⟩

# Retrieve cookies telling us needed information on the previous data.
my $userID  = cookie('userID');
# Write data with last quad of IP addr appended for user identification.
$file = $file.$userID;

# Get Parameters
($Ao, $Bo, $TotalMols, $maxN) = getParams();

# Initialise values for a Monomolecular reaction
($NA, $NB, $NC) = init(0, $Ao, $Bo, $TotalMols, $maxN);
$NAo = $NA;
$NBo = $NB;

# Simulate reactions
for ($i=0; $i<$maxN; $i++)
{
    if ($NA != 0) # If number of atoms of A is not zero.
    {
        ($NA, $NB) = runDice0($NA, $NB);
    }
    $A[$i] = $NA;
    $B[$i] = $NB;
}

writeData("$tmp_dir/$file", \@A, \@B);

my $order_title = 0;
⟨mc create graph 29⟩
return_form(0);

⟨mc common subroutines 24⟩
⟨mc0 subroutines 27a⟩
⟨mc0 mc1 subroutines 32⟩
⟨file write subroutines 31⟩
⟨error handling subroutines 39⟩
⟨debugging subroutines 40⟩
```

4.1.2 First Order

21 ⟨*mc1* 21⟩≡

```
⟨common preamble 23a⟩ # use modules, global file locations and time.
⟨mc preamble 23b⟩

# Retrieve cookies telling us needed information on the previous data.
my $userID  = cookie('userID');
# Write data with last quad of IP addr appended for user identification.
$file = $file.$userID;

# Get Parameters
($Ao, $Bo, $TotalMols, $maxN) = getParams();

# Initialise values for a Monomolecular reaction
($NA, $NB, $NC) = init(1, $Ao, $Bo, $TotalMols, $maxN);
$NAo = $NA;
$NBo = $NB;

# Simulate reactions
for ($i=0; $i<$maxN; $i++)
{
    if ($NA != 0) # If number of atoms of A is not zero.
    {
        ($NA, $NB) = runDice1($NA, $NB);
    }
    $A[$i] = $NA;
    $B[$i] = $NB;
}

writeData("$tmp_dir/$file", \@A, \@B);

my $order_title = 1;
⟨mc create graph 29⟩
return_form(1);

⟨mc common subroutines 24⟩
⟨mc1 subroutines 27b⟩
⟨mc0 mc1 subroutines 32⟩
⟨file write subroutines 31⟩
⟨error handling subroutines 39⟩
⟨debugging subroutines 40⟩
```

4.1.3 Second Order

22 ⟨*mc2* 22⟩≡

```
⟨common preamble 23a⟩ # use modules, global file locations and time.
⟨mc preamble 23b⟩

# Retrieve cookies telling us needed information on the previous data.
my $userID  = cookie('userID');
# Write data with last quad of IP addr appended for user identification.
$file = $file.$userID;

# Get Parameters
($Ao, $Bo, $TotalMols, $maxN) = getParams();

# Initialise values for a Monomolecular reaction
($NA, $NB, $NC) = init(2, $Ao, $Bo, $TotalMols, $maxN);
$NAo = $NA;
$NBo = $NB;

# Simulate reactions
for ($i=0; $i<$maxN; $i++)
{
    if ( ($NA != 0) && ($NB != 0) ) # If number of atoms of A is not zero.
    {
        ($NA, $NB, $NC) = runDice2($NA, $NB, $NC);
    }

    $A[$i] = $NA;
    $B[$i] = $NB;
    $C[$i] = $NC;
}

writeData("$tmp_dir/$file", \@A, \@B, \@C);

my $order_title = 2;
⟨mc create graph 29⟩
return_form2();

⟨mc common subroutines 24⟩
⟨mc2 subroutines 28⟩
⟨file write subroutines 31⟩
⟨error handling subroutines 39⟩
⟨debugging subroutines 40⟩
```

4.2 Perl Preamble

This software is being developed on one machine and run onanother. In order to easily run it on either computer the variable `$server` is set by a hostname lookup.

Common preamble for all CGI scripts. Describes what modules are used, global file locations and time().

23a ⟨*common preamble* 23a⟩≡ (20–22 42 51 77c)

```
#!/usr/bin/perl -w

# Web-based Kinetics Experiment using Monte-Carlo Technique.
# Copyright (C) 2000  Michael Lake, mikel@speleonics.com.au
#
# 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.

use strict;
use CGI qw/:standard/;  # load CGI routines
use Statistics::OLS;

my $server = `hostname`;
chomp($server);     # we have to remove the newline or "A HREF's" are broken
                    # over two lines
my $html_base = "http://$server/monte";
my $file_base = "/home/kinetics/public_html/monte_carlo";
my $tmp_dir = "/tmp";
my $file = "Kinetics."; # label any files created by these programs so you
                        # know where they come from

my $seconds = time();
```

Common preamble for Monte Carlo simulation scripts. This is not used by the analysis and viewdata scripts.

23b ⟨*mc preamble* 23b⟩≡ (20–22)

```
my ($i, $j);
my (@molArray1, @molArray2, @A, @B, @C);

my $TotalMols;  # Maximum size of the array of molecules ie
                # No. molecules in the beaker.
my $Ao;         # Percentage initial concentration of reactant A.
my $Bo;         # Percentage initial concentration of reactant B.
my $NAo;        # No. of molecules of A initially ie $Ao * $TotalMols
my $NBo;        # No. of molecules of B initially ie $Bo * $TotalMols
my $maxN;       # Max number of reactions.
my ($NA, $NB, $NC); # No. of molecules at any later time.


# We must reinit this hash to null before each invocation.
my %form_input_errors = ();

# Declare hash for error messages and set each one.
my %error_messages;
$error_messages{"Ao"} = "The concentration of A must be numeric.";
$error_messages{"Bo"} = "The concentration of B must be numeric.";
$error_messages{"TotalMols"} = "The total number of molecules must be an integer.";
$error_messages{"maxN"} = "The maximum number of tries must be an integer.";
```

4.3 Reading Command Line Parameters

When the static HTML page (eg. order1.html) fires off the cgi scripts there will be several parameters. Parameters such as Ao etc will be taken from the defaults setup when the variables were declared in the perl preamble.

24 ⟨*mc common subroutines* 24⟩≡ (20–22) 26 ▷

```
###############
# Subroutines
###############

sub getParams
{
    # TODO - get a hash back of all keys and check for specific keys for
    # each case.

    # Get any parameters the user has entered from the form or that
    # static pages have provided. If there are parameters param()
    # will return true. This doesn't mean the params are meaningful !
    if ( param() )
    {
        $Ao = param("Ao");
        $Bo = param("Bo");
        $TotalMols = param("TotalMols");
        $maxN = param("maxN");

        # The user may have sent rubbish either in the text entry boxes
        # or directly entered in the URL.
        if ( ($Ao =~ /[^0-9]/) || ($Ao eq "") )
        {
            # User has entered something other than just digits OR
            # they have not entered anything ie Ao=null.

            # This hash with key "Ao" is set to 1 which signals an error.
            $form_input_errors{"Ao"} = 1;   # raise error flag
            $Ao = 10;
        }

        if ( ($Bo =~ /[^0-9]/) || ($Ao eq "") )
        {
            $form_input_errors{"Bo"} = 1;
            $Bo = 0;
        }

        if ( ($Ao+$Bo) > 100 )
        {
                $form_input_errors{"Ao"} = 1;
                $form_input_errors{"Bo"} = 1;
                $Ao = 10;
                $Bo = 10;
        }

        if ( ($TotalMols =~ /[^0-9]/) || ($TotalMols eq "") )
        {
            # User has entered something other than just digits.
            $form_input_errors{"TotalMols"} = 1;
            $TotalMols = 1000;
        }

        if ( ($maxN =~ /[^0-9]/) || ($maxN eq "") )
        {
            # User has entered something other than just digits.
            $form_input_errors{"maxN"} = 1;
            $maxN = 1000;
        }
```

```
    }
    else
    # There were no command line parameters at all.
    # This will occur when the user presses the Defaults button whether
    # they are in 1st or 2nd order.
    {
        $Ao = cookie('Ao');
        $Bo = cookie('Bo');
        $TotalMols = cookie('TotalMols');
        $maxN = cookie('maxN');
    }

    return ($Ao, $Bo, $TotalMols, $maxN);
}
```

4.4 Initialising the Arrays

This section is where we initialise the arrays.

26 ⟨*mc common subroutines* 24⟩+≡ (20–22) ◃24 88▹

```
sub init
{
    my ($order, $Ao, $Bo, $TotalMols, $maxN) = @_;
    my ($i, $NAo);

    # Initialise values for a Bimolecular reaction
    # Ao is in percentage, NAo is the number of A molecules.
    $NAo = 0.01 * $Ao * $TotalMols;
    $NBo = 0.01 * $Bo * $TotalMols;

    $NA = $NAo;
    $NB = $NBo;
    $NC = 0;

    # Creating molecular arrays for the solvent.
    for ($i=0; $i<$TotalMols; $i++)
    {
        $molArray1[$i] = 0;
        $molArray2[$i] = 0;
    }

    # Filling with molecules A...
    for ($i=0; $i<$NAo; $i++)
    {
        $molArray1[$i] = "1";
    }

    # Filling with molecules B...
    for ($i=0; $i<$NBo; $i++)
    {
        $molArray2[$i] = "2";
    }

    # Randomise the positions of A molecules.
    fisher_yates_shuffle( \@molArray1 );
    fisher_yates_shuffle( \@molArray2 );

    # Initialise the  concentration arrays [A] and [B] to zero.
    for ($i=0; $i<$maxN; $i++)
    {
        $A[$i] = 0;
        $B[$i] = 0;
        $C[$i] = 0;
    }

    return ($NA, $NB, $NC);
}
```

4.5 Running the Simulation

Elements of the array molArray can be either 0, 1 or 2. Solvent molecules are represented by 0, type A molecules by 1 and type B molecules by 2.

27a ⟨*mc0 subroutines* 27a⟩≡ (20)

```
sub runDice0
{
    my ($NA, $NB) = @_;

    my $rand1 = sprintf "%u", rand $TotalMols;

    # We have a molecule of A to change to B
    # Solvent molecules = 0
    #       A molecules = 1
    #       B molecules = 2
    $molArray1[$rand1] = 2;
    $NA = $NA - 1;
    $NB = $NB + 1;

    return ($NA, $NB);
}
```

27b ⟨*mc1 subroutines* 27b⟩≡ (21 84)

```
sub runDice1
{
    my ($NA, $NB) = @_;

    my $rand1 = sprintf "%u", rand $TotalMols;

    # Solvent molecules = 0
    #       A molecules = 1
    #       B molecules = 2
    if ($molArray1[$rand1] == 0)
    {
        # Do nothing as we hit a solvent molecule.
    }
    elsif ($molArray1[$rand1] == 1)
    {
        # We have a molecule of A to change to B
        $molArray1[$rand1] = 2;
        $NA = $NA - 1;
        $NB = $NB + 1;
    }
    elsif ($molArray1[$rand1] == 2)
    {
        # Do nothing as we hit a molecule of B.
    }
    else
    {
        my $tmp = "Error in function runDice().";
        print STDERR $tmp, "\n";
        $tmp = $tmp."<BR>Contact the programmer.";
        printDebuggingPage($tmp);
    }

    return ($NA, $NB);
}
```

28 ⟨*mc2 subroutines* 28⟩≡ (22) 36 ▷

```
sub runDice2
{
    my ($NA, $NB, $NC) = @_;

    # Get two integer random numbers.
    my $rand1 = sprintf "%u", rand $TotalMols;  # rand number for grid 1
    my $rand2 = sprintf "%u", rand $TotalMols;  # rand number for grid 2

    # Solvent molecules = 0
    #       A molecules = 1
    #       B molecules = 2
    if ( ($molArray1[$rand1] == 1) && ($molArray2[$rand2] == 2) )
    {
        # We have a molecule of A and B to change to C
        $molArray1[$rand1] = 3;
        $molArray2[$rand2] = 3;
        $NA = $NA - 1;
        $NB = $NB - 1;
        $NC = $NC + 1;
    }

    return ($NA, $NB, $NC);
}
```

4.6 Create Graph of Data

Two utilities are used here; *graph* takes raw data and creates a meta data file while *plot* reads this meta data and creates a *gif* image which can be displayed on the web browser.

Example 1: Use graph to read data and create meta data with label "Test".

```
graph -a -L 'Test' /tmp/data >|
/home/kinetics/public_html/monte/tmp/temp.meta
```

Example 2: Use plot to read meta data and create a gif image data.

```
plot -T gif --bitmap-size 400x400 temp.meta >| order.gif
```

Note that in the code we use absolute pathnames like `/usr/bin` when we do a system call for security reasons. The `- a` before the input data file indicates tht we want graph to automatically generate abscissa values as the data file only contains ordinate values.

29 ⟨*mc create graph* 29⟩≡ (20–22)

```
#####################
# Create the Graphs
#####################

# Notice the trailing space required at the end of all but the last command.

my ($program, $command, $label, $axis, $type, $size, $infile, $outfile);

# First create a GNU metafile of the input data using graph.
$program = "/usr/bin/graph ";
$label   = "-L 'Monte Carlo Simulation, Order $order_title' ";
$axis    = "-X 'Number of Runs' -Y 'Number of Molecules' ";
$infile  = "-a $tmp_dir/$file ";
$outfile = "$tmp_dir/$file.meta ";
$command = $program.$label.$axis.$infile.">|".$outfile;
system("$command");

# Now create a gif image of the data from this metafile using plot.
$program = "/usr/bin/plot ";
$type    = "-T gif ";
$size    = "--bitmap-size 400x400 ";
$infile  = "$tmp_dir/$file.meta ";
$outfile = "$file_base/tmp/$file.gif ";
$command = $program.$type.$size.$infile.">|".$outfile;
system("$command");
```

4.7 File IO Routines

30 ⟨*file read subroutines* 30⟩≡ (42 51)

```
sub readData
{
    my $file = $_[0];

    my (@A, @B, @C);
    my $blank = 0;

    open (IN, $file) || die "Cannot open file for reading\!";

    while(<IN>)
    {
        chomp();    # Get rid of trailing new line otherwise processed data
                    # will have blank lines between all data values.
        s/^\s*//;   # Get rid of all leading spaces so that
                    # blank lines will be null.

