REDUCE interface to the CUBA integration library

Kostas N. Oikonomou
AT&T Labs Research, Middletown, NJ, U.S.A.
ko@research.att.com

January 31, 2015

1 Introduction

The `cuba` package is an interface between REDUCE (CSL) and the CUBA library for multi-dimensional numerical integration. The library can be found at http://www.feynarts.de/cuba and offers a choice of four integration methods: `Vegas`, `Suave`, `Divonne`, and `Cuhre`. The first three are Monte Carlo-based and the fourth is deterministic. It is recommended to read the CUBA manual, and, optionally, to look at the other documentation provided on the site.

Here is some basic information on the algorithms, copied from the CUBA web site.

ROUTINE	BASIC INTEGRATION METHOD	ALGORITHM	VARIANCE REDUCTION
Vegas	Sobol quasi-random sample, *or* Mersenne Twister pseudo-random sample, *or* Ranlux pseudo-random sample	Monte Carlo Monte Carlo Monte Carlo	importance sampling
Suave	Sobol quasi-random sample, *or* Mersenne Twister pseudo-random sample, *or* Ranlux pseudo-random sample	Monte Carlo Monte Carlo Monte Carlo	globally-adaptive subdivision + importance sampling
Divonne	Korobov quasi-random sample, *or* Sobol quasi-random sample, *or* Mersenne Twister pseudo-random sample, *or* Ranlux pseudo-random sample, *or* cubature rules	Monte Carlo Monte Carlo Monte Carlo Monte Carlo deterministic	stratified sampling, aided by methods from numerical optimization
Cuhre	cubature rules	deterministic	globally-adaptive subdivision

`Vegas` is the simplest of the four. It uses importance sampling for variance reduction, but is only in some cases competitive in terms of the number of samples needed to reach a prescribed accuracy. Nevertheless, it has a few improvements over the original algorithm and comes in handy for cross-checking the results of other methods.

`Suave` is a new algorithm which combines the advantages of two popular methods: importance sampling as done by `Vegas` and subregion sampling in a manner similar to `Miser`. By dividing into subregions, `Suave` manages to a certain extent to get around `Vegas`' difficulty to adapt its weight function to structures not aligned with the coordinate axes.

`Divonne` is a further development of the CERNLIB routine D151. `Divonne` works by stratified sampling, where the partitioning of the integration region is aided by methods from numerical optimization. A number of improvements have been added to this algorithm, the most significant being the possibility to supply knowledge about the integrand. Narrow peaks in particular are difficult to find without sampling very many points, especially in high dimensions. Often the exact or approximate location of such peaks is known from analytic considerations, however, and with such hints the desired accuracy can be reached with far fewer points.

`Cuhre` employs a cubature rule for subregion estimation in a globally adaptive subdivision scheme. It is hence a deterministic, not a Monte Carlo method. In each iteration, the subregion with the largest error is halved along the axis where the integrand has the largest fourth difference. `Cuhre` is quite powerful in moderate dimensions, and is usually the only viable method to obtain high precision, say relative accuracies much below 10^{-3}.

2 The Reduce package

The `cuba` package evaluates integrals *only* over hyper-rectangles[1]. As an example of what can be done in REDUCE, say f is a function $\mathbb{R}^3 \to \mathbb{R}$ and we want to compute

$$\int_{a_1}^{b_1}\int_{a_2}^{b_2}\int_{a_3}^{b_3} f(x_1,x_2,x_3)\,dx_1\,dx_2\,dx_3$$

using the `Vegas` algorithm, one of the choices provided by CUBA. This is done by saying

```
load_package cuba;
on rounded;
cuba_int(f,{{a1,b1},{a2,b2},{a3,b3}},Vegas);
```

Notes

- If you have `on lower`, then `Vegas` above has to be substituted by `!Vegas`.
- Although quite a bit of effort has gone into making the package work even when not in rounded mode, it is probably best to have `on rounded`.

[1] CUBA itself evaluates all integrals over the unit hypercube, but the REDUCE interface provides a small extension, allowing the user to integrate over an arbitrary hyper-rectangle.

The REDUCE function `f` defining the integrand is assumed to take a 3-element *list* x as input and return the value $f(x)$ of the integrand at the point $x \in \mathbb{R}^3$. If so, the above call to `cuba_int(...)` will return a list of the form

{`value, error, probability, number of regions, number of evaluations, status`}

where `value` is the value of the integral, `error` is an indication of the probable error, and `status` indicates whether the algorithm terminated successfully or not. Consult the CUBA manual for the other quantities.

