`fparamin`: code to find all local minima of a certain parameter-dependent function set

A. H. Barnett, Dartmouth College

version 0.9: August 14, 2006

1 Installation

Un-tar the file `fparamin.tar`, and add the resulting created directory to your Matlab path.

2 Background

We wish to locate all local minima of a positive function $f(x)$ in an interval $x \in [a, b]$, using as small a number of function evaluations as possible. In practise, we care about functions whose square $F(x) := [f(x)]^2$ locally take the form of a parabola

$$F(x) = A + C(x - B)^2 + O(x - B)^3 + \cdots , \tag{1}$$

where $A \geq 0$ is an offset controlling the local minimum value, $B \in [a, b]$ is the location of a minimum, and C is a 'slope' which controls how steep the graph is either side of the minimum. This particular form emerges in the Method of Particular Solutions (MPS), the designed application; here f is the 'tension', or lowest (generalized) singular value of some operator, and x is the eigenvalue parameter (frequency, wavenumber, or 'energy'). The minima found can be interpreted as eigenvalues of a certain PDE problem, for instance the drum problem.

Given such a function, we may use the locally parabolic nature to locate the nearest local minimum as follows: start with three ordinates x_1, x_2, x_3 and their corresponding (squared) function values F_1, F_2, F_3. A parabola

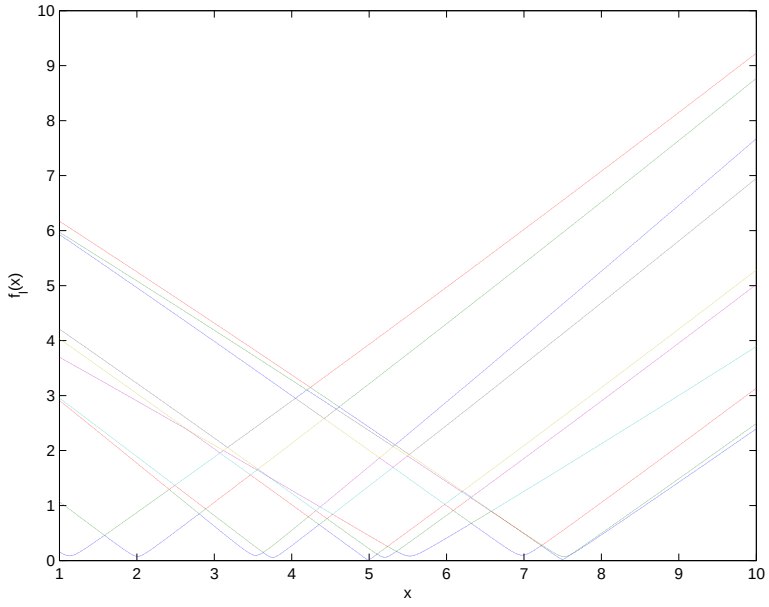

Figure 1: Model function set $\{f_l(x)\}$. The minimum function $f(x)$ is the lowest curve, shown in blue, whose local minima we wish to find in the interval $[1, 10]$. Apart from avoided crossings, this is a realistic model of MPS singular value curves.

may be uniquely fit to these 3 points, that is, A, B and C may be found such that $F(x) = A + C(x - B)^2$ holds for the three points. B is the new approximation to the ordinate of the minimum; the furthest from B of the three original x values is discarded, and the procedure may now repeat until a convergence criterion is met. This is similar to Newton's method. This is roughly what is done by the function `min_parabola`.

Now assume we have more information available: let the original function be the minimum of a set of scalar functions $\{f_l(x)\}_{l=1,2,\ldots}$, that is,

$$f(x) = \min_l f_l(x), \tag{2}$$

where each $f_l^2(x) := F_l(x)$ has the same parabolic form as above. Suppose that at each x it is no more expensive to compute the complete list of function $\{F_l(x)\}$ than is is to compute $F(x)$. Can we then use the information about the higher function values to locate the minima? If each $F_l(x)$ is of known character, the answer is yes. This is what `fparamin` does. Fig. 1 gives an example model function set of the type whose minima we wish to find.

