Week 5

Monday, April 12, 2021 10:48 AM

RoboWeek 5

Chapter 25

Robo Week 5: Gradient Ascent and The Flatland Challenge

Schedule

25.1 Mini-Lecture: The Gradient

- *Contours* of a function are curves of constant function value, $f(\mathbf{r}) = c$.
- We can use the chain rule to prove that
  - the *gradient* is orthogonal to the contours of a function.
  - the *gradient* points in the direction of steepest ascent for a given function $f(\mathbf{r})$.

25.2 The Leisure-Seeker

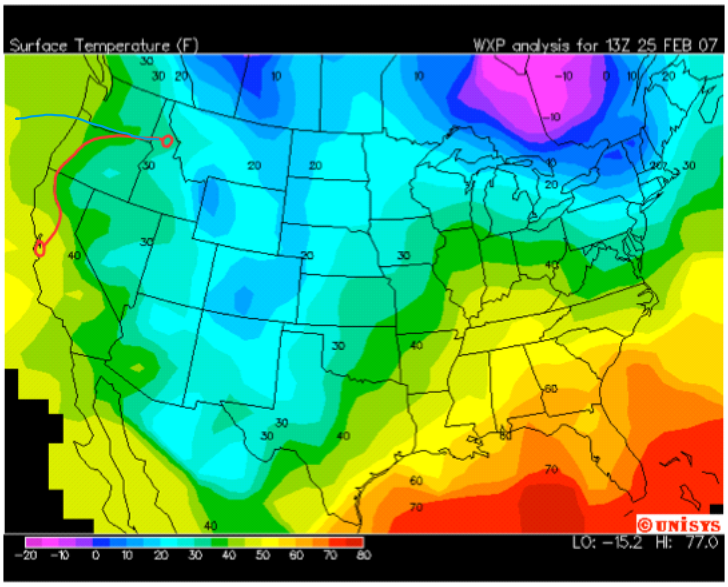


Figure 25.1: This trajectory is an example of *the techie's gambit*—making an unconsidered beeline for Silicon Valley. Can you explain why this is an incorrect example of gradient ascent?

Exercise 25.1

1. Figure 25.1 depicts regions of relatively similar temperature values; the *boundaries* between these colored regions are contours. Draw in a few gradient vectors along contours in the Pacific Northwest region of the map (near Washington, Oregon, and northern California).
2. Figure 25.1 is an example of an **incorrect** trajectory for gradient ascent. There's a very obvious reason why this trajectory is incorrect—what is it? Parallel to contour
3. Draw an updated version of the curve, starting from northern Idaho and following the gradient of the temperature field.

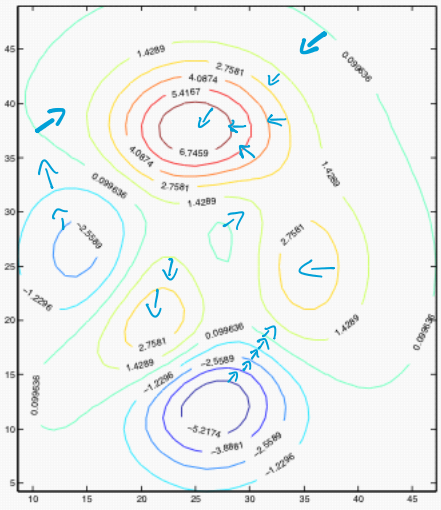
25.3 Conceptual Questions about Partials and Gradients

Exercise 25.2

1. What is meant by f_x? By $\frac{\partial^2 f}{\partial x^2}$? By $D_u f$? By ∇f?
2. The idea of gradient is often discussed primarily in two dimensional and three dimensional "physical" spaces, using $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, and so forth. But more generally, you can have a gradient of a function of any number of variables. Give a real-world example of a gradient for a situation that involves more than 3 variables.
3. "The gradient always points in the direction of fastest increase." Can you think of physical examples where there is *more than one* direction of fastest increase? What's going on here?
4. "The gradient is always normal to level curves/surfaces." Explain. *Note:* In this context, "level curves/surfaces" is referring to contour lines on a surface. Each contour line indicates a particular value or 'level' of the scalar function.
5. "The directional derivative in the direction of $\mathbf{u}$ is given by $\nabla f \cdot \hat{\mathbf{u}}$." Why does this make sense?
6. If the gradient is zero, does that imply that you are at a max or a min? Why or why not?
7. What does it mean for $\frac{\partial^2 f}{\partial x^2} > 0$? What about $\frac{\partial^2 f}{\partial x \partial y} > 0$?

