Using MATLAB to Generate HTML

R. S. SCHESTOWITZ*
IMAGING SCIENCE AND BIOMEDICAL ENGINEERING
UNIVERSITY OF MANCHESTER
UNITED KINGDOM

*roy.schestowitz@isbe.man.ac.uk

About this Document

I decided to share my experience outputting my MATLAB experiments onto public HTML files. Autonomous experimentation becomes very trivial in this way. The great benefits can possibly be understood by viewing the example on the next few slides.

Disclaimer

This document was written in haste. Please report typos and mistakes when found.

Motivation

- Many experiments performed day by day
- Input variables change
- Output data disappears
- Manual work needed for useful documentation
- Experiment reconstruction is *hard*

The Solution

1. Save all input values automatically

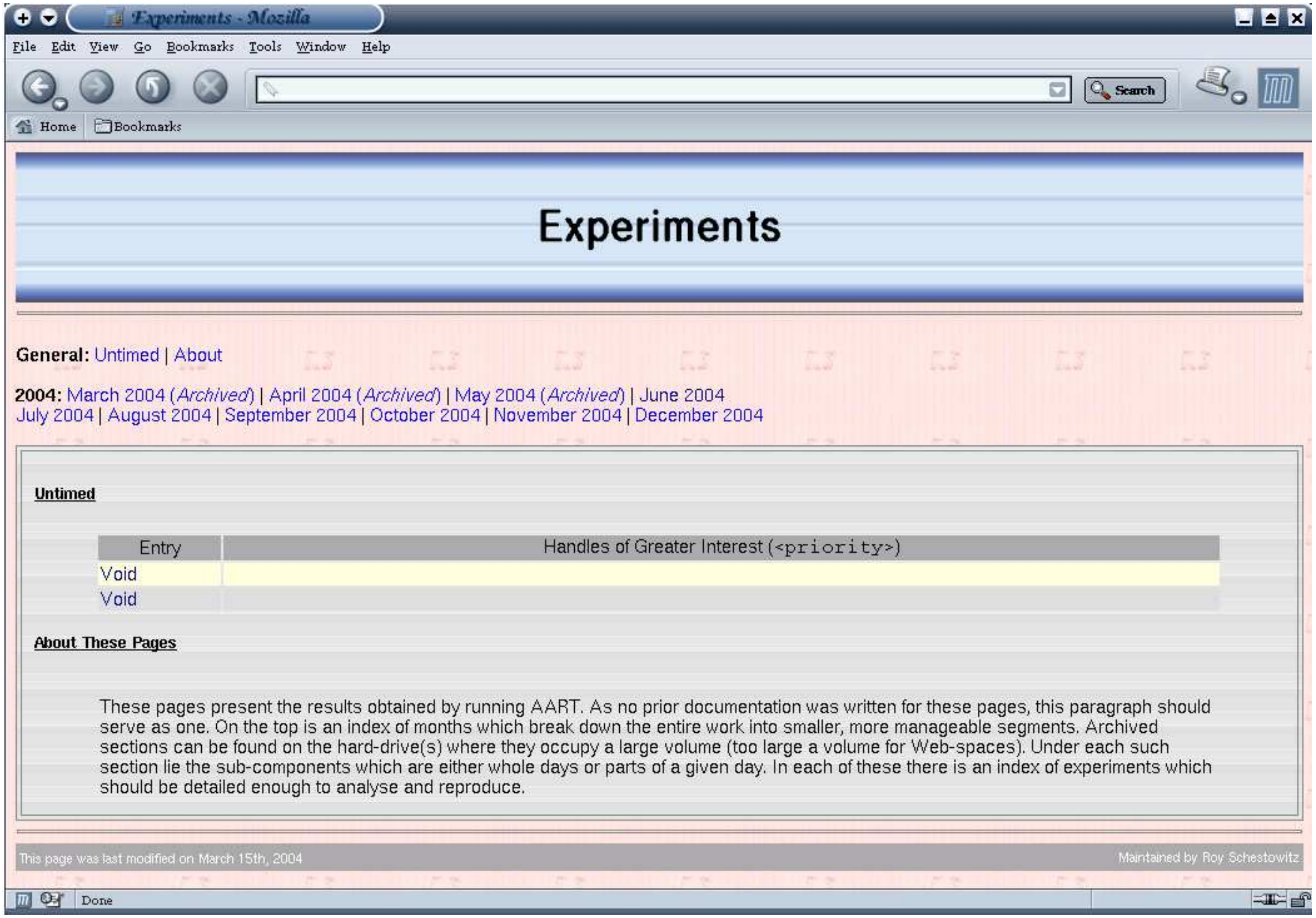
2. Save all figures automatically

3. Save all videos automatically

4. Record data for context, e.g. date, version, description

5. Collate (1)-(4) in a coherent interlinked manner

Example - Top Level



Experiments

General: Untimed | About

2004: March 2004 (*Archived*) | April 2004 (*Archived*) | May 2004 (*Archived*) | June 2004
July 2004 | August 2004 | September 2004 | October 2004 | November 2004 | December 2004

Untimed

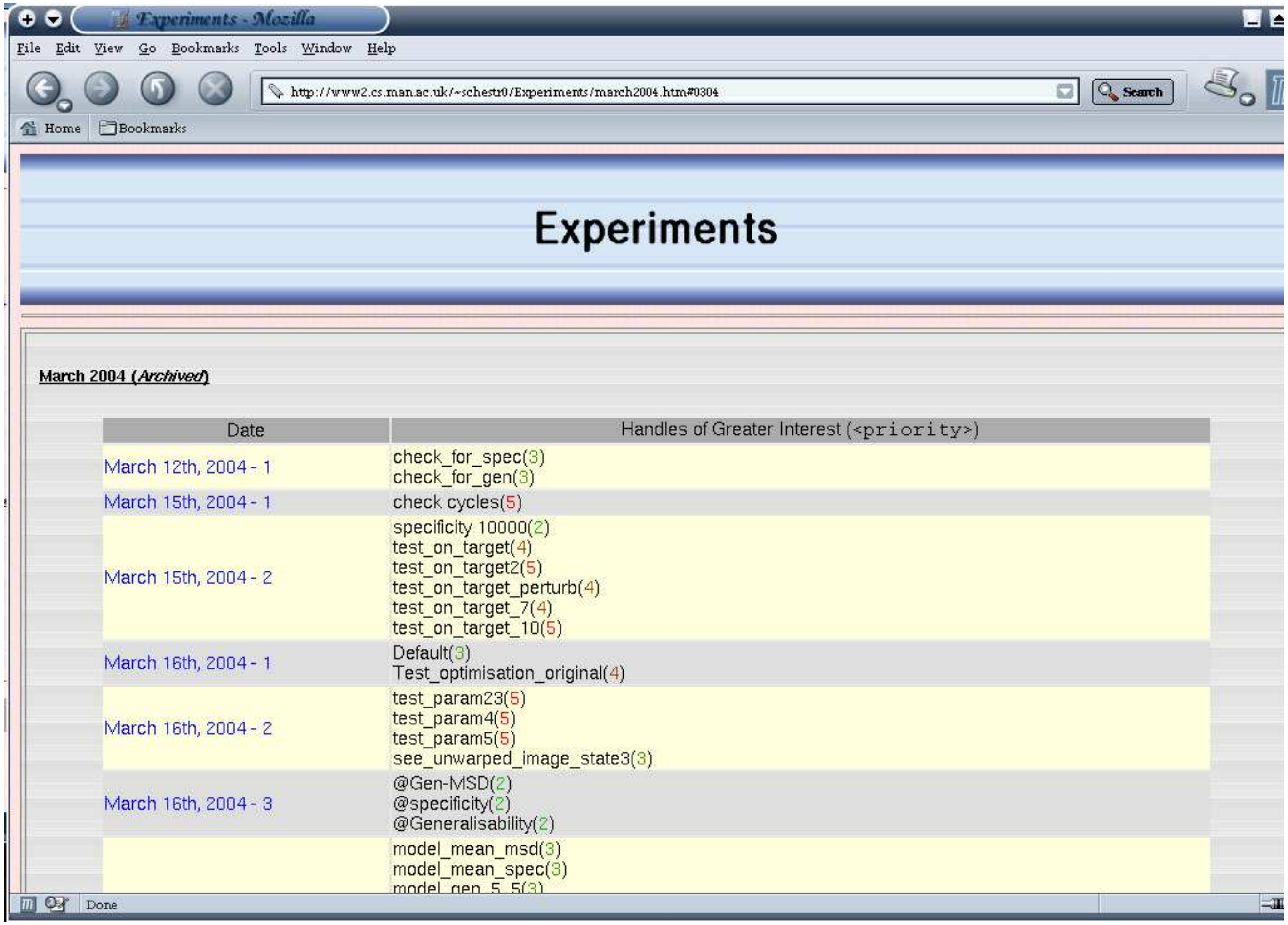
Entry	Handles of Greater Interest (`<priority>`)
Void	
Void	

