A Fortran Contour Integral Package

Ralph Baker Kearfott

Abstract

A series of Fortran routines for computing points on given contours of a function f(x, y) appear. Fortran routines for evaluating the integral of f(x, y) over the area bounded by a given closed contour of f also appear. The routines to locate contour levels and to find points on a given contour level are based on a predictor-corrector version of the arc-continuation techniques known as the Chow-Yorke algorithm or Davidenko's method. Other routines are borrowed, in modified form, from Alan Cline's spline under tension package (Dept. of Computer Sciences, University of Texas at Austin). Fortran functions and test drivers appear.

Computing Reviews categories: 5.12, 5.13, 5.15, 5.16

Key Words: contours, statistical computing, graphing functions, curvature, arc continuation, Davidenko's method, Chow-Yorke algorithm.

A Fortran Contour Analysis Package

1. Introduction and purpose.

Statistical analyses and graphics procedures often involve analysis of the contours of a function of two variables. Thus, there should be convenient, reliable, and efficient routines for dealing with contours. For this reason we developed our Contour Analysis Package (CAP).

2. Structure and usage.

The following subordinate tasks were singled out: (i) given a real-valued function $f(x, y)$ and a contour level ℓ, find a point (x_0, y_0) on the ℓ-contour of f, i.e., find numbers x_0 and y_0 such that $f(x_0, y_0) = \ell$; (ii) given a point (x_0, y_0) on a closed contour $f(x_0, y_0) = \ell$, find a set of points $\mathscr{S}_\ell = \{(x_i, y_i) \mid f(x_i, y_i) = \ell\}_{i=0}^{m}$ such that $\mathscr{S}_\ell$ adequately describes that contour; (iii) given a set $\mathscr{S}_\ell$, obtain a one-parameter representation of the (approximate) corresponding contour; (iv) given a one-parameter representation of a closed contour of f, compute the integral of f over the area bounded by that contour.

Package module ORTHTJ performs task (i), module CNTOUR performs task (ii), module KURVP1 (borrowed from Alan Cline [2]) performs task (iii), and module AREAFP (modified from Alan Cline's package (ibid.)) performs task (iv).

Various auxiliary modules may also be of interest. Module KURVP2 (from Alan Cline (ibid.)) may be used to evaluate points on the contour produced by KURVP1. Module INDF2D may be used to compute an indefinite integral table of f, evaluated at an arbitrary set of points $\{x_i, y_i\}_{i=0}^{m}$.

A Fortran double precision function PROBGT conveniently combines several tasks. Its input parameters include (in addition to certain tolerances and workspace arrays) a contour level ℓ and an initial guess (x_0, y_0) thought to be near the ℓ-contour. Certain tolerances for ORTHTJ, CNTOUR, and AREAFP are chosen automatically in PROBGT; the return value of PROBGT contains an approximate integral of f over the area bounded by $f(x, y) = \ell$.

Numerous uses are envisioned. For example, repeated calls to ORTHTJ, CNTOUR, KURVP1, and KURVP2 can be used to plot neighboring contours of a function; such plotting schemes are more flexible, efficient, and produce nicer plots than, say, "poor man's contours." The module PROBGT may be used directly to analyse the shape of unusual probability densities ([1]).

Module ORTHTJ is applicable to relatively general f, but modules CNTOUR and PROBGT function correctly only when $\{(x, y) \mid f(x, y) = \ell\}$ is homeomorphic to a circle.

Sample drivers are included for PROBGT, AREAFP, CNTOUR, and ORTHTJ. In addition, a main test routine, sample f, and results of running the routine on a Honeywell 68/80 (Multics system) are included.

3. Principles and implementation details.

Routines in the package stem from several sources. The modules KURVP1, KURVP2, CURVP1 and CURVP2, and the auxiliary modules SNHCSH, INTRVP, and TERMS are borrowed essentially unaltered from Alan Cline's spline-under-tension package([2]). (All routines in CAP, however, are written in or have been converted to double precision.)

The modules AREAFP and AREAF are modified versions of modules AREAP and AREA in the spline-under-tension package (ibid.). The module AREAFP returns an approximation to the integral of f over the region bounded by $f(x, y) = \ell$. This is accomplished by integrating $F^x(x(s), y(s))y'(s)$ relative to arclength s, over the contour defined by points $\{(x_i, y_i) \mid f(x_i, y_i) \doteq \ell\}_{i=0}^{m}$. Here $F^x(x, y)$ is the indefinite integral relative to x of f such that $F^x(x_0, y_0) = 0$; thus, by Green's Theorem, AREAFP obtains the integral of f. The module AREAF simply computes $y'(s)$ for a given s, while the module INDF2D supplies $\{F^x(x_i, y_i)\}_{i=0}^{m}$. The module INDF2D accomplishes this by employing a composite five point Gauss formula to integrate $f(x, y_i)$ from x_0 to x_k, $k = 1, \ldots, m$; different numbers of subintervals are used depending on $|x_k - x_0|$. The

actual quadrature is performed by the module CMPGS, which is a modified Fortran version of an algorithm by Don Freeman ([3]).