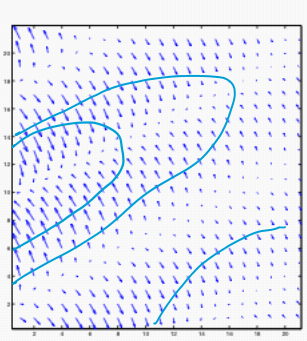
Exercise 25.3

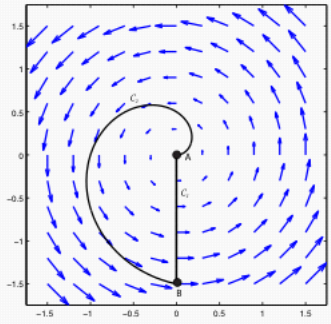
The diagram below shows a contour plot for a function. Sketch in the gradient field.



Exercise 25.4

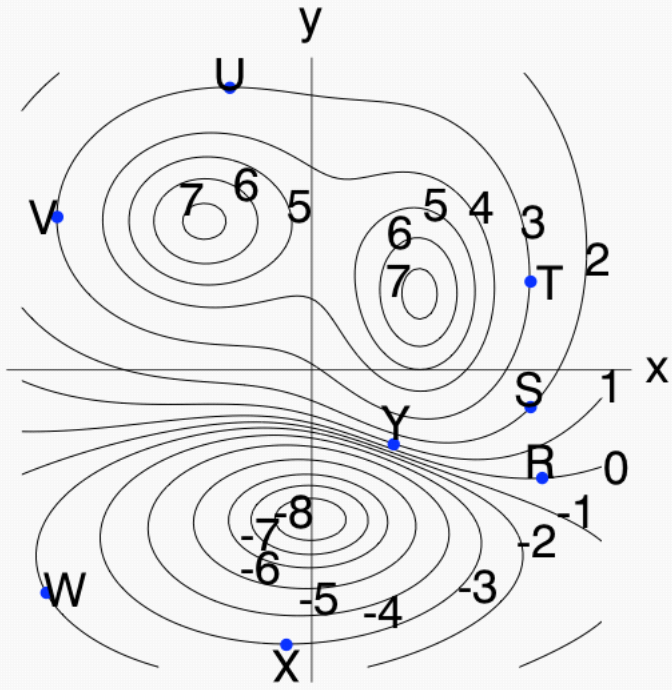
The diagrams below show two vector fields. One is the gradient of a function of two variables; the other is not. Which one could be a gradient? Why? Sketch in the level curves for the one that works.





Exercise 25.5

The diagram below shows some level curves of a function $g(x, y)$. The numbers indicate the g-values of these level curves, and the letters indicate points on the level curves. Note that point Y is on the level curve $g = 1$ and point W is on the level curve $g = -2$.



	x	y
R	−	+
S	−	+
T	−	−
U	−	−
V	+	−
W	−	−
X	+	−
Y	+	+

1. What are the signs of the partial derivatives $\frac{\partial g}{\partial x}$ and $\frac{\partial g}{\partial y}$ at each of the points marked (R,S,T,U,V,W,X and Y)?

2. At which of the points marked does the gradient vector ∇g have the greatest magnitude? Explain. Y, b/c it's rapidly increasing

3. Let

$$\mathbf{u} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The directional derivative $D_{\hat{\mathbf{u}}} g$ is zero at exactly one of the points marked. Which point is it?

S

25.4 Gradient Ascent

Gradient Ascent (or descent) is a technique to determine the maximum or minimum of a function of many variables by taking steps in the direction of the gradient (or negative gradient). If the height of a surface is

described by $z = f(\mathbf{r})$, where $\mathbf{r}$ is the position vector in the plane, and we begin at $\mathbf{r}_0$, the points determined by Gradient Ascent are given by

$$\mathbf{r}_{i+1} = \mathbf{r}_i + \lambda_i \nabla f(\mathbf{r}_i), \ i = 0, 1, 2, \ldots$$

where λ_i is the relative size of the step that we take in the direction of the gradient. (Incidentally, this λ has nothing to do with eigenvalues; we've just run out of Greek letters!) There are various schemes for choosing these, and one of the simplest is to determine the next step with a simple proportionality

$$\lambda_{i+1} = \delta \lambda_i$$

where $0 < \delta \leq 1$. To choose λ_0 and δ, we have to consider the particulars of the problem. For example, if $\delta = 1$, the simplest scenario, we increment each step by λ. The choice of λ depends on how the surface changes. We are going to develop a method and implementation to drive your NEATO on the floor of the classroom in a way that physically realizes the method of Steepest Ascent. First, we are going to introduce the mountain you will climb, and you will think through the steps involved.

