Combustion and Flame 194 (2018) 37–51

Contents lists available at ScienceDirect

Combustion and Flame

journal homepage: www.elsevier.com/locate/combustflame

Fast approximations of exponential and logarithm functions combined with efficient storage/retrieval for combustion kinetics calculations

Federico Perini*, Rolf D. Reitz

University of Wisconsin–Madison 1500 Engineering Drive, Madison, WI 53706, United States of America

ARTICLE INFO

Article history:
Received 19 July 2017
Revised 2 October 2017
Accepted 12 April 2018

Keywords:
Exponential
Logarithm
Floating-point algebra
Table interpolation
chemical kinetics

ABSTRACT

We developed two approaches to speed up combustion chemistry simulations by reducing the amount of time spent computing exponentials, logarithms, and complex temperature-dependent kinetics functions that heavily rely on them. The evaluation of these functions is very accurate in 64-bit arithmetic, but also slow. Since these functions span several orders of magnitude in temperature space, some of this accuracy can be traded with greater solution speed, provided that the governing ordinary differential equation (ODE) solver still grants user-defined solution convergence properties. The first approach tackled the `exp()` and `log()` functions, and replaced them with fast approximations which perform bit and integer operations on the exponential-based IEEE-754 floating point number machine representation. The second approach addresses complex temperature-dependent kinetics functions via storage/retrieval. We developed a function-independent piecewise polynomial approximation method with the following features: it minimizes table storage requirements, it is not subject to ill-conditioning over the whole variable range, it is of arbitrarily high order $n > 0$, and is fully vectorized. Formulations for both approaches are presented; and their performance assessed against zero-dimensional reactor simulations of hydrocarbon fuel ignition delay, with reaction mechanisms ranging from 10 to 10^4 species. The results show that, when used concurrently, both methods allow global speed-ups of about one order of magnitude even with an already highly-optimized sparse analytical Jacobian solver. The methods also demonstrate that global error is within the integrator's requested accuracy, and that the solver's performance is slightly positively affected, i.e., a slight reduction in the number of timesteps per integration is seen.

© 2018 The Combustion Institute. Published by Elsevier Inc. All rights reserved.

1. Introduction

The computational cost associated with the solution of stiff Ordinary Differential Equations (ODEs) describing chemical kinetics is still one of the major factors limiting the usage of detailed chemistry in multidimensional combustion simulations [1]. In a chemically reactive gas-phase environment, conservation equations for the closed system's mass and energy appear as rates of change of species mass fractions Y_i and temperature T:

$$\frac{dY_i}{dt} = \dot{Y}_i = \frac{W_i}{\rho}\sum_{j=1}^{n_r}\left[\left(\nu''_{j,i} - \nu'_{j,i}\right)q_j\right],$$
$$\frac{dT}{dt} = \dot{T} = -\frac{1}{\bar{c}_v}\sum_{i=1}^{n_s}\frac{U_i\dot{Y}_i}{W_i}, \tag{1}$$

when the mixture of $i = 1, \ldots, n_s$ species M_i, is subject to a reaction mechanism, i.e., a network of $j = 1, \ldots, n_r$ chemical reactions:

$$\sum_{i=1}^{n_s}\nu'_{j,i}M_i \rightleftharpoons \sum_{k=1}^{n_s}\nu''_{j,k}M_k,\ \ j = 1, \ldots, n_r; \tag{2}$$

$\boldsymbol{\nu}'$ and $\boldsymbol{\nu}''$ are sparse matrices containing stoichiometric reaction coefficients of reactants and products respectively [2]. The system usually exhibits very stiff behavior because of both the exponential form of the reaction rates, and the strongly nonlinear coupling between species concentrations caused by the law of mass action, as witnessed by the species' mutual excitation rate [2]:

$$\frac{\partial\dot{Y}_i}{\partial Y_j} = \frac{W_i}{\rho}\sum_{k=1}^{n_r}\left\{\frac{\nu_{k,i}}{Y_j}\left[\nu'_{k,j}k_{f,k}\prod_{r=1}^{n_s}\left(\frac{\rho Y_r}{W_r}\right)^{\nu'_{k,r}} - \nu''_{k,j}k_{r,k}\prod_{s=1}^{n_s}\left(\frac{\rho Y_s}{W_s}\right)^{\nu''_{k,s}}\right]\right\}, \tag{3}$$

where ρ is the system's density, k_f and k_r the forward and reverse reaction rates, W the species' molecular weights. Because of its stiffness, time integration of the reactive system of Eq. (1) is usually performed as an independent ODE system even in multidimensional simulations, where an operator splitting scheme is

* Corresponding author.
E-mail address: perini@wisc.edu (F. Perini).

https://doi.org/10.1016/j.combustflame.2018.04.013
0010-2180/© 2018 The Combustion Institute. Published by Elsevier Inc. All rights reserved.

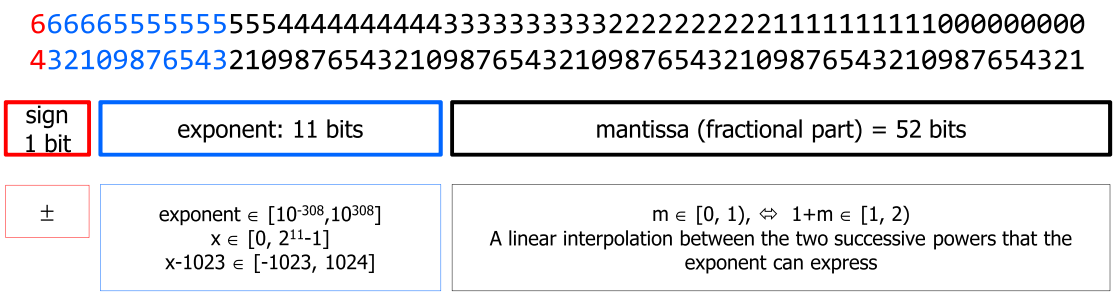

Fig. 1. IEEE-754 representation of a 64-bit (`double`) real number.

employed to relax the solver integration constraints towards the flow time scales (see [3]).

