Center for Simulation of Advanced Rockets

University of Illinois at Urbana-Champaign

CSAR Computer Code Documentation Guide

Center for Simulation of Advanced Rockets
University of Illinois at Urbana-Champaign
2260 Digital Computer Laboratory
Urbana, IL

Title:	CSAR Computer Code Documentation Guide
Author:	Documentation Committee
Subject:	Guide for producing basic documentation for computer codes developed directly in support of the Center for Simulation of Advanced Rockets.
Revision:	2
Revision History	Revision 0: Initial Release of Guide. March 2001 Revision 1: Reformatted ; December 2001 Revision 2: Comments from VAST committee incorporated. Code comments section on Local variables changed. Added comment on limiting use of local variables. Added notes in user's guide description to document required utility codes. Introduction shortened and reformatted. Title changed from Procedure to Guide. December 2002
Effective Date:	12/06/2002

1.0 Introduction

The purpose of this document is to provide guidance to code authors on the documentation needed for all computer programs that are produced in support of the Center for Simulation of Advanced Rockets.

2.0 Guide

This procedure defines three types of documentation that are to be produced for each computer code or distinct module:

- Source Code Comments.
- User's Guide.
- Developer's Guide.

Each of these will be described in the following sections. The outline of each document should be followed as closely as possible to assist in review and use of the information. If the outline is not followed, then the content shall still be included in the documentation produced.

2.1 Source Code Comments

Source code comments are to be included in every source file, notwithstanding the language used. Comments should be included in source files, make files, UNIX scripts, or any other files that result in executable code or are used to produce that code. This includes any utility codes used in production of input files for major codes.

The paper documents described in sections 2.2 and 2.3 are meant to supplement the generous comments that should be in each subroutine of a module. The comments are absolutely essential (even for the original developer) to code readability, maintenance, and correctness. The separate written documents provide the "big picture," including figures and equations that are not possible to include in comments.

Each subroutine/procedure/method shall have a comment header that describes:

- Author, creation date, modifications
- What does the routine do?
- What methods/algorithms are used (references)?
- What are the input and output parameters/arguments.
- What are the local variables that are important or difficult to understand from context?

In the body of the routine, there should be at least a brief comment before each task/step the subroutine performs. More is better up to a point. Familiarity with the programming language and a background in the discipline (e.g., fluids) should be assumed.

2.2 User's Guide

The following annotated outline describes the suggested format and required content of the Users Guide that is to be produced for each code.

1. Purpose and Methods

- Briefly describe what the code does, i.e., what problems it solves or what services it provides. (e.g., 3-D Navier-Stokes solver.)
- Briefly describe the algorithm, including important assumptions or limitations. (e.g., Method of Jameson, 1985, AIAA-2341. Second order TVD scheme and 4th order Kunge-Rutta explicit time integration. Valid for Reynolds numbers from 10 to 10^5.)
- List the main features or capabilities that make the code useful. (e.g., MPI parallel, multiblock, LES turbulence model of Moser, et al., 2001 AIAA-0012, mesh adaptivity).
- More detailed descriptions should be provided in the Developer's Guide under "Algorithm and Features."

2. Building and Running

- Tell how to compile and run the code. Typically a makefile is supplied, which can be modified for a number of platforms. Describe all command-line switches and/or parameters to be used.
- Describe the directories used, where the source, executable, and (sample) input data are located, etc., so that a user unfamiliar with the code could compile and run it without looking at the code. Describe any utility codes needed.

3. Input and Output (User Interface)

- Provide a brief description of the purpose of all input parameters, input arguments, or input data used by the application.
- Also describe all output arguments or data. If files are written, what is the format? Are there tools to analyze/view the output? What are they and how are they used?
- If the application is part of the software integration framework, a detailed description of the API (library subroutine calls) should go here.

- Describe any utility codes needed to produce the input, and describe the use of these codes.

4. Examples and Test Problems

This section could be supplemented by a README file in each of a set of test problem directories. It is included in the User's Guide because the test problems can help illustrate how the code is used as well as verify that the code is working properly on a new platform. This section can include equations or figures that cannot easily be given in a simple text file.

For each test problem, provide the following information:

- What specific problem is being solved?
- Why is this problem being solved? What aspects of the code does it exercise?
- What is the solution? Give a reference or provide a data file for comparison with the code's results.
- What degree of difference between the "true" solution and the numerical results is acceptable?

The test directories should contain any input files needed to run the tests, as well as a directory with results from at least the latest released version of the code. They will be stored in the CVS repository with the application.

2.3 Developers Guide

The following annotated outline describes the suggested format and required content of the Developers Guide that is to be produced for each code.

1. Algorithm and Features

- Describe the algorithm and the capabilities of the code in more detail than in the User's Guide. You may refer to method papers here – the level of detail typical of a method paper would be ideal.

2. Organization

- Provide an overview of the structure of the code, e.g., a call graph.
- Briefly describe the control flow and the function of the important subroutines or classes for a developer who is new to the code. Comments in minor routines should complete this functional description.

3. Data Structures

- Give the rationale for the important data structures in the code, e.g., why did you define the arrays and data types as they appear in the code -- was it for efficiency or to make it easier to extend the code later?

4. Interface

This section is intended to help a new developer couple a physics module with the GENx system code. If the application is part of the software integration framework, this section can be skipped.

If the application is a physics module, tell how the code fits into the GENx system code.

- What data does it need to receive from other modules, and what data is it intended to provide?
- In what routines does this happen?
- What arrays are copied into buffers?
- How does the code know what to do when called by the orchestration routine, i.e., how is it controlled from the outside?
- What global data does it access (if any)? Is the use of global data minimized?

5. Implementation Details

This section addresses the numerical aspects of the algorithm. It should include information such as:

- Assumptions and limitations -- approximations and regimes of validity
- Treatment of any Boundary Conditions -- this is crucial! Explain how the boundary conditions are enforced, e.g., numerically how do you enforce $dP/dn = 0$?