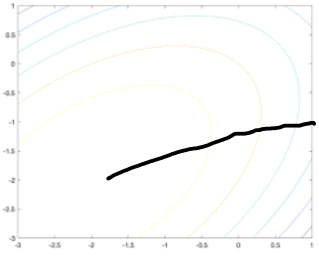
Exercise 25.6

The mountain you will "climb" is defined as follows

$$f(x, y) = xy - x^2 - y^2 - 2x - 2y + 4$$

where x, y, and f are measured in meters. You are required to start at $(1, -1)$, and work your way to the top by method of steepest ascent.

1. Visualize the contours of this function on the domain $(-3, 1) \times (-3, 1)$. This notation means the set of points that have an x and y components, $-3 \leq x, y \leq 1$.
2. Draw the path of steepest ascent if we were moving continuously from a starting point at $(1, -1)$.
3. Find the gradient of this function.
4. Assuming $\mathbf{r}_0 = (1, -1)$, what is the initial gradient at $\mathbf{r}_0$? What would be a reasonable choice for λ_0 so that $\mathbf{r}_1$ is not too far from the continuous path? Plot $\mathbf{r}_1$ on your contour plot.
5. Assuming you place your NEATO at $(1, -1)$ pointing along the y-axis, how much do you have to rotate it in order to align it with the gradient at $\mathbf{r}_0$? What would be a reasonable angular speed?
6. Assuming that you are going to drive your NEATO at $0.1m/s$, how long would you drive in order to reach $\mathbf{r}_1$?
7. What is the gradient at $\mathbf{r}_1$? What value of δ should you use so that λ_1 and $\mathbf{r}_2$ are reasonable? Plot $\mathbf{r}_2$ on your contour plot.
8. Assuming your NEATO is now at $\mathbf{r}_1$, how much do you have to rotate it in order to align it with the new gradient? What would be a reasonable angular speed?
9. Assuming that you are going to drive your NEATO at $0.1m/s$, how long would you drive in order to reach $\mathbf{r}_2$?



<- 25.6.2

25.5 Gradient Ascent in MATLAB

In this exercise we would like you to implement gradient ascent in MATLAB. Recall that if $f(\mathbf{r})$ is a scalar function of a position vector $\mathbf{r}$, the points determined by gradient ascent are given by

$$\mathbf{r}_{i+1} = \mathbf{r}_i + \lambda_i \nabla f(\mathbf{r}_i),\ i = 1, \ldots$$

where λ_i is the relative size of the step that we take in the direction of the gradient. There are various schemes for choosing these—one of the simplest is a proportionality rule that modifies the scaling factor λ_i at each step

$$\lambda_{i+1} = \delta \lambda_i$$

where both δ and λ_0 are *thoughtfully chosen* for the problem at hand.

Exercise 25.7

1. Remind me; what must be true about the gradient at a maximum of a function? What does that tell you about the norm (length) of the gradient at that point? All are 0
2. Develop pseudo-code for the gradient ascent algorithm. Keep it general: Assume that you have f, a general scalar function of a vector $\mathbf{r}$. Suppose you have a MATLAB function that provides the function value `fun(r)` and gradient `grad(r)`. You are going to want to use a loop - a "while" loop would be a good choice. Introduce any parameters you need to make this pseudo-code work, and make sure to return the entire history of points r_i. What would be a reasonable stopping criterion?
3. When you are happy with your pseudo-code, develop a script or function that:
   (a) Automatically determines the discrete points $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n$ given an initial point $\mathbf{r}_0$.
   (b) Can be tuned by varying δ and λ_0.
   (c) Takes any other parameters that you deemed necessary in writing your pseudo-code.
   (d) Returns the entire history of tested points $\mathbf{r}_i$.
4. Test your gradient ascent algorithm on the function $f(x, y) = xy - x^2 - y^2 - 2x - 2y + 4$. To start, use the parameter values $r_0 = [0.7, 0.5], \lambda_0 = 0.9, \delta = 0.9$. Does your algorithm reach the maximum of the function? How do you know?
5. Visualize the sequence of points $\mathbf{r}_i$ for $i = 1, \ldots, n$ returned from your algorithm in the previous part. Do the points travel "directly" to their endpoint? Or do they "wander"?
6. Test your gradient ascent algorithm on the function $f(x, y) = xy - x^2 - y^2 - 2x - 2y + 4$. To start, use the same parameter values as before, but set $\delta = 1.1$. Does your algorithm reach the maximum of the function? If not, what went wrong?

