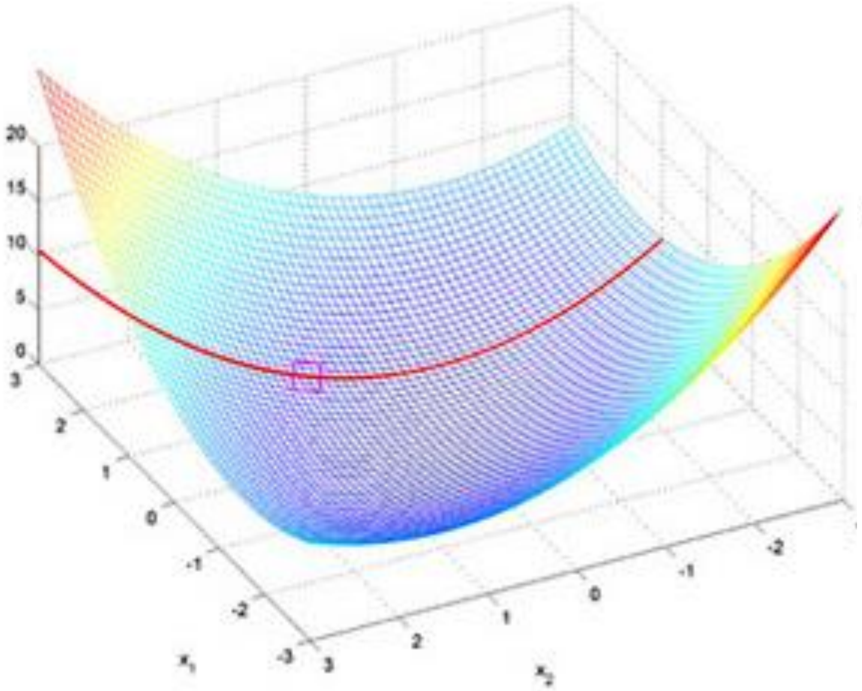
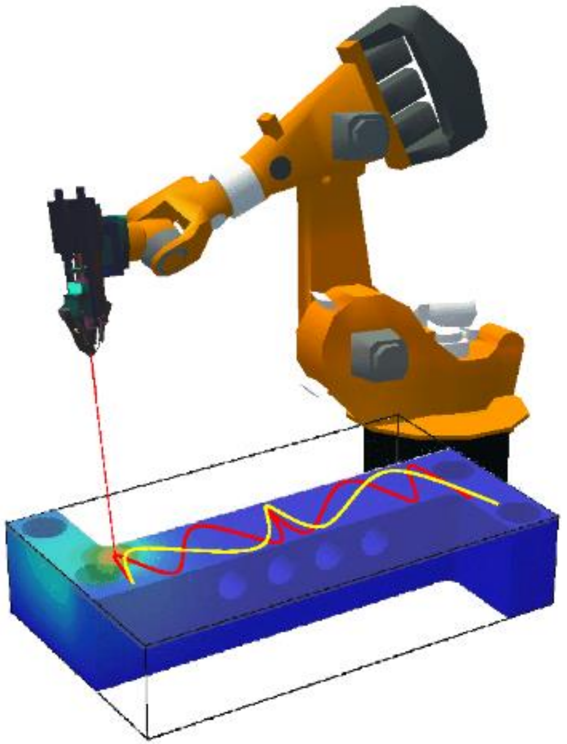
MEGR 3090/7090/8090: Advanced Optimal Control



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

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

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



Lecture 13

October 3, 2017

Preliminary Announcements

Exam 1: $\underbrace{\text{Thursday, Oct. 12}}_{}$ – Tuesday, Oct. 17
(in class)

I won't be here
⇒ Recorded lecture
⇒ Exam posted by 5pm

HW → help session Sunday, Oct. 8 from 6-8pm – Canvas

Recap

Optimization problems

- Constrained linear programs
- Unconstrained quadratic programs
- Constrained quadratic programs
- Unconstrained nonlinear programs
- Constrained nonlinear programs.

