Numerical Computing II

Homework 16: Optimization

Marty Fuhry

February 22, 2010

Exercise 16.1

```
a = 0;
b = pi;

p = 1/2*(3-sqrt(5));

xl = a + p*(b-a);
xr = a + (1-p)*(b-a);

fleft  = f(xl);
fright = f(xr);

for j = 1:4
	% set fright to fleft
	% need to evaulate new fleft
	if fleft >= fright
		b  = xr;
		xr = xl;
		xl = a + p*(b-a);
		fright = fleft;
		fleft = f(xl);
	% set fleft to fright
	% need to evaulate new fright
	else
		a = xl;
		xl = xr;
		xr = a + (1-p)*(b-a);
		fleft = fright;
		fright = f(xr);
	endif
endfor
```

Our endpoints after the fourth iteration of the Golden Section Search were:

$$a = 0.45835$$
$$b = 0.91670$$

We find that the true max, $\pi/4$, is well between this interval. The estimated positions and values were as follows:

$$x_l = 0.63343$$
$$f(x_l) = 0.31417$$
$$x_r = 0.74163$$
$$f(x_r) = 0.32167$$

These values very accurately approximate the max after only four iterations.

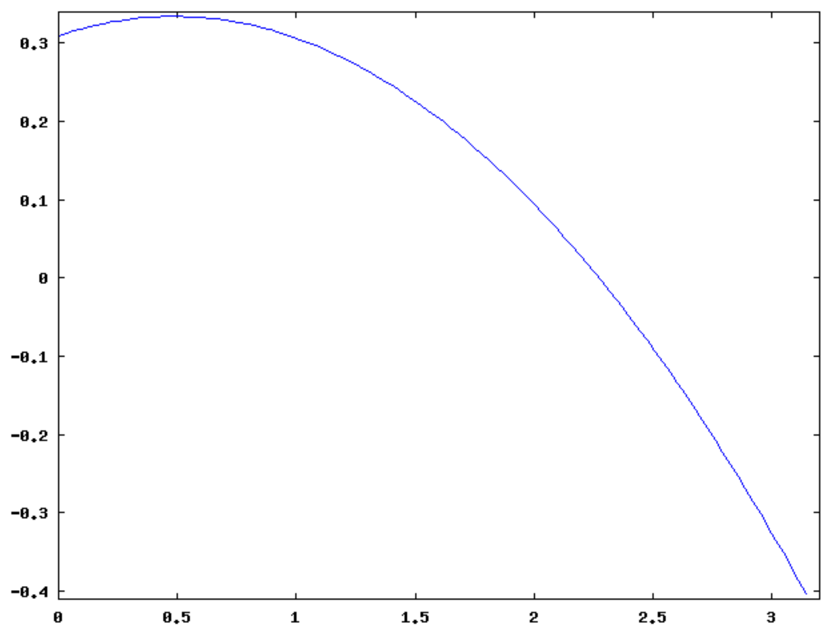
Exercise 16.2

```
% define endpoints and row
a = 0;
b = pi;
r = 1/2*(3-sqrt(5));
x = r*(b-a);

for j = 1:4
        % quadratic polynomial fit
        p = polyfit([a,x,b], [f(a),f(x),f(b)], 2);
        % if there's a maximum
        if p(1) < 0
                % take the derivative
                q = polyderiv(p);
                % find the max value
                xmax = -q(2)/q(1);
                % discard farthest term from xmax
                adif = abs(a - xmax);
                bdif = abs(b - xmax);
                xdif = abs(x - xmax);
                discard = max([adif,bdif,xdif]);
                if discard == adif
                        a = xmax;
                elseif discard == bdif
                        b = xmax;
                elseif discard == xdif
                        x = xmax;
                endif
        % else, GSS
        else
                % find the endpoints
                a = min([a,b,x]);
                b = max([a,b,x]);
                % set xleft and xright
                xl = a + r*(b-a);
                xr = a + (1-r)*(b-a);
                % solve for the GCC
                if f(xl) >= f(xr)
```

```
                        b = xr;
                        xr = xl;
                        xl = a + r*(b-a);
                else
                        a = xl;
                        xl = xr;
                        xr = a + (1-r)*(b-a);
                endif
        endif
        %j
        %xmax
end
xs = linspace(0,pi,100);
plot(xs,polyval(p,xs))
```

The graph made by the fourth interpolating polynomial looks like this:



This code interpolates a function quadratically in the least squares sense using the endpoints a, b, and x. After interpolation, if the interpolated polynomial has a maximum value, the first coefficient is negative. If the first coefficient is not negative, then we can use the Golden Section Search to produce a "close enough" approximation for a new value of x to be used for our third interpolation point. After each "successful" polynomial interpolation, the furtheset point from the polynomial's max will be discarded.

Running this code, we receive values:

$$x_m = .48300$$
$$f(x_m) = 0.28653$$

This is curiously no closer than the Golden Section Search. Though, after many iterations, it does converge properly to the actual max value, $\pi/4$.