        # We have to match $_ on it being not equal to a "" because if we
        # happen to have a data line = numerical 0 then $_ would be
        # false and the if would be skipped.
        if ($_ ne "")
        {
            if ($blank==0)
            {
                push(@A, $_);
            }
            elsif ($blank==2)
            {
                push(@B, $_);
            }
            elsif ($blank==4)
            {
                push(@C, $_);
            }
        }
        else
        {
            $blank = $blank + 1;
        }
    }

    close(IN);
    return (\@A, \@B, \@C);
}
```

31 ⟨*file write subroutines* 31⟩≡ (20–22 42)

```
sub writeData
{
    my ($file, $refA, $refB, $refC) = @_;
    # There will be at least the filename and $refA, $refB as args.
    # If there was no $refC arg passed then $refC will just be undef.

    my $i;

    open (OUT, ">$file") || die "Cannot open file $file for writing\!";
    my $oldhandle = select OUT;

    # Print data to output.
    for ($i=0; $i<$#{$refA}; $i++)
    {
        print $refA->[$i], "\n";
    }

    if ($refB)  # If not undefined print array.
    {
        print "\n\n";   # required by gnuplot - separates data series
        for ($i=0; $i<$#{$refB}; $i++)
        {
            print $refB->[$i], "\n";
        }
    }

    if ($refC)  # If not undefined print array.
    {
        print "\n\n";   # required by gnuplot - separates data series
        for ($i=0; $i<$#{$refC}; $i++)
        {
            print $refC->[$i], "\n";
        }
    }

    select $oldhandle;
    close(OUT);
}
```

4.8 Create Dynamic Forms and Send Back to Browser

4.8.1 Form for First Order Reaction

Note: For debugging table design add a 'border' to the tables like so:
`<TABLE border=1>`

A problem was encountered in the initial development of this script when it was found that Netscape was not reading the newly created gif image but fetching the old gif from it's cache. Even clicking SHIFT-Reload didn't convince Netscape to use the new gif. The solution was to append a ? followed by a unique tag to each newly created gif so that Netscape interpreted the gif as requiring a parameter. This forced Netscape to reload the gif.

`src="http://$server/kinetics/tmp/order.gif?$seconds"`

This is in perldoc CGI which covers use of the *override value*.

> Another Note: The default values that you specify for the forms are only used the first time the script is invoked (when there is no query string). On subsequent invocations of the script (when there is a query string), the former values are used even if they are blank.
>
> If you want to change the value of a field from its previous value, you have two choices:
>
> (1) call the param() method to set it.
>
> (2) use the -override (alias -force) parameter (a new feature in version 2.15). This forces the default value to be used, regardless of the previous value:

32 ⟨*mc0 mc1 subroutines* 32⟩≡ (20 21)

```
##################################################
# Create the Result Form and send back to Browser
##################################################

sub return_form
{
    my $order = $_[0];

    my ($title, $scriptName);
    if ($order == 0)
    {
        $title = "Zeroth Order Kinetics";
        $scriptName = "mc0";
    }
    elsif ($order == 1)
    {
        $title = "First Order Kinetics";
        $scriptName = "mc1";
    }
    else
    {
        # TODO - Handle error.
    }

    my $q = new CGI;                    # create new CGI object
    my ($userID, $remote_ip);

    # Get the remote IP address to use as a session ID.
    # returns the string "192.168.1.4"
    $remote_ip = $q->remote_host(); # eg if 192.168.1.4
    # TODO we really need to split(/./, $remote_ip)
    $remote_ip =~ s/138.25.82.//;   # then $remote_ip = 4

    # create cookies
    my $userIDC     = $q->cookie(-name=>'userID',   -value=>$remote_ip);
    my $orderC      = $q->cookie(-name=>'order',    -value=>$order);
    my $NAoC        = $q->cookie(-name=>'NAo',      -value=>$NAo);
    my $NBoC        = $q->cookie(-name=>'NBo',      -value=>$NBo);
```

```
    my $TotalMolsC  = $q->cookie(-name=>'TotalMols',-value=>$TotalMols);
    my $maxNC       = $q->cookie(-name=>'maxN',     -value=>$maxN);

    # You must incorporate the cookie into the HTTP header.
    print $q->header(-cookie=>[$userIDC,$orderC,$NAoC,$NBoC,$TotalMolsC,$maxNC]);

    print $q->start_html({
            -title=>"Kinetics",
            -style=>{'src'=>"$html_base/styles.css"} });

    print "\n";
    print $q->h1($title);

    print $q->startform("GET", "$scriptName");

    print $q->hidden(-name=>'Bo', -default=>'0');

    # Main navigation bar
    # (Don't indent this next code chunk as it uses a 'print here' !)
⟨html top navbar2 60d⟩

    # Main table for form.
    print "<TABLE>";

# -------------
# ROW 1
# -------------

print <<here;
<TR>
    <TD rowspan="7"><IMG border="1" width="400" height="400"
         src="$html_base/tmp/$file.gif?$seconds"></TD>
    <TD><B>Input</B></TD>
</TR>
here

# -------------
# ROW 2
# -------------

print "<TR><TD>";
if ($form_input_errors{"Ao"})
{
    print "<BLINK><FONT color=red>";
    print "Initial concentration of reactant A:</FONT></BLINK>";
}
else
{
    print "Initial concentration of reactant A:";
}

print "<BR>(between 0 and 100 &#037;)</TD><TD>";
print $q->textfield(-name=>'Ao',
                    -default=>'40',
                    -override=>'0',
                    -size=>3,
                    -maxlength=>3);
print "&#037;</TD></TR>";

# -------------
# ROW 3
# -------------

print "<TR><TD>";
```

```
if ($form_input_errors{"TotalMols"})
{
    print "<FONT color=red>Total number of molecules:</FONT>";
}
else
{
    print "Total number of molecules:";
}
print "<BR>(between 100 and 10,000)</TD>";

print "<TD>";
print $q->textfield(-name=>'TotalMols',
                    -default=>'2000',
                    -override=>'0',
                    -size=>5,
                    -maxlength=>5);
print "</TD></TR>";

# -------------
# ROW 4
# -------------

print "<TR><TD>";
if ($form_input_errors{"maxN"})
{
    print "<FONT color=red>Number of runs:</FONT>";
}
else
{
    print "Number of runs:";
}
print "<BR>(between 1000 and 50,000)</TD><TD>";
print $q->textfield(-name=>'maxN',
                    -default=>'1000',
                    -override=>'0',
                    -size=>5,
                    -maxlength=>5);
print "</TD></TR>";

# -------------
# ROW 5
# -------------

print <<here;
<TR>
    <TD align="center" colspan=2>
here

# defaults() creates a button that, when invoked, will cause
# the form to be completely reset to its defaults, wiping
# out all the changes the user ever made.
print $q->defaults(' Default Values ');

print <<here;
    <INPUT type=submit value=" Start Reaction "></TD>
</TR>
here

# -------------
# ROW 6
# -------------

print <<here;
<TR>
```

```
    <TD colspan="2">
    <TABLE>
        <TR><TD><B>Data Used</B>        </TD><TD></TD></TR>
        <TR><TD>Initial concentration of A was: </TD><TD>$Ao &#037;</TD></TR>
        <TR><TD>Initial number of molecules of A was: </TD><TD>$NAo</TD></TR>
        <TR><TD>Total number of molecules was:</TD><TD>$TotalMols</TD></TR>
        <TR><TD>Number of runs was:     </TD><TD>$maxN</TD></TR>
    </TABLE>
    </TD>
</TR>
here

# -------------
# ROW 7
# -------------

print <<here;
<TR><TD colspan=2><B>Observations</B><BR>
        The number of molecules of <B>A</B> should be decreasing.<BR>
        The number of molecules of <B>B</B> should be increasing.<BR>
</TR>
</TABLE>
<P>
<HR>
here

print_error_details();

# MRL July 2001: moved "Back" and "Analysis" links to top as often students
# dont have the browser window tall enough to see the links.
# Current section navigation bar
#print <<here;
#<TABLE width="100%">
#<TR>
#   <TD align="left"><A HREF="/kinetics/cgi-bin/$scriptName">Back</A>
#   <TD align="right"><A HREF="/kinetics/cgi-bin/analysis">Analysis</A>
#</TR>
#</TABLE>
#here


    print $q->endform;
    print $q->end_html; # end the HTML
}
```

4.8.2 Form for Second Order Reaction

36 ⟨*mc2 subroutines* 28⟩+≡ (22) ◁28

```
sub return_form2
{
    my $q = new CGI;                    # create new CGI object

    my ($userID, $remote_ip);
    my $order = 2;
    my $scriptName = "mc2"; # In mc0 and mc1 we get this from the if/else block.

    # Get the remote IP address to use as a session ID.
    # returns the string "192.168.1.4"
    $remote_ip = $q->remote_host();
    $remote_ip =~ s/138.25.82.//;

    # create cookies
    my $userIDC     = $q->cookie(-name=>'userID',   -value=>$remote_ip);
    my $orderC      = $q->cookie(-name=>'order',    -value=>$order);
    my $NAoC        = $q->cookie(-name=>'NAo',      -value=>$NAo);
    my $NBoC        = $q->cookie(-name=>'NBo',      -value=>$NBo);
    my $TotalMolsC  = $q->cookie(-name=>'TotalMols',-value=>$TotalMols);
    my $maxNC       = $q->cookie(-name=>'maxN',     -value=>$maxN);

    # You must incorporate the cookie into the HTTP header.
    print $q->header(-cookie=>[$userIDC,$orderC,$NAoC,$NBoC,$TotalMolsC,$maxNC]);

    # start the HTML
    print $q->start_html({
            -title=>"Kinetics",
            -style=>{'src'=>"$html_base/styles.css"} });
    print "\n";

    print $q->h1('Second Order Kinetics');

    print $q->startform("GET","mc2");

    # Main navigation bar
⟨html top navbar2 60d⟩

# Main table for form.
print <<here;
<TABLE>
here

# -------------
# ROW 1
# -------------

print <<here;
<TR>
    <TD rowspan="7"><IMG border="1" width="400" height="400"
         src="$html_base/tmp/$file.gif?$seconds"></TD>
    <TD><B>Input</B></TD>
</TR>
here

# -------------
# ROW 2
# -------------

print "<TR><TD>";
# Column 1 is the gif

# Column 2 for Description.
```

```
if ($form_input_errors{"Ao"} || $form_input_errors{"Bo"})
{
    print "<BLINK><FONT color=red>";
    print "Initial concentration of reactants:</FONT></BLINK>";
}
else
{
    print "Initial concentration of reactants:";
}
print "<BR>(between 0 and 100 &#037;)</TD>";

# Column 3 for Reactant A
print "<TD>A:";
print $q->textfield(-name=>'Ao',
                    -default=>'40',
                    -override=>'0',
                    -size=>3,
                    -maxlength=>3);

# Column 4 for Reactant B
print "<TD>B:";
print $q->textfield(-name=>'Bo',
                    -default=>'40',
                    -override=>'0',
                    -size=>3,
                    -maxlength=>3);
print "&#037;</TD></TR>";

# -------------
# ROW 3
# -------------

print "<TR><TD>";
# Column 1 is the gif

# Column 2 for Description.
if ($form_input_errors{"TotalMols"})
{
    print "<BLINK><FONT color=red>";
    print "Total number of molecules:</FONT></BLINK>";
}
else
{
    print "Total number of molecules:";
}
print "<BR>(between 100 and 10,000)</TD>";

# Column 3-4 for input "Total No. of Mols"
print "<TD colspan=\"2\">";
print $q->textfield(-name=>'TotalMols',
                    -default=>'1000',
                    -override=>'0',
                    -size=>5,
                    -maxlength=>5);
print "</TD></TR>";

# -------------
# ROW 4
# -------------

print "<TR><TD>";
# Column 2 for Description.
if ($form_input_errors{"maxN"})
{
```

```
    print "<BLINK><FONT color=red>";
    print "Number of runs:</FONT></BLINK>";
}
else
{
    print "Number of runs:";
}
print "<BR>(between 1000 and 10,000)</TD>";

# Column 3-4 for input "Number of runs"
print "<TD colspan=\"2\">";
print $q->textfield(-name=>'maxN',
                    -default=>'5000',
                    -override=>'0',
                    -size=>5,
                    -maxlength=>5);
print "</TD></TR>";

# -------------
# ROW 5
# -------------

print <<here;
<TR>
    <TD align="center" colspan=3>
here
print $q->defaults(' Default Values ');
print <<here;
    <INPUT type=submit value=" Start Reaction "></TD>
</TR>
here

# -------------
# ROW 6
# -------------

print <<here;
<TR>
    <TD colspan="3">
    <TABLE>
        <TR><TD><B>Data Used</B>        </TD></TR>
        <TR><TD>Initial concentrations: </TD>
            <TD>A: $Ao &#037;</TD>
            <TD>B: $Bo &#037;</TD>
        </TR>
        <TR><TD>Initial number of molecules was:</TD>
            <TD>A: $NAo</TD>
            <TD>B: $NBo</TD>
        </TR>
        <TR><TD>Total number of molecules was:</TD><TD>$TotalMols</TD></TR>
        <TR><TD>Number of runs was:     </TD><TD>$maxN</TD></TR>
    </TABLE>
    </TD>
</TR>
here


