ASEN 3502 Computational Methods
Lab 4

Exact Due Date on Gradescope

General Instructions

1. You may not use any built-in optimization methods in your code unless explicitly permitted by the problem.
2. This lab is designed so that each member of your group can complete different tasks in parallel. This is intended to allow each student time to gain experience with programming and optimization.
3. You may use any programming language, and you may use different languages for different parts. However, the primary goal of this lab is understanding, and *you should use MATLAB unless every team member is enthusiastic about using something else.*
4. Adhere to the AI policy at asen3502.com/ai-policy.

Deliverables (at end of Week 3)

1. To the **Lab 4 Writeup** assignment on Gradescope: A **single pdf** writeup that contains all of the items marked `writeup`.
   - Make sure to clearly label all plots and figures and use correct units where appropriate.
   - The lab writeup does *not* need to include narrative text unless specifically requrested; it can just include the plot or results from each `writeup` task labeled with which task they are from.
2. To the **Lab 4 Code** assignment on Gradescope: Your **results.json** file and **source code** for all `code` items.
   - You may combine multiple parts into one file. However, please help the TAs by clearly labelling where everything is, e.g. use filenames like `week_1.jl`.
   - In this lab, you will *not* be graded based on documentation or comments in your code. (Though in practice, documentation is extremely important!)

Compiled 2026-11-06 12:06:57

1 Week 1: Optimization Techniques

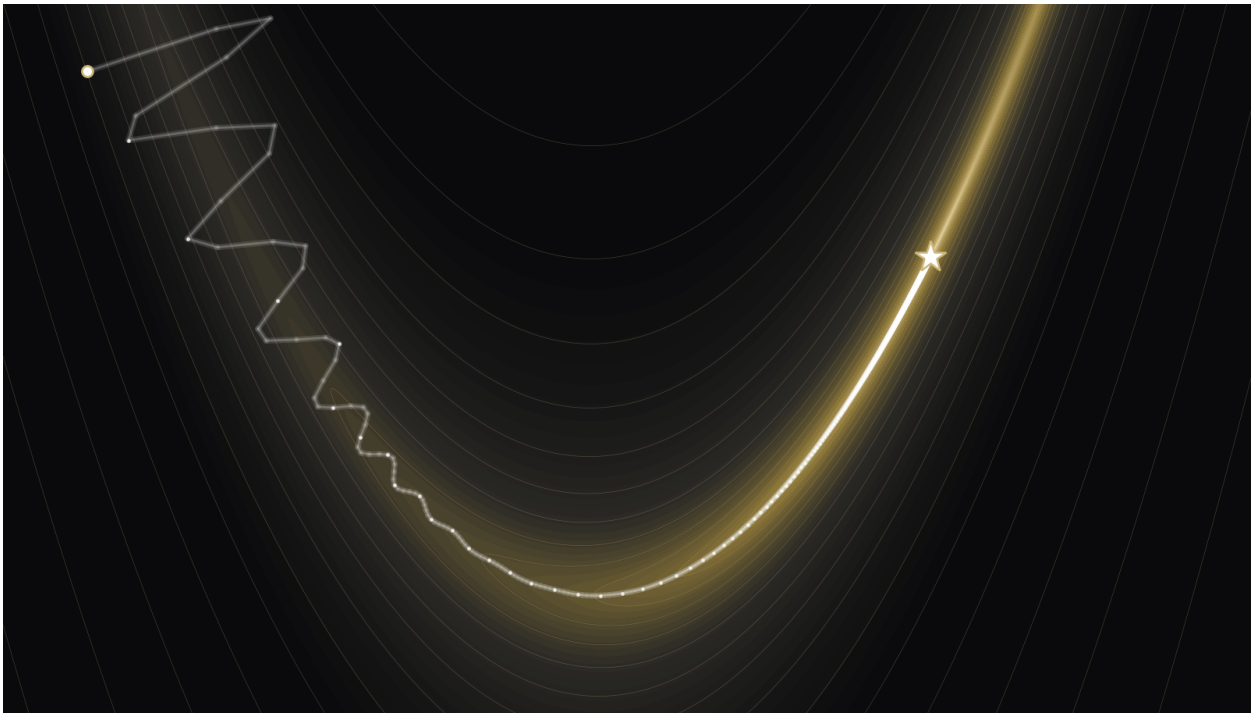


Figure 1: Gradient descent with momentum on the Rosenbrock function – overshooting the valley walls before converging on the minimum at (1, 1). This is one of the algorithms you will implement!