About These Pages

These pages present the results obtained by running AART. As no prior documentation was written for these pages, this paragraph should serve as one. On the top is an index of months which break down the entire work into smaller, more manageable segments. Archived sections can be found on the hard-drive(s) where they occupy a large volume (too large a volume for Web-spaces). Under each such section lie the sub-components which are either whole days or parts of a given day. In each of these there is an index of experiments which should be detailed enough to analyse and reproduce.

This page was last modified on March 15th, 2004 Maintained by Roy Schestowitz

Example - Month Level


Experiments - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://www2.cs.man.ac.uk/~schestr0/Experiments/march2004.htm#0304
Search
Home
Bookmarks
Done

Experiments

March 2004 (*Archived*)

Date	Handles of Greater Interest (`<priority>`)

March 12th, 2004 - 1	check_for_spec(3)
	check_for_gen(3)

March 15th, 2004 - 1	check cycles(5)

March 15th, 2004 - 2	specificity 10000(2)
	test_on_target(4)
	test_on_target2(5)
	test_on_target_perturb(4)
	test_on_target_7(4)
	test_on_target_10(5)

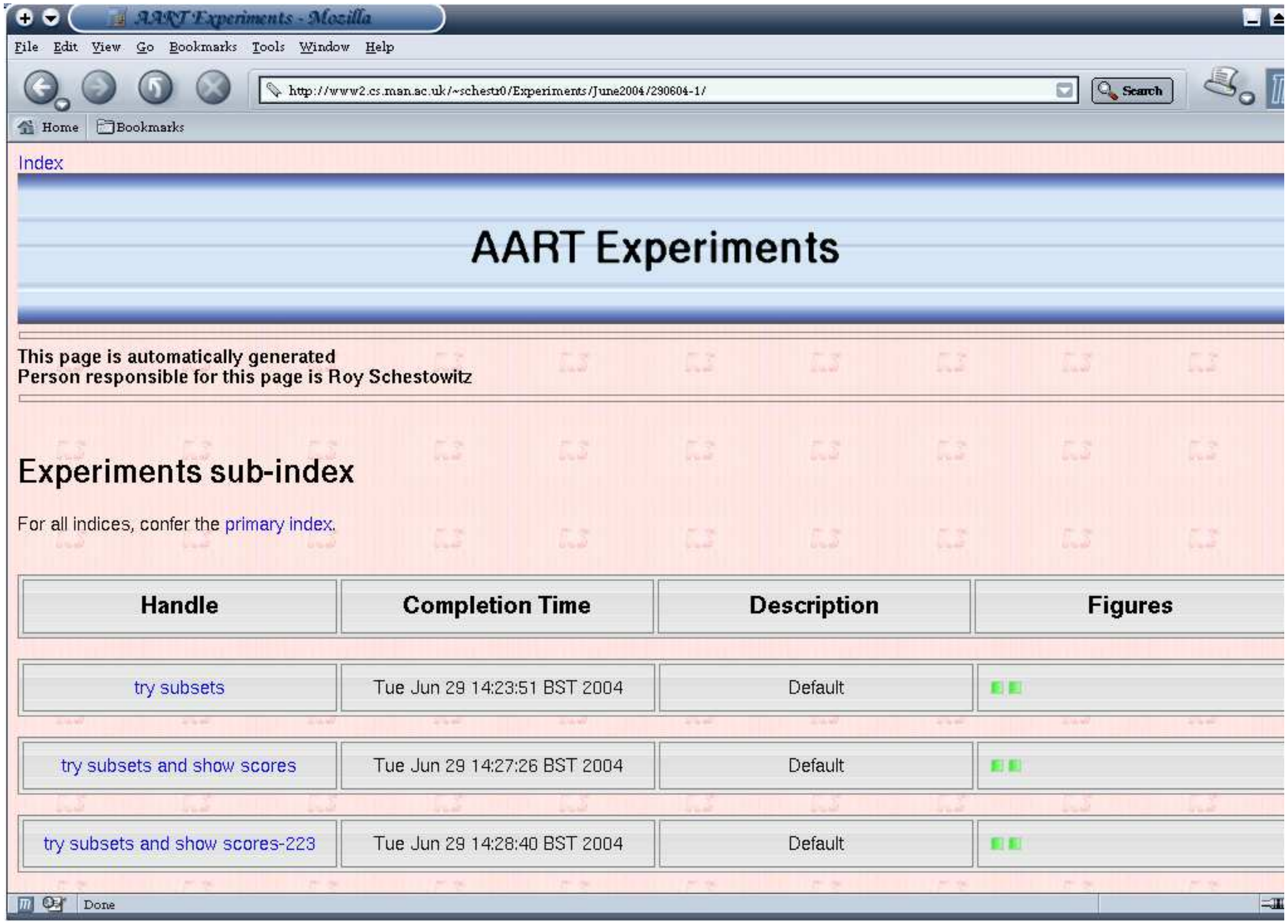
March 16th, 2004 - 1	Default(3)
	Test_optimisation_original(4)