# -------------
# ROW 7
# -------------

print <<here;
<TR><TD colspan=3><B>Observations</B><BR>
        The number of molecules of <B>A</B> and <B>B</B> should be decreasing.<BR>
```

```
        The number of molecules of <B>C</B> should be increasing.<BR>
</TR>

</TABLE>
<P>
<HR>
here

print_error_details();

# Current section navigation bar
#print <<here;
#<TABLE width="100%">
#<TR>
#   <TD align="left"><A HREF="/kinetics/cgi-bin/mc2">Back</A>
#   <TD align="right"><A HREF="/kinetics/cgi-bin/analysis">Analysis</A>
#</TR>
#</TABLE>
#<HR>
#here

print $q->endform;
print $q->end_html; # end the HTML
}
```

4.9 Error Handling Subroutines

39 ⟨*error handling subroutines* 39⟩≡ (20–22 42)

```
sub print_error_details
{
    my $i;

    if ( keys(%form_input_errors) )
    {
        foreach $i ( keys(%form_input_errors) )
        {
            print "<FONT color=red>";
            print "Error: $error_messages{$i}</FONT><BR>";
        }
    }
}
```

4.10 Debugging Subroutines

These subroutines are only used for debugging the cgi scripts and this code chunk can be removed from the main chunk for release versions.

4.10.1 printDebuggingLine

The `printDebuggingLine()` takes pairs of arguments; a string describing the variable and the variable itself. Any number of pairs can be appended to the argument list. To get the name/values pairs to appear on different lines in the HTML output `<BR>` tags can be included in the string.

eg. `printDebuggingLine("Ao=", $Ao, " Bo=", $Bo, "<BR>Order=", $order);`

Will produce;

```
Debugging line:
Ao=40
Bo=
userID=138.25.82.4; path=/cgi-bin/
```

Do not expect that the order of what is printed in the HTML will be the same as the order of the arguments. The arguments are read in and stored as a hash so the order might be anything.

Other examples of what you can do are:

```
printDebuggingLine("IP:", $remote_ip, "I am: ", $userID);
printDebuggingLine("Ao=", $Ao, "NBo=", $q->cookie('NBo'));
```

TODO - how do we do this now? `print " Errors: ", keys(%form_input_errors);`

40 ⟨*debugging subroutines* 40⟩≡ (20–22 42 51) 41a ▷

```
sub printDebuggingLine
{
    # Usage ("Descriptive text", "variable", etc...)
    # eg ("Ao=", $Ao, " Bo=", $Bo, "<BR>Order=", $order)
    my %hash = @_;

    # The newlines before and after the while loop just make the
    # debugging section stand out clearer in the HTML source
    print "\n\n";

    # The <BR> makes the Debugging line start on a new line in the HTML.
    print "<BR>Debugging line: ";
    while ( my($name, $value)=each(%hash) )
    {
        print $name, $value, " ";
    }

    print "\n\n";
}
```

4.10.2 printDebuggingPage

41a ⟨*debugging subroutines* 40⟩+≡ (20–22 42 51) ◁40 41b▷

```
sub printDebuggingPage
{
    my $error_string = $_[0];

    print header(), "\n";
    print start_html("Error Information"), "\n";
    print h1("HTML Error"), "\n";
    print $error_string;
    print end_html();

    exit(0);
}
```

4.10.3 printSquareArray

41b ⟨*debugging subroutines* 40⟩+≡ (20–22 42 51) ◁41a

```
sub printSquareArray
{
    # Subroutine used for debugging only
    # Print the array.
    # Syntax: $array_size = printSquareArray(@array);

    my @array = @_;
    my ($i, $j, $array_size);

    my $side = sqrt($#array+1);
    print STDERR "Side=", $side, "\n";

    for ($i=0; $i<$side; $i++)
    {
        for ($j=0; $j<$side; $j++)
        {
            print STDERR $array[$i*$side+$j], " ";
        }
        print STDERR "\n";
    }

    return $#array;
}
```

5 Data Analysis Scripts

The main analysis program deals with both 1st and 2nd order data.

42 ⟨*analysis* 42⟩≡

```
⟨common preamble 23a⟩ # use modules, global file locations and time.

my $interval = 10;
my $i;
my ($refA, $refB, $refC, $refX, $refY);
my ($button, $from, $to, $maxTo);
my ($intercept, $slope, $R_squared, $sample_size);

# Create a hash to store out error message text in.
my %error_messages;
$error_messages{"From"} = "The starting value for the regression must be \
 an integer and equal to or greater than zero.";
$error_messages{"To"} = "The ending value for the regression must be an \
 integer and not more than the number of data points.";

# Must reinit this hash to null before each invocation.
my %form_input_errors = ();

# Retrieve cookies telling us needed information on the previous data.
my $userID  = cookie('userID');
$button     = cookie('button');
my $order   = cookie('order');
my $NAo     = cookie('NAo');
my $NBo     = cookie('NBo');

# Read data before getting parameters so we can check that
# $from is less than data set size.
$file = $file.$userID;
($refA, $refB, $refC) = readData("$tmp_dir/$file");

# Read in CGI parameters from form.
($button, $from, $to) = getParams();

# Pass these references to the process data subroutine. The equation
# type is passed so it know whether to do a 0th, 1st or 2nd order
# analysis.
$refY = processData($button, $order, $NAo, $NBo, $refA, $refB, $refC);

# Write out a new file of the processed data.
if ($button == 1 && ($order != 2))
{
    # Button 1 means there has been no processing of data AND
    # if the order is zeroth or 1st we just write out original A and B data.
    writeData("$tmp_dir/$file.processed", $refA, $refB);
}
elsif( ($button == 1) && ($order == 2) )
{
    # Button 1 means there has been no processing of data AND
    # if the order is 2nd we write out original A, B and C data.
    # ie same as above but we write out C data as well.
    writeData("$tmp_dir/$file.processed", $refA, $refB, $refC);
}
else
{
    # User selected button 2 or 3 so write out the processed data.
    writeData("$tmp_dir/$file.processed", $refY);
}

# Create a smaller set of data for the regression to speed things up.
($refX, $refY) = mySlice($from, $to, $interval, $refY);
```

```
# Perform linear regression.
($intercept, $slope, $R_squared, $sample_size) = regression($refX, $refY);

# Declarations for the graph and plot.
my ($program, $command, $label, $axis, $type, $size, $infile, $outfile);

# Now create a GNU metafile of the processed data using graph.
$program = "/usr/bin/graph ";
$label   = "-L 'Monte Carlo Simulation' ";
# TODO - diff Y axis for each button.
$infile  = "-a $tmp_dir/$file.processed ";
$outfile = "$tmp_dir/$file.processed.meta ";
$command = $program.$label.$axis.$infile.">|".$outfile;
system("$command");

# Now create a gif image of the data from this metafile using plot.
$program = "/usr/bin/plot ";
$type    = "-T gif ";
$size    = "--bitmap-size 400x400 ";
$infile  = "$tmp_dir/$file.processed.meta ";
$outfile = "$file_base/tmp/$file.processed.gif ";
$command = $program.$type.$size.$infile.">|".$outfile;
system("$command");

return_analysis($button, $order, $NAo, $NBo);


#########################################################
# Subroutines here.
#########################################################
⟨analysis general subroutines 44⟩
⟨analysis form subroutines 47⟩
⟨file read subroutines 30⟩
⟨file write subroutines 31⟩
⟨error handling subroutines 39⟩
⟨debugging subroutines 40⟩
```

 ⟨*analysis general subroutines* 44⟩≡ (42)

```
sub getParams
{
    # button is the equation we want to plot ie selected from
    # the radio buttons.
    my ($button, $from, $to);

    # Get any parameters the user has entered from the form.
    # TODO - error checking that $to < size of A
    if ( param() )
    {
        $button = param("button");
        $from = param("From");
        $to = param("To");

        if ( ($from =~ /[^0-9]/) || ($from eq "") ||
        ($from < 0) || ($from > $#{$refA}) )
        {
            $form_input_errors{"From"} = 1;
            $from = 0;
        }

        if ( ($to =~ /[^0-9]/) || ($to eq "") || ($to > $#{$refA}) )
        {
            $form_input_errors{"To"} = 1;
            $to = $#{$refA};
        }
    }
    else
    {
        $button = 1;
        $from = 0;
        #$to = $#{$refA}; # 05-07-20 commented out and added code below
        $i=0;
        while ($refA->[$i] != 0)
        {
            $i++;
        }
        $to = $i - 1;
    }

    return ($button, $from, $to);
}


sub processData
{
    my ($button, $order, $NAo, $NBo, $refA, $refB, $refC) = @_;
    my ($i, $tmp1, $tmp2, @tmp);

    # Case for each equation we want to do.

    if ($button == 1)       # Radio button 1
    {
        # Do nothing
        $axis = "-X 'Number of Runs' -Y 'Number of A' ";
        for ($i=0; $i<=$#{$refA}; $i++)
        {
            # $tmp[$i] = $refA->[$i];
            push(@tmp, $refA->[$i]);
        }
    }
    elsif ($button == 2)    #  Radio button 2
    {
```

```
        $axis = "-X 'Number of Runs' -Y 'ln (Number of A)' ";
        for ($i=0; $i<=$#{$refA}; $i++)
        {
            if ($refA->[$i] >= 1)
            {
                # $tmp[$i] = log($refA->[$i]); 05-07-20
                # otherwise if $refA[20] is zero then $tmp[20] is undefined.
                push(@tmp, log($refA->[$i]) );
            }
        }
    }
    elsif ($button == 3)    # Radio button 3
    {
        if ( ($order==1) || ($order==0) )
        {
            $axis = "-X 'Number of Runs' -Y '1/A' ";
            for ($i=0; $i<=$#{$refA}; $i++)
            {
                if ($refA->[$i] != 0)
                {
                    # $tmp[$i] = 1 / $refA->[$i]; 05-07-20
                    push(@tmp, 1/($refA->[$i]) );
                }
            }
        }
        if ( ($NAo != $NBo) && ($order==2) )    # Case 1: [A] not = [B]
        {
            $axis = "-X 'Number of Runs' -Y 'ln{Bo(Ao-x)/Ao(Bo-x)}' ";
            for ($i=0; $i<=$#{$refA}; $i++)
            {
                # NBo (NAo - x)  where x = number reacted at time N
                # --------------
                # NAo (NBo - x)
                $tmp1 = $NBo*($refA->[$i]);
                $tmp2 = $NAo*($refB->[$i]);
                if ( ($tmp1 != 0) && ($tmp2 != 0) )
                {
                    # $tmp[$i] = log($tmp1/$tmp2); 05-07-20
                    push(@tmp, log($tmp1/$tmp2) );
                }
            }
        }
        if ( ($NAo == $NBo) && ($order==2) )    # Case 2: [A] = [B]
        {
            $axis = "-X 'Number of Runs' -Y '1/A' ";
            for ($i=0; $i<=$#{$refA}; $i++)
            {
                if ($refA->[$i] != 0)
                {
                    $tmp[$i] = 1 / $refA->[$i];
                }
            }
        }
    }
    else
    {
        # TODO -  handle error.
    }

    return \@tmp;
}

sub mySlice
{
```

```
    my ($from, $to, $interval, $refA) = @_;
    my ($i, $element, @X, @Y);

    $i = 0;
    for ($element=$from; $element<=$to; $element=$element+$interval)
    {
        push(@X, $element);
        push(@Y, $refA->[$element]);
        $i = $i + 1;
    }

    return (\@X, \@Y);
}

sub regression
{
    my ($refX, $refY) = @_;
    my ($sample_size, $intercept, $slope, $R_squared);
    my $ls = Statistics::OLS->new();

    $ls->setData ($refX, $refY);
    $ls->regress(); # TODO - returns 1 for success, check this.
    ($intercept, $slope) = $ls->coefficients();
    $R_squared = $ls->rsq();
    $sample_size = $ls->size();

    return ($intercept, $slope, $R_squared, $sample_size);
}
```

47 ⟨*analysis form subroutines* 47⟩≡ (42)

```
sub return_analysis
{
    # $order was passed in as an arg which itself came from reading
    # a cookie left over from the mc run.
    my ($button, $order, $NAO, $NBo) = @_;

    my ($title, $scriptName);
    if ($order == 0)
    {
        $title = "Zeroth Order Kinetics";
        $scriptName = "mc0";
    }
    elsif ($order == 1)
    {
        $title = "First Order Kinetics";
        $scriptName = "mc1";
    }
    elsif ($order == 2)
    {
        $title = "Second Order Kinetics";
        $scriptName = "mc2";
    }
    else
    {
        # TODO - Handle error.
    }

    my $q = new CGI;                    # create new CGI object

    # create cookies
    my $buttonC = $q->cookie(-name=>'button', -value=>$button);

    # You must incorporate any cookies into the HTTP header.
    print $q->header(-cookie=>$buttonC);
    # print $q->header();

    # start the HTML
    print $q->start_html({
            -title=>"Kinetics",
            -style=>{'src'=>"$html_base/styles.css"} });
    print "\n";

    print $q->h1("Analysis of the Data<BR>$title");

    print $q->startform("GET", "analysis");

    #printDebuggingLine(" Size B=", $#{$refB}, " Size C=", $#{$refC},);
    #printDebuggingLine("NAo= ",$NAo,"NBo= ",$NBo);
    #printDebuggingLine("button=",$button);