Model Formulation

For weeks 1 and 3 of this lab, we will be considering a fixed wing aircraft trim optimization problem. Consider a scenario where a Cessna 172 Skyhawk is climbing shortly after takeoff at $V = 30$ m/s. To fly steadily the aircraft must be at *trim*: the lift must balance the weight, and the pitching moment about the center of gravity must be zero (we will only consider longitudinal trim for this assignment). The pilot works with two variables to achieve this, the angle of attack α and the elevator deflection δ_e as shown in figure 2.

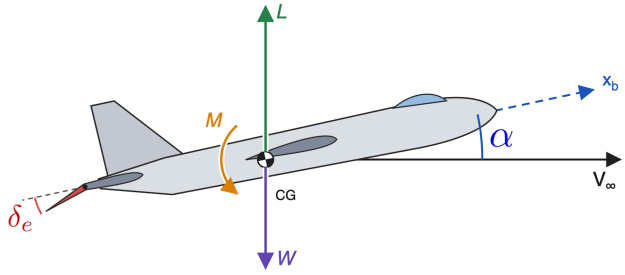


Figure 2: The aircraft model to be analyzed.

The lift and pitching moment coefficients are modeled as:

$$C_L(\alpha, \delta_e) = C_{L_0} + C_{L_\alpha}\alpha + C_{L_\delta}\delta_e + \Delta C_{L,p}(\alpha), \qquad C_M(\alpha, \delta_e) = C_{M_0} + C_{M_\alpha}\alpha + C_{M_\delta}\delta_e.$$

Compiled 2026-11-06 12:06:57

The first three terms of C_L are a linear model built from stability derivatives (if you don't recognize these equations - don't worry, the focus of this lab is not on understanding the equations). The last term is due to fast-moving air after takeoff behind the propeller which adds noticeable lift to the inboard wing. Wind-tunnel data taken with the propeller running is fitted by

$$\Delta C_{L,p}(\alpha) = \Delta C_{L,p0}\, e^{-(\alpha/\alpha_p)^2},$$

which is largest when the wing is aligned with the slipstream. We can now find trim by minimizing the sum of the squared lift error (as a fraction of the weight) and the squared pitching moment coefficient

$$f(\alpha, \delta_e) = \left(\frac{qSC_L(\alpha, \delta_e) - W}{W} \right)^2 + C_M(\alpha, \delta_e)^2,$$

so that both terms are zero at a trim point. Thus, our goal will be to **minimize the function** $f(\alpha, \delta_e)$, and ideally find values for α and δ_e such that $f = 0$.

Parameters (dynamic pressure $q = \frac{1}{2}\rho V^2$)

$\rho = 1.225$ kg/m^3	$V = 30$ m/s	$S = 16.2$ m^2	$W = 11346$ N
$C_{L_0} = 0.247$	$C_{L_\alpha} = 6.0$ rad^{-1}	$C_{L_\delta} = 0.43$ rad^{-1}	
$C_{M_0} = 0.1$	$C_{M_\alpha} = -1.3$ rad^{-1}	$C_{M_\delta} = -1.28$ rad^{-1}	
$\Delta C_{L,p0} = 0.6$	$\alpha_p = 0.05$ rad		

All angles are in radians. Convert to degrees only for plots and reported values. Throughout Weeks 1 and 3, use a convergence tolerance tol $= 10^{-6}$ on the step length $\|\vec{x}_{k+1} - \vec{x}_k\|$ and at most 1000 iterations, where $\vec{x} = [\alpha;\ \delta_e]$.