March 16th, 2004 - 2	test_param23(5)
	test_param4(5)
	test_param5(5)
	see_unwarped_image_state3(3)

March 16th, 2004 - 3	@Gen-MSD(2) @specificity(2) @Generalisability(2)

	model_mean_msd(3)
	model_mean_spec(3)
	model_gen_5_5(3)

Example - Day Level



AART Experiments - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://www2.cs.man.ac.uk/~schestr0/Experiments/June2004/290604-1/
Search
Home
Bookmarks
Done

Index

AART Experiments

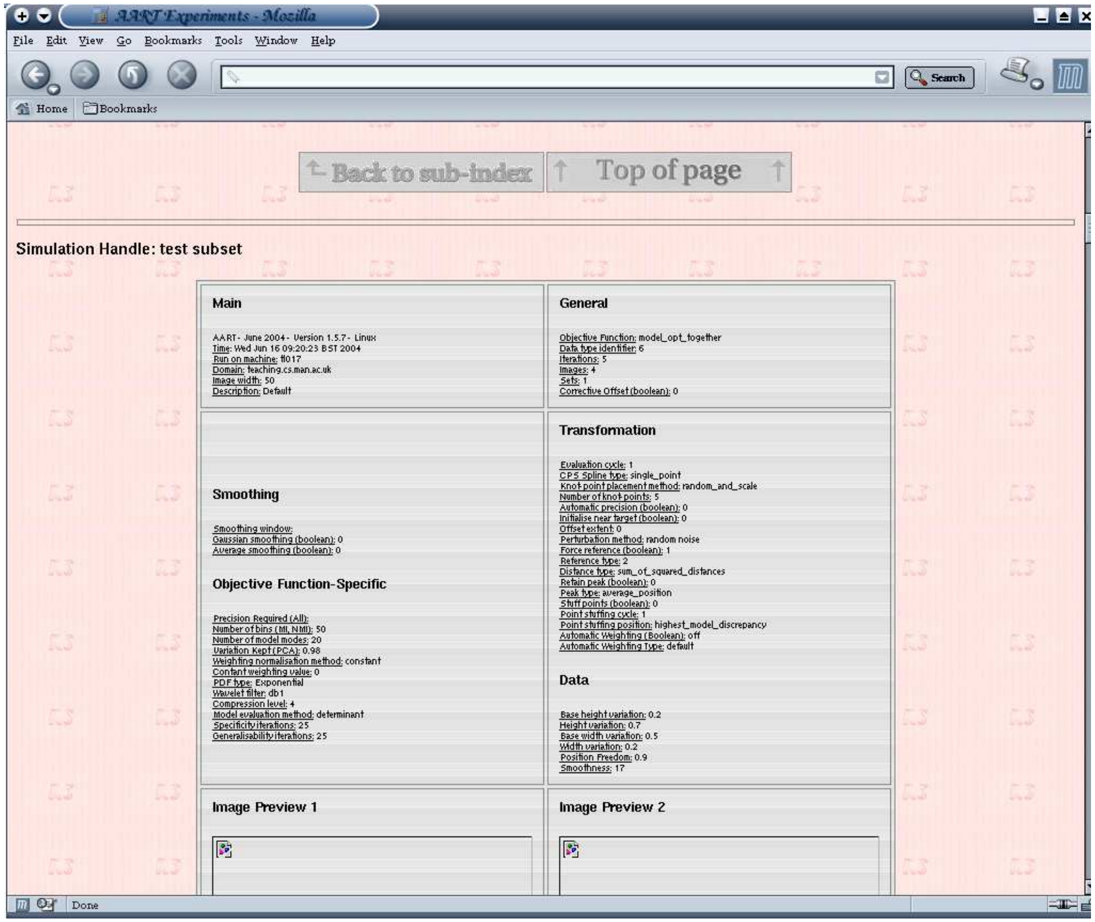
This page is automatically generated
Person responsible for this page is Roy Schestowitz