Optimization tools

- Active set methods
- Gradient descent
- $\nabla J(\underline{u}^*) = 0$, ensure convexity
- KKT conditions, ensure convexity
- Newton's method (unconstrained SQP)
- Sequential quadratic programming (SQP)
- Interior point

Sequential Quadratic Programming (SQP) – Overall Process Overview

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 - Review

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:

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

Problems with the QP Problem and Nonlinear Constraints - Example

UNC CHARLOTTE

Consider the following optimization problem:

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

Subject to $u_1^2 + u_2^2 = 1$

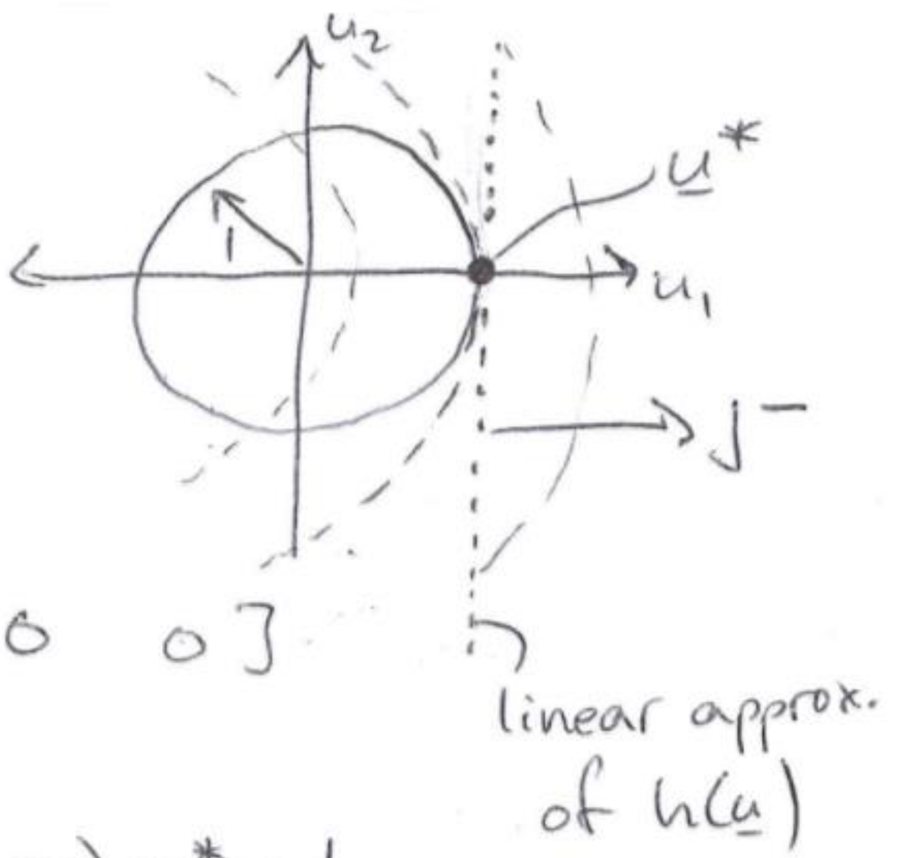
Do the following:

- Compute the minimizer, $\mathbf{u}^*$, analytically
- Take the initial guess to be $\mathbf{u}_0 = \mathbf{u}^* + [\varepsilon \quad 0]^T$, and perform one iteration of SQP by hand
- Note any issues that appear to have arisen in this process

Problems with the QP Problem and Nonlinear Constraints - Example

Ex. Minimize $J(\underline{u}) = -u_1 - \frac{1}{2}u_2^2$

subject to $\underbrace{u_1^2 + u_2^2}_{h(\underline{u})} = 1$



KKT: $\nabla J(\underline{u}^*) + \lambda \nabla h(\underline{u}^*) = \underline{0}$

$[-1 \quad -u_2^*] + \lambda[2u_1^* \quad 2u_2^*] = [0 \quad 0]$

$-1 + 2\lambda u_1^* = 0$

$\underbrace{-u_2^* + 2\lambda u_2^* = 0}_{u_2^*(2\lambda - 1) = 0} \Rightarrow u_2^* = 0 \Rightarrow u_1^* = 1$