1.1 Define the objective function

writeup code

Code up $f(\alpha, \delta_e)$ as a function handle and plot its contours with the provided starter function `plot_contours(fig, f)`, where `fig` is the figure number and `f` is your function handle. Include the contour plot in your writeup and describe, in a couple sentences, how many local minima you see.

1.2 Determine the optimum with Matlab

writeup code

Find the true location of the minimum of f with MATLAB's `fminsearch` function (explicitly permitted for this question only). Use the options below when you call `fminsearch`:

```
opts = optimset('TolX', 1e-12, 'TolFun', 1e-16);
```

Report the reference optimum α^* and δ_e^* in degrees to four decimal places, and f^*. Explain in one or two sentences why f^* is essentially zero and what that means physically.

1.3 Derive and verify the gradient

writeup code

Derive the gradient $\nabla f = G = [\partial f/\partial\alpha;\ \partial f/\partial\delta_e]$ analytically and include the derivation in your writeup. You may use a symbolic derivative calculator, but you must document your steps and final derivation. Implement

```
G = trim_gradient(alpha, delta_e, params)
```

where G is a 2×1 vector, alpha is α, delta_e is δ_e (radians), and params is a struct containing any necessary constants.

Write a script `verify_gradient` that compares the gradient to a forward finite difference approximation

$$\frac{\partial f}{\partial x_k} \approx \frac{f(\vec{x} + h\vec{e}_k) - f(\vec{x})}{h}$$

Use $h_k = \sqrt{\epsilon}\,|x_k|$, where $\epsilon \approx 2.2 \times 10^{-16}$ is machine precision (`eps` in MATLAB).

1.4 Implement gradient descent

writeup code

Implement the gradient descent algorithm with a backtracking line search. This and every method in the next section should use the interface

```
[position, path] = method_name(g,initial_guess,tol,max_iterations,{method-specific})
```

where `position` is the final result $[\alpha;\ \delta_e]$ (2×1, radians), `path` is a $2 \times (k+1)$ array holding every $[\alpha;\ \delta_e]$ position iteration, `g` is a function handle for your `trim_gradient` function, the starting point `initial_guess`, the minimum step size `tol`, the maximum iterations `max_iterations`, and `{method-specific}` is where you can add any method-specific parameter values (for example `l_init`, the initial step size).

Using a `tol` $= 1 \cdot 10^{-6}$, `max_iterations` $= 1000$, backtracking factor $\rho = 0.5$, and sufficient decrease factor $\mu_1 = 10^{-4}$, use your gradient descent function to answer the following questions:

a. Start at $\vec{x}_0 = [0;\ 0]$ with $l_{\text{init}} = 0.01$. Plot the path on your contour plot and report where it stops, and the number of iterations. Is this a minimum? Is it a trim point? Explain.

b. Start at $\vec{x}_0 = [0.2;\ 0.1]$ with $l_{\text{init}} = 0.045$. Report the endpoint and iteration count.

1.5 Implement Newton's method

writeup code

Implement Newton's method with a backtracking line search, using the interface above. The Hessian is provided in `trim_hessian.m`. You may tune the line search parameters as you see fit, just report the values you used. Start from $\vec{x}_0 = [0.2;\ 0.1]$ rad.

In your writeup include one contour plot clearly showing the paths of gradient descent from Section 1.4 part b) and of Newton's method. Also, include a table with each method's final α and δ_e (degrees), the number of iterations, and the distance from the reference optimum of Section 1.2. Which method stopped farthest from the reference optimum? Why does a convergence test on the step length not guarantee that you are close to the minimum?

2 Week 2: UAV Range Design Under Uncertainty

This week, we will use a *library* optimizer (such as `fmincon` in MATLAB) rather than building an optimizer by hand. Consider a small electric fixed-wing UAV which flies an out-and-back survey mission: out to a target against the wind, back with the wind, and lands with a 20% battery reserve. We want to find the airframe and cruise speed parameters that maximize the total ground distance (R) flown.