Pseudo-code for exercise above.

```
Provide: grad, delta, lambda0, tolerance, n_max, r_0
Set: n=0, r_i = r_0, grad_i = grad(r_i), lambda = lambda_0, R = [r_0]
While: n < n_max and norm(grad_i) > tolerance do
    r_i = r_i + lambda * grad_i
    R(end + 1) = r_i
```

```
    grad_i = grad(r_i)
    lambda = delta * lambda
    n = n + 1
Return: R
```

25.6 Flatland Challenge

You will develop a method and implementation to drive your NEATO on the (virtual) floor of the classroom in a way that physically realizes the method of Steepest Ascent. The mountain you will "climb" is defined as follows: (this should look somewhat familiar—when have you seen this equation before and what does it describe?)

$$z = f(x, y) = xy - x^2 - y^2 - 2x - 2y + 4$$

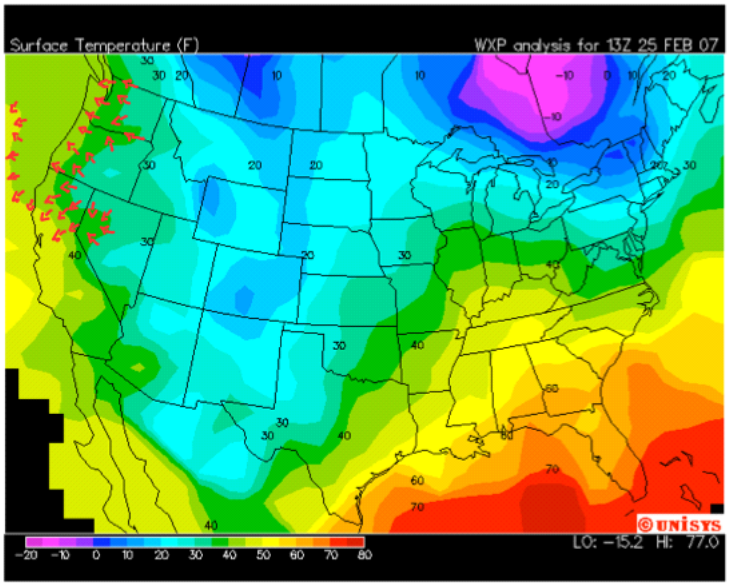
where x, y, and z are measured in meters. You are required to start your NEATO at $(1, -1)$ pointing in the $+y$ direction, and work your way to the top. In order to position your NEATO with the correct orientation, you can use the `placeNeato.m` function, which we have provided (**Note: that the placeNeato function is already inside of the simulated environment, you do not need to download it in order to use it**).

Exercise 25.8

1. Decompose this problem. What are the steps involved? You should have a decomposition that clearly explains the process of driving the NEATO along a discrete approximation to the path of steepest ascent.
2. Now develop the code for the NEATO. It may help to review your Bridge of Doom code....
3. Now drive your NEATO! You will startup the Neato simulator using the `flatland` environment. Remember, full instructions for using the simulator are available on the Meeto Your Neato page. For this challenge you'll want to run the flatland world using `qeasim`. In MATLAB, run this command. `>> qeasim start flatland_no_spawn`