    # Main navigation bar
# Must NOT indent this next line!
⟨html top navbar2 60d⟩

    # Main table for form.
    print "<TABLE>";

    # ---------------------
    # There is only ONE ROW
    # ---------------------

# Column 1 for gif
```

```
print <<here;
<TR>
    <TD valign="top" width=300>
    <IMG border="1" width="400" height="400"
    src="$html_base/tmp/$file.processed.gif?$seconds">
    </TD>
here

# Column 2 for radio buttons
print "<TD>Select<BR>Plot<P>\n";

# Radio button 1
print "<INPUT type=radio override=0 name=button value=1";
if ($button==1){ print " checked";} print ">\n";
print "<IMG align=top width=75";
print " src=\"$html_base/images/analysis_rbutton1.gif\"><BR>\n";

# Radio button 2
print "<INPUT type=radio override=0 name=button value=2";
if ($button==2){ print " checked";} print ">\n";
print "<IMG align=top width=75";
print " src=\"$html_base/images/analysis_rbutton2.gif\"><BR>\n";

# Radio button 3
print "<INPUT type=radio override=0 name=button value=3";
if ($button==3){ print " checked";} print ">\n";

if ( ($order==0) || ($order==1) )
{
    print "<IMG align=top width=75 src=\"$html_base/images/analysis_rbutton3.gif\"><BR>\n";
}
if ( ($order==2) && ($NAo == $NBo) )    # Case 1: [A] not = [B]
{
    print "<IMG align=top width=75 src=\"$html_base/images/analysis_rbutton3.gif\"><BR>\n";
}
if ( ($order==2) && ($NAo != $NBo) )    # Case 2: [A] = [B]
{
    print "<IMG align=top width=100 src=\"$html_base/images/analysis_rbutton4.gif\"><BR>\n";
}
else
{
    # TODO handle error.
}

print "<INPUT type=submit value=\" Plot \"></TD>\n";

    # Column 3 for entry of data and results.
    print "<TD valign=top width=400>";

    # Nest a table within this data column.
    print "<TABLE>";
        print "<TR><TD colspan=2><B>Input</B><P>";
        print "Calculate Regression<BR>";
        print "(values can range from 0 to ",$#{$refA}, ")";
        print "</TD></TR>";

        print "<TR><TD>";
        if ($form_input_errors{"From"})
        {
            print "<FONT color=red>From:</FONT>";
        }
        else
        {
            print "From:";
```

```
        }
        print $q->textfield(-name=>'From',
                    -default=>'0',
                    -override=>'0',
                    -size=>5,
                    -maxlength=>5);
        print "</TD>";
        print "<TD>To:";
        print $q->textfield(-name=>'To',
                    -default=>"$to",
                    -override=>'0',
                    -size=>6,
                    -maxlength=>6);
        print "</TD>";
        print "</TR>";

        print "<TR><TD colspan=2> <BR><B>Results</B><P></TD></TR>";

        print "<TR>";
        print "<TD>Regression from:</TD><TD>$from</TD>";
        print "</TR>";

        print "<TR>";
        print "<TD>Regression to:</TD><TD>$to</TD>";
        print "</TR>";

        print "<TR>";
        print "<TD>Slope:</TD>";
        printf "<TD>%.4f",  1000*$slope;
        print " x 10<SUP>-3</SUP></TD>";
        print "</TR>";

        print "<TR>";
        print "<TD>Intercept:</TD>";
        printf "<TD>%.5f</TD>",  $intercept;
        print "</TR>";

        print "<TR>";
        print "<TD>Correlation coefficient:</TD>";
        printf "<TD>%1.6f</TD>", $R_squared;
        print "</TR>";

        print "<TR>";
        print "<TD>Size of data set used:</TD>";
        print "<TD>$sample_size</TD>";
        print "</TR>";

    print "</TABLE>";

        print "<P>To see a table of the current data click";
        print " <A HREF=\"/kinetics/cgi-bin/viewdata\">here</A>.";

print <<here;
</TD>
</TR>
</TABLE>
<P>
here

print_error_details();

# Current section navigation bar
print <<here;
<TABLE width="100%">
```

```
<TR>
    <TD align="left"><A HREF="/kinetics/cgi-bin/mc$order">Back</A>
    <!-- TODO <TD align="right"><A HREF="$html_base/report.html">Report</A> -->
</TR>
</TABLE>
<HR>
here

    print $q->endform;
    print $q->end_html; # end the HTML
}
```

6 Other CGI Scripts

6.1 viewdata

This script allows us to view a table of the data created from the last Monte Carlo simulation run.

¡INPUT type=”submit” name=”action” value=”Previous”¿

51 ⟨*viewdata* 51⟩≡

```
⟨common preamble 23a⟩ # use modules, global file locations and time.

# Retrieve cookies telling us needed information on the previous data.
my $userID  = cookie('userID');
$file = $file.$userID;

my $i;
my $dataset;
my $action;
my $series = 2;

my ($start, $end);
my $max_lines = 100;
my $last_data_line;
my ($refA, $refB, $refC);
my (@A, @B, @C);

($refA, $refB, $refC) = readData("$tmp_dir/$file");
@A = @$refA;
@B = @$refB;

$last_data_line = $#A;
my $last_dataset = int( $last_data_line / $max_lines );

if ($#{$refC} != -1)
{
    @C = @$refC;
    $series = 3;
}

if (param())
{
    $dataset = param("dataset");
    $action = param("action");

    if ($action eq "First")
    {
        $dataset = 1;
    }
    elsif ( ($action eq "Previous") && ($dataset > 1) )
    {
        $dataset = $dataset - 1;
    }
    elsif ( ($action eq "Next") && ($dataset < $last_dataset) )
    {
        $dataset = $dataset + 1;
    }
    elsif ($action eq "Last")
    {
        $dataset = int( $last_data_line / $max_lines );
    }
    else
    {
        # Do nothing.
    }
}
```

```
else
# There were no parameters because this is the first time the form has
# been invoked.
{
    $dataset = 1;
}

$start = ($dataset-1) * $max_lines;
$end   = $dataset * $max_lines;

# We now have all the infomation to create the form and send it to the
# users browser.
return_view();
```

⟨*viewdata subroutines* 53⟩
⟨*file read subroutines* 30⟩
⟨*debugging subroutines* 40⟩

6.2 viewdata Subroutines

53 ⟨*viewdata subroutines* 53⟩≡ (51) 55 ▷

```
sub return_view
{
    my $q = new CGI;                    # create new CGI object
    print $q->header();                 # create the HTTP header

    # start the HTML
    print $q->start_html({
            -title=>"Kinetics",
            -style=>{'src'=>"$html_base/styles.css"} });
    print "\n";
    print $q->startform("GET", "viewdata");

print <<here;
<INPUT type='hidden', name="dataset", value="$dataset">
here

print <<here;
<A NAME="start_of_table">
here

    print $q->h1('Current Monte Carlo Data');

    # Main navigation bar
⟨html top navbar2 60d⟩
    #printDebuggingLine("Series=", $series, " size B=",$#{$refB}, " size C=",$#{$refC});


    ##################################################################
    # Begin a wrapper table to contain the data table and some help.
    print "<TABLE width=\"85%\">\n<TR>";

    # This is the left hand data cell.
    print "<TD>\n";
    return_data_table($series, $start, $end);
    print "\n</TD>";

    # This is the right hand data cell.
print <<here;
<TD valign="top">
The table on the left shows the data for the last Monte Carlo experiment run.
Only $max_lines lines of data are retrieved at a time. You can scroll down the
table to see these $max_lines values or go directly to the end of the table via
the hyperlink.
<P>
Click here to return to the
<A HREF="/kinetics/cgi-bin/analysis">Analysis</A> section.
<P>

<TABLE>
<TR>
    <TD>To show the first $max_lines lines of data click here:</TD>
    <TD><INPUT type="submit" name="action" value="First"></TD>
    <TD></TD>
</TR>
<TR>
    <TD>To show the previous/next $max_lines lines of data click here:</TD>
    <TD><INPUT type="submit" name="action" value="Previous"></TD>
    <TD><INPUT type="submit" name="action" value="Next"></TD>
</TR>
<TR>
    <TD>To show the last $max_lines lines of data click here:</TD>
```

```
    <TD><INPUT type="submit" name="action" value="Last"></TD>
    <TD></TD>
</TR>
</TABLE>

<P>
<A HREF="#end_of_table">&lt; Jump to the end of this table.</A>
here

    print "</TR></TABLE>";
    # This is the end of the wrapper table
    ##################################################################


    # Make the name and hyperlinks at the bottom.
print <<here;
<P>
<A NAME="end_of_table">
<A HREF="#start_of_table">Return to start of Table</A><BR>
<P>
<HR>
here

    print $q->endform();
    print $q->end_html; # end the HTML
}
```

This function returns a formated HTML table of the data. Three integer arguments must be given in the following order:

1. the number of data series - this will be 2 for a 1st order (ie A and B) and 3 for a 2nd order reaction (ie A, B and C)
2. a starting number which will determine where in the data array the table will start at
3. an ending number which will be the last value of the data array.

The number of data series was determined earlier by the size of the C array.

55 ⟨*viewdata subroutines* 53⟩+≡ (51) ◁53

```
sub return_data_table
{
    my ($series, $start, $end) = @_;

    # If there was only A and B then series = 2 else if there is
    # A, B and C then series = 3.

    # Print table headings
print <<here;
    <TABLE border=1>
    <TR>
    <TD><B>Run<BR>number</B></TD>
    <TD><B>Number of<BR>molecules of A</B></TD>
    <TD><B>Number of<BR>molecules of B</B></TD>
here

    if ($series == 3)
    {
        print "\t<TD><B>Number of<BR>molecules of C</B></TD>";
    }
    print "\t</TR>";

    # Print each data line of the table.
    for ($i=$start; $i<=$end; $i++)
    {
        print "\n\t<TR><TD>$i</TD><TD>$A[$i]</TD><TD>$B[$i]</TD>";
        if ($series ==3)
        {
            print"\t<TD>$C[$i]</TD>";
        }
        print "</TR>";
    }

    print "\n\t</TABLE>";
}
```

6.3 Temporary Page for when Scripts are Off-line

Copy this script to `/home/http/cgi-bin` and rename it to whatever script is currently unavailable.

56 ⟨*mc-offline* 56⟩≡

```
#!/usr/bin/perl
use CGI qw/:standard/;          # load CGI routines

$q = new CGI;                   # create new CGI object
print $q->header();             # create the HTTP header

print $q->start_html('Kinetics');   # start the HTML
print $q->h1('Kinetics');           # level 1 header

print <<here;
<P>
This page is temporarily unavailable while modifications
are being made.
<P>
Click your browser's &quot;Back&quot; button to return to the
Kinetics page.
<P>

Mike
here

print $q->end_html;                     # end the HTML
```

6.4 Script to Generate Table of Random numbers

57 ⟨*print-random* 57⟩≡

```
#!/usr/bin/perl
# Generate a table of random numbers and typeset it for LaTeX.
my $N = 600;
my $max_value = 100;
my $rand;

print <<here;
\\documentclass[a4paper,10pt]{article}
\\usepackage{supertabular}
\\usepackage{anysize}

% To get a linespace between paragraphs define and set my own paragraph
% separation and indent.
\\newlength{\\myparskip}
\\setlength{\\myparskip}{0.4\\baselineskip}
\\setlength{\\parskip}{\\myparskip}
\\setlength{\\parindent}{0pt}

\\begin{document}
\\\marginsize{3cm}{2cm}{1cm}{2cm}
\\pagestyle{empty}

{\\Large \\bfseries Random Numbers for Kinetics Experiment

}
\\vspace{\\baselineskip}

Generated on \\today

The following is a list of random numbers to use in picking places on the grid.
Select them in order down the list. Do not select them randomly as that can result
in a bias towards numbers at the centre of the table.
As an example, if the random number was 91 then we pick grid coordinate (x,y)
equal to (9,1).
\\vspace{\\baselineskip}

\\tablefirsthead{%
\\textbf{Try Number}&\\textbf{Random Number}%
&\\textbf{Run Number}&\\textbf{Remaining A or B}\\\\
}
\\tablehead{%
\\multicolumn{2}{l}{\\small \\sl continued from previous page}\\\\
\\textbf{Try Number}&\\textbf{Random Number}%
&\\textbf{Run Number}&\\textbf{Remaining A or B}\\\\
}
\\tabletail{%
\\multicolumn{2}{l}{\\small \\sl continued on next page}\\\\
}
\\tablelasttail{\\hline}
\\begin{supertabular}{llll}
here

for ($i=1; $i<$N; $i++)
{
    $rand = sprintf "%u", rand $max_value;
    # Print cols 1 and 2
    # MRL 10-08-20: If the rand number is a single digit the students
    # didn’t know whether to plot it as 05 or 50 so commented out next
    # line and replaced with a formatted printf statement.
    # print "$i\t&$rand";
    printf "%3d\t&%2d", $i, $rand;
    if ( (($i/6) =~ /\./) ) # Not evenly divisible by 6
```

```
    {
        # Print cols 3 and 4 (both empty)
        print "&&\\\\\n";
    }
    else
    {
        print "&", $i/6, "&\\\\\n"; # Print cols 3 (run number) and 4
        print "\\hline\n";          # horizontal line
        print "&&&\\\\\n";          # blank line
    }
}
$rand = sprintf "%u", rand $max_value;
print "$i\t&$rand&", $i/6, "&\\\\";

print <<here;
\\end{supertabular}
\\vspace{\\fill}

{\\small These random numbers were generated from a Perl script which also
formatted the numbers into a \\LaTeX document for printing.}

\\end{document}
here
```

7 HTML Interface

This section covers the static HTML pages ie. the pages which are not generated on-the-fly by the CGI scripts. Like the code these pages are generated from this noweb source document by noweb.

Note: After updating the HTML files, and copying them to the server directory, you may have to hold the SHIFT key down when you click Netscapes Reload button to force Netscape to flush its cache.

7.1 List of Static HTML Files

`index.html`	
`manual.html`	A Monte Carlo Simulation done manually
`computer.html`	A Monte Carlo Simulation by Computer
`about.html`	
`help.html`	

7.2 The Main Contents Page

The root chunk `index.html` is just a table of contents as follows:

59a ⟨*index.html* 59a⟩≡

```
⟨html head 59b⟩
<BODY>
<H1>Computer Based Experiments in Kinetics<BR>
using Monte Carlo Techniques</H1>
<P>
Welcome to the Web-based Monte-Carlo experiment. You will learn how a
Monte Carlo experiment works and use it to simulate chemical reactions
of different kinetic orders.<P>
Some Notes before you start:<BR>
* Use <B>Netscape</B> rather than Microsoft Explorer as Netscape correctly
displays this experiment at the present time. <BR>
* The use of ‘‘cookies’’ <B>must</B> also be enabled as this program stores your
simulation data as cookies. The lecturer/laboratory officer may have already
set this but you wish to check.