Design variables $\vec{x} = [S,\ AR,\ m_b,\ V]$

symbol	definition	units	bounds
S	wing area	m^2	$0.3 \le S \le 3$
AR	aspect ratio	–	$4 \le AR \le 20$
m_b	battery mass	kg	$0.2 \le m_b \le 10$
V	cruise airspeed	m/s	$8 \le V \le 40$

Uncertain parameters $\vec{\theta} = [m_{\text{pay}},\ C_{D_0},\ e_b,\ V_w]$. For now, use the nominal values provided below. We will later use the distributions that appear in Section 2.3 instead.

symbol	meaning	units	nominal
m_{pay}	payload mass	kg	2.0
C_{D_0}	zero-lift drag coefficient	–	0.0325
e_b	battery specific energy	J/kg	648000
V_w	wind speed along the route	m/s	4.0

Model

$$m_w = k_w\, S^{1.2}\, AR^{0.9}, \qquad W = mg = g(m_{\text{pay}} + m_{\text{fix}} + m_b + m_w)$$

$$q = \tfrac{1}{2}\rho V^2, \qquad e = 1.78\left(1 - 0.045\, AR^{0.68}\right) - 0.64, \qquad D = qSC_{D_0} + \frac{W^2}{qS\,\pi e\, AR}$$

$$\eta = \eta_{\max}\frac{2r}{1+r^2}, \qquad r = \frac{V}{V_\eta}, \qquad E = \frac{(1 - f_{\text{res}})\, e_b\, m_b\, \eta}{D\, V}$$

$$R = E\, V\left(1 - \frac{V_w^2}{V^2}\right) \tag{1}$$

Here m_w is the empirical wing mass, W is the total weight of the UAV, q is the dynamic pressure, e is Raymer's straight-wing Oswald efficiency, D is the total drag, η is the propulsive efficiency of at design speed V_η, r is the flight-to-slipstream velocity ratio, E is the endurance, and R is the range.

Constraints All four must satisfy $g_i(\vec{x}) \le 0$:

$$g_1 = k_{\text{stall}}\sqrt{\frac{2W}{\rho\, S\, C_{L_{\max}}}} - V \qquad \text{(stall margin)} \tag{2}$$

$$g_2 = RC_{\min} - \frac{\eta P_{\max} - DV}{W} \qquad \text{(rate of climb)} \tag{3}$$

$$g_3 = \sqrt{AR\, S} - b_{\max} \qquad \text{(wingspan limit)} \tag{4}$$

$$g_4 = m - m_{\max} \qquad \text{(maximum take-off mass)} \tag{5}$$

Fixed parameters

$g = 9.81$ m/s^2	$\rho = 1.225$ kg/m^3	$m_{\text{fix}} = 1.5$ kg	$k_w = 0.35$
$\eta_{\max} = 0.70$	$V_\eta = 18$ m/s	$f_{\text{res}} = 0.20$	$C_{L_{\max}} = 1.3$
$k_{\text{stall}} = 1.25$	$P_{\max} = 500$ W	$RC_{\min} = 2.0$ m/s	$b_{\max} = 4.0$ m
$m_{\max} = 15$ kg			

2.1 Build the UAV model

writeup code

Implement the model provided above with the following interface

```
[R, g] = uav_range(x, theta, params)
```

where `R` is an $n \times 1$ range vector, `g` is an $n \times 4$ constraint matrix, `x` is a 1×4 vector, and `theta` is an $n \times 4$ matrix (each row is a parameter set for which to calculate a range and set of constraint values). You can retrieve the `params` struct by calling the provided `default_params.m` function. Vectorize over the rows of `theta`. You will later need to call `uav_range` with many rows inside an optimizer.

Now, use your model to calculate the range, R, in meters as well as the calculated constraints, $\vec{g}$, for each of the three test points below to six significant digits. Report your values in a table. It is a good idea to check this answer with the instructional team!

