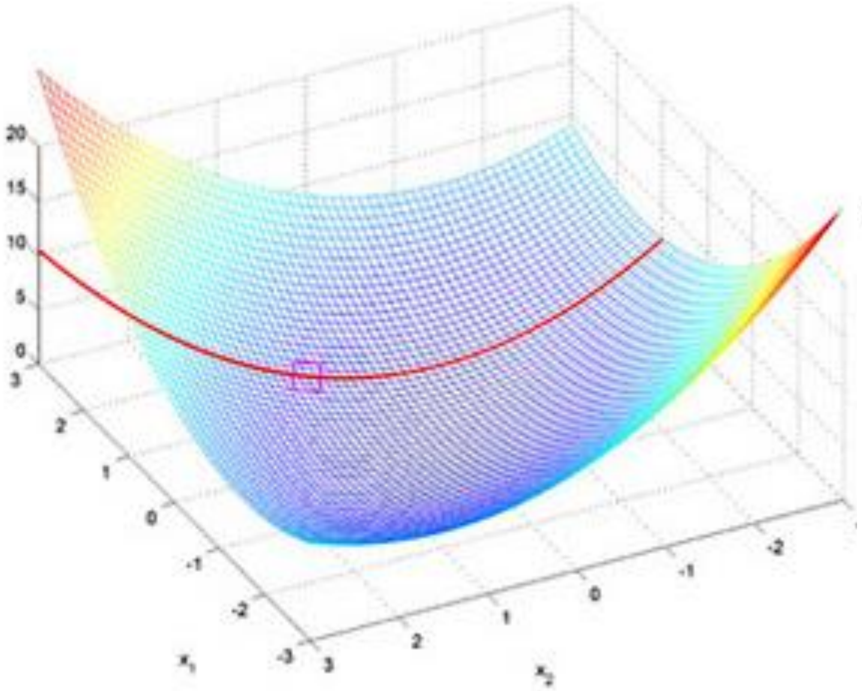
# MEGR 3090/7090/8090: Advanced Optimal Control

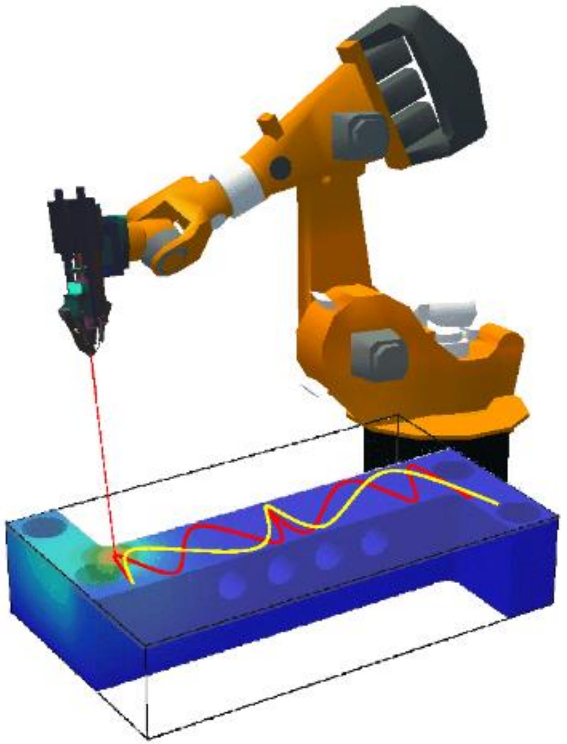
UNC CHARLOTTE



$$\boxed{V_n(\mathbf{x}_n) = \min_{\{\mathbf{u}_n,\mathbf{u}_{n+1},\cdots,\mathbf{u}_{N-1}\}} \left[\frac{1}{2}\sum_{k=n}^{N-1}\left(\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k\right) + \frac{1}{2}\mathbf{x}_N^T\mathbf{Q}_N\mathbf{x}_N\right]}$$

$$\begin{aligned}
V_n(\mathbf{x}_n) &= \min_{\{\mathbf{u}_n,\mathbf{u}_{n+1},\cdots,\mathbf{u}_{N-1}\}} \left[\frac{1}{2}\sum_{k=n}^{N-1}\left(\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k\right) + \frac{1}{2}\mathbf{x}_N^T\mathbf{Q}_N\mathbf{x}_N\right] \\
&= \min_{\mathbf{u}_n}\left[\frac{1}{2}\left(\mathbf{x}_n^T\mathbf{Q}_n\mathbf{x}_n + \mathbf{u}_n^T\mathbf{R}\mathbf{u}_n\right) + \underbrace{\min_{\{\mathbf{u}_{n+1},\cdots,\mathbf{u}_{N-1}\}}\left[\frac{1}{2}\sum_{k=n+1}^{N-1}\left(\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k\right) + \frac{1}{2}\mathbf{x}_N^T\mathbf{Q}_N\mathbf{x}_N\right]}_{V_{n+1}(\mathbf{x}_{n+1})}\right] \\
&= \min_{\mathbf{u}_n}\left[\frac{1}{2}\left(\mathbf{x}_n^T\mathbf{Q}_n\mathbf{x}_n + \mathbf{u}_n^T\mathbf{R}\mathbf{u}_n\right) + V_{n+1}(\mathbf{x}_{n+1})\right]
\end{aligned}$$

$$\boxed{V_n(\mathbf{x}_n) = \min_{\mathbf{u}_n}\left[\frac{1}{2}\left(\mathbf{x}_n^T\mathbf{Q}_n\mathbf{x}_n + \mathbf{u}_n^T\mathbf{R}\mathbf{u}_n\right) + V_{n+1}(\mathbf{x}_{n+1})\right]}$$



**Lecture 12**

**September 28, 2017**

# Review – Unconstrained Optimization Problem and Optimality Conditions

UNC CHARLOTTE

**General optimization problem:** Minimize $J(\mathbf{u})$

**Necessary condition for optimum:** $\nabla J(\mathbf{u}^*) = 0$

**Sufficiency conditions:**

- $\mathbf{u}^*$ is a local minimizer if $J(\mathbf{u})$ is ***locally convex*** around $\mathbf{u}^*$
- $\mathbf{u}^*$ is a global minimizer if $J(\mathbf{u})$ is ***globally convex***

# Review – Solution Techniques for Unconstrained Problems

## Special case – quadratic objective function given by $J(\mathbf{u}) = \mathbf{u}^T Q \mathbf{u} + \mathbf{r}^T \mathbf{u}$ , $Q \succ 0$:

- Unique global minimizer given by $\mathbf{u}^* = -\frac{1}{2} Q^{-1} \mathbf{r}$

## General case – Gradient descent method:

- At each iteration (k), approximate $J(\mathbf{u})$ linearly around $u_k$: $J(\mathbf{u}) \approx J(\mathbf{u}_k) + \nabla J(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k)$
- Choose the next candidate point ($\mathbf{u}_k$) in the direction of $\nabla J(\mathbf{u}_k)$, using a line search to determine $\alpha_k$:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \alpha_k \nabla J(\mathbf{u}_k)$$

## Newton's method (unconstrained sequential quadratic programming (SQP)):

- At each iteration (k), approximate $J(\mathbf{u})$ quadratically around $\mathbf{u}_k$:

$$J(\mathbf{u}) \approx J(\mathbf{u}_k) + \nabla J(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) + 0.5(\mathbf{u} - \mathbf{u}_k)^T H(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k)$$

- Choose the next candidate point ($\mathbf{u}_k$) in the direction of the minimizer of the quadratic approximation, either using a line search to choose $\alpha_k$ or taking $\alpha_k = 1$ ("pure" Newton's method):

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \alpha_k H^{-1}(\mathbf{u}_k) \nabla J(\mathbf{u}_k)$$

# Review – Solution Techniques for Unconstrained Problems

Suppose that $J(\underline{u}) = J_0 + \underline{r}^T \underline{u} + \underline{u}^T Q \underline{u}$

$$\nabla J = \underline{r}^T + 2\underline{u}^T Q$$

$$\Rightarrow \underline{r}^T + 2\underline{u}^{*T} Q = \underline{0}$$

$$\Rightarrow 2\underline{u}^{*T} Q = -\underline{r}^T$$

$$\Rightarrow \underline{u}^* = -\frac{1}{2} Q^{-1} \underline{r}$$

# Review – Constrained Optimization Problem and Optimality Conditions

**Optimization problem:** Minimize $J(\mathbf{u})$

Subject to: $g(\mathbf{u}) \leq \mathbf{0}$

$h(\mathbf{u}) = \mathbf{0}$

**Optimality requirements:** $\nabla J(\mathbf{u}^*) + \boldsymbol{\mu}^T \nabla g(\mathbf{u}^*) + \boldsymbol{\lambda}^T \nabla h(\mathbf{u}^*) = 0$

Complementary slackness: $\mu_i g_i(\mathbf{u}^*) = 0, \forall i$

Combines additional terms from inequality and equality constraints

Feasibility: $\mu_i \geq 0, \forall i$

Constraint satisfaction: $g(\mathbf{u}^*) \leq 0$

$h(\mathbf{u}^*) = \mathbf{0}$

**Note**: If $\mathbf{u}^*$ is a minimizer, then the KKT conditions will be satisfied (i.e., they are necessary). In general, however, satisfaction of the KKT conditions does not guarantee that $\mathbf{u}^*$ is a unique global minimizer (or even a minimizer)!

# Review – Constrained Optimization Problem and Optimality Conditions

The Lagrangian, $L(\underline{u}, \underline{\mu}, \underline{\lambda})$, is given by:

$$L(\underline{u}, \underline{\mu}, \underline{\lambda}) = J(\underline{u}) + \underline{\mu}^T g(\underline{u}) + \underline{\lambda}^T h(\underline{u})$$

$$\nabla L_{\underline{u}} = \nabla J(\underline{u}) + \underline{\mu}^T \nabla g(\underline{u}) + \underline{\lambda}^T \nabla h(\underline{u})$$

# Review – Optimization Techniques for *Special Case* Constrained Optimization Problems

**Linear program (LP):** Minimize: $J(\mathbf{u}) = \boldsymbol{k}^T\mathbf{u}$ Subject to: $A_1\mathbf{u} - \mathbf{b}_1 \leq \mathbf{0}$

$A_2\mathbf{u} - \mathbf{b}_2 = \mathbf{0}$

## LP solution – synopsis:

- Simplex method – Start at a vertex of the constraint set, then move to a vertex in the direction of decreasing $J$
- Implemented in MATLAB using the syntax: `[u_opt,J_opt] = linprog(k,A1,b1,A2,b2,[],[],options);`

**Quadratic program (QP):** Minimize: $J(\mathbf{u}) = \mathbf{u}^T Q\mathbf{u} + R\mathbf{u}$ Subject to: $A_1\mathbf{u} - \mathbf{b}_1 \leq \mathbf{0}$

$A_2\mathbf{u} - \mathbf{b}_2 = \mathbf{0}$

## QP solutions:

- Active set methods – Generalization of the simplex method…solutions can lie on constraint surfaces, not just vertices
- Interior point method – Soften inequality constraints with a barrier function
- QP can be implemented in MATLAB using the syntax:

```
[u_opt,J_opt] = quadprog(Q,r,A1,b1,A2,b2);
```

# Review – Optimization Techniques for ***Special Case*** Constrained Optimization Problems

QP: Minimize $J(\underline{u}) = \underline{u}^T Q \underline{u} + \underline{r}^T \underline{u}$

subj. to: $A_1 \underline{u} - \underline{b}_1 \leq \underline{0}$

$A_2 \underline{u} - \underline{b}_2 = 0$

$$L(\underline{u}, \underline{\mu}, \underline{\lambda}) = \underline{u}^T Q \underline{u} + \underline{r}^T u + \underline{\mu}^T (A_1 \underline{u} - \underline{b}_1) +$$

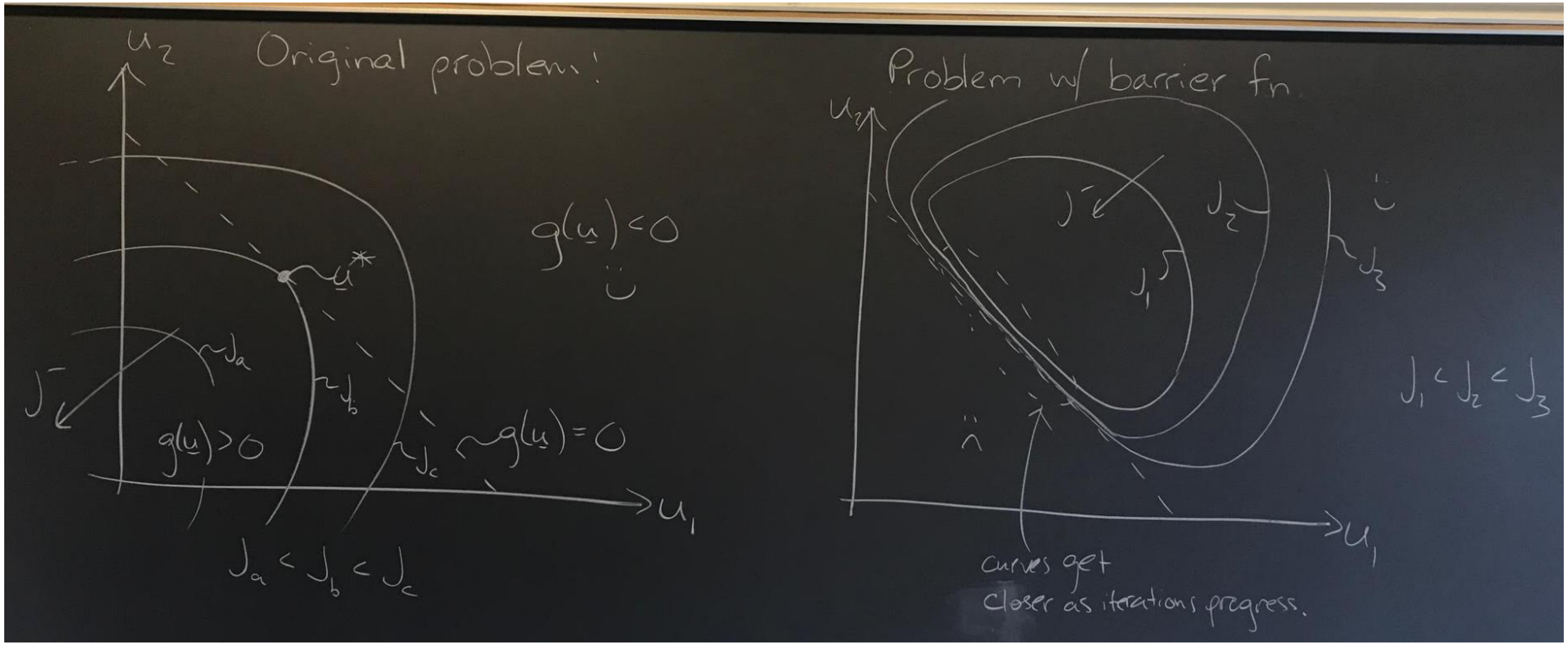
$$\underline{\lambda}^T (A_2 \underline{u} - b_2)$$

# Review – Optimization Techniques for ***Special Case*** Constrained Optimization Problems

$$\nabla L_{\underline{u}} = 2\underline{u}^T Q + \underline{\mu}^T A_1 + \underline{\lambda}^T A_2$$

$$\nabla L_{\underline{u}}(\underline{u}^*) = \underbrace{2\underline{u}^{*T} Q + \underline{\mu}^T A_1 + \underline{\lambda}^T A_2}_{\text{Linear in } \underline{u}} = 0$$

# Review – Optimization Techniques for ***Special Case*** Constrained Optimization Problems




Active set methods & interior point methods work very well when the only nonlinearities in the KKT conditions come from complementary slackness.

# Sequential Quadratic Programming (SQP) – A ***General Case*** Technique for Constrained Optimization Problem

**General nonlinear optimization problem (NLP):** Minimize $J(\mathbf{u})$

Subject to: $g(\mathbf{u}) \leq \mathbf{0}$

$h(\mathbf{u}) = \mathbf{0}$

**Sequential quadratic programming (SQP) – basic process:**

- At each iteration (k), approximate the **optimization problem** (objective function and constraints) as a quadratic program (QP) – this is called the ***QP subproblem***
- Solve the QP (we have tools for this from last lecture)
- Apply a correction to deal with possible constraint violations (due to the fact that the original optimization problem was ***approximated*** as a QP) – this is tricky!
- Repeat

# Formulating the QP Subproblem - Details

**General optimization problem (reminder):** Minimize $J(\mathbf{u})$ Subject to: $g(\mathbf{u}) \leq \mathbf{0}$

$h(\mathbf{u}) = \mathbf{0}$

**Local approximations of $J(\mathbf{u})$, $g(\mathbf{u})$, and $h(\mathbf{u})$ (based on a Taylor expansion):**

$$J(\mathbf{u}) \approx J(\mathbf{u}_k) + \nabla J(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) + 0.5(\mathbf{u} - \mathbf{u}_k)^T H(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k)$$

$$g(\mathbf{u}) \approx g(\mathbf{u}_k) + \nabla g(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) \qquad h(\mathbf{u}) \approx h(\mathbf{u}_k) + \nabla h(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k)$$

**Resulting QP subproblem:**

$$\mathbf{u}_{k+1} = \arg\min_{\mathbf{u}} (J(\mathbf{u}_k) + \nabla J(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) + 0.5(\mathbf{u} - \mathbf{u}_k)^T H(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k)\ )$$

Subject to:

$$g(\mathbf{u}_k) + \nabla g(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) \leq 0 \qquad h(\mathbf{u}_k) + \nabla h(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) = 0$$

# Formulating the QP Subproblem - Details

Remember: QP requires (i) quadratic obj. fn

& linear constraints

Given $J(\underline{u})$, $2^{nd}$ order Taylor expansion around

$\underline{u}_k$ is: $J(\underline{u}) \approx J(\underline{u}_k) + \nabla J(\underline{u}_k)(\underline{u} - \underline{u}_k) +$

iter.

$\frac{1}{2}(\underline{u} - \underline{u}_k)^T H(\underline{u}_k)(\underline{u} - \underline{u}_k)$

# Formulating the QP Subproblem - Details

UNC CHARLOTTE

Given $g(\underline{u})$, 1st order Taylor expansion around $\underline{u}_k$ is:

$$g(\underline{u}) \approx g(\underline{u}_k) + Dg(\underline{u}_k)(\underline{u} - \underline{u}_k)$$
$$h(\underline{u}) \approx h(\underline{u}_k) + Dh(\underline{u}_k)(\underline{u} - \underline{u}_k)$$

QP subproblem:

decision var.

$$\text{Minimize} \quad \nabla J(\underline{u}_k)(\underline{u} - \underline{u}_k) + \frac{1}{2}(\underline{u} - \underline{u}_k)^T H(\underline{u}_k)(\underline{u} - \underline{u}_k)$$

$$\text{Subj. to} \quad g(\underline{u}_k) + Dg(\underline{u}_k)(\underline{u} - \underline{u}_k) \leq \underline{0}$$
$$h(\underline{u}_k) + Dh(\underline{u}_k)(\underline{u} - \underline{u}_k) = \underline{0}$$

# Formulating the QP Subproblem - Details

$$\underline{u}_{k+1}^* = \underline{u}_k + (\underline{u} - \underline{u}_k)^*$$

← MATLAB will output this when you solve the QP subprobl

# Correction Mechanisms for Constraint Violations

## Option 1 – Tighten inequality constraints:

- Suppose that $g(\mathbf{u}_{k+1}) > 0$ (a constraint violation)… It will be necessary to choose $\mathbf{u}_{k+2}$ such that $\nabla g(\mathbf{u}_{k+1})(\mathbf{u}_{k+2} - \mathbf{u}_{k+1}) \leq -g(\mathbf{u}_{k+1})$. Note that the inequality constraint is ***automatically tightened*** at step k+2, so constraint violations won't accumulate

- To protect against any constraint violations, we can modify an original problem with only inequality constraints by imposing stricter constraints than the original problem. Thus, a small constraint violation in the modified problem may not lead to a constraint violation in the original problem.

**Original problem:**

Minimize $J(\mathbf{u})$

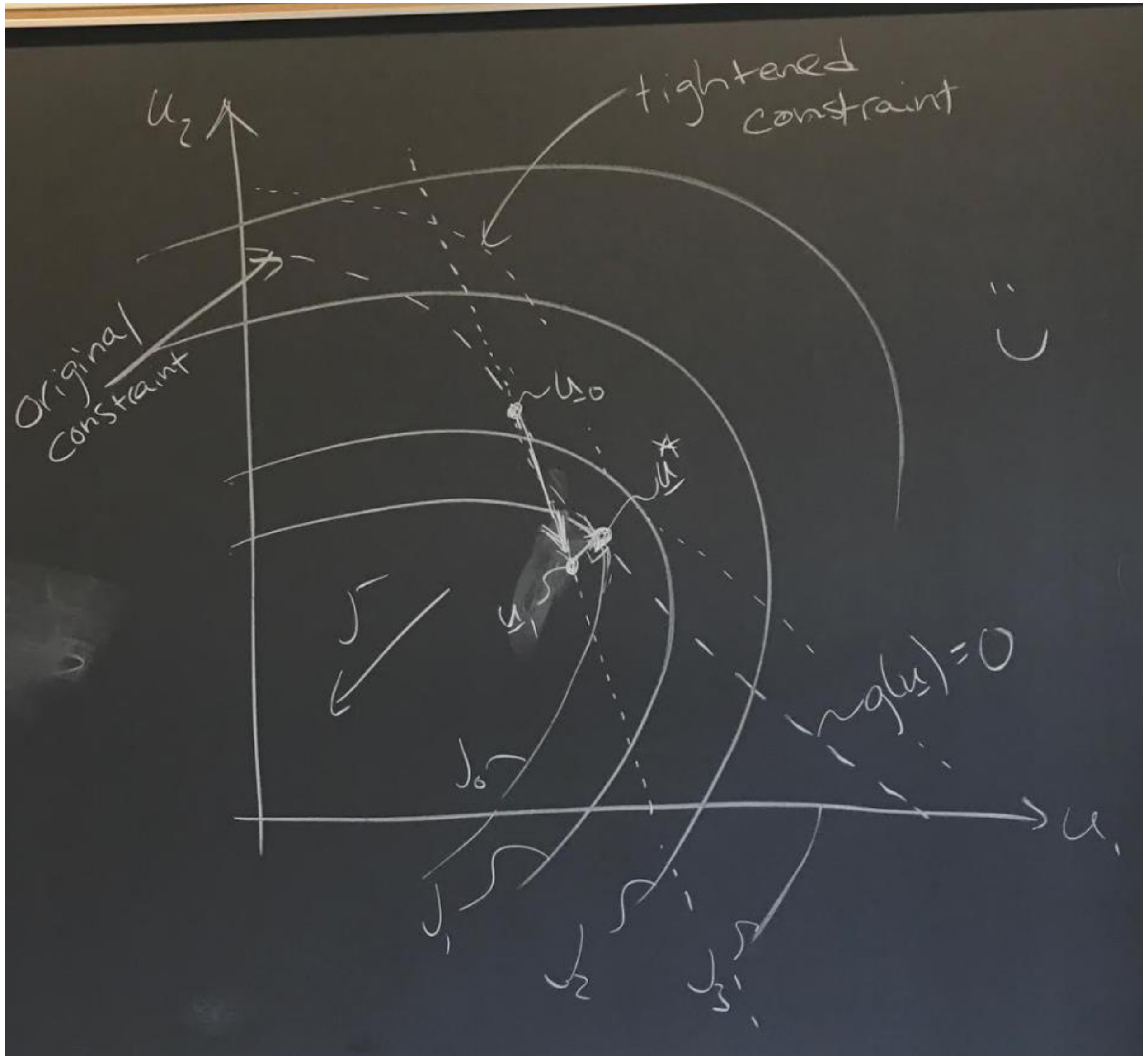
Subject to: $g(\mathbf{u}) \leq \mathbf{0}$

**Modified problem:**

Minimize $J(\mathbf{u})$

Subject to: $g(\mathbf{u}) \leq -K$
where $K > 0$

# Correction Mechanisms for Constraint Violations



# Correction Mechanisms for Constraint Violations

**Option 2 – Line search** (very similar to the line search for gradient descent and Newton's method):

- Solution to each iteration's QP subproblem represents a ***search direction***:

$$\mathbf{s}_k = \arg\min_{\mathbf{s}} (J(\mathbf{u}_k) + \nabla J(\mathbf{u}_k)\mathbf{s} + 0.5\mathbf{s}^T H(\mathbf{u}_k)\mathbf{s}\,)$$

Subject to:

$$g(\mathbf{u}_k) + \nabla g(\mathbf{u}_k)\mathbf{s} \le 0 \qquad\qquad h(\mathbf{u}_k) + \nabla h(\mathbf{u}_k)\mathbf{s} = 0$$

- Perform a one-dimensional search along $\mathbf{s}_k$, verifying constraint satisfaction for candidate points:

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k \mathbf{s}_k$$

where: $\alpha_k = \arg\min_{\alpha} (J(\mathbf{u}_k + \alpha \mathbf{s}_k))$ Subject to:

$$g(\mathbf{u}_k + \alpha \mathbf{s}_k) \le 0$$

$$|h(\mathbf{u}_k + \alpha \mathbf{s}_k)| \le \varepsilon$$

User-defined

# Correction Mechanisms for Constraint Violations

## Option 3 – Project the iterate ($\mathbf{u}_k$) onto the constraint surface(s):

- First, solve the QP subproblem as usual:

$$\mathbf{u}_{k+1}^{prelim} = \arg\min_{\mathbf{u}} (J(\mathbf{u}_k) + \nabla J(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) + 0.5(\mathbf{u} - \mathbf{u}_k)^T H(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k)\ )$$

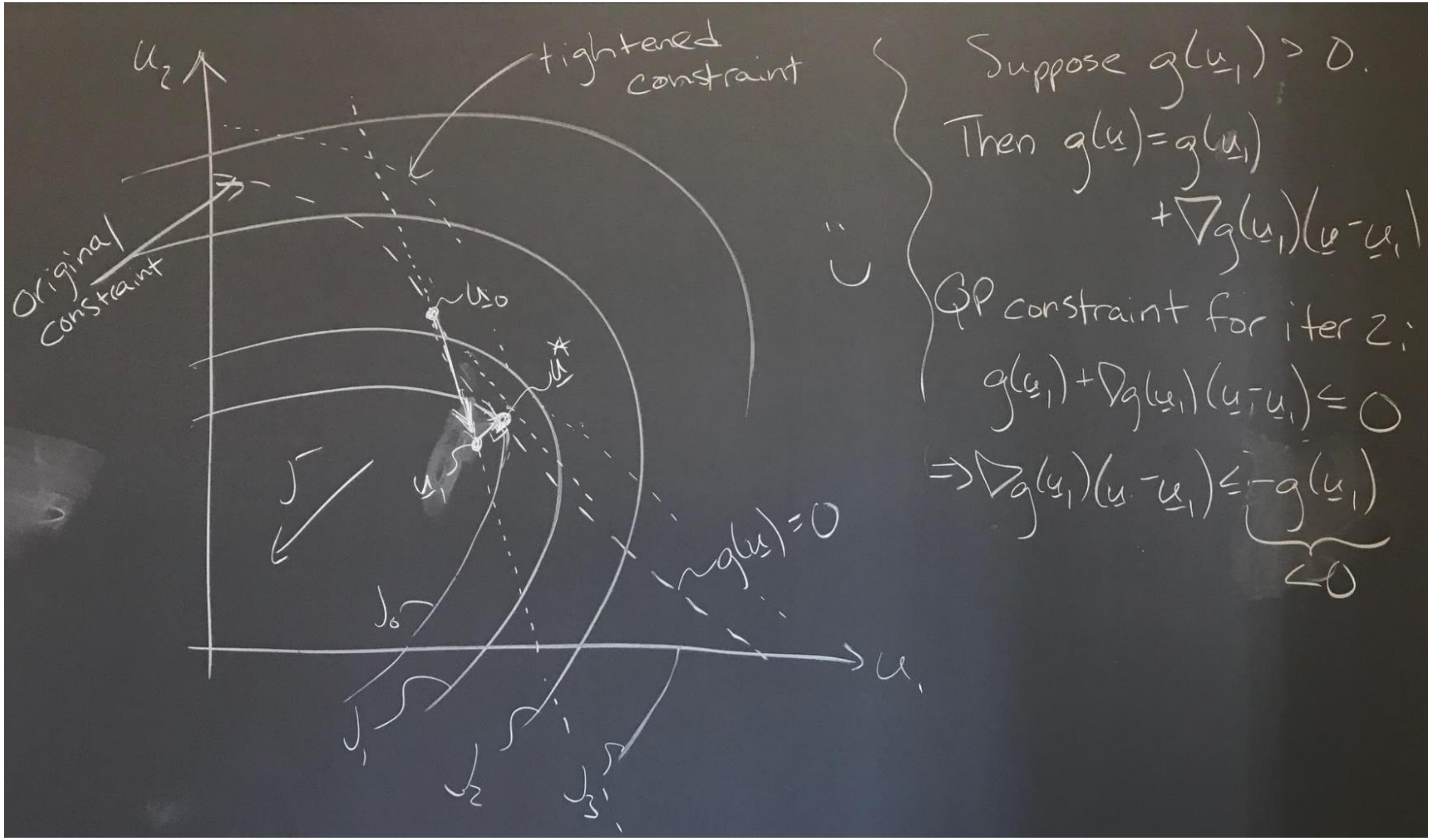
Subject to: $g(\mathbf{u}_k) + \nabla g(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) \leq 0 \qquad h(\mathbf{u}_k) + \nabla h(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) = 0$

- Next, identify all inequality constraints that are violated…denote this vector of constraints by $g'(u)$, and define $z(\mathbf{u}) \triangleq \left[g'^T(\mathbf{u}) \quad h^T(\mathbf{u})\right]^T$

- Orthogonally project the preliminary iterate ($\mathbf{u}_{k+1}^{prelim}$) onto the active constraint surface:

$$\mathbf{u}_{k+1} = \underbrace{\boxed{\left(I - \nabla z(\mathbf{u})\left(\nabla z(\mathbf{u})^T \nabla z(\mathbf{u})\right)^{-1} \nabla z(\mathbf{u})\right)}}_{\text{Projection operator}} \mathbf{u}_{k+1}^{prelim}$$

# Correction Mechanisms for Constraint Violations




**Important point**: Regardless of whether a correction mechanism is in place, SQP works in such a way that constraint violations aren't designed to accumulate. If a constraint is violated at one iteration, SQP will work to meet the original constraint at the next iteration.

# Unconstrained SQP = Newton's Method

UNC CHARLOTTE

**Unconstrained nonlinear program (NLP):** Minimize $J(\mathbf{u})$ Subject to NOTHING

**Local approximation of the objective function:**

$$J(\mathbf{u}) \approx J(\mathbf{u}_k) + \nabla J(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) + 0.5(\mathbf{u} - \mathbf{u}_k)^T H(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k)$$

**Resulting QP subproblem:**

$$\mathbf{u}_{k+1} = \arg\min_{\mathbf{u}} \left( J(\mathbf{u}_k) + \nabla J(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) + 0.5(\mathbf{u} - \mathbf{u}_k)^T H(\mathbf{u}_k)(\mathbf{u} - \mathbf{u}_k) \right)$$

**Solution (if this doesn't look familiar, review your Newton's method notes!):**

$$\mathbf{u}_{k+1} = (H(\mathbf{u}_k))^{-1} \nabla J(\mathbf{u}_k)$$

# (Relatively) Simple SQP Example Problem (Papalambros Example 7.10)

Perform one iteration of SQP for the following optimization problem:

Minimize $J(\mathbf{u}) = u_1 u_2^{-2} + u_2 u_1^{-1}$

Subject to:

$$u_1 + u_2 \geq 1$$

$$u_1 u_2 = 1$$

For your initial guess, take $\mathbf{u}_{init} = [2 \quad 0.5]^T$

# (Relatively) Simple SQP Example Problem (Papalambros Example 7.10)

SQP steps:

1) Set up QP subproblem
2) Solve QP subproblem
3) Apply corrections for constraint violations

↑

Today we will "roll the dice" on this one

# (Relatively) Simple SQP Example Problem (Papalambros Example 7.10)

Ex 1: $J(\underline{u}) = u_1 u_2^{-2} + u_2 u_1^{-1}$

Constraints: $u_1 + u_2 \geq 1 \Rightarrow \underbrace{-u_1 - u_2 + 1}_{g(\underline{u})} \leq 0$

$u_1 u_2 = 1 \Rightarrow \underbrace{u_1 u_2 - 1}_{h(\underline{u})} = 0$

# (Relatively) Simple SQP Example Problem (Papalambros Example 7.10)

$$\nabla J = \begin{bmatrix} u_2^{-2} - u_2 u_1^{-2} & -2u_1 u_2^{-3} + u_1^{-1} \end{bmatrix}$$

$$H = \begin{bmatrix} 2u_2 u_1^{-3} & -2u_2^{-3} - u_1^{-2} \\ -2u_2^{-3} - u_1^{-2} & 6u_1 u_2^{-4} \end{bmatrix}$$

$$\nabla g = \begin{bmatrix} -1 & -1 \end{bmatrix}, \quad \nabla h = \begin{bmatrix} u_2 & u_1 \end{bmatrix}$$

# (Relatively) Simple SQP Example Problem (Papalambros Example 7.10)

$$\nabla g(\underline{u}_k)(\underline{u} - \underline{u}_k) + g(\underline{u}_k) \leq 0$$

$$\Rightarrow \underbrace{\nabla g(\underline{u}_k)(\underline{u} - \underline{u}_k) \leq -g(\underline{u}_k)}_{\text{MATLAB quadprog works w) constraints in this form.}}$$

See m-file on Canvas for MATLAB implementation.

# Another (Relatively) Simple SQP Example Problem

Perform four iterations of SQP for the following optimization problem:

Minimize $J(\mathbf{u}) = e^{-u_1} + (u_2 - 2)^2$

Subject to:

$$u_1 u_2 \leq 1$$

For your initial guess, take $\mathbf{u}_{init} = [1 \quad 1]^T$

# Another (Relatively) Simple SQP Example Problem

Ex. 2: $J(\underline{u}) = e^{-u_1} + (u_2 - 2)^2$

$$u_1 u_2 \le 1 \Rightarrow \underbrace{u_1 u_2 - 1}_{g(\underline{u})} \le 0$$

Initial guess: $\underline{u}_0 = [1 \quad 1]^T$

$\nabla J = [-e^{-u_1} \quad 2(u_2 - 2)]$, $H = \begin{bmatrix} e^{-u_1} & 0 \\ 0 & 2 \end{bmatrix}$

$\nabla g = [u_2 \quad u_1]$

See m-file on Canvas for MATLAB implementation.

# Preview of Upcoming Lectures

**Using SQP for more complex optimal control problems, using off-the-shelf SQP solvers:**

- Modifying the QP subproblem to deal with problematic nonlinear constraints
- fmincon (MATLAB)

**Dynamic programming:**

- Leads to a ***globally optimal*** solution for very general discrete-time optimal control problems
- Can be very computationally intensive, but still more efficient than an exhaustive grid search