3 Installation

At present the REDUCE parts of this package can be built only using the CSL version of REDUCE, but in that context get compiled automatically as part of the full standard system. However the code for CUBA and the C-coded interface between it and REDUCE have to be built by hand, and currently this works when all the REDUCE sources have been installed and REDUCE is built from scratch.

In that case you should identify the directory `packages/foreign/cuba` in the REDUCE source tree and select it as current. Ensure that the command `wget` is available on your platform and then you can go `make` to fetch CUBA from its home site, compile it, and create the dynamic library that forms a link between CUBA and REDUCE.

This should work on any sufficiently modern Unix-like system, including either the 32- or 64-bit version of Cygwin. The term "modern" here refers to Linux systems using releases no older then the very end of 2011: any such will probably provide a version of the gcc C compiler (i.e. one from 4.6.x onwards) sufficient for CUBA. This corresponds to Ubuntu from release 11.10 onwards or Fedora from about version 15.

To use the `cuba` package on Windows you must run a Cygwin version of CSL Reduce, not a native Windows one. That means that if you want the benefit of a GUI you must have an X server running and the environment variable DISPLAY set up for it. Passing the command-line flag `--cygwin` to the CSL version of REDUCE should cause a suitable version of the system to be loaded, and this probably needs to be done from the command line of a Cygwin terminal. This limitation is because the main CUBA library does not support native Windows.

Anybody with either and older version of an operating system or one other than (Free)BSD, OSX, Linux or Cygwin may need to identify a C compiler that can handle CUBA (any that support enough of the features of the 2011 C standard should suffice) and edit the Makefile to set the C compiler and any flags or options that it needs. Slightly bigger alterations will be needed if the linking command that makes the dynamic interface library needs changing.

4 The interface

Currently, the interface provides the functions listed in Table 1. The table gives minimal explanations, consult the CUBA manual for details.

`cuba_gen_par(name,value)`	Set the generally-applicable parameter *name* (a string) to *value*
`cuba_vegas_par(name,value)`	Set a Vegas-specific parameter
`cuba_suave_par(name,value)`	Set a Suave-specific parameter
`cuba_divonne_par(name,value)`	Set a Divonne-specific parameter
`cuba_cuhre_par(name,value)`	Set a Cuhre-specific parameter
`cuba_verbosity(v)`	For $v = 0, 1, 2$ `cuba_int` will provide more informative output
`cuba_set_flags_bit(i)`	Set the ith bit of the global `flags`
`cuba_clear_flags_bit(i)`	Clear the ith bit of the global `flags`
`cuba_statefile(fname)`	file `fname` will be used for checkpointing a long-running integration
`cuba_int(f,`$\{\{a_1, b_1\}, \dots\}$`,alg)`	Integrate the REDUCE function f over the hyper-rectangle $\{a_1, b_1\} \times \cdots \times \{a_m, b_m\}$ using algorithm `alg`

Table 1: Functionality of the REDUCE interface to the CUBA library.

There are some features of CUBA that are not handled by this version of the interface: vector integrands, i.e. functions from $\mathbb{R}^n \to \mathbb{R}^m$ with $m > 1$, integration routines that can do more than 2^{32} evaluations, and some of the parallelization features.

5 Implementation of the `cuba` package

5.1 Structure

This is not of interest to most users, but the package consists of the following files[2]:

`redcuba.c`	Builds `libredcuba.so`, a "glue" library between the actual CUBA library `libcuba.a` and REDUCE/CSL
`C_call_CSL.h`	The "procedural" interface from C to CSL, used in the above
`cuba.red`	The module defining the CUBA package
`cuba_main.red`	The REDUCE module (symbolic procedures) implementing the interface
`alg_intf.red`	Utilities for interfacing between algebraic and symbolic modes
`cuba.tst`	A REDUCE test file.

5.2 Debugging

To debug the interface, there is a variable `DEBUG` in `redcuba.c`, normally set to 0. By setting it to 1 or 2 and re-making `libredcuba.so` the package will produce various debugging messages that should be useful.

Acknowledgments

Thanks to Arthur Norman for his invaluable support in navigating the intricacies of REDUCE, algebraic and symbolic mode, RLISP, Standard Lisp, CSL, etc.

[2] If the list of files and comments is confusing, refer to the Acknowledgments.