<H2>Contents</H2>

<A HREF="intro/intro.html">General Introduction</A><P>
<A HREF="theory/theory.html">Review of Kinetics Theory</A><P>
<A HREF="manual/manual.html">Monte Carlo Experiments done Manually</A><P>
<A HREF="computer/computer.html">Monte Carlo Experiments by Computer</A><P>
<A HREF="about.html">About this Software</A><P>
<A HREF="help.html" >Help</A><P>
<HR>
⟨html tail 60a⟩
```

7.2.1 Head and Tail Chunks

All the HTML files contain the following head and tail chunks. These contain the HTML declarations, head and body tags.

59b ⟨*html head* 59b⟩≡ (59a 66 71 74 75)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
    <TITLE>Kinetics</TITLE>
    <LINK rel="stylesheet" type="text/css" href="styles.css">
</HEAD>
<BODY>
```

The postamble code chunk contains the normal HTML termination.

60a ⟨*html tail* 60a⟩≡ (59a 66 71 74 75)

```
</BODY>
</HTML>
```

7.2.2 Cascading Style Sheet

60b ⟨*styles.css* 60b⟩≡

```
/* Cascading Styles Sheet for Kinetics Experiment */
H1 {font-style:sans-serif; color:#800000; font-size:large; text-align:center }
H2 {color:#000000 }
BODY {background:#FFF9EF}
```

7.2.3 Top Navigation Bar

There are two top navigation bars defined here:

The first one is used by all the static pages.

The “Main Contents” link will always point to the index.html one level up whereas the help file link will point to the local help file for that section. That way we can have a different help file for each section. The Help files are in Section 7.8.

60c ⟨*html top navbar* 60c⟩≡ (61 64 66 71)

```
<TABLE cellpadding="0" cellspacing="0" width="100%">
<TR><TD align="left" ><A HREF="../index.html" >Main Contents</A>
    <TD align="right"><A HREF="help.html"  >Help</A>
</TR>
</TABLE>
<P>
```

The next navigation bar is used by the dynamically generated pages; “Zero”, “First” and “Second Order Kinetics”. It differs in that it has a few more links and uses the `$scriptName` variable to determine the script name that created the page.

60d ⟨*html top navbar2* 60d⟩≡ (32 36 47 53)

```
print <<here;
<TABLE width="100%">
<TR>
    <TD width="33%" align="left" valign="top">
        <A HREF="$html_base/index.html">Main Contents</A><P>
        <A HREF="$html_base/computer/computer.html">
        Computer Experiment Contents</A></TD>
    <TD width="33%" align="center" valign="top">
        <A HREF="/kinetics/cgi-bin/$scriptName">Back to Data</A><P>
        <A HREF="/kinetics/cgi-bin/analysis">Analyse your Data</A></TD>
    <TD width="33%" align="right" valign="top">
        <A HREF="$html_base/help.html">Help</A></TD>
</TR>
</TABLE>
<P>
here
```

7.3 Introduction HTML Pages

These are LaTeX files which are converted to HTML files using *latex2html*.

The file intro.tex will become intro.html

61 ⟨*intro.tex* 61⟩≡

```
\documentclass{article}
\usepackage{html}

\begin{document}

\section{General Introduction}

\begin{rawhtml}
⟨html top navbar 60c⟩
\end{rawhtml}

\subsection{What is Monte Carlo?}

The Monte Carlo method is an important simulation technique in science. It is
often used where the underlying fundamental processes are random in nature and
indeed it derives its name from the district of Monte Carlo in Monaco - a place
well known for its games of chance.

Monte Carlo simulations are particularly useful when it is inconvenient or
impossible to carry out a physical experiment and instead a mathematical
experiment that closely simulates the physical one is studied. For example, it
would be prohibitively expensive in a study of nuclear reactor design to
actually build a reactor and study the neutron processes physically. Since the
fundamental particles of nuclear physics obey probabilistic laws, their actual
behaviour can be simulated by carrying out mathematical experiments using
random numbers, whose behaviour repesents the behaviour of the neutrons.
Simulation experiments using random numbers to represent physical quantities
have been studied for decades in such diverse areas as population genetics,
chemical engineering operation research and chemical kinetics \cite{Hammer},
\cite{Rab}.

Rather than try and explain theoretically, or through mathematics, how Monte
Carlo works you will gain an understanding of the method by performing a Monte
Carlo simulation manually and later using the computer.


\subsection{Application to Kinetics}

In the experiments to be performed in the next section the Monte Carlo method
will be used to provide data which models the course of a chemical reaction and
the rate of appearance and disappearance of products and reactants. Three
reactions are to be studied:

\begin{enumerate}
\item A first order reaction of type:\\
      $\mathrm{A} \longrightarrow \mathrm{Products}$

\item A second order reaction of type:\\
      $\mathrm{A + B} \longrightarrow \mathrm{Products}$

\item A first-order consecutive reaction commencing initially with pure
      A only (
      \begin{rawhtml}
      k<sub>1</sub> and k<sub>2</sub>
      \end{rawhtml}
      are the rate constants for the two steps.\\
      $\mathrm{A} \stackrel{k1}{\longrightarrow} \mathrm{B}
      \stackrel{k2}{\longrightarrow} \mathrm{C}$
```

```
\end{enumerate}

This division of reactions into first and second order is because from a
theoretical point of view they proceed via different kinetic mechanisms.
To describe the kinetics in a precise mathematical way we represent the
reactions with their respective differential rate laws.

For first order reaction (1) the differential rate law is given by:
\[
-\frac{d[A]}{dt} = k[A]
\] where [A] is the concentration of A at time t and
k is the specific rate constant.

The rate of reaction
is proportional to the concentration at any given time.
Note that rate is always a positive value. Since concentration of A decreases
with time the rate is given by:
\[
-\frac{d[A]}{dt}
\]

For a second order reaction (2), the rate is:
\[
-\frac{d[A]}{dt}  = k[A][B]
\]
where k is now the second order specific rate constant.

However, experimentally it is the concentration of a reactant or product
that is measured - not the time rate of change as given by the
differential rate laws. For instance, a spectrophotometer may measure
the absorbance at particular times thus providing us with the the
concentration of a reactant versus time.

To theoretically predict how the concentration varies with
time we must integrate the differential rates laws to obtain the
integrated rate equations. For each of these reactions an analytical
expression for the integrated rate equation can be derived.

For example, by integrating the differential rate equation for the
first order reaction we can obtain the equation which
describes the variation of concentration with time:

\[
ln [A]  =  -kt + ln [A]_0
\]

where $[A]_0$ is the concentration of A at time=0.

Integration of the second order rate equation is more complex and there
are different equations depending on whether the initial concentration of
A is the same as B or not.

If $[A]_0 \ne [B]_0$ then:
\[
\frac{1}{B_o - A_o} \ln \frac{A_o(B_o-x)}{B_o(A_o-x)}  =  kt
\]

For the case where $[A]_0 = [B]_0$ the reaction behaves as if it were
second order in a single reactant and we obtain:
\[
\frac{1}{A} - \frac{1}{A_0} = kt
\]

again $[A]_0$ is the concentration of A or B at time=0.
```

```
For the consecutive reaction the closed form integrated rate equation
can only be derived with extreme difficulty~\cite{Frost}. Herein lies
the elegance of the Monte Carlo method. The computer can simulate the
reaction taking place and save the concentration data as a function of
time.  The data can then be plotted in the same way as you would for a
real experiment.  For more complex reactions than the above, no closed
form rate equations can be obtained, yet the Monte Carlo method still
enables one to obtain the variation of concentration with time.

\begin{thebibliography}{99}
    \bibitem{Hammer} J.M. Hammersley and D.C. Handscomb,
        \emph{Monte Carlo Methods}, Methuen, London (1964).
    \bibitem{Rab} B. Rabinovitch, \emph{Journal of Chemical Education},
        46, 262 (1969).
    \bibitem{Frost} A.A. Frost and R.G. Pearson,
        \emph{Kinetics and Mechansim}, Wiley, New York (1961).
\end{thebibliography}

\begin{rawhtml}
<A HREF="#top_of_page">Return to Top of Page</A>
\end{rawhtml}

\end{document}
```

7.4 Theory HTML Pages

64 ⟨*theory.tex* 64⟩≡

```
\documentclass{article}
\usepackage{html}

\begin{document}


\section{Review of Kinetic Theory}

\begin{rawhtml}
⟨html top navbar 60c⟩
\end{rawhtml}

\subsection{First Order Reactions}

We start with the differential rate equation for a first order reaction. This
equation is a mathematical statement that the rate of change of the
concentration of A is proportional to the concentration of A at that
time.
\begin{equation}
-\frac{dA}{dt} = kA
\end{equation}

\noindent this can be rearranged to:
\begin{equation}
-\frac{1}{A} dA = k dt
\end{equation}

\noindent which can now be integrated over time.
For the integration limits we will use the information
that initially $A = A_0$ at $t=0$.
\begin{eqnarray}
-\int_{A_0}^A\frac{1}{A}\mathrm{d}A  &=& k\int_{0}^t\mathrm{d}t\\
-(\ln{A} - \ln{A_0})                 &=& kt\\
\ln{A}                               &=& -kt + \ln{A_0}
\end{eqnarray}

This is of the form
y = \texttt{slope} x + \texttt{intercept}
which is the equation for a straight line.

Therefore, for a first order reaction, if we plot the natural
logarithm of [A] versus time we should obtain a straight line of
slope = ``-k'' and intercept = ln[Ao].
With suitable data we can therefore calculate the specific rate constant k.


\subsection{Second Order Reactions}

\noindent For second order kinetics the differential rate law is given by:

\begin{equation}\label{diff2}
-\frac{dA}{dt}  = kAB
\end{equation}

The integrated rate law for second order reactions where $[A]_0 \ne
[B]_0$ is:

\begin{equation}
\frac{1}{B_0 - A_0} \ln \frac{A_0(B_0-x)}{B_0(A_0-x)}  =  kt
\end{equation}

where x is the quantity of A or B that has reacted at time t.
```

```latex
\noindent In the special case that $[A]_0 = [B]_0$, the integration of
equation~\ref{diff2} is quite easy:

\begin{equation}
-\frac{dA}{dt}  = kA^2
\end{equation}

\noindent each side can be integrated like so:
\begin{eqnarray}
-\int_{A=A_0}^A\frac{1}{A^2}\mathrm{d}A  &=& k\int_{t=0}^t\mathrm{d}t\\
-\left| -\frac{1}{A} \right|_{A_0}^{A}  &=& kt\\
\frac{1}{A} - \frac{1}{A_0}             &=& kt\\
\frac{1}{A}                             &=& kt - \frac{1}{A_0}
\end{eqnarray}
Thus for a second order reaction where $[A]_0 = [B]_0$, if we plot
the reciprocal of [A] versus time we should obtain a straight line of
slope = ``k''.

\begin{rawhtml}
<A HREF="#top_of_page">Return to Top of Page</A>
\end{rawhtml}


\end{document}
```

7.5 Manual Monte Carlo Simulation HTML Pages

The drawing of the whiteboard used for the manual Monte Carlo simulations was done using Xfig. This can be converted to a gif for the HTML pages below using: `fig2dev -L gif grid.fig grid.gif`