$\vec{x}$	$\vec{\theta}$
[0.5, 12, 4, 20]	[2.0, 0.0325, 648000, 4]
[1.0, 8, 2, 15]	[2.3, 0.035, 612000, 6]
[0.4, 18, 5.5, 22]	[1.7, 0.028, 684000, 2]

2.2 Find the deterministic optimum

writeup code

With $\vec{\theta}$ at its nominal values, solve

$$\max_{\vec{x}} \; R(\vec{x}) \quad \text{s.t.} \quad g_i(\vec{x}) \le 0 \text{ for } i = [1,4] \quad \text{within the bounds on } \vec{x}.$$

with your library optimizer from at least four starting points ($\vec{x}_0$) inside the bounds. Hint: your optimizer *minimizes*, so it will behave better if you minimize a modified range function such as $-R/10^5$. If you use `fmincon`, here are some algorithm options to start with:

```
options = optimoptions('fmincon', 'Algorithm', algorithm, 'Display', 'off', ...
    'OptimalityTolerance', 1e-10, 'StepTolerance', 1e-12, 'ConstraintTolerance', 1e-10, ...
    'MaxIterations', 500, 'MaxFunctionEvaluations', 5000);
```

Report your optimized $\vec{x}^*$, R^*, and which constraints are active (which $g_i = 0$). The optimum is $R^* = 273.170$ km, your solution should match this very closely.

2.3 Generate the theta distribution

writeup code

Introduce uncertain parameters as follows (the numbers below are the *nominal scenario*; the challenge will change them):

parameter	distribution	nominal scenario
m_{pay} [kg]	normal $\mathcal{N}(\mu, \sigma)$	$\mu = 2.0$, $\sigma = 0.3$
C_{D_0}	uniform $\mathcal{U}(\min, \max)$	$\min = 0.025$, $\max = 0.040$
e_b [J/kg]	normal $\mathcal{N}(\mu, \sigma)$	$\mu = 648000$, $\sigma = 54000$
V_w [m/s]	uniform $\mathcal{U}(0, V_{w,\max})$	$V_{w,\max} = 8$

Write a sampler that returns N rows of $\vec{\theta}$ for a given scenario (the scenario file `scenario.m` gives you the field names and a MATLAB version you may translate). Run your sampler for $N = 10,000$ and plot the resulting distributions of $m_{\text{pay}}, C_{D_0}, e_b$, and V_w. Are these distributions what you expect?

2.4 Challenge problem

code

Your UAV design will now fly 10^6 survey missions, each of which requires a range of 300 km. The parameters of each mission are random, distributed as

parameter	distribution	challenge missions
m_{pay} [kg]	normal $\mathcal{N}(\mu, \sigma)$	$\mu = 1.5$, $\sigma = 0.1$
C_{D_0}	uniform $\mathcal{U}(\min, \max)$	$\min = 0.025$, $\max = 0.0325$
e_b [J/kg]	normal $\mathcal{N}(\mu, \sigma)$	$\mu = 648000$, $\sigma = 54000$
V_w [m/s]	uniform $\mathcal{U}(0, V_{w,\max})$	$V_{w,\max} = 8$

A mission whose parameters violate any constraint fails and flies 0 km. Your score is the mean *excess* range over the 300 km requirement,

$$\text{score} = \mathbb{E}\big[\, R(\vec{x}, \vec{\theta}) \cdot \mathbf{1}\{\vec{g}(\vec{x}, \vec{\theta}) \le 0\} \,\big] - 300 \text{ km},$$

and a design with 100 or more failed missions (out of 10^6) is too risky and is not ranked. Submit a design $\vec{x} = [S,\ AR,\ m_b,\ V]$ with the provided function

```
evaluate([S AR m_b V], 'your.gradescope.email@colorado.edu')
```

which prints your mean excess range and number of failed missions locally and, if the design is not too risky, saves your results to `results.json`. Submit `results.json` to Gradescope, where your mean excess range appears on the class leaderboard. You may call `evaluate` as often as you like, and the missions are the same on every call. SStart by submitting your deterministic optimum from Section 2.2. *Hint: look into running a Monte Carlo with your sampler for a higher score.*

