Lab #2

1. (a) Since f is monotone, when $y > 0$ the condition $f(t) \leq y$ is equivalent to $t \leq g(y)$.

   (b) The fact that g is monotone and increasing derives from the fact that when y increases, the condition $f(t) \leq y$ becomes less stringent. It is concave by the following argument. Take $y_1, y_2 > 0$ and let $\theta \in [0, 1]$, and define $y = \theta y_1 + (1 - \theta)y_2$. Let $t_i = g(y_i)$, so that $y_i = f(t_i)$, $i = 1, 2$, and define $t = \theta t_1 + (1-\theta)t_2$. We have

   $$y = \theta y_1 + (1 - \theta)y_2 = \theta f(t_1) + (1 - \theta)f(t_2) \geq f(t).$$

   Hence, since f is monotone increasing, $g(y) \geq t$, that is, $g(\theta y_1 + (1 - \theta)y_2) \geq \theta g(y_1) + (1 - \theta)g(y_2)$. This concludes the proof.
   The following CVX snippet yields the function value $g(y)$ for given y.

```
function cvx_optval = g(y)
cvx_begin
variable t;
maximize(t)
subject to
2*pos(t) + 3*pow_pos(t, 1.2) + 4.1*pow_pos(t, 2.3) <= y;
cvx_end
```

   (c) The following CVX snippet solve the problem. Note that the constraint $y > 0$ is implicit and need not be explicitly declared.

```
cvx_begin
variables x y;
minimize(quad_over_lin(x, y) + 4*x + 5*y)
subject to
g(y) + 2*g(y) >= 2;
cvx_end
```

2. (a) The following Matlab script finds the optimal solution.

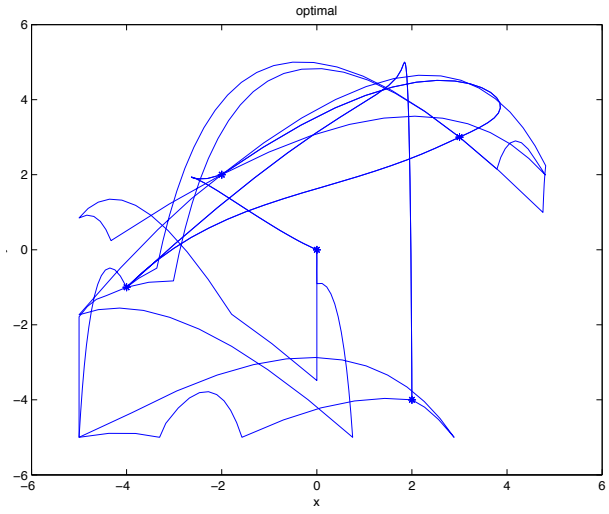
```
cvx_quiet(true);
thrusters_data;
F = [ cos(theta1) cos(theta2);...
      sin(theta1) sin(theta2)];

% finding optimal solution
cvx_begin
    variables u(2,K-1) p(2,K) v(2,K)
```

```
    minimize ( sum(sum(u)))
    p(:,1) == 0;            % initial position
    v(:,1) == 0;            % initial velocity
    % way-point constraints
    p(:,k1) == w1;
    p(:,k2) == w2;
    p(:,k3) == w3;
    p(:,k4) == w4;
    for i=1:K-1
        p(:,i+1) == p(:,i) + h*v(:,i);
        v(:,i+1) == (1-alpha)*v(:,i) + h*F*u(:,i)/m + [0; -g*h];
    end
    u >= 0;
    % constaints on positions (x,y)
    p <= pmax;
    p >= -pmax;
cvx_end

display('The optimal fuel use is: ');
optval = cvx_optval
plot(p(1,:),p(2,:));
hold on
ps = [zeros(2,1) w1 w2 w3 w4];
plot(ps(1,:),ps(2,:),'*');
xlabel('x'); ylabel('y'); title('optimal');
axis([-6 6 -6 6]);
```

This Matlab script generates the following optimal trajectory.