66 ⟨*manual.html* 66⟩≡

```
⟨html head 59b⟩

<A NAME="top_of_page">
<H1>Monte Carlo Experiments done Manually</H1>

⟨html top navbar 60c⟩

<H2>Introduction</H2>

In this experiment you will carry out some Monte Carlo experiments
manually.  This should provide you with an understanding of what
actually occurs on a molecular level, why the concentration of a species
versus time has a particular form and how that is related to the
differential rate equations.  Subsequent experiments are done using a
computer program which just repeats exactly what you will do here except that
the computer is able to run with many more &quot;molecules&quot; and for
a much longer simulation time.

<H2>The Monte Carlo Method</H2>

Consider first a monomolecular reaction A ---&gt; B carried out in a
non-reacting solvent.
<!-- This type of reaction occurs for example when -->
<P>

In the Monte Carlo simulation, the reaction vessel is represented by a
grid in which each square is numbered.  <A HREF="#fig1">Figure 1</A>
shows this virtual reaction vessel containing 20 mole &#037 of
reactant A (ie. there are 20 molecules marked A in the 100 available
positions). The blank squares represent solvent molecules.  If we wished
to consider the reaction of a pure substance then all positions of the
grid would be occupied with reactant A.
<P>

<A NAME="fig1"/>
<TABLE>
<TR><TD><IMG src="../images/grid.gif"></TD>
    <TD valign="top">Figure 1. Virtual reaction vessel.
    <P>Initially there was 20
       mole &#037 of reactant A but some has reacted
       (crossed out squares).</TD></TR>
</TABLE>
<P>

<!-- <TR><TD>Figure 1. Virtual reaction vessel. Initially there was 20 mole &#037 of
reactant A but some has reacted (crossed out squares).</TD></TR> -->

To perform a Monte Carlo simulation the grid is repeatedly pinpointed in
a random fashion by picking a random number. This is callled a
&quot;try&quot;.When a pinpointed position is occupied by a letter ‘A’
which represents an unreacted molecule the given molecule is considered
to have reacted to form molecule ‘B’.  This is represented in Figure 1
by marking off the letters with a cross.  If the random selection of a
position in the grid hits a crossed-out letter representing a reacted
molecule or a vacancy representing a solvent molecule, then such a
selection will represent a failed event, or the passage of an element of
time without any reaction taking place. The number of letters of A
remaining after N tries is equivalent to the concentration [A] and the
number of tries N is equivalent to time t. Another way to look at this
```

```
is that each <I>try</I> is a &quot;<I>tick of the clock</I>&quot;.
<P>

Now let's consider how we do a Monte Carlo simulation of a bimolecular
reaction involving the collision of two reactant molecules A + B ---&gt;
C.  The progress of such a reaction again depends at any instant upon
the probability of a reactive collision taking place.  Like the
monomolecular reaction we use a grid with the letters A marked on it to
represent molecules of type A. This time though we use a second grid of
the same size but marked with the letters B to represent molecules of type
B.  If both reactants are pure substances, that is, not in solution,
then all positions on the both grids would be occupied with letters.
<P>

This time the procedure is to pick two random numbers for each
&quot;try&quot; and pinpoint both grids. A reaction event occurs (ie A +
B -> C) if both A and B letters are hit. If only one unreacted molecule
is hit or a solvent molecule is hit or a reacted molecule is hit then no
reaction takes place.
<P>

<B>Materials Supplied</B>
<UL>
    <LI>White board grid marked off into 100 squares
    <LI>White board markers
    <LI>Graph paper
    <LI>Printed table of random number pairs
</UL>


<H2>Manual Simulation of a Monomolecular Reaction
in Solution</H2>

<B>Populate the grid with molecules of A</B><BR>
The grid you will be using is a whiteboard marked out with 100 numbered
squares and the initial concentration of reactant A that you should use
is 40 mole &#037;.  First allocate molecules A to random positions on
the board.  Just mark with the whiteboard marker 40 A's anywhere on the
board. The choice is unimportant, since the act of pinpointing of sites
is entirely random.  Don't use the printed table of random numbers as
that's going to be used later.
<P>

To pinpoint positions in the grid randomly we use a set of uniformly
distributed random numbers whose range covers the number of positions in
the grid. For a grid of a hundred positions, a set of random digits
ranging from 00 to 99 is employed. A <B>printed table</B> of random
numbers is provided. An example table, like the printed out one, is
shown below.
<P>

<TABLE border="1">
<TR><TH valign=top>Try Number</TH>
    <TH>Random Number</TH>
</TR>
<TR><TD>1<TD>12</TR>
<TR><TD>2<TD>28</TR>
<TR><TD>3<TD>36</TR>
<TR><TD>4<TD>49</TR>
<TR><TD>5<TD>51</TR>
<TR><TD>6<TD>65</TR>
<TR><TD><BR><TD><BR></TR>
<TR><TD>7<TD>73</TR>
<TR><TD>8<TD>86</TR>
```

```
<TR><TD>9<TD colspan=2>etc...</TR>
<CAPTION align=bottom>Example of random data for tries.</CAPTION>
</TABLE>
<P>

For each two digit random number take the first digit as the x
coordinate and the second digit as the y coordinate for pinpointing the
grid.  Starting from the top of the table choose the first random
number.  For subsequent random numbers use the next row down, do
not pick random numbers from the table randomly. This can bias the
choice towards numbers located near the centre of the table.
<P>

So that you don't have too many data points to graph this procedure will
use a <B>set</B> of random numbers and we will call this a <B>run</B>.
We will then plot the numbers of A remaining after each run.  For each
run a small set (6 in this case) of random numbers is chosen from the
table of random numbers.  Pinpoint the appropriate positions on the grid
and if an A is hit cross off that position with the marker pen. At the
end of each set or run a tally is made of the total number of successful
reactions for this run and the remaining number of A's.  One run then
represents a unit of time and the number of runs N, the elapsed time.
The elapsed time simulates the number of attempts of a molecule to
decompose in a monomolecular reaction, or for a biomolecular reaction,
the number of collisions taking place.
<P>

For instance, in the first run you might have picked 23 as the first
random number. If a letter A is at site (2,3) cross it off using the
whiteboard marker. Again pick a random number pair and repeat the
procedure another 5 times. After a total of 6 tries we have done one
run.
<P>

<B>Record your results</B><BR>
Record the remaining number of unreacted molecules A (ie 40 minus
the number that you crossed off) and the time (which will be 1 for the
first run). An example of what your table of results might be like is
shown in the table below.
<P>

<TABLE border="1"> <TR><TH>Run Number</TH> <TH>No. Reacted</TH> <TH>No.
Remaining</TH>
</TR>
<TR><TD>0<TD>0<TD>40</TR>
<TR><TD>1<TD>4<TD>36</TR>
<TR><TD>2<TD>4<TD>32</TR>
<TR><TD>3<TD>4<TD>28</TR>
<TR><TD>4<TD>3<TD>25</TR>
<TR><TD>5<TD>3<TD>22</TR>
<TR><TD>6<TD>2<TD>20</TR>
<TR><TD>7<TD>1<TD>19</TR>
<TR><TD>8<TD>0<TD>19</TR>
<TR><TD colspan=3>etc...</TR>
<CAPTION align=bottom>Example of how you might record the data.
</CAPTION>
</TABLE>
<P>

<B>Plot results as you go</B><BR>
Plot your results as <I>No. Remaining</I> versus <I>Run Number</I> as you
gather the data. That way you will be able to see when the reaction rate
has effectively slowed and when to stop the simulation.
For the monomolecular reactions, around 100 runs should be sufficient.
```

```
<H2>Bimolecular Reaction of Two Pure Substances</H2>

For this biomolecular reaction mark off the squares so that all the
squares of one grid contain A's and all the squares of the
other grid contain B's (ie we have two pure reactants and no
solvent).
<P>

Read a <B>pair</B> of two digit numbers from the table, ie two
rows.  Apply the first number of the
pair to one grid for A molecules and the second to the other grid for B
molecules.  If the first number of the pair selects a molecule A, but the second
number applied to the second grid only selects a reacted
molecule, then the try is unsuccessful. This represents the occurrence
of an unsuccessful collision.  If the two tries hit an A <B>and</B> a B
then this represents a successfule reaction and we cross off the A and B
on both grids.
Record your results in a table of Run number, Remaining A and Remaining B.
<P>

<H2>Consecutive First Order Reaction</H2>

In the case of the consecutive reaction, pinpointing of a position
containing molecule A leads to the production of B. Likewise, if B is
pinpointed it is marked off and a molecule C is created. Use a separate
grid for A and B molecules. The ratio of the number of runs at the grid
containing A to the number of runs at the grid containing B is
proportional to the ratio k1 / k2 of the rate constants. For the
consecutive reaction, 200 runs should be sufficient.

<H2>Report</H2>

In the following N is the number of runs.
When you plot the following draw a smooth curve through your plots.

<OL>
    <LI>For the monomolecular reaction plot your experimental results
        as <CODE>ln [A]</CODE> versus N where <CODE>ln</CODE> is the natural
        logarithm. <BR>
        Determine the Monte Carlo rate constant for this
        reaction using the line of best fit.
    <LI>For the bimolecular reaction plot the quantity
        1/[A] versus N.<BR>
        Determine the
        Monte Carlo rate constant for this reaction using the line of
        best fit.
    <LI>For the consecutive reaction plot the following on the same
        graph:
    <UL>
        <LI>[A] / [A]<sub>0</sub> versus N
        <LI>[B] / [A]<sub>0</sub> versus N
        <LI>[C] / [A]<sub>0</sub> versus N
    </UL>
    <LI>Why does the plot of concentration versus time exhibit
        &quot;noise&quot;?
    <LI>What parameter(s) in the experiment would you vary to reduce
        the noisyness of the data?
    <LI>Would you expect the size of the data set (ie number of tries
        in a run) to effect the results? If so in what way?
    <LI>Discuss your results, including a comment on the accuracy of
        this method.
</OL>
Your discussion of the results for this section of the experiment
```

```
should only take a few paragraphs.
<P>

<!--
<H2>Advanced Questions</H2>
<UL>
    <LI>Why are two grids used for a bimolecular reaction instead of
        just one grid?
</UL>
<P>
-->

<A HREF="#top_of_page">Return to Top of Page</A>
<P>

<HR>
⟨html tail 60a⟩
```

7.6 Computer Monte Carlo Experiments HTML Pages

71 ⟨*computer.html* 71⟩≡

```
⟨html head 59b⟩

<H1>Monte Carlo Experiments by Computer</H1>

⟨html top navbar 60c⟩

<H2>Before you Begin</H2>
Read over the section below &quot;Report on your Results&quot; now so
that that you will know what to look for and what to think about while
doing the experiments. Then run the experiment.

<H2>Run The Monte Carlo Experiments</H2>

This is where we get the computer to do all the work of running a Monte
Carlo simulation. You will have control over how many virtual molecules
there are in the reaction vessel and how long you want to run the
simulation to run.
<P>

The aim is to obtain experimental data that exhibits the behaviour that
you expect for this form of reaction. Insufficient molecules in the
simulation or inadequate time will not give you good data for the
analysis. However you choose numbers that are too large ie too many
molecules the simulation will take too long. You have to choose a
compromise through some experimentation.  If you need help at any time
click the Help link which is always located at the top right.
<P>

<TABLE>
<TR><TD rowspan="4" valign="top">
    To run the experiments and analyse the results click here:</TD>
    <TD>
    <A HREF="/kinetics/cgi-bin/mc0?Ao=40&Bo=0&TotalMols=2000&maxN=1000">
    Run Zeroth Order Reaction</A></TD>
</TR><TD>
    <A HREF="/kinetics/cgi-bin/mc1?Ao=40&Bo=0&TotalMols=2000&maxN=5000">
    Run First Order Reaction</A></TD>
</TR>
<TR><TD>
    <A HREF="/kinetics/cgi-bin/mc2?Ao=40&Bo=40&TotalMols=2000&maxN=8000">
    Run Second Order Reaction</A></TD>
</TR>
<!-- TODO
<TR><TD>
    Run Consecutive First Order Reaction</TD>
</TR>
-->
</TABLE>
<P>

When you are finished return back to here and report on your results.
<P>

<H2>Report on your Results</H2>

Report briefly about what you observed and what were the major
conclusions that your group came to regarding the reaction kinetics.
This should take no more than a page or two for all the experiments.

These are questions that you my wish to cover in your report:
<UL>
    <LI>How many molecules and how many runs did it take to get good
```

```
        data?
    <LI>What was the expected mathematical form of the raw data and did
        the data show this?
    <LI>What about the analysed data? Did this backup your expectations
        about the raw data?
    <LI>How close is the experimental data to the predictions from
        theory?
</UL>

<HR>
⟨html tail 60a⟩
```

7.7 LaTeX Code for Creating Radio Button gifs in Analysis Section

These images are for the radio buttons between the two graphs. The LaTeX code here is reproduced below as code chunk `equations.tex`.

Radio button 1. No change

Radio button 2. ln[A] *vs.* N

Radio button 3 for [A] = [B]

$$\frac{1}{[\mathrm{A}]}\; vs.\ \mathrm{N}$$

Radio button 4 for [A] $\ne$ [B]

$$\ln \frac{B_o - x}{A_o - x}\; vs.\ \mathrm{N}$$

73 ⟨*equations.tex* 73⟩≡

```
\documentclass{article}
\begin{document}
\section{Images for the Analysis Form}

These images are for the radio buttons between the two graphs.

Radio button 1. $\mathrm{No\; change}$

Radio button 2.
$\ln[\mathrm{A}]\; vs.\ \mathrm{N}$

Radio button 3 for [A] = [B]
\[
\frac{1}{[\mathrm{A}]}\; vs.\ \mathrm{N}
\]

Radio button 4 for [A] $\ne$ [B]
\[
\ln \frac{A_o(B_o-x)}{B_o(A_o-x)}
\]

\end{document}
```

7.8 The General HTML Help File

74a ⟨*help.html* 74a⟩≡

```
⟨html head 59b⟩

<H1>General Help</H1>

<TABLE width="100%">
<TR><TD align="left" ><A HREF="index.html" >Main Contents</A>
</TR>
</TABLE>
<P>
There is no help here yet :-) <BR>
If you have suggestions for what might go here
please contact the Author of this software. <BR>
See the <A HREF="about.html">About this Software</A>
<P>

Click on the &quot;Main Contents&quot; link above or click on your browser's
&quot;Back&quot; button to return to the front page.<P>

<HR>
⟨html tail 60a⟩
```

Help for the sections Intro, Theory, Manual and Computer are similar except for the location of the Main Contents which links to ../index.html instead of index.html.