Solution 25.1

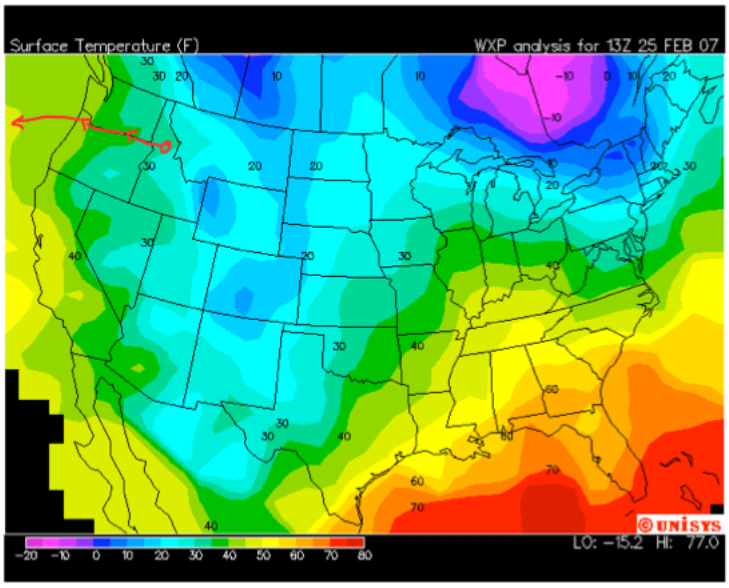
1. Figure 25.1 depicts regions of relatively similar temperature values; the *boundaries* between these colored regions are contours. Draw in a few gradient vectors along contours in the Pacific Northwest region of the map (near Washington, Oregon, and northern California).



   Remember that the gradient lies *orthogonal* to contours.

2. Figure 25.1 is an example of an **incorrect** trajectory for gradient ascent. There's a very obvious reason why this trajectory is incorrect—what is it? In its middle, the trajectory runs parallel to a contour line. Remember that the gradient lies orthogonal to contours. In order for this trajectory to be a gradient ascent, it must be orthogonal to contours of the map.

3. Draw an updated version of the curve, starting from northern Idaho and following the gradient of the temperature field.

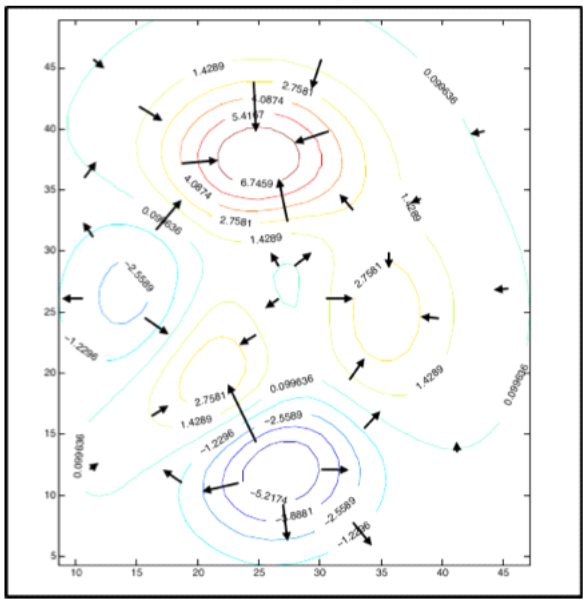


This curve is correct because it lies orthogonal to every contour it crosses.

Solution 25.2

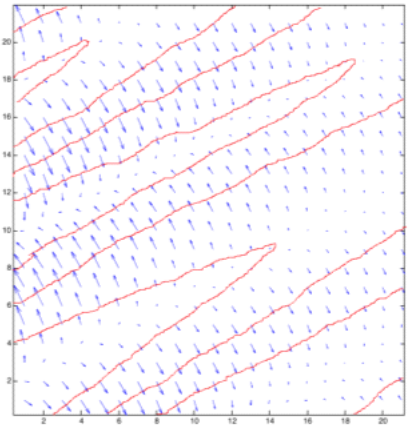
1. The partial derivative with respect to x, meaning the slope in the x direction (where every other independent variable is held constant); the second partial derivative with respect to x, meaning the curvature in the x direction (where every other independent variable is held constant); the directional derivative with respect to vector u, meaning the slope in the u direction; the gradient, meaning the maximum slope in its corresponding direction.
2. For example, the temperature depends on three spatial variables and time, i.e. T(x,y,z,t). The gradient would tell you in which direction (including time!) to move so as to increase the temperature the fastest.
3. At the bottom of a cone, the increase is equally fast in all directions.
4. The gradient points in the direction of most rapid increase—this direction must be perpendicular to the level curves of the function since the function is unchanged on a level curve.
5. The gradient could be decomposed into a component in $\hat{\mathbf{u}}$ direction and a component normal to that. Here, $\hat{\mathbf{u}}$ is an arbitrary direction that we are choosing. So the rate of change of the function in the $\hat{\mathbf{u}}$ direction is $\nabla f \cdot \hat{\mathbf{u}}$.
6. No, because you could be at a saddle point, for example.
7. Upward curvature in the x-direction; slope in y-direction increases with increasing x (and vice versa).