In recent years, several studies have addressed aspects of the chemical kinetics ODE system to increase its computational efficiency. Some researchers have focused on ODE solution methods for stiff systems [4–7] aimed at achieving time advancement of the solution with the least number of integrator timesteps. Perhaps the greatest CPU time savings have been achieved by adopting fine-tuned analytical formulations of the chemistry system and its Jacobian [2,8,9], coupled with sparse matrix algebra [10,11] to take advantage of the reaction mechanism's sparsity. Some recent efforts have also attempted to exploit graphics processing units (GPUs) to increase throughput of the kinetics calculations [12].

This study focuses on reducing the cost of evaluating reaction kinetics functions involving exponentials and logarithms. Two approaches are presented: first, several fast approximations of the `exp()` and `log()` functions exploiting the IEEE-754 floating-point number representation [13] were developed. Second, a storage/retrieval approach for costly temperature-dependent functions was introduced. A simulation matrix featuring 11 reaction mechanisms from 10 to 10^4 species, simulating ignition of fuel–air mixtures at conditions relevant to combustion devices was established, in order to assess the robustness, the accuracy and the speed of the proposed approaches. Combustion CFD simulations, including 2D and 3D cases, were validated as well. The results demonstrate significant speed-ups of almost an order of magnitude for the total CPU time even in presence of analytical Jacobian and sparse algebra; plus, a stabilizing effect of the smoothed function approximations on the ODE solvers' performance.

The major contributions of this work can be summarized as follows:

- A fast equally-spaced tabulation/polynomial interpolation approach for exponentially-varying functions which has limited storage needs, is continuous in both function and derivative evaluations up to an arbitrary order, is defined piecewise like a spline, but its accuracy is not affected by what happens far from the interpolation point.
- New, improved methods for fast evaluation of exponential and logarithm functions, that provide not only a continuous function, but also continuous derivatives. These methods improve on the fast exponential approach based on floating point representation manipulation methodology developed by Schraudolph [14] by using an arbitrary-order spline reconstruction of the mantissa, which compares favorably even to the most recent formulations [15].

2. Algorithm description

We developed methods to approximate the exponential and logarithm functions for 64-bit (`double precision`) floating-point numbers complying with the IEEE-745 standard [13]. This format represents a real number by subdividing the 64-bit space into three integer strings, as reported in Fig. 1:

$$r = (-1)^s 2^{x-b}(1+m), \tag{4}$$

where:

- s (1 bit) is the sign bit;
- x (11 bits) $\in [0, 2047]$ is an integer exponent, shifted by a fixed bias $b = 1023$, such that both negative and positive integer powers can be represented: $2^{x-b} \in [2^{-1023}, 2^{+1024}] \approx [10^{-308}, 10^{+308}]$;
- m (52 bits) is the *mantissa* or a fractional part: $m \in [0, 1)$, or $[1+m) \in [1, 2)$.

This model produces an exact represenation of any integer powers of 2, which have empty mantissa $m = 0$; the mantissa acts as a truncated linear interpolation between subsequent powers of two, hence allowing any real numbers r in the range to be represented within an accuracy of approximately 15 decimal digits.

2.1. Fast exponential function

Approximations of the exponential function exploiting the IEEE-754 standard were developed by Schraudolph [14] and later extended by other researchers [16]. These formulations target 32-bit numbers, and are not suitable for double precision integration of highly stiff problems. However, a recent paper by Malossi et al. [15] targeted 64-bit numbers and, while coefficients are not given, employs a McLaurin series expansion, and is included here for completeness. The exponential function is naturally defined as a series:

$$\exp(x) = e^x = \sum_{n=1}^{+\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots \tag{5}$$

and this feature is exploited by accurate exponential evaluation methods [17]. However, in [14] it was demonstrated that a fast approximation of the exponential can be achieved by just manipulating the IEEE-754 number representation of Eq. (4) using simple bit shift and integer algebra operations. First, the exponential operation is reduced to a power-of-two operation with a change of basis:

$$e^x = 2^{x/\log 2} = 2^y; \tag{6}$$

if number y is represented as an integer and a fractional part, $y = y_i + y_f$, then its machine representation fits its power-of-two exponentiation well:

$$2^y = 2^{y_i} 2^{y_f} = (-1)^s 2^{x-b}(1+m). \tag{7}$$

Sign $s = 0$ is always positive. The integer term 2^{y_i} can be computed by simply fitting a suitable integer $x = 1023 + y_i$ into the exponent part of the floating point representation, i.e., by left-shifting integer x by 52 positions, and casting it into the real number representation:

$$2^{y_i}\ (\texttt{real64}) \leftarrow 2^{52} \cdot (1023 + y_i)\ (\texttt{int64}); \tag{8}$$

Download English Version:

https://daneshyari.com/en/article/6593448

Download Persian Version:

https://daneshyari.com/article/6593448

Daneshyari.com