We make the following assumption: each component function F_l is close to parabolic over the whole interval $[a, b]$. By 'close to', we mean formally

$$A_l + C_{\min}(x - B_l)^2 \;\leq\; F_l(x) \;\leq\; A_l + C_{\max}(x - B_l)^2 \tag{3}$$

holds for all $x \in [a, b]$, for some $A \geq 0$, $B_l \in [a, b]$. In other words, each F_l may deviate from parabolic, as long as it is bounded by two parabolas of given curvatures $C_{\min}$, $C_{\max}$. In the MPS situation, deviations from an exact parabola are due to: i) avoided crossings, and ii) cubic and higher Taylor terms in (1). We will also assume that in some neighborhood of B_l, F_l is closely approximated by a parabola of curvature C_l, with $C_{\min} \leq C_l \leq C_{\max}$, $\forall l$. Of course, C_l for each l is unknown at the outset, and, as shown in Fig. 1, may vary.

We may now restate our goal as the identification of the complete set of B_l lying in the interval $[a, b]$, using as few function set evaluations as possible. Looking at the rightmost 'V'-shape in Fig. 1, we see that one minimum swallows another, a *degeneracy problem*. We would like to identify both, therefore the B_l are the desired objects, rather than simply the minima of f. (PUT THIS UP FRONT?) Given the above assumptions, the following holds.

Proposition 2.1. *Fix x, and an interval $I = [x - \epsilon, x + \epsilon]$ for some $\epsilon > 0$. Suppose n is the number of values of l satisfying $f_l(x) < C_{min}^{1/2}\epsilon$, and N is the*

number satisfying $f_l(x) < (C'_{max})^{1/2}\epsilon$. *Then* n_I, *the number of* B_l *values lying in* I, *satisfies* $n < n_I < N$.

Here $C'_{\max}$ needs to be chosen slightly larger than $C_{\max}$, but this effect is only important when $\epsilon \approx (A_l/C_l)^{1/2}$. (Only if this change is made, can a rigorous result hold). However we will ignore this difference for now. It is not known if the assumptions hold in any precise fashion for the MPS, however they hold extremely well in practise.

`fparamin` therefore has three phases:

- The code `split_interval` makes use of Prop. 2.1 to do binary subdivision of $[a, b]$ into subintervals j which contain between 0 and 1 B_l values. This subdivision is continued only until f values of `opts.ttyp` are reached, therefore it may reach an interval containing between 0 and n_j values, where $n_j > 1$.

- In each subinterval, `min_parabola` locates a single minimum of the single function $F(x)$ using parabolic iterations as above, or returns emptyhanded if no such minimum is found. If the fitted C values are less than $C_{\min}$, then this code does binary subdivision to a maximum recursion depth of `opts.recur` until valid C values are reached, from which parabolic iterations are done as above.

- Valid B_j values are gathered together. For each j, identify the degeneracy as the number of l such that $f_l(B_j) < t_{\text{typ}}$.

3 Usage

Let

`f = fun(x, P1, P2, ...)`

be a Matlab function which returns either a scalar or a list.

Then a list of minima `x` and corresponding function f values `f` and curvatures `C` is found via

`[x, f, C, info] = fparamin(a, b, cmin, cmax, opts, fun, P1, P2, ...)`

which has inputs `a`, `b` defining the interval. `cmin` and `cmax` are $C_{\min}^{1/2}$ and $C_{\max}^{1/2}$. `opts` sets advanced options (see `help fparamin` and code). The `info` output contains total function evaluation count, etc.

The code `test_fparamin.m` gives example usage.

Choose `ttyp` to be the largest minimum values of $f(x)$ you expect to find.

Choose `cmin` and `cmax` to enclose the range of slopes you expect. Performance when their ratio becomes much larger than 3 is not known.

4 Bugs

The code does not count degeneracies closer than $(A_l/C_l)^{1/2}$ very reliably, *i.e.* in the regime where the minimum values $A_l^{1/2}$ cause two or more minima to effectively coelesce. Work needs to be done to make use of the information about how many are within various intervals to do a consistency check on this.

No sanity checks are done if all the minima found have $C_j < C_{\min}$. Currently they are all discarded.

About 20 function evaluations per minimum are needed to locate them to 6 or so digits of accuracy. This counld be improved but not by more than factor 2.

5 List of codes

`fparamin.m`	user-called driver routine
`split_interval.m`	binary subdivision of interval to isolate minima
`min_parabola.m`	iterative parabolic minimization in subinterval
`para_fit.m`	fit parabola to three (x, F) pairs
`test_paramin.m`	tests driver routine with random model functions
`test_split_interval.m`	tests `split_interval.m` with random model functions