Experiments sub-index

For all indices, confer the primary index.

Handle	Completion Time	Description	Figures
try subsets	Tue Jun 29 14:23:51 BST 2004	Default	🟩 🟩
try subsets and show scores	Tue Jun 29 14:27:26 BST 2004	Default	🟩 🟩
try subsets and show scores-223	Tue Jun 29 14:28:40 BST 2004	Default	🟩 🟩

Example - Lower Level


AART Experiments - Mozilla
File Edit View Go Bookmarks Tools Window Help
Search
Home
Bookmarks
Back to sub-index
Top of page
Done

Simulation Handle: test subset

Main

AART - June 2004 - Version 1.5.7 - Linux
Time: Wed Jun 16 09:20:23 BST 2004
Run on machine: tl017
Domain: teaching.cs.man.ac.uk
Image width: 50
Description: Default

General

Objective Function: model_opt_together
Data type identifier: 6
Iterations: 5
Images: 4
Sets: 1
Corrective Offset (boolean): 0

Smoothing

Smoothing window:
Gaussian smoothing (boolean): 0
Average smoothing (boolean): 0

Objective Function-Specific

Precision Required (All):
Number of bins (MI, NMI): 50
Number of model modes: 20
Variation Kept (PCA): 0.98
Weighting normalisation method: constant
Contant weighting value: 0
PDF type: Exponential
Wavelet filter: db1
Compression level: 4
Model evaluation method: determinant
Specificity iterations: 25
Generalisability iterations: 25

Transformation

Evaluation cycle: 1
CPS Spline type: single_point
Knot point placement method: random_and_scale
Number of knot points: 5
Automatic precision (boolean): 0
Initialise near target (boolean): 0
Offset extent: 0
Perturbation method: random noise
Force reference (boolean): 1
Reference type: 2
Distance type: sum_of_squared_distances
Retain peak (boolean): 0
Peak type: average_position
Stuff points (boolean): 0
Point stuffing cycle: 1
Point stuffing position: highest_model_discrepancy
Automatic Weighting (Boolean): off
Automatic Weighting Type: default

Data

Base height variation: 0.2
Height variation: 0.7
Base width variation: 0.5
Width variation: 0.2
Position Freedom: 0.9
Smoothness: 17

Image Preview 1

Image Preview 2

Initial Implementation Steps

1. Recording Inputs
2. Recording Output (Images/Videos)
3. Linking

Step 1: Recording Inputs

A good starting point would be to print out all outputs to the user, e.g.:

```
disp(’|== Inputs =================================’);
disp([’| Input String: ’, some_string)]);
disp([’| Input Boolean: ’, num2str(some_boolean)]);
disp([’| Input Number: ’, num2str(some_value)]);
disp(’|==============================================’);
```

Step 1: Recording Inputs Ctd.

