NEW SIMULATION CODE FOR SYNCHROTRON RADIATION BASED ON A REAL BEAM ORBIT

Tetsuo Abe*,
High Energy Accelerator Research Organization (KEK), 1-1 Oho, Tsukuba, Ibaraki 305-0801, Japan

Abstract

A computer code to simulate synchrotron-radiation power and spatial distributions has been developed based on the method by Abe and Yamamoto [1], where a real beam orbit is obtained by fitting measurements of beam-position monitors (BPMs) with some offset corrections for BPMs and quadrupole-magnet alignments. In this paper, the status and basic performance are presented. This code has been rewritten in `Fortran2003` so as to obtain expectable maximal speed-up by parallel computing, aiming at online alarm systems to take precautions against synchrotron-radiation damage, toward higher beam current accelerators.

INTRODUCTION

Synchrotron radiation (SR) is useful for beam monitoring / diagnostics in accelerator science, and for experimental studies in material science. However, it sometimes causes serious damage to accelerator or detector components of such as a high luminosity factory machine. In the previous work [1], a simulation method to protect the detector against the SR was developed, where the method is based on a real beam orbit obtained by fitting measurements of beam-position monitors (BPMs) with some offset corrections for BPMs and quadrupole-magnet alignments. The validity of the method was demonstrated for the gain-drop accident of Belle Silicon Vertex Detector [2]. Based on this method, an old version of a SR alarm software was written in `Fortran77`. However, the turnaround time would increase proportional to the number of magnets. Since there is almost no room for improvement on CPU clock frequencies, we need to introduce parallel computing in order to make the online software useful. For this purpose, the old code has been completely rewritten in `Fortran2003` in an object-oriented form, which is adequate to parallel computing.

PROGRAM COMPONENTS

The following modules have been newly written, and combined to form an online software with a graphical user interface (GUI).

Figure 1 shows the flowchart for the new code.

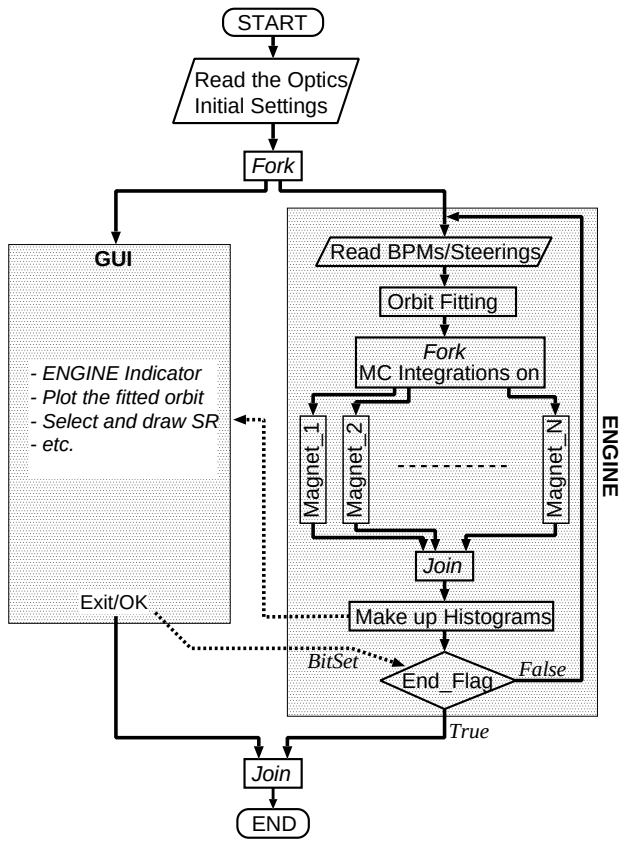

Figure 1: Flowchart for the new code. Dotted arrows indicate data transfer between the two main threads.

Numerical Integration Module

Since the dimension of integration to obtain SR power and spatial distributions is seven at maximum, including a simulation of a beam size, a Monte Carlo (MC) method is appropriate. In this module, the iterative importance-sampling method is employed as in `VEGAS` [3] and `BASES` [4]. The data structure is designed to have an array containing one thousand different functions at maximum, each of which has a distributed-memory in parallel computing. This module possesses one- or two-dimensional histogramming and a function to generate unweighted events of SR photons for subsequent analyses, e.g. detector simulations.