Advanced Instead of flying one design on every mission, you may keep one airframe (S and AR) and choose the battery mass and airspeed for each mission from estimates of that mission's parameters. Submit with

```
evaluate({S, AR, @plan}, 'your.gradescope.email@colorado.edu')
```

where `plan` is a function with the interface `[m_b, V] = plan(theta)`. Here `theta` is an $n \times 4$ matrix ($n = 10^6$, one row per mission) of estimates of $[m_{\text{pay}},\ C_{D_0},\ e_b,\ V_w]$, equal to the true values plus noise with standard deviations [0.02 kg, 0.001, 20000 J/kg, 1 m/s], and `m_b` and `V` are $n \times 1$ vectors (or scalars) of the battery mass loaded and the airspeed flown on each mission. Vectorize your plan over the rows of `theta_est`.

The general instructions for the lab apply to this section: You must write the code yourself, and the AI policy applies. The winner will explain their solution to the class, so make sure to understand it!

3 Week 3: ADAM and Constrained Optimization

Note on notation. The symbols used in lecture for ADAM and the augmented Lagrangian differ from the ones in this lab. Wherever they differ, a restatement in the lecture notation follows the original text in purple. The algorithms are identical; only the names of the symbols change.

This week returns to the aircraft trim problem of Week 1, with the same f, `trim_gradient`, convergence tolerance and maximum number of iterations.

3.1 Implement ADAM

writeup code

Implement the ADAM method with the same interface as your Week 1 methods. Start at $\vec{x}_0 = [0.2;\ 0.1]$ rad with $l = 0.02$, $\beta_1 = 0.3$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$. Plot the path on your contour plot, and report the final α and δ_e in degrees, the number of iterations, and the distance from the reference optimum of Section 1.2.

Lecture notation: step factor $\alpha = 0.02$ (lab l), decay rates $\gamma_v = 0.3$ (lab β_1) and $\gamma_s = 0.99$ (lab β_2), and small constant $\epsilon = 10^{-8}$. The ADAM step factor α is unrelated to the angle of attack α in $\vec{x}$. With the biased and corrected momentum $\vec{v}$, $\hat{\vec{v}}$ and squared gradient $\vec{s}$, $\hat{\vec{s}}$ from the lecture pseudocode, the update is

$$\vec{x}_{k+1} \leftarrow \vec{x}_k - \frac{\alpha}{\epsilon + \sqrt{\hat{\vec{s}}_{k+1}}} \odot \hat{\vec{v}}_{k+1},$$

with all products, square roots, and divisions elementwise.

3.2 Implement the Augmented Lagrangian Method

writeup code

Now suppose the angle of attack must stay below $\alpha_{\max} = 10°$ and the elevator deflection below $\delta_{\max} = -5°$. The trim point of Section 1.2 violates the first of these, so the constrained optimum is not a trim point. Write the two constraints as $\vec{c}(\vec{x}) \le \vec{0}$ with

$$\vec{c}(\alpha, \delta_e) = \begin{bmatrix} \alpha - \alpha_{\max} \\ \delta_e - \delta_{\max} \end{bmatrix}.$$

Lecture notation: these are two inequality constraints $\vec{g}(\vec{x}) \le \vec{0}$ with

$$\vec{g}(\alpha, \delta_e) = \begin{bmatrix} \alpha - \alpha_{\max} \\ \delta_e - \delta_{\max} \end{bmatrix},$$

and there are no equality constraints $\vec{h}(\vec{x}) = \vec{0}$.

For equality constraints $c_i(\vec{x}) = 0$, the augmented Lagrangian method minimizes, at outer iteration k,

$$\Phi_k(\vec{x}) = f(\vec{x}) + \sum_{i \text{ active}} \left[\lambda_i c_i(\vec{x}) + \frac{\mu_k}{2} c_i(\vec{x})^2\right] - \sum_{i \text{ inactive}} \frac{\lambda_i^2}{2\mu_k}, \tag{6}$$

where the λ_i are estimates of the Lagrange multipliers and μ_k is a penalty parameter. Constraint i is treated as *active* when $\lambda_i + \mu_k c_i(\vec{x}) > 0$, which covers every violated constraint and any constraint whose multiplier is still positive. The gradient is

$$\nabla\Phi_k(\vec{x}) = \nabla f(\vec{x}) + \sum_{i \text{ active}} \left(\lambda_i + \mu_k c_i(\vec{x})\right) \nabla c_i. \tag{7}$$