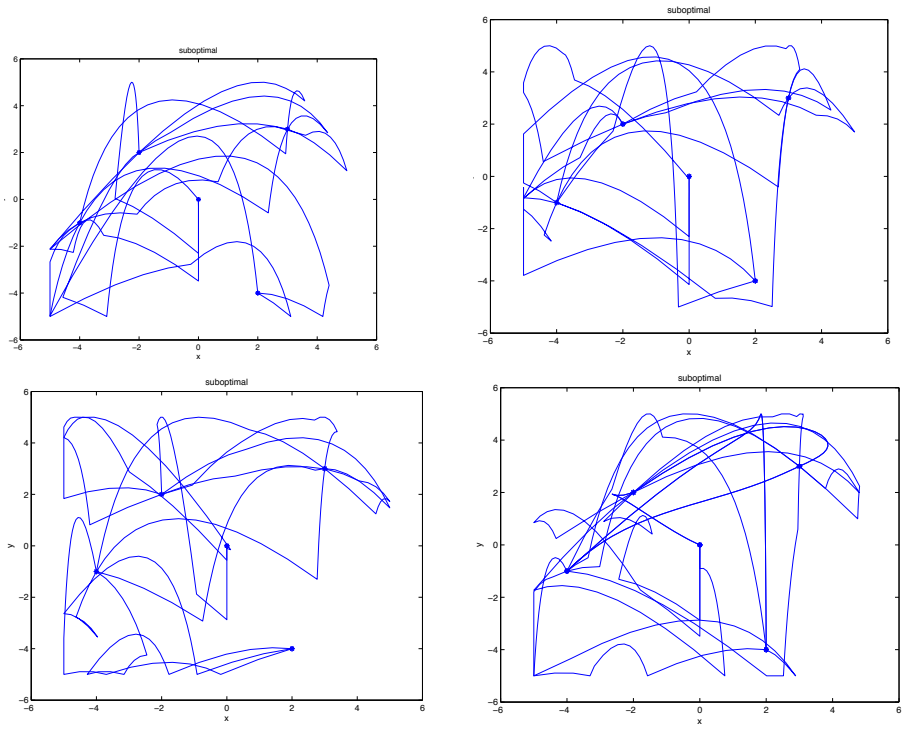
The optimal value fuel use is found to be 1055.3.

(b) The following script finds 1% suboptimal solutions.

```
% finding nearly optimal solutions
cvx_begin
    variables u(2,K-1) p(2,K) v(2,K)
    minimize ( sum ( sum ( randn(2,K-1).*u ) ) + ...
               sum ( sum ( randn(2,K).*p ) ) + ...
               sum ( sum ( randn(2,K).*v ) ) )
    p(:,1) == 0;            % initial position
    v(:,1) == 0;            % initial velocity
    % way-point constraints
    p(:,k1) == w1;
    p(:,k2) == w2;
    p(:,k3) == w3;
    p(:,k4) == w4;
    for i=1:K-1
        p(:,i+1) == p(:,i) + h*v(:,i);
        v(:,i+1) == (1-alpha)*v(:,i) + F*u(:,i) + [0; -g*h];
    end
    u >= 0;
    sum(sum(u))<=1.01*optval;
    % constaints on positions (x,y)
    p <= pmax;
    p >= -pmax;
cvx_end

figure;
plot(p(1,:),p(2,:));
hold on
ps = [zeros(2,1) w1 w2 w3 w4];
plot(ps(1,:),ps(2,:),'*');
xlabel('x'); ylabel('y'); title('suboptimal');
axis([-6 6 -6 6]);
```

This script returns 4 randomly-generated nearly optimal trajectories.



We see that these nearly optimal trajectories are very, very different. So in this problem there is a weak minimum, *i.e.*, a very large 1%-suboptimal set.