Solution 25.3



Solution 25.4

The one on the left could be a gradient; the right one could not because a path of gradient ascent would go in a circle. That is a problem because any point would have two gradient values associated with it—there is no way that it can come from the derivatives of a function. See Fig. 25.3 for a sketchy sketch of level curves for the left one.



Solution 25.5

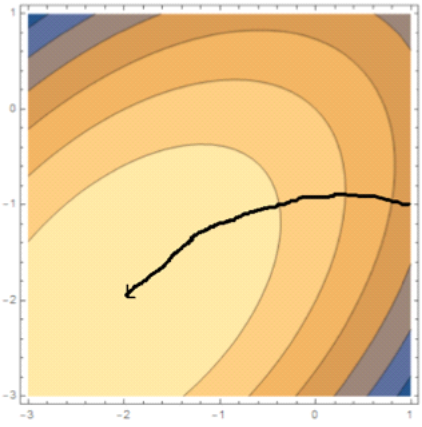
1.

Point	Sign of g_x	Sign of g_y
R	- or 0	+
S	-	+
T	-	0
U	0	-
V	+	0
W	-	-
X	0	-
Y	+	+

2. At Y because the contours are closest together, indicating the greatest change in the function value per unit distance normal to the level curves.

3. We are looking for a point on a contour line that has a tangent (or equivalently, slope) that is = $\mathbf{u}$. At that point, when we take the dot product of $\mathbf{u}$ and the gradient (perpendicular to the contour line), we will get 0. Point S, where the contour line is parallel to $\mathbf{u}$.

Solution 25.6

1. 

2. See the path on the contour plot.

3. $(-2-2x+y, -2+x-2y)$

4. Initial gradient is (-5,1), with a magnitude of roughly 5.1. To move about 0.5 meters, the initial multiplier λ_0 should be about 0.1 meters. This gives $\mathbf{r}_1 = \mathbf{r}_0 + \lambda_0 \nabla f(\mathbf{r}_0) = (0.5, -0.9)$.

5. We have to rotate from the y-axis to the direction of the gradient (-5,1) - this is roughly 1.37 rad (78.7 degrees) CCW. 0.5 rad/s would not exceed 0.3 m/s if rotated in place.

6. The time taken to reach $\mathbf{r}_1$ is determined by $t = \frac{|\mathbf{r}_1 - \mathbf{r}_0|}{V} \frac{1\text{m}}{3.28\text{ft}}$, where we remember to keep our units the same. The distance traveled is roughly $\sqrt{0.26} = 0.5$ meters which is roughly 0.15 m. At a speed of 0.1 m/s this would take 1.5 seconds.

7. The approach to these is similar to the first step.

8. The approach to these is similar to the first step.

9. The approach to these is similar to the first step.

Solution 25.7

1. Remind me; what must be true about the gradient at a maximum of a function? What does that tell you about the norm (length) of the gradient at that point? The gradient must be zero at the maximum of an objective function (in the unconstrained case). Therefore, the norm of the gradient is zero at this point.

2. Develop pseudo-code for the gradient ascent algorithm. Keep it general: Assume that you have f, a general scalar function of a vector $\mathbf{r}$. Suppose you have a MATLAB function that provides the function value `fun(r)` and gradient `grad(r)`. You are going to want to use a loop - a "while" loop would be a good choice. Introduce any parameters you need to make this pseudo-code work, and make sure to return the entire history of points r_i. What would be a reasonable stopping criterion? My stopping criterion is a tolerance on the norm of the gradient. See below for pseudo-code.

3. When you are happy with your pseudo-code, develop a script or function that:

   (a) Automatically determines the discrete points $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n$ given an initial point $\mathbf{r}_0$.

   (b) Can be tuned by varying δ and λ_0.

   (c) Takes any other parameters that you deemed necessary in writing your pseudo-code.

   (d) Returns the entire history of tested points $\mathbf{r}_i$.

   Zach prepared a solution available here, as well as an example of its use.