Lecture notation: the augmented Lagrangian is written $\hat{f}(\vec{x};\,\bar{\vec{\lambda}},\mu)$ (lab Φ_k) and is built from the *modified* constraint vector $\bar{\vec{g}}$,

$$\bar{g}_j(\vec{x};\,\bar{\vec{\lambda}}) = \begin{cases} g_j(\vec{x}) & \text{if } g_j(\vec{x}) \geq -\bar{\lambda}_j/\mu \\ -\bar{\lambda}_j/\mu & \text{otherwise,} \end{cases} \qquad \hat{f}(\vec{x};\,\bar{\vec{\lambda}},\mu) = f(\vec{x}) + \bar{\vec{\lambda}}^{\mathsf{T}}\bar{\vec{g}}(\vec{x};\,\bar{\vec{\lambda}}) + \frac{\mu}{2}\,\|\bar{\vec{g}}(\vec{x};\,\bar{\vec{\lambda}})\|_2^2.$$

The first case of $\bar{g}_j$ is the lab's *active* set ($g_j \geq -\bar{\lambda}_j/\mu$ is the same test as $\bar{\lambda}_j + \mu\, g_j \geq 0$). In the second (*inactive*) case, $\bar{\lambda}_j\bar{g}_j + \frac{\mu}{2}\bar{g}_j^2$ reduces to $-\bar{\lambda}_j^2/(2\mu)$, which is the lab's inactive sum. The gradient is

$$\nabla\hat{f}(\vec{x};\,\bar{\vec{\lambda}},\mu) = \nabla f(\vec{x}) + \sum_{j:\ \bar{g}_j = g_j} \big(\bar{\lambda}_j + \mu\, g_j(\vec{x})\big)\,\nabla g_j(\vec{x}),$$

since the inactive terms do not depend on $\vec{x}$.

Start the first outer iteration from $\vec{x}_0 = [0.15;\ -0.1]$ rad with $\bar{\vec{\lambda}} = \vec{0}$ and $\mu_0 = 10$ and use $\mu_{k+1} = 4\mu_k$. Minimize each Φ_k with your ADAM function, but pass it in as a function handle with the interface

```
[position, path] = augmented_lagrange(g,initial_guess,tol,max_iterations,f,...)
```

(where `g` and `f` are now the gradient and value of Φ_k). Inside the augmented Lagrangian use $l = 0.0005$, $\beta_1 = 0.7$ and $\beta_2 = 0.99$.

Lecture notation: initial multipliers $\bar{\vec{\lambda}}_0 = \vec{0}$, initial penalty $\mu_0 = 10$, and penalty increase factor $\rho = 4$, so that $\mu_{k+1} \leftarrow \rho\,\mu_k$ (this ρ is not the backtracking factor of Week 1). Each outer iteration minimizes $\hat{f}(\vec{x};\,\bar{\vec{\lambda}}_k,\mu_k)$ with ADAM starting from $\vec{x}_k$ to get $\vec{x}_k^*$, then updates the multipliers with $\bar{\vec{\lambda}}_{k+1} \leftarrow \bar{\vec{\lambda}}_k + \mu_k\,\bar{\vec{g}}(\vec{x}_k^*;\,\bar{\vec{\lambda}}_k)$ and sets $\vec{x}_{k+1} \leftarrow \vec{x}_k^*$. In the interface above, the handles `g` and `f` return $\nabla\hat{f}$ and $\hat{f}$; the handle `g` is not the constraint vector $\vec{g}$. Inside the augmented Lagrangian the ADAM parameters are $\alpha = 0.0005$, $\gamma_v = 0.7$ and $\gamma_s = 0.99$.

Plot the path on your contour plot with both constraint lines drawn, and report the final α and δ_e in degrees, the total number of iterations, and which constraint is active. Show that both constraints are satisfied. Does your path ever leave the feasible region? Where?