We now wish to do the same, but record the text in a file so that it remains accessible afterwards.

```
fid = fopen(’log.htm’,’a’);
fprintf(fid, [’<H2>Inputs</H2>’]);
fprintf(fid, [’\n<BR><U>Input String:</U> ’, ...
  some_string]);
fprintf(fid, [’\n<BR><U>Input Boolean:</U> ’, ...
  num2str(some_boolean)]);
fprintf(fid, [’\n<BR><U>Input Number:</U> ’, ...
  num2str(some_value)]);
fclose(fid);
```

Step 2: Recording Outputs

What is important is to keep track of filenames and be systematic about it. We begin by setting a counter.

```
figure_handle_number = 1;
   % A number to be incremented every time a figure is
   % saved. It manages the number of figures that
   % are generated.
```

Step 2: Recording Outputs Ctd.

Whenever a figure is generated, it needs to be saved as follows:

```
figure(current_figure_handle);
plot(data);
saveas(current_figure_handle, ...
[[current_experiment_handle],’-’, ...
num2str(figure_handle_number)], ’jpg’);
figure_handle_number = figure_handle_number + 1;
```

Step 2: Recording Outputs Ctd.

Q: What about videos?

They can be saved as AVI files in a uniform way and later be linked to.

Q: What about data files?

Just save them and remember the filenames.

```
save([[current_experiment_handle],['.mat']] ...
  , 'data1','data2');
```

Step 3: Linking

Possibly the most important and difficult steps. The current directory now has a collection of files: MATLAB data files, JPEG files, HTML files and AVI files. We need to group them together in a meaningful way.

How can they be grouped? Use HTML. What follows is some sample code.

Low Level linkage - Images

```
fprintf(fid, ’\n<TABLE><TR><TD><H2>Image Preview 1...
  </H2><BR><A HREF="’);
fprintf(fid, [[handle], ’-1.jpg"><IMG HEIGHT=200 ...
  WIDTH=300 BORDER=0 SRC="’]);
fprintf(fid, [[handle], ’-1.jpg"> ...
  </A></TR></TABLE>’]);
```

Low Level linkage - Videos

```
fprintf(fid, ’\n<TABLE><TR><TD><A HREF="’]);
fprintf(fid, [[handle],’-1.avi">Video #1 ...
  </A></TD></TR></TABLE>’]);
```

Low Level linkage - Data and Statistics

```
fprintf(fid, ’\n<TABLE><TR><TD><A HREF="’);
fprintf(fid, [[handle],’.mat">Input Data ...
</A></TD><TD><A HREF="’]);
fprintf(fid, [[handle],’.htm">Statistics ...
</A></TD></TR>’]);
fprintf(fid, ’\n</TABLE>’);
```

Page Headers

Since the file is opened in append mode (’a’), it will record all data sequentially and create a large file without a title. Use the following to add a header only at the start.

```
if (strcmp(which(’log.htm’), ”)),
  add_headers = 1;
else
  add_headers = 0;
end
fid = fopen(’log.htm’,’a’);
if (add_headers == 1),
  add_html_headers(fid);
end
```

Header Example

```
fprintf(fid, ['<HTML><HEAD><TITLE> ...
  My Experiments</TITLE>\n']);
fprintf(fid, ['<link rel="stylesheet" ...
  href="exp.css" type="text/css">\n']);
fprintf(fid, ['</head>\n']);
fprintf(fid, ['<BODY ...
  background="../../bg.gif">\n']);
fprintf(fid, ['<A NAME="top" ...
  HREF="index.htm">Index</A>\n']);
fprintf(fid, ['<H1>My Experiments ...
  </H1>\n']);
fprintf(fid, ['<HR SIZE=5>\n']);
```

A Word on Indexing

At some stage it would be sensible to create an index file dynamically. This way, all experiments are put in a central page which allows a broader scope.

```
fid = fopen('index.htm','a');
fprintf(fid, ['\n<P><CENTER> ...
  <TABLE WIDTH=100%% BORDER=BOX CELLPADDING=10 ...
  CELLSPACING=3 BACKGROUND="bg.gif"> ...
  <TD ALIGN=CENTER WIDTH=25%%><A ...
  HREF="log.htm#', [handle], '">', ...
  [handle], '</A></TD><TD ALIGN=CENTER ...
  WIDTH=25%%>']);
```

```
fprintf(fid, [’\n’, [description], ...
  ’</TD><TD 25%%> ’]);
fclose(fid);
```

Final Word

- Start with simple text file outputs.
- Build up towards HTML formatted files.
- Link to external outputs, e.g. pictures.
- Create index to all indices to progressively build a useful hierarchy.