Algorithms supplement

Algorithm 109
GOLDEN SECTION SEARCH

I. D. Hill
Clinical Research Centre
Harrow, Middlesex
HA1 3UJ

Author's note
The Fibonacci and Golden Section methods of searching for the minimum (or maximum) of a function appear deceptively simple, but are full of traps for the unwary. *The Computer Journal's* Algorithm 2 (Pike and Pixner, 1965), Algorithm 2 modified (Pike, Hill and James, 1967), and Algorithms 16 and 17 (Overholt, 1967a, b) all calculate each point of function evaluation very precisely, as an exact fraction of the interval of uncertainty (within which the required minimum is known to lie), rather than by the obvious method of placing it symmetrically to the point where evaluation has already been made.

It has recently been found, however, that this precaution is not sufficient to prevent trouble, which occurs when a point calculated early in the process happens to be extremely close to the minimum. Subsequent points then all give larger function values and the early point remains as one of the two that are relevant at each stage. As the interval of uncertainty gets narrower it is being assumed that this early point remains in the correct relative position, and this can lead to a breakdown of the process even though each new point calculated is being correctly positioned.

The obvious way to overcome this difficulty would be to stop the process every so often, and restart 'from scratch' using the latest interval of uncertainty and correctly positioning each of the two points. However this would not only call for additional function evaluations, which it is the aim of the method to minimise, but would also mean that the point eventually reported as the minimum might not be the best one evaluated.

The procedure *golda* below therefore adopts an alternative approach of watching the ratio of the distance between the two points to the total interval of uncertainty and, if this gets too big, leaving the points as they are and slightly expanding the interval of uncertainty to put the ratio right again. There is no need to look to see whether the ratio gets too small as, if it does, this always leads to too large a ratio on the next iteration. Within this technique the individual points are positioned by simple symmetry as this can now be done without danger.

The technique could be modified to end in a Fibonacci sequence instead of a Golden Section sequence, with the advantage of a final point in the middle of the interval of uncertainty, but the extra complication seems too great to be worth it. The formal parameters are identical in specification, and in purpose, to those of Algorithm 2 (modified) for which *golda* may be substituted with just the change of procedure name at each call.

I thank a referee for pointing out that, in exceptional circumstances, the method could call for a function evaluation outside the *a:b* range. Precautions are therefore taken to prevent this from happening.

```
real procedure golda (a, b, eps, fval, prem, f);
value a, b, eps; real a, b, eps, fval;
Boolean prem; real procedure f;
comment finds within plus or minus eps the position of the minimum
of the function f(x) in the range a < x < b by the Golden Section
search method. f(x) must be monotonic decreasing from x = a to the
minimum position and then monotonic increasing to x = b.
On exit fval contains the function value at the position found, while
prem is set to true if a premature exit has been made or false other-
wise. A premature exit indicates either that the minimum is so flat
that the accuracy to which the calculations are performed is insuf-
ficient to cut the interval of uncertainty down to plus or minus eps,
or that the minimum is so close to a or b that the method has called
for a function evaluation outside the a:b range;

begin real x, y, c, d, epc, g, h, fg, fh, fx, fy;
Boolean equal, adjust, prim;

procedure initiate;
comment finds h and g as the golden section points within the
interval x - y, and evaluates the function at each of them;
 begin
 h := x + (c - 1·0) × (y - x); g := x + y - h;
 fh := f(h); fg := f(g)
 end initiate;

comment c has value (1 + √5)/2. An accuracy of 7 significant figures
is quite sufficient;
c := 1·618034; fx := fy := maxreal;
x := a; y := b;
epc := eps × (c - 1·0); equal := prim := false;
initiate;
for d := h - g while d > epc ∧ ¬ prim do
if fg = fh then
 begin
 if equal ∧ fx = fg then prim := true
 else
  begin
  equal := true; fx := fy := fh;
  x := g; y := h; initiate;
  if fg > fx ∨ fh > fx then prim := true
  end
 end fg = fh
else
 begin
 equal := false;
 comment d/(y - x) should have value √5 - 2 = 0·236068. Adjust-
 ment is needed if rounding errors have made it much bigger than
 this;
 adjust := d > 0·237 × (y - x);
 if fg > fh then
  begin
  if adjust then y := h + c × d;
  x := g; g := h;
  fx := fg; fg := fh;
  h := x + y - g;
  if h < b then
   begin
   fh := f(h);
   if fh > fg ∧ fh > fy then prim := true
   end
  else
   begin
   h := b; fh := f(b);
   prim := true
   end
  end
 else
  begin
  if adjust then x := g - c × d;
  y := h; h := g;
  fy := fh; fh := fg;
  g := x + y - h;
  if g > a then
   begin
   fg := f(g);
   if fg > fh ∧ fg > fx then prim := true
   end
```