4. Test your gradient ascent algorithm on the function $f(x, y) = xy - x^2 - y^2 - 2x - 2y + 4$. To start, use the parameter values $r_0 = [0.7, 0.5]$, $\lambda_0 = 0.9, \delta = 0.9$. Does your algorithm reach the maximum of the function? How do you know? My code reaches its gradient termination criterion, and lies at the center of a high-value contour. This is evidence that the algorithm has converged to a local maximum of the function.

5. Visualize the sequence of points $\mathbf{r}_i$ for $i = 1, \ldots, n$ returned from your algorithm in the previous part. Do the points travel "directly" to their endpoint? Or do they "wander"? With the parameters above, my gradient ascent algorithm produces iterations $\mathbf{r}_i$ that tend to "wander"; this is common behavior for gradient ascent.

6. Test your gradient ascent algorithm on the function $f(x, y) = xy - x^2 - y^2 - 2x - 2y + 4$. To start, use the same parameter values as before, but set $\delta = 1.1$. Does your algorithm reach the maximum of the function? If not, what went wrong? My code produces an iteration sequence that diverges; the gradient does not converge to zero but rather grows very large.

Solution 25.8

1. Decompose this problem. What are the steps involved? You should have a decomposition that clearly explains the process of driving the NEATO along a discrete approximation to the path of steepest ascent.

2. Now develop the code for the NEATO. It may help to review your Bridge of Doom code....

3. Now drive your NEATO! You will startup the Neato simulator using the `flatland` environment. Remember, full instructions for using the simulator are available on the Meeto Your Neato page. For this challenge you'll want to run the flatland world using `qeasim`. In MATLAB, run this command. `>> qeasim start flatland_no_spawn` Paul has put together a solution file here

hw5

hw5

HW5

Exercise 25.6

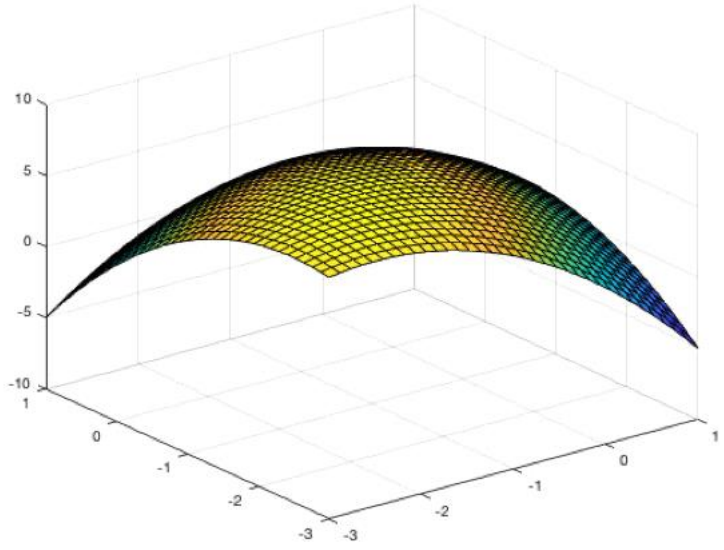
```
xyinterval = [-3, 1]
```

```
xyinterval = 1×2
    -3     1
```

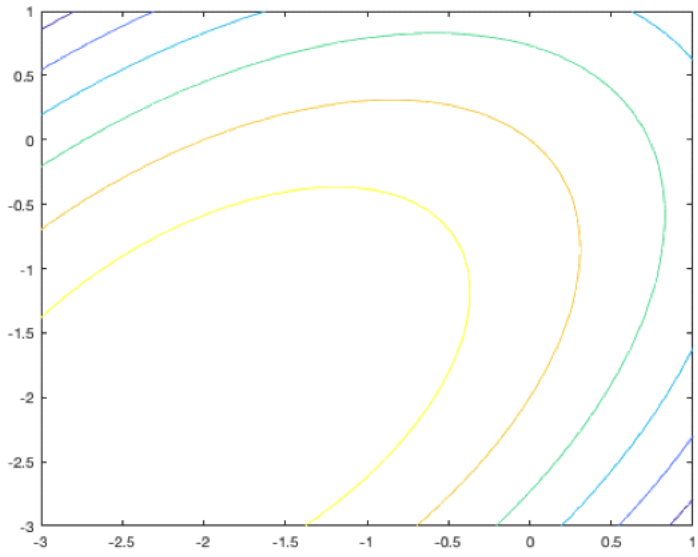
```
d = 0.235; % m, diameter of the neato
syms f(x,y) r(i) lambda(i) sigma
f(x,y) = (x * y) - x^2 - y^2 - (2 * x) - (2 * y) + 4
```

$$f(x, y) = -x^2 + x\,y - 2\,x - y^2 - 2\,y + 4$$

```
figure(1); clf;
fsurf(f, xyinterval)
zlim([-10, 10])
```




```
%%% Exercise 25.6.1
figure(2); clf;
fcontour(f, xyinterval)
```