74b ⟨*help2.html* 74b⟩≡

```
⟨html head 59b⟩

<H1>Specific Help</H1>

<TABLE width="100%">
<TR><TD align="left" ><A HREF="../index.html" >Main Contents</A>
</TR>
</TABLE>
<P>

There is no help here yet :-) If you have suggestions for what might go here
please contact the Author of this software. <BR>
See the <A HREF="about.html">About this Software</A>
<P>

Click on the &quot;Main Contents&quot; link above or click on your browser's
&quot;Back&quot; button to return to the front page.<P>

Click on your browser's &quot;Back&quot; button to return to
the last section.<P>

<HR>
⟨html tail 60a⟩
```

8 About this Software

75 ⟨*about.html* 75⟩≡

```
⟨html head 59b⟩

<H1>About this Software</H1>

<TABLE width="100%">
<TR><TD align="left" ><A HREF="index.html" >Main Contents</A>
</TR>
</TABLE>
<P>
<H2>Introduction</H2>
This software was developed whilst at the University of Technology, Sydney for
teaching Kinetics in the Department of Chemistry, Materials and Forensic
Science.

<H2>How does it work?</H2>
This program is web based and will run on any modern graphical web
browser.  When you enter input data for a Monte Carlo simulation and
click the <CODE>Start</CODE> button to commence the simulation the web
browser submits your request as a HTML form to another computer running
a web server. That web server runs a CGI script which does the
calculations and returns the results to you as another form - ready to
be filled out again with new parameters and submitted.
<P>

<H2>What's it written in and what runs it?</H2>
The CGI script is written in a programming language called Perl and runs on a
low end Intel processor (233MHz and 32Meg RAM) running a Linux/GNU operating
system with the Apache Web Server. This PC easily provides sufficient
computational power for the Monte Carlo calculations.
Most of the delay that you will notice is your Web Browser rendering the graphics
rather than the server doing the Monte Carlo simulation.
The Introduction and Theory sections are typeset using LaTeX and
automatically converted into HTML using <CODE>latex2html</CODE>. This
generates the gif images of the equations that you see in the text.
<P>

<A HREF="http://www.perl.org/">Perl</A>, Linux and the
<A HREF="http://www.apache.org/">Apache</A> Web Server are open source
software as are <A HREF="http://www.latex-project.org/">LaTeX</A> and
<A HREF="http://www-texdev.mpce.mq.edu.au/l2h/docs/manual/">latex2html</A>.
<P>

This Kinetics software is also open source and covered by
the <A HREF="http://www.gnu.org">GNU</A> copyright. A copy is available
<!-- TODO check that COPYING is accessable -->
<A HREF="COPYING">here</A>.
<P>

<A HREF="mailto:Mike.Lake@uts.edu.au">
Michael Lake</A><BR>
July 2001
<P>

<HR>
⟨html tail 60a⟩
```

9 Test Code

76a ⟨*tests.html* 76a⟩≡ 76b ▷

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
    <TITLE>Tests</TITLE>
    <STYLE type="text/css">
    <!--
    H2 {font-size: 12pt}
    -->
    </STYLE>
</HEAD>

<BODY>

<H1 align="center">Apache Server and CGI Test Routines</H1>
```

9.1 Test - Apache Web Server

If `srm.conf` contains `Alias /monte/ /home/kinetics/public_html/monte_carlo`

76b ⟨*tests.html* 76a⟩+≡ ◁ 76a 76c ▷

```
<H2>Alias Tests</H2>

Testing aliases in <TT>srm.conf</TT> such as
&quot;<TT>Alias /monte/ /home/kinetics/public_html/monte_carlo</TT>&quot;.<BR>
Tail Apache error.log to see what is happening as you click on each URL.
<P>

<A HREF="http://martel/monte/">http://martel/monte/</A>
<P>

<A HREF="http://martel/~kinetics/monte_carlo">
http://martel/~kinetics/monte_carlo</A>

<HR>
```

9.2 Test - Apache Web Server

The first part just tests the Apache web server with a plain text text form.

76c ⟨*tests.html* 76a⟩+≡ ◁ 76b 77b ▷

```
<H2>Test Apache Web Server</H2>
<FORM   method=get
        action="/kinetics/cgi-bin/testApache">

        <INPUT type="submit" value=" Test Apache ">
        <tt>method=get action=/kinetics/cgi-bin/testApache</tt>
</FORM>
<HR>
```

Here is the cgi script that the `Test Apache` button fires off.

77a ⟨*testApache* 77a⟩≡
```
#!/usr/bin/perl -w

use strict;

# Note that there MUST be a blank line after the MIME type header.
print <<End_of_text;
Content-type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
This is the script 'testApache'<BR>
The test suceeded.
<P>
<A HREF="http://martel/monte/tests.html">Return to Tests Page</A>

</BODY>
</HTML>
End_of_text
```

This script should return the text:

```
This is the script 'testApache'
The test suceeded.
```

9.3 Test - Report on Document Locations

This reports on the values of various locations that documents will be served from. Very useful for debugging if the browser reports that a script cannot be found.

77b ⟨*tests.html* 76a⟩+≡ ◁76c 78a▷
```
<H2>Report on Document Locations</H2>
<FORM   method=get
        action="/kinetics/cgi-bin/test-locations">

        <INPUT type="submit" value=" Report Locations ">
        <tt>method=get action=/kinetics/cgi-bin/test-locations</tt>
</FORM>
<HR>
```

77c ⟨*test-locations* 77c⟩≡
⟨*common preamble* 23a⟩
```
print header(), "\n";
print start_html("Document Locations"), "\n";
print h1("Report on Document and Script Locations"), "\n";

print "Server:      $server", br;
print "HTML base:   $html_base", br;
print "File base:   $file_base", br;
print "Temp dir:    $tmp_dir", br;
print "File prefix: $file", br;
print "Time:        $seconds";

print end_html();
```

9.4 Test - CGI Push

The documentation on this can be found here: `perldoc CGI::Push`

> "INSTALLING CGI::Push SCRIPTS Server push scripts must be installed as no-parsed-header (NPH) scripts in order to work correctly. On Unix systems, this is most often accomplished by prefixing the script's name with "nph-"."

The documentation `perldoc CGI` also states...

> "Servers use a variety of conventions for designating CGI scripts as NPH. Many Unix servers look at the beginning of the script's name for the prefix "nph-". The Macintosh WebSTAR server and Microsoft's Internet Information Server, in contrast, try to decide whether a program is an NPH script by examining the first line of script output."

78a ⟨*tests.html* 76a⟩+≡ ◁77b 79▷

```
<H2>Test CGI Push Module</H2>

<FORM   method=get
        action="/kinetics/cgi-bin/nph-testCGI">

        <INPUT type=submit value=" Test CGI Push ">
        <tt>action=/kinetics/cgi-bin/nph-testCGI</tt>
</FORM>
<HR>
```

The documentation on this can be found here: `perldoc CGI`

> "CGI.pm provides three simple functions for producing multipart documents of the type needed to implement server push. To import these into your namespace, you must import the ":push" set. You are also advised to put the script into NPH mode and to set $— to 1 to avoid buffering problems."

This is the script that is fired off.

78b ⟨*nph-testCGI* 78b⟩≡

```
#!/usr/bin/perl -w
use CGI qw/:push -nph/;

$| = 1;
print multipart_init(-boundary=>'----------------here we go!');
while (1) {
    print multipart_start(-type=>'text/plain');
    print "The current time is ", scalar(localtime),"\n",
    multipart_end;
    sleep 1;
}
```

It will return a string like `The current time is Sun Jul 15 13:11:54 2001` which continually updates.

9.5 Test - CGI Functions Interface

Append some code to test out form submission.

79 ⟨*tests.html* 76a⟩+≡ ◁78a 81▷

```
<H2>Test CGI Functions Interface</H2>
<FORM   method=get
        action="/kinetics/cgi-bin/testCGI-1">

<INPUT type=submit value=" Test CGI Functions Interface">

<H4>Text Boxes</H4>
Concentration: <INPUT type="text" size="2" name="conc" value="20"> &#037;
<!-- This sends: testForm?conc=20 -->

<H4>Check Boxes</H4>
<em>Each name selected will send a value. This sends: chem=A if A is selected AND
chem=B if B is selected etc...</em>
<P>
Display in plot:<br>
A: <INPUT type="checkbox" checked name="chem" value="A"> Chemical A<BR>
B: <INPUT type="checkbox"         name="chem" value="B"> Chemical B<BR>
C: <INPUT type="checkbox"         name="chem" value="C"> Chemical C

<H4>Radio Boxes</H4>
<em>Those of the same name are members of a group.
The value is included as part of the submitted string.</em>
<P>

What experiment do you want to run?<br>
Order 0: <INPUT type="radio" checked name="order" value="zero">
    Zeroth order reaction<BR>
Order 1: <INPUT type="radio"         name="order" value="mono">
    Monomolecular reaction<BR>
Order 2: <INPUT type="radio"         name="order" value="bi">
    Bi-molecular reaction<BR>

Button object yet to do.
<BUTTON type=submit>
    Click to test: <!-- IMG src="http://martel/~kinetics/images/next.gif" -->
</BUTTON>
</FORM>

<HR>
```

Here is the cgi script that the submit button fires off.

80 ⟨*testCGI-1* 80⟩≡

```
#!/usr/bin/perl -w

use strict;
use CGI qw(:standard);

my ($i, $concA);
my ($chemA, $chemB, $chemC);
my @list;
my $file = "crap.data";

print header(), "\n";
print start_html("Test Form"), "\n";
print h1("Results of Test Form"), "\n";

$concA = param("conc");
if ( $concA =~ /[^0-9]\s/) {
    print "User entered something other than a digit or white space.";
}
else
{
    print "Concentration of reactant A = ", $concA, "&#037;", p;
}

$chemA = param("chem");
print "Chemicals that will be plotted: ", $chemA, p;

print "List of all parameters passed to script is:", br;
@list = param();
foreach $i (@list)
{
    print "Name=<tt>",$i, "</tt> Value=",param($i), br;
}

print "<P><B>Testing file writing ...</B>";

if (open (OUT, ">/tmp/$file") )
{
    print OUT "This is a test from the Kinetics Test suite\n";
    print br, "File &quot;<TT>/tmp/$file</TT>&quot; written OK";
}
else
{
    print "Cannot open file $file for writing\!";

}
close(OUT);

print end_html();
```

9.6 Test - CGI Objects Interface

81 ⟨*tests.html* 76a⟩+≡ ◁79

```
<H2>Test CGI Objects Interface</H2>

<FORM   method=get
        action="/kinetics/cgi-bin/testCGI-2">

        <INPUT type=submit value=" Test CGI Objects Interface">
</FORM>

</BODY>
</HTML>
```

This is the script.

82 ⟨*testCGI-2* 82⟩≡
```
#!/usr/bin/perl
use CGI qw/:standard/;          # load CGI routines

my $now = gmtime;
my $seconds = time();
my $server  = `hostname`; chomp($server);

$q = new CGI;                   # create new CGI object
print $q->header(               # create the HTTP header
            -xbase=>"http://$server/kinetics/cgi-bin/",
            -type=>'text/html',
            -expires=>'-1d',
            -pragma=>'no-cache',
            -last_modified=>"$now",
            -cache_control=>'no-cache');

print $q->start_html('Test CGI'),   # start the HTML
      $q->h1('Test CGI');           # level 1 header

print $q->startform("GET","testCGI-2");
print $q->textfield(-name=>'Ao',
                    -default=>'20',
                    -override=>0,
                    -size=>5,
                    -maxlength=>5),p;

# $q->param('Ao', "12");

print $q->defaults(' Default Values ');
print $q->hidden(-name=>'order', -default=>'1');

print $q->submit(-name=>'button_name',
                 -value=>' Start ');

# If script invoked for the first time there will be no parameters.
# So don't show any results.
if (param())
{
    $Ao = param('Ao');
    print "Value was: ", param('Ao'), " at ",$now, p;
    print "Seconds = ", $seconds, p;
    $filename = "graph.$seconds.gif", p;
    print "Filename = ", $filename, p;

    # Create graph.

    system("/usr/bin/graph -L 'Test gif $Ao' \
    /home/kinetics/public_html/monte_carlo/tmp/data >| /tmp/temp.meta");
    system("/usr/bin/plot -T gif --bitmap-size 400x400 \
    /tmp/temp.meta>| /tmp/order.gif");

print $q->img({-expires=>'+30s',
            -pragma=>'no-cache',
            -cache_control=>'no-cache',
            -last_modified=>"$now",
            -src=>"/tmp/order.gif?$seconds"});
}
else
{
    print "First run at ", $now;
}
```

```
print $q->endform;
print $q->end_html;                     # end the HTML
```

9.7 Tests - TODO still

9.7.1 nph-mc

84 ⟨*nph-mc* 84⟩≡

```
#!/usr/bin/perl -w
use CGI qw/:standard :push -nph/;

$| = 1; # Only required if doing server push.

my ($i, $j, @molArray, @A, @B);
my $rand1;

my $TotalMols = 400; # Maximum size of the array of molecules ie
                     # No. molecules in the beaker. Try 2500
my $Ao = 40;        # Percentage initial concentration of reactant A.
my $Bo = 40;        # Percentage initial concentration of reactant B.
my $NAo;            # No. of molecules of A initially ie 0.01 * $Ao * $TotalMols
my $NBo;            # No. of molecules of B initially ie 0.01 * $Bo * $TotalMols
my ($NA, $NB);      # No. of mols at any later time.
my $maxN = 10000;   # Max number of reactions. Try 10000

my $server  = `hostname`; chomp($server);
my $html_base = "http://$server/monte";
my $file_base = "/home/kinetics/public_html/kinetics";
my $tmp_dir = "/tmp";
my $file = "Kinetics.data";