The routines ORTHTJ and CNTOUR are both based on the arc-following algorithm explained in [5]. There, arcs of $H(y) = \theta$, where $H : \mathbb{R}^{n+1} \to \mathbb{R}^{n}$ were followed by integrating

$$H'(y(s))y'(s) = \theta .$$

The integration in [5] is performed by an implicit Euler method followed by quasi-Newton corrector steps in directions orthogonal to the Euler predictor step.

Some of the ideas and techniques used are also found in [8], [9], and elsewhere. A similar approach, developed independently, appears in [4], while an alternate approach appears in [10] and in [11]. We adopt the predictor-corrector approach since: (i) it has been shown to be competitive in efficiency ([5]); (ii) precise control over the error at each point on the arc is possible; and (iii) numerous simplifications and resulting gains in efficiency are possible for the two-dimensional case considered here.

In ORTHTJ, an initial guess (x_g, y_g) and a desired contour level ℓ are input. Curves orthogonal to the contour $f(x, y) = c$ are then followed approximately, in the direction of increasing f if $f(x_g, y_g) < \ell$, and in the direction of decreasing f otherwise. Corrector iteration is unnecessary, since only the final point, and not points on the curve, are

desired. Module HPRIME computes ∇f whenever needed, using finite differences. Step size control is as in [5]. Upon bracketing the contour $f(x, y) = \ell$, linear interpolation is used iteratively to refine the location of a point (x_0, y_0) with $f(x_0, y_0) = \ell$. We note the method in ORTHTJ reduces to a method of steepest descent in certain instances. However, its behavior is quite adequate in most cases.

In CNTOUR, a level ℓ and a point (x_0, y_0) with $f(x_0, y_0) = \ell$ are input. The algorithm then proceeds as in [5], except that the termination criterion requires $\|(x_m, y_m) - (x_0, y_0)\|$ be small in relation to the step size. The values of ∇f are computed approximately using Broyden updates. It is unnecessary to make special "Powell" corrections to these updates as in [5], since the directions of successive updates to ∇f span $\mathbb{R}^2$ when both a predictor and a corrector step are executed.

Termination in CNTOUR is signalled provided $\|(x_m, y_m) - (x_0, y_0)\|$ is small, $\|(x_m, y_m)\|$ is larger than a prescribed tolerance (in which case the contour may not be closed), $\|\nabla' f(x_i, y_i)\| \doteq 0$ for some i, or provided the maximum allowed number of function evaluations has been exceeded. Convergence is signalled in the routine TRNFND.

It is assumed that ∇f is non-zero on the curve $f(x, y) = \ell$. If there are singularities or bifurcations, ideas such as those in [6] must be implemented.

The Basic Linear Algebra Subroutine DNRM2 ([7]) is used.

Additional details may be found as comments in the Fortran code. This includes explanation of various parameters the user must select.

Bibliography

[1] Anderson, Charles L., private communication.

[2] Cline, A.K., FITPACK - A software package for curve and surface fitting employing splines under tension, Department of Computer Sciences, University of Texas at Austin, 1981.

[3] Freeman, R.D., MULTINT, Algorithm 32, Communications of the A.C.M., February, 1961, p. 106.

[4] Georg, Kurt, On tracing an implicitly defined curve by quasi-Newton steps and calculating bifurcation by local perturbations, S.I.A.M. Journal on Scientific and Statistical Computing, vol. 2 no. 1, pp. 35-50.

[5] Kearfott, R. Baker, A derivative-free arc continuation method and a bifurcation technique, to appear in the proceedings of a conference on simplicial and continuation methods, held in Bremen, July, 1980.

[6] Kearfott, R. Baker, Some general derivative-free bifurcation techniques, submitted to the S.I.A.M. Journal on Scientific and Statistical Computing, 1981.

[7] Lawson, C.L., Hanson, R.J., Kincaid, D.R., and Krogh, F., Basic linear algebra subroutines for Fortran usage, Algorithm 539, A.C.M. Transactions on Mathematical Software, vol. 5, no. 3, pp. 308-325.

[8] Li, T.Y., and Yorke, J.A., A simple reliable numerical algorithm for following homotopy paths, Technical Summary Report no. 1984, Mathematics Research Center, University of Wisconsin at Madison, 1979.

[9] Li, T.Y., and Garcia, C.B., On a path following method for systems of equations, Technical Summary Report no. 1983, Mathematics Research Center, University of Wisconsin at Madison, 1979.

[10] Watson, Layne T., A globally convergent algorithm for computing fixed points of C^2 maps, Applied Mathematics and Computing, vol. 5, pp. 297-311, 1979.

[11] Watson, Layne T., and Fenner, Dan, The Chow-Yorke algorithm for fixed points or zeros of C^2 maps, Algorithm 555, A.C.M. Transactions on Mathematicsl Software, vol. 6, pp. 252-260, 1980.