~\Documents\documents_general\structured_courses\math564\evaluations\projects\p05\pathtime.m

```
function [f,g]=pathtime(x,par)

% parse the input data
% n is the number of sinusoid coefficients for x and for y
% w,z are the decision variable weights
% v is the velocity map array (rows are y, cols are x)
% A,B are the (x,y) coordinates of the beginning and ending points
n=length(x)/2;
w=x(1:n);
z=x(n+1:end);
v=par.v;
[my,mx]=size(v);
A=par.A;
B=par.B;

% construct a piecewise linear path approximation for computation
% xx,yy are the x and y coordinates along the path on [0,1]x[0,1]
% s is the variable that parametrically defines the path
s=linspace(0,1,1000)';
xx=(1-s)*A(1)+s*B(1);
yy=(1-s)*A(2)+s*B(2);
for k=1:n               % you can do this without a loop
    S=sin(k*pi*s);
    xx=xx+w(k)*S;
    yy=yy+z(k)*S;
end

% xxr,yyr are the coordinates of the midpoint of each line segement
% in the velocity array index units.  Any points outside of the array
% are set at the boundary using max/min functions.
xxm=1+xx*(mx-1);
yym=1+yy*(my-1);
xxm=(xxm(2:end)+xxm(1:end-1))/2;
yym=(yym(2:end)+yym(1:end-1))/2;
xxm=max(min(xxm,mx),1);
yym=max(min(yym,my),1);

% compute the travel time.  dist is the distance on [0,1]x[0,1]
% between line segment end points -- summed.  vel is the velocity
% at the midpoint interpolated from array data.  f is travel time.
dist=sqrt(diff(xx).^2+diff(yy).^2);
vel=interp2(v,xxm,yym);
f=sum(dist./vel);

% compute the gradient by approximation
if nargout>1
    del=sqrt(eps);
    g=zeros(2*n,1);
    for j=1:2*n
        y=x;
        y(j)=y(j)+del;
```

```
52          df=pathtime(y,par);
53          g(j)=(df-f)/del;
54      end
55  end
56
57  return
```