Applied Mathematics and Computation 320 (2018) 754–768

Contents lists available at ScienceDirect

Applied Mathematics and Computation

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

Nonlinear function inversion using *k*-vector

David Arnas[a,*], Daniele Mortari[b]

[a] Centro Universitario de la Defensa Zaragoza, Crta. Huesca s/n, 50090 aragoza, Spain
[b] Aerospace Engineering, Texas A&M University, College Station, TX 77843-3141, USA

ARTICLE INFO

Keywords:
Root finders
Nonlinear functions
Convergence acceleration
Computational efficiency

ABSTRACT

This work introduces a general numerical technique to invert one dimensional analytic or tabulated nonlinear functions in assigned ranges of interest. The proposed approach is based on an “optimal” version of the *k*-vector range searching, an ad-hoc modification devised for function inversion. The optimality consists of retrieving always the same number of data $(1, 2, \ldots)$ for a specified searching range to initiate the root solver. This provides flexibility to adapt the technique to a variety of root solvers (e.g., bisection, Newton, etc.), using a specified number of starting points. The proposed method allows to build an inverse function toolbox for a set of specified nonlinear functions. In particular, the method is suitable when intensive inversions of the same function are required. The inversion is extremely fast (almost instantaneous), but it requires a one-time preprocessing effort.

1. Introduction

Many problems in science involve computing the inverse of nonlinear functions with unknown analytical expressions [1]. Examples of these are: the Airy function [2,3] (in optics, fluid mechanics, elasticity, or quantum physics), the Bessel’s integrals [4,5] (in many scientific problems, as in quantum field, and condense matter theory problems), the Fresnel’s integrals [6,7] (in quantum mechanics and optics), the Dawson’s integrals [8,9] (in relative hydrodynamics and wave problems), and the error function [10,11]. A great number of numerical algorithms have been developed over the years to calculate these functions, while their inverses are computed by classic root finders and proper initial guesses. The key point here is: *how to obtain a very accurate initial guess as fast as possible*? The purpose of this paper is, indeed, the development of an algorithm providing (almost instantaneously) initial guesses with accuracy increasing with the memory available.

The most classic and common root finders are Newton–Raphson, regula-falsi, bisection, secant, and fixed-point. Recently, other general root finders were developed to increase performance, based on Adomian decomposition method [12,13] and modifications of the Newton–Raphson algorithm [14,15]. However, these methodologies only allow to find one root at a time in nonlinear equations, with increasing complexity when multiple roots are to be computed.

In this work, we introduce a general algorithm to invert one-dimensional functions, which is based on generating a first approximation to the root using the *k*-vector range searching methodology. Then, the final root is obtained by the application of another root finder such as Newton–Raphson or regula-falsi. The advantages of this methodology over the existing methods are that *it allows to find all the roots of the function at the same time* and that *it can be adapted to each problem in study*. On the other hand, its disadvantage is that it requires to perform an initial preprocessing for each function,

* Corresponding author.
E-mail address: darnas@unizar.es (D. Arnas).

https://doi.org/10.1016/j.amc.2017.10.009
0096-3003/© 2017 Elsevier Inc. All rights reserved.

making the method suitable when intensive inversions of the same function are required and/or when a toolbox of prescribed inverse functions is needed.

The k-vector technique has its origin in the spacecraft attitude estimation problem using star trackers, where the identification of the observed stars (Star-ID) is required in order to estimate the spacecraft orientation. This process must be done as fast as possible (to improve the performance of the control system) by an onboard computer with limited capabilities. To solve this problem, a fast range searching technique, called k-vector, was developed [16,17]. Currently, the k-vector is at the heart of *Pyramid* [18], the state-of-the-art of Star-ID algorithms, currently operating in several satellites.

The most important property of the k-vector is that, once the preprocessing effort is done for a given database, *the searching process becomes independent from the database size*, which makes the method especially suitable for large databases. Compared with the most common searching algorithm, the binary-search technique, the k-vector has a complexity of $\mathcal{O}(3)$, whilst the binary-search technique has a complexity of $\mathcal{O}(2\log_2 n)$, where n is the database size.