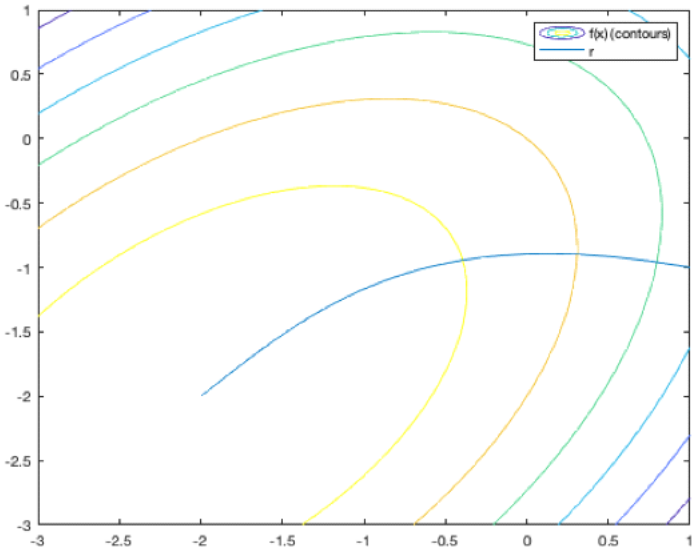



```
%%% Exercise 25.6.2 is drawn in OneNote

%%% Exercise 25.6.3
grad = matlabFunction([diff(f, x); diff(f, y)])
```

```
grad = function_handle with value:
    @(x,y)[x.*-2.0+y-2.0;x-y.*2.0-2.0]
```

```
%%% Exercise 25.6.4
r_0 = [1, -1];

grad(r_0(1), r_0(2)); % Gradient at r(0)


%%% Calculate the path

% These values were chosen experimentally
dtdi = 1; % every i is this many seconds
i_max = 200; % We'll calculate this many time steps. It may take fewer to reach the pe
sigma = 1;
lambda_0 = 0.05;
rs = zeros(i_max, 2);
lambdas = zeros(i_max, 1);

rs(1, :) = r_0;
lambdas(1) = lambda_0;
```

```
for i=2:i_max
    lambdas(i) = lambdas(i - 1) .* sigma;
    gradient = grad(rs(i - 1, 1), rs(i - 1, 2))';
    rs(i, :) = rs(i - 1, :) + (lambdas(i - 1) .* gradient);
end

figure(3); clf;
fcontour(f, xyinterval); hold on; % draw the contours again
plot(rs(:, 1), rs(:, 2)); legend("f(x) (contours)", "r"); hold off;
```




```
%%% Exercise 25.6.5

% To figure out the needed turning, we need to calculate the tangent line at i = 0
% I think there's a better, linear algebra way to do this
drs = diff(rs);
dts = ones(size(drs)) * dtdi;
vs = drs ./ dts;
thetas = atan(vs(:, 2) ./ vs(:, 1)) + pi;
theta_0 = pi / 2;

dtheta_0 = thetas(1) - theta_0;

fprintf("If the Neato started at (1, -1) pointing in the +Y direction, you'd need to r
```

```
If the Neato started at (1, -1) pointing in the +Y direction, you'd need to rotate it 1.37 radians CCW to
```

```
max_speed = 0.1; % m/s, max wheel velocity
max_omega = max_speed * (2 / d) % max rotational speed for given wheel velocity
```

```
max_omega = 0.8511
```

```
t_to_align = dtheta_0 / max_omega
```

```
t_to_align = 1.6137
```

```
%%% Exercise 25.6.6
speeds = vecnorm(vs, 2, 2)
```

```
speeds = 199×1
    0.2550
    0.2249
    0.1995
    0.1780
    0.1598
    0.1443
    0.1312
    0.1199
    0.1102
    0.1018
     ⋮
```