Example 3.13 ("Mesh-Based Optimization")

An important class of optimization problems is introduced through this example. We use the term "mesh-based optimization" problem to refer to problems that require the determination of a function, say $y(x)$, on a mesh. Thus, the number of variables depends on how finely the mesh is discretized. The *Brachistochrone* problem involves finding the path or shape of the curve such that an object sliding from rest and accelerated by gravity will slip from one point to another in the least time. While this problem can be solved analytically, with the minimum time path being a *cycloid*, we will adopt a discretized approach to illustrate mesh-based optimization, which is useful in other problems.

Referring to Fig. E3.13a, there are n discretized points in the mesh, and heights y_i at points $2, 3, \ldots, n-1$ are the variables. Thus,

$$\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$$
$$\mathbf{x} = [y_2, y_3, \ldots, y_{n-1}]^T$$

Importantly, a coarse mesh with, say, $n = 5$ points is first used and corresponding optimum heights are determined. Linear interpolation on the coarse mesh solution to obtain the starting point for a fine mesh is then used. The Matlab function *interp1* makes this task simple indeed. This way, optimum solutions with any size n is obtainable. Directly attempting a large value of n generally fails – that is, the optimizer cannot handle it.

Consider a line segment in the path as shown in Fig. E3.13a. From mechanics, we have

$$-\mu mg(\cos\theta)(\Delta L) = \tfrac{1}{2}m(v_{i+1}^2 - v_i^2) + mg(y_{i+1} - y_i) \qquad \text{(a)}$$

where m is the mass, v refers to velocity and $g > 0$ is acceleration due to gravity. For frictionless sliding, $\mu = 0$. Further, the time Δt_i taken to traverse this segment is given by

$$\Delta t_i = \frac{(v_{i+1} - v_i)\Delta L}{g(y_i - y_{i+1})}$$

$$\text{where } \Delta L = \sqrt{h^2 + (y_{i+1} - y_i)^2} \qquad \text{(b)}$$

Equations (a) and (b) provide the needed expressions. The objective function to be minimized is $T = \sum_{i=1}^{n-1} \Delta t_i$. A starting shape (say, linear) is first assumed. Since the object starts from rest, we have $v_1 = 0$. This allows computation of v_2 from Eq. (a), and Δt_1 from Eq. (b). This is repeated for $i = 2, \ldots, n-1$, which then defines the objective function $f =$ total time T.

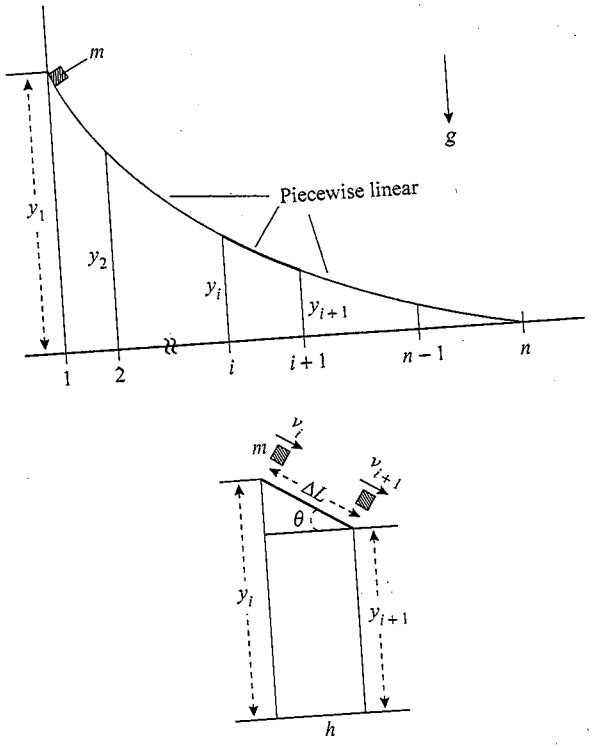
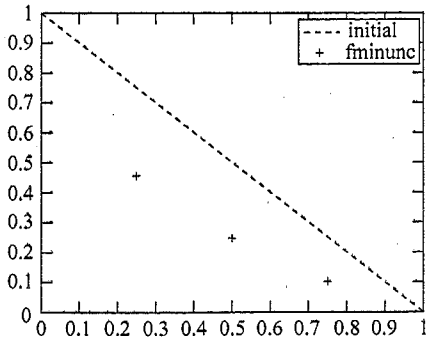


Figure E3.13a



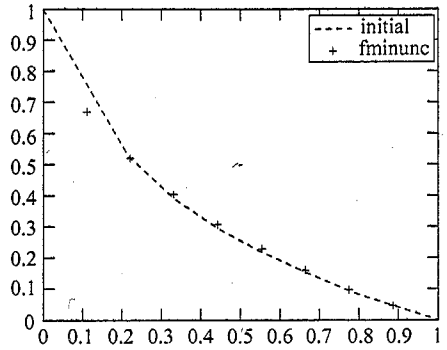
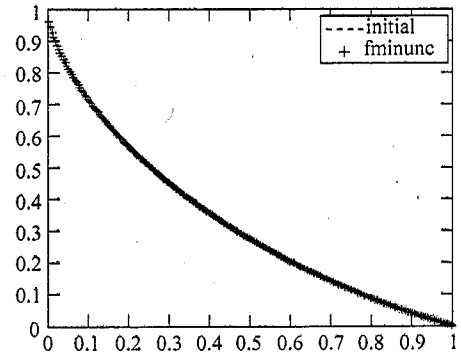


Figure E3.13b



Results using Matlab *fminunc* routine yield the optimum paths in Fig. E3.13b, between two fixed points (0, 1) and (1, 0), with $n = 5, 10, 250$, successively.

3.9 Approximate Line Search

In lieu of solving the "exact" line search problem in (3.16), approximate line search strategies are presented here. The first danger in taking steps that result