###############
# Run Reaction
###############

# Initialise values for a Monomolecular reaction
init_mono();

my $side = sqrt($#molArray+1);
my $i;
my $n = 1;
my ($NA, $NB);
my $string = "";

print multipart_init(-boundary=>'start----------------end');
while (1)
{
    $string = "";
    $n = $n + 1;
    for ($i=0; $i<$side; $i++)
    {
        $string = $string . "@molArray[$i*$side .. $i*$side+$side]";
        # replace solvent molecules with a space, or for a . use \.
        $string =~ s/0/ /g;
        $string =~ s/1/A/g;     # replace A molecules with 'A'
        $string =~ s/2/B/g;     # replace B molecules with 'B'
        $string = $string . "\n";
    }
    runDice1();
    print multipart_start(-type=>'text/plain');
    print $string;

    print "\n\nStep 0  A=", $NAo, " B=", $NBo;
    print   "\nStep ", $n, " A=", $NA, " B=", $NB;

    print "\n\nTo stop click the Browser's Stop button";
    print   "\nTo start again click the Browser's Reload button";
    print   "\nTo return click the Browser's Back button";
```

```
    print multipart_end;
    sleep 1;
}

⟨mc1 subroutines 27b⟩
```

9.7.2 Test - readdata

85 ⟨*test-readdata* 85⟩≡

```
#!/usr/bin/perl -w

use strict;

my ($i, $j, $tmp, @array);
my $series=1;
my $blank=0;
my (@A, @B, @C);

open (IN, "/tmp/Kinetics.data") ||
    die "Cannot open file for reading\!";

while(<IN>)
{
    s/^\s*//; # get rid of all leading spaces
    if ($_)
    {
        ($i, $tmp) = split;
        if ($series < 2)    # First time through
        {
            $array[$i] = $tmp;
        }
        else
        {
            $array[$i] = $array[$i]."\t".$tmp;
        }
    }
    else
    {
        # First time series = 1 then incremented to
        # 1 on the first blank line and 2 on the second blank line
        # following.
        print "Blank line\n";
        $series++;
    }
}
close(IN);

print "----------------------\n";
    for ($i=0; $i<=$#array; $i++)
    {
        print $i, "\t", $array[$i], "\n";
    }
```

9.7.3 Test - readdata 2

See: man perlreftut

86 ⟨*test-readdata2* 86⟩≡

```
#!/usr/bin/perl -w

use strict;

my ($i, $j, @tmp);
my $series=1;
my $blank=0;
my (@A, @B, @C);
my ($refA, $refB, $refC);

($refA, $refB, $refC) = readData("data");
@A = @$refA;
@B = @$refB;
@C = @$refC;

#print "A data\n";
#for ($i=0; $i<=$#{$refA}; $i++)
#{
#   print $i, "\t", $refA->[$i];
#}

print "A data again\n";
for ($i=0; $i<=$#A; $i++)
{
    print $i, "\t", $A[$i];
}

print "\nB data\n";
for ($i=0; $i<=$#{$refB}; $i++)
{
    print $i, "\t", $refB->[$i];
}

print "\nC data\n";
for ($i=0; $i<=$#{$refC}; $i++)
{
    print $i, "\t", $refC->[$i];
}


sub readData
{
    my $file = $_[0];
    my $i=0;
    my $j=0;
    my (@A, @B, @C);
    my $blank=0;

    open (IN, $file) || die "Cannot open file for reading\!";

    while(<IN>)
    {
        s/^\s*//;   # get rid of all leading spaces
                    # Now blank lines will be null
        if ($_ && $blank==0)
        {
            push(@A, $_);
            print $i, "\t", $_;
            $i++;
        }
```

```
        elsif ($_ && $blank==2)
        {
            push(@B, $_);
            #print $j, "\t", $B[$j];
        }
        elsif ($_ && $blank==4)
        {
            push(@C, $_);
            #print $k, "\t", $C[$k];
        }
        else
        {
            $blank = $blank + 1;
            #print "Blank line\n";
        }
    }

    close(IN);
    return (\@A, \@B, \@C);
}
```

10 Algorithms used and other Techniques

10.1 Obtaining a Random Number

See the Perl documentation (perldoc -f rand):

> rand EXPR
>
> Returns a random fractional number greater than or equal to C¡0¿ and less than the value of EXPR. (EXPR should be positive.) If EXPR is omitted, the value C¡1¿ is used. Automatically calls C¡srand()¿ unless C¡srand()¿ has already been called. See also C¡srand()¿.
>
> (Note: If your rand function consistently returns numbers that are too large or too small, then your version of Perl was probably compiled with the wrong number of RANDBITS.)

10.2 How do I select a random element from an array?

Use the rand() function (perldoc -q rand):

```
sub select_random
{
    my ($index, $element);

    $index   = rand @array;
    $element = $array[$index];
}
```

Make sure you only call srand once per program. If you are calling it more than once (such as before each call to rand), you're almost certainly doing something wrong.

10.3 How do I shuffle an array randomly?

This is found in "perldoc -q rand". One uses the Fisher-Yates shuffle algorithm:

88 ⟨*mc common subroutines* 24⟩+≡ (20–22) ◁26

```
# Generate a random permutation of @array in place
sub fisher_yates_shuffle
{
    my $array = shift;
    my $i;
    for ( $i = @$array; --$i; )
    {
        my $j = int rand ($i+1);
        next if $i == $j;
        @$array[$i,$j] = @$array[$j,$i];
    }
}
```

Usage is like this;

```
# permutes @array in place
fisher_yates_shuffle( \@array );
```

The perldocs also mention not to use shuffling algorithms that use splice, randomly picking another element to swap the current element with such as:

```
    srand;
    @new = ();
    @old = 1 .. 10;  # just a demo
    while (@old)
    {
        push(@new, splice(@old, rand @old, 1));
    }
```

> "This is bad because splice is already O(N), and since you do it N times, you just invented a quadratic algorithm; that is, O(N**2). This does not scale, although Perl is so efficient that you probably won't notice this until you have rather largish arrays."

10.4 Preventing Caching

Don't need to use this now

```
print $q->header(-type=>'text/html',
                 -expires=>'+30s',
                 -pragma=>'no-cache',
                 -cache-control=>'no-cache');

<IMG border="1" width="400" height="400"
        cache-control="no-cache" pragma="no-cache"
        src="http://$server/tmp/order.gif?$seconds">
```

10.5 Creating gifs on the fly

This method creates the gif immediately. Instead of using:

```
    system("graph -T gif data >|
        /home/kinetics/public_html/tmp/order.gif");
```

use this:

```
    cat data | graph -T gif -x 0 20 -y 0 0.2 > crap.gif
    graph data > crap
    plot -T gif --bitmap-size 400x400 crap > crap.gif
```

Appendices

A Script for Converting png image files to gifs

90 ⟨*allpngs2gifs* 90⟩≡

```
#!/bin/bash

# The latex2html will only create png files because the
# netpbm utility does not have ppmtogif in the Debian disro.
# To get around this use convert to chnage all the pngs to gifs.
# Then change all references in the HTML files from pngs to gifs.

for i in intro/*.png equations/*.png theory/*.png; do
    convert $i `echo $i | sed s/png/gif/`
    rm -f $i
done

cat intro/intro.html | sed s/png/gif/g > intro/tmp.html
cat theory/theory.html | sed s/png/gif/g > theory/tmp.html
cat equations/equations.html | sed s/png/gif/g > equations/tmp.html

mv intro/tmp.html intro/intro.html
mv theory/tmp.html theory/theory.html
mv equations/tmp.html equations/equations.html
```

B Installing the Statistics::OLS CPAN Package

Statistics::OLS performs ordinary least squares and associated statistics. The version used was version 0.07 from CPAN file `Statistics-OLS-0.07.tar.gz`

```
$ tar xvf Statistics-OLS-0.07.tar
Statistics-OLS-0.07/
Statistics-OLS-0.07/Makefile.PL
Statistics-OLS-0.07/OLS.pm
Statistics-OLS-0.07/Changes
Statistics-OLS-0.07/test.pl
Statistics-OLS-0.07/README
Statistics-OLS-0.07/MANIFEST

$ cd Statistics-OLS-0.07

Statistics-OLS-0.07$ ls
Changes  MANIFEST  Makefile.PL  OLS.pm  README  test.pl
```

README says...

“Untar and unzip the distribution and cd into its directory. Then the usual…”

```
perl Makefile.PL
make
make test
make install

$ pwd
/home/admin/Statistics-OLS-0.07

$ perl Makefile.PL
Checking if your kit is complete...
    Looks good
    Writing Makefile for Statistics::OLS

# make
mkdir blib
mkdir blib/lib
mkdir blib/lib/Statistics
mkdir blib/arch
mkdir blib/arch/auto
mkdir blib/arch/auto/Statistics
mkdir blib/arch/auto/Statistics/OLS
mkdir blib/lib/auto
mkdir blib/lib/auto/Statistics
mkdir blib/lib/auto/Statistics/OLS
mkdir blib/man3
cp OLS.pm blib/lib/Statistics/OLS.pm
Manifying blib/man3/Statistics::OLS.3pm

Statistics-OLS-0.07$ make test
PERL_DL_NONLAZY=1 /usr/bin/perl -Iblib/arch -Iblib/lib \
  -I/usr/lib/perl5/5.005/alpha-linux \
  -I/usr/lib/perl5/5.005 test.pl
1..2
ok 1
ok 2
Statistics-OLS-0.07$
```

```
# make install
Installing /usr/local/lib/site_perl/Statistics/OLS.pm
Installing /usr/local/man/man3/Statistics::OLS.3pm
Writing /usr/local/lib/site_perl/i386-linux/auto/Statistics/OLS/.packlist
Appending installation info to /usr/lib/perl5/5.005/i386-linux/perllocal.pod
#

$ perldoc Statistics::OLS
```

Documentation displays OK so package is installed.

C Changes During 2001 to Incorporate

C.1 Install noweb onto Debian Box at Work

An "`apt-cache show nowebm`" says that this replaces noweb.

```
# apt-get install nowebm
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  iconx
The following NEW packages will be installed:
iconx nowebm
0 packages upgraded, 2 newly installed, 0 to remove and 3 not upgraded.
Need to get 1133kB of archives. After unpacking 2673kB will be used.
Do you want to continue? [Y/n] y
Get:1 http://mirror.aarnet.edu.au stable/main iconx 9.3.2-2 [146kB]
Get:2 http://mirror.aarnet.edu.au stable/main nowebm 2.9a-5 [986kB]
Fetched 1133kB in 27s (40.6kB/s)
Selecting previously deselected package iconx.
(Reading database ... 27675 files and directories currently installed.)
Unpacking iconx (from .../iconx_9.3.2-2_i386.deb) ...
Selecting previously deselected package nowebm.
Unpacking nowebm (from .../nowebm_2.9a-5_i386.deb) ...
Setting up iconx (9.3.2-2) ...

Setting up nowebm (2.9a-5) ...
texhash: Updating /var/lib/texmf/ls-R...
texhash: Updating /var/spool/texmf/ls-R...
texhash: Done.

#
```

C.2 Install netpbm Package onto Debian Box at Work

Do I really need this package or not ?

```
# apt-get install netpbm-nonfree
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  netpbm-nonfree
0 packages upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 47.5kB of archives. After unpacking 205kB will be used.
Get:1 http://mirror.aarnet.edu.au stable/non-free netpbm-nonfree 1:19940301.1-8 [47.5kB]
Fetched 47.5kB in 1s (26.4kB/s)
Selecting previously deselected package netpbm-nonfree.
(Reading database ... 27811 files and directories currently installed.)
Unpacking netpbm-nonfree (from .../netpbm-nonfree_1%3a19940301.1-8_i386.deb) ...Setting up netpbm-nonfre

#
```

C.3 Install GNU plotutils Package oto Debian Box at Work

This provides the programs `graph` and `plot`.

```
# apt-get install plotutils
```

```
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  plotutils
0 packages upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 2051kB of archives. After unpacking 5622kB will be used.
Get:1 http://mirror.aarnet.edu.au stable/main plotutils 2.3-1 [2051kB]
Fetched 2051kB in 5m51s (5830B/s)
Selecting previously deselected package plotutils.
(Reading database ... 27832 files and directories currently installed.)
Unpacking plotutils (from .../plotutils_2.3-1_i386.deb) ...
Setting up plotutils (2.3-1) ...

#
```

C.4 Create a New User for the Kinetics Experiment

From the man page for adduser:

> "adduser and addgroup add users and groups to the system according to command line options and configuration infor mation in /etc/adduser.conf. They are friendlier front ends to the useradd and groupadd programs, choosing Debian policy conformant UID and GID values, creating a home directory with skeletal configuration, running a custom script, and other features."

```
# adduser kinetics
Adding user kinetics...
Adding new group kinetics (1003).
Adding new user kinetics (1003) with group kinetics.
Creating home directory /home/kinetics.
Copying files from /etc/skel
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for kinetics
Enter the new value, or press return for the default
Full Name []: Kinetics Experiment
Room Number []: B4114a
Work Phone []: 9514 1724
Home Phone []:
Other []:
Is the information correct? [y/n] y
#

# mkdir public_html
# mkdir cgi-bin
```

C.5 Security

During semester, when I won't need to edit any files do the following for security:

Change the entry in `/etc/passwd` from

`kinetics:x:1003:100::/home/kinetics:/bin/bash`

to

`kinetics:x:1003:100::/home/kinetics:/bin/FALSE`

References

[1] Literate Programming Using Noweb, A. Johnson & B. Johnson, *Linux Journal*, October 1997, Issue 42

[2] "Monte Carlo Experiments in Chemical Kinetics", Author unknown.