Function Minimization Module

Beam-orbit fitting is performed by minimizing a χ^2 function of a beam orbit and BPM measurements, using a quasi-Newton method with a variable metric, where

* tetsuo.abe@kek.jp

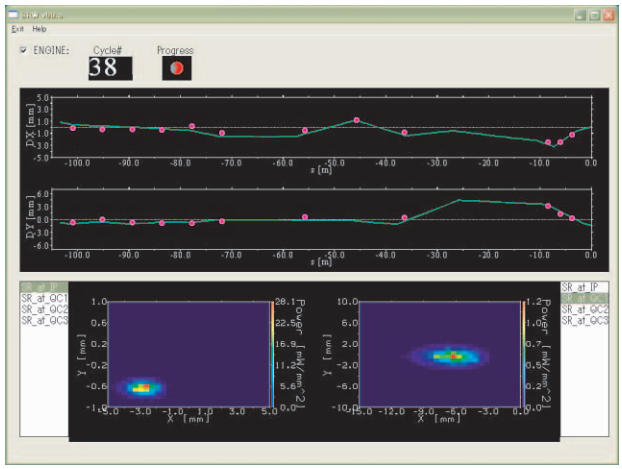

Figure 2: GUI (tentative version). Green lines indicate a fitted orbit for BPM measurements (magenta circles). Two SR spatial distributions are shown as examples. Only temporary offset corrections are applied here.

two matrix-updating formulae (DFP and BFGS) are used [5]. This method is basically the same as implemented in `MINUIT` [6] / `MIGRAD` routine, except for the special treatment when a Hessian matrix is not positive definite. Another difference is that an argument of the function to be minimized can be any user-defined structure (derived-type) because many and various types of parameters are needed for beam orbit calculations.

Parallelism

The code has been complied using `gfortran` from GNU Compiler Collection (GCC) [7] version 4.3. Since its version 4.2, GCC has officially supported `OpenMP` [8], an application program interface for multi-platform shared-memory parallel programming in `C/C++` and `Fortran` on all architectures. The parallel computing of the new code is performed by `OpenMP` implemented in `gfortran`.

GUI

For user inputs and graphical outputs, `DISLIN` [9] version 9.1 is used, which is a high-level plotting library for displaying data as curves, pie charts, contours, maps, etc, including a large variety of parameter setting routines called to create individually customized graphics. This software is available for not only several `C`, but also `Fortran 77/90/95` compilers, so that the interface to `DISLIN` is written also in `Fortran`.

Figure 2 is an example of snapshots of the GUI, showing a fitted orbit and SR spatial distributions.

PERFORMANCE

The first version of the new code has been just completed. The turnaround times were measured for the 100 m straight section of the KEKB [10] / HER upstream the interaction point, using a personal computer with an Intel Core Duo processor (2 GHz). The typical average turnaround time for this example is 14.2 seconds for "`OMP parallel num_threads(1)`", and 9.2 seconds for "`OMP parallel num_threads(2)`", excluding the time of orbit fitting, where the contribution from orbit fitting to the whole turnaround time is small. Actually, this MC integration module can provide almost two times faster computations with "`OMP parallel num_threads(2)`" compared to "`OMP parallel num_threads(1)`" if the integrands are almost the same. Therefore, some scheduling algorithm using "`OMP do schedule()`" clause is needed to make the turnaround time minimum.

In any case, we certainly reap the benefit of parallel computing.

FUTURE PROSPECTS

The first version includes only monitoring. The most important function, which is to take precautions against SR in order to protect accelerator components or detectors, is to be implemented soon.

REFERENCES

[1] T. Abe and H. Yamamoto, Phys. Rev. ST Accel. Beams **7**, 072802 (2004) [arXiv:physics/0404043].

[2] G. Alimonti *et al.* [BELLE Collaboration], Nucl. Instrum. Meth. A **453**, 71 (2000).

[3] G. P. Lepage, J. Comput. Phys. **27**, 192 (1978).

[4] S. Kawabata, Comput. Phys. Commun. **88**, 309 (1995).

[5] R. Fletcher, Comput. J. **13**, 317 (1970).

[6] F. James and M. Roos, Comput. Phys. Commun. **10**, 343 (1975).

[7] `http://gcc.gnu.org/`

[8] `http://www.openmp.org/drupal/`

[9] `http://www.mps.mpg.de/dislin/`

[10] S. Kurokawa, Nucl. Instrum. Meth. A **499**, 1 (2003).