The k-vector is a general range searching technique, particularly suitable for retrieving data from static databases (such as star catalogs) [19]. However, this technique has recently been applied to solve a variety of different problems, such as sampling [20], interpolation, and estimation [21].

This work focuses on the application of the k-vector technique to fast function inversion, introducing a general methodology to invert any one-dimensional nonlinear function in specific ranges of interest. In particular, the technique allows to speed the convergence of other root finders, providing an initial root approximation, or for bracketing all the roots of a given function. This study shows how to solve particular problems the technique may deal with, such as singularities of the function, piecewise function definition, and functions with regions with very different derivative values. Moreover, and in order to improve the performance of the function inversion, an *optimal* k-vector technique is here introduced, which can be applied in combination with any root solver to obtain machine error precision.

The optimal k-vector is an ad-hoc improvement of the k-vector methodology, specifically developed for function inversion. It is based on the idea of finding the distribution of points, for a specific function, in order *to always retrieve the same number of elements* to initiate the root finder. This way, the searching algorithm becomes adaptive, by retrieving the initial guess(es) as the selected root solver asks for. One example of this is to get two points bracketing the roots, a property that is required for bisection and regula-falsi algorithms. In case a Newton root solver is selected, the k-vector approach is able to provide just one point – the closest to the root – for the iteration initialization.

Furthermore, when the function to invert is computationally intensive, the inverse process can be performed *with no function evaluations*. The process requires a more intensive preprocessing, producing in general a larger database that will be used to get fast approximations of the roots with no function evaluations. This will reduce the root accuracy but it increases the inverse process speed considerably.

Simple and detailed examples are presented in order to show the procedure and potential of the methodology in a clear manner. These examples include the Gamma function, the Airy function of the first kind, the Bessel integral, Kepler's equation and the Gaussian integral.

2. Background on the *k*-vector

The range searching problem consists on retrieving, in a database of size n, all the elements that are contained in a given interval, $[y_a, y_b]$, where y_a and y_b are the lower and upper bounds of the interval, respectively. The most common searching algorithms are based on the *binary search* technique, that has complexity $\mathcal{O}(2\log_2 n)$, and the *search by hashing*, which is fast on the average case while it can be linear in the worst case due to the collision problem[1] affecting the method. The k-vector range searching technique is as fast as a hashing method (with best case complexity $\mathcal{O}(3)$), which has no collision problem but that may include some of the *closest elements* to the query range.[2] However, it requires a very fast preprocessing effort (provided in the appendix), as fast as just reading the sorted database. These properties make the k-vector particularly suitable for searching in large databases, where the worst case complexity goes asymptotically to $\mathcal{O}(3)$ as more memory is available to store the k-vector elements [22].

The k-vector is built using a sorted database ordered in the ascending mode and then generating a vector of indexes containing the information of the number of elements below a mapping function for specific values of the function. In this document, a straight line is assumed to be used as mapping function, since it provides the best speed performance for the methodology. Fig. 1 shows a random database (left), the sorted database (right), and the k-vector mapping function (line).

Let n be the number of elements of the database, being $\boldsymbol{y}(i)$ the ith element, and $\boldsymbol{s}$ the vector containing the sorted database $\boldsymbol{y}$, that is:

$$\boldsymbol{s}(i) \le \boldsymbol{s}(i+1), \quad \forall i \in [1, n-1]. \tag{1}$$

Therefore, the minimum and maximum values of the database are $y_{\min} = \boldsymbol{s}(1)$ and $y_{\max} = \boldsymbol{s}(n)$. Let $\boldsymbol{I}$ be the sorting indexes vector relating $\boldsymbol{y}$ and $\boldsymbol{s}$, that is:

$$\boldsymbol{s}(i) = \boldsymbol{y}(\boldsymbol{I}(i)) \quad \text{where} \quad i = \{1, 2, \ldots, n\}. \tag{2}$$

[1] The collision issue is due to the use of modular arithmetic to assign bins to elements. As a consequence of that, elements with values far from each other may be assigned to the same bin.

[2] The expected number of these extraneous elements depends on the nonlinearity of the database.