Problems with the QP Problem and Nonlinear Constraints - Example

UNC CHARLOTTE

Take $\underline{u}_0 = [1+\varepsilon \quad 0]^T$

QP subproblem:

$$\hat{J}(\underline{u}) = J(\underline{u}_0) + \nabla J(\underline{u}_0)(\underline{u}-\underline{u}_0) + \frac{1}{2}(\underline{u}-\underline{u}_0)^T H(\underline{u}_0)(\underline{u}-\underline{u}_0)$$

$$\nabla J = [-1 \quad -u_2] \Rightarrow \nabla J(\underline{u}_0) = [-1 \quad 0]$$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \Rightarrow H(\underline{u}_0) = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\Rightarrow \hat{J}(\underline{u}) = -(1+\varepsilon) + [-1 \quad \overset{0}{\cancel{-u_{20}}}] \begin{bmatrix} u_1 - u_{10} \\ u_2 - u_{20} \end{bmatrix} + \frac{1}{2}[(u_1-u_{10}) \; (u_2-u_{20})] H(\underline{u}_0) \begin{bmatrix} u_1 - u_{10} \\ u_2 - u_{20} \end{bmatrix}$$

$$= -(1-\varepsilon) - (u_1 - (1+\varepsilon)) - \frac{1}{2}(u_2 - u_{20})^2$$

Problems with the QP Problem and Nonlinear Constraints - Example

UNC CHARLOTTE

$$\hat{h}(\underline{u}) = h(\underline{u}_0) + \nabla h(\underline{u}_0)(\underline{u} - \underline{u}_0)$$

$$= (1+\varepsilon)^2 + [2u_{10} \quad \cancel{2u_{20}}^{\,0}] \begin{bmatrix} u_1 - u_{10} \\ u_2 - u_{20} \end{bmatrix}$$

$$= (1+\varepsilon)^2 + 2(1+\varepsilon)(u_1 - u_{10}) = 1$$

$$\Rightarrow u_1 - u_{10} = \frac{1 - (1+\varepsilon)^2}{2(1+\varepsilon)}$$

$$= \frac{-\varepsilon(2+\varepsilon)}{2(1+\varepsilon)}$$

$u_2 - u_{20}$ = anything! (based on constraint)

Given $\hat{J}(\underline{u})$, take $u_2 - u_{20} = \boxed{\pm\infty}$ ← Clearly not correct!

Resolving Issues with Nonlinear Constraints by Reformulating the QP Subproblem

Main idea: Instead of deriving (and minimizing) a quadratic approximation of $J(\mathbf{u})$, derive (and minimize) a quadratic approximation of $L(\mathbf{u}, \boldsymbol{\mu}, \boldsymbol{\lambda})$, where, as a reminder:

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

Resulting QP subproblem:

Hessian of the Lagrangian

$$\mathbf{u}_{k+1} = \arg\min_{\mathbf{u}} \left(L(\mathbf{u}_k, \boldsymbol{\mu}_k, \boldsymbol{\lambda}_k) + \nabla_u L(\mathbf{u}_k, \boldsymbol{\mu}_k, \boldsymbol{\lambda}_k)(\mathbf{u} - \mathbf{u}_k) + 0.5(\mathbf{u} - \mathbf{u}_k)^T \boxed{\nabla_u^2 L(\mathbf{u}_k, \boldsymbol{\mu}_k, \boldsymbol{\lambda}_k)} (\mathbf{u} - \mathbf{u}_k) \right)$$

Subject to:

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

Same constraint expressions as before

Note: We need to initialize $\boldsymbol{\mu}_k$ and $\boldsymbol{\lambda}_k$, in addition to $\mathbf{u}_k$

Resolving Issues with Nonlinear Constraints by Reformulating the QP Subproblem

$$L(\underline{u}, \lambda, \cancel{\mu}) = -u - \frac{1}{2}u_2^2 + \lambda(u_1^2 + u_2^2)$$

