Generating Adjoint of Industrial Codes with Limited Memory

CHRISTÈLE FAURE
INRIA Sophia Antipolis, 2004 route des L Lucioles, BP 93, F-06902 Sophia-Antipolis Cedex, France

Received 13 December 1999; accepted in revised form 27 November 2000

Abstract. The challenge is to generate an adjoint code of Thyc-3D by means of the automatic differentiator Odyssee. This adjoint code must compute the gradient using at most three times more core memory than the original function. The other constraint is to do as few manual interventions as possible. The original code contains nearly 500 sub-programs (70000 lines, 1000 parameters) and calls black-box functions whose codes are not available. Those functions are only available in the form of a compiled library. We discuss various storage/retrieval and precomputation options. The results suggest further improvements and interactive features of automatic differentiation tools.

Key words: automatic differentiation, adjoint codes, memory limited, thermal hydraulic, two-phase flow, condenser, steam generator.

1. Introduction

Automatic Differentiation [1, 11, 12] is a set of techniques that are aimed at differentiating functions that are given as programs for their evaluation. The differentiated code computes the same values as the original function plus some derivative information. The method differs from finite differences in that the value of the derivatives are computed exactly (up to rounding errors) in a much more efficient way. This is particularly true when gradients (set of partial derivatives) are to be computed. The reverse mode of automatic differentiation (AD) that can be viewed as a method for generating adjoint codes automatically is considered here. The theoretical complexities of the reverse mode AD are studied in [5], this paper focuses on practical aspects of automatic adjoining.

Thyc-3D is an industrial 3D thermal hydraulic in bundles code developed at EDF-DER and associated to a 1D mockup named Thyc-1D. Thyc-1D is a one dimensional thermal hydraulic module for two-phase flow modeling. It consists of: three conservation equations for the two-phase mixture (mass, momentum and energy), one conservation equation for the vapor mass and one conservation equation for the relative liquid-vapor velocity between the two phases. Thyc-3D allows the study of various flows in bundles such as tube heat transfer exchangers and rob bundles like reactor cores. We do not give details on these softwares, but a complete description can be found in [2, 16]. We have first worked on the mockup to evaluate
the feasibility of automatic adjoining using Odyssée [8] and have compared the various adjoining strategies. In [4] we show that the ratio of the execution time of the adjoint code of Thyc-1D with respect to the original code is 7.1 which is acceptable. The ratio 9 in terms of memory requirement is acceptable on the 1D code, but not on the 3D code. Faure and Duval [7] show that the use of the optimal checkpointing technique makes the generated adjoint code more flexible. For example, if the chosen ratio in terms of memory is 3 (instead of 9), the ratio in terms of execution time is 7.5 (compared to 7.1). Using optimal loop checkpointing, the memory requirement is divided by 3 whereas the execution time is slightly augmented.

The use of the automatic differentiation tool Odyssée to compute some gradients of Thyc-3D is presented. The challenge is to differentiate the program with as few manual interventions as possible, in such a way that the resulting code needs only a moderate amount of core memory. Using automatic differentiation we are sure of the accuracy of the computed derivatives, only execution time and memory requirement are to be studied. For this study,* the end user (namely EDF-DER) imposed no constraint on the execution time, but limited the adjoint core memory to at most 3 times the original memory need.

Section 2 presents briefly the reverse mode from a theoretical point of view. The strategical choices used for automatic generation of the adjoint codes of Thyc-3D are explained in Section 3. The execution time and memory requirements of the generated adjoint codes are analysed in Section 4.

2. Reverse Mode Problems

The evaluation of derivatives in the opposite order to the computation of the original variables makes the reverse mode much more complicated than the direct mode. The straight-forward solution is to record every intermediate computation (named trajectory) during a first run of the original program and during a second phase to retrieve all these values before computing the elementary Jacobian matrices in the opposite order of the original execution. This basic strategy is optimal in terms of theoretical execution time but cannot be applied on many real world programs because of the large amount of storage required. A trade-off between storing the whole trajectory but never repeating even parts of the original calculation and storing parts of the trajectory but partially recomputing the original function is to be applied. When the differentiation of a multi-level program is concerned, a variety of strategies can be applied for generating the adjoint code. In [5] these strategies are described and compared in terms of time and memory complexities. Odyssée and Tamc [9] are the only AD tool that allow for automatic adjoining of industrial size programs. They are both based on source transformation techniques which makes compile time optimizations of the adjoint code possible. The operator overloading technique used within Adolc [13] makes compile time optimizations impossible. In

---

* This study was initiated and financed in part by EDF-DER.