QP approximation:

$$\hat{L}(\underline{u}, \lambda) = L(\underline{u}_0, \lambda_0) + \nabla_{\underline{u}}(L(\underline{u}_0, \lambda_0))(\underline{u} - \underline{u}_0) + \frac{1}{2}(\underline{u} - \underline{u}_0)^T H(L(\underline{u}_0, \lambda_0))(\underline{u} - \underline{u}_0)$$

$$\nabla_{\underline{u}}(L(\underline{u}_0, \lambda_0)) = [-1 + 2\lambda u_1 \quad -u_2 + 2\lambda u_2]\Big|_{\underline{u}_0, \lambda_0}$$

$$= [-1 + 2\lambda_0 u_{10} \quad 0]$$

$$H(L(\underline{u}_0, \lambda_0)) = \begin{bmatrix} 2\lambda & 0 \\ 0 & -1 + 2\lambda \end{bmatrix}$$

$$\Rightarrow \hat{L}(\underline{u}, \lambda) = L(\underline{u}_0, \lambda_0) + (2\lambda_0 u_{10} - 1)(u_1 - u_{10}) + \frac{1}{2}(u_1 - u_{10})^2 2\lambda_0 + \underbrace{\frac{1}{2}(u_2 - u_{20})^2(2\lambda_0 - 1)}_{\text{If } \lambda_0 \geq \frac{1}{2},\ \text{then } (u_2 - u_{20})^* = 0}$$

Observations and Challenges Regarding the QP Subproblem Reformulation

Key points:

- Since $\boldsymbol{\mu}^{*T} g(\mathbf{u}^*) = 0$ and $\boldsymbol{\lambda}^{*T} g(\mathbf{u}^*) = 0$ by the KKT conditions, it follows that $J(\mathbf{u}^*) = L(\mathbf{u}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)$
- It follows that minimizing $L(\mathbf{u}, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)$ with respect to $\mathbf{u}$ is the same as minimizing $J(\mathbf{u})$
- Initializing estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ remains a challenge
- Lagrangian-based SQP is performed in commercial SQP packages, which we will work with from here on…we'll focus specifically on MATLAB's fmincon

fmincon – Basic Setup

Main idea of fmincon: Perform a nonlinear optimization/program (NLP) using SQP, where the NLP is given, in general, by:

Minimize $J(\mathbf{u})$

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

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

Syntax (see also https://www.mathworks.com/help/optim/ug/fmincon.html):

```
options = optimoptions('fmincon','Display','iter','Algorithm','sqp');
```

SQP is specified as the chosen solver

```
u_opt = fmincon(obj_fun,u0,[],[],[],[],[],[],nonlinear_constraints,options);
```

- The objective function (obj_fun) is specified as a separate MATLAB function whose input is $\mathbf{u}$ and whose output is $J(\mathbf{u})$
- The constraints (nonlinear_constraints) are specified through a MATLAB function whose input is $\mathbf{u}$ and whose outputs are $g(\mathbf{u})$ and $h(\mathbf{u})$, in that order

fmincon – Basic Setup

u = fmincon(obj-fn, u0, A_ineq, b_ineq, A_eq, b_eq, u_min, u_max, g-and-h, options);

Minimize $J(\underline{u})$

subject to: $A_{ineq}\underline{u} - \underline{b}_{ineq} \le 0$

$$A_{eq}\underline{u} - \underline{b}_{eq} = 0$$

$$u_{min}\underline{1}_{N\times 1} - \underline{u} \le 0$$

$$\underline{u} - u_{max}\underline{1}_{N\times 1} \le 0$$

$$g(\underline{u}) \le 0$$

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

fmincon – Example 1

Use fmincon to compute the solution to the following optimization problem that we previously examined:

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

Subject to $u_1^2 + u_2^2 = 1$

Example code available on Canvas.

fmincon – Example 2

Use fmincon to compute the solution to the following optimization problem that we previously examined:

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

Subject to:

$$u_1 u_2 \leq 1$$

Example code available on Canvas.

fmincon – Example 3

Suppose a vehicle's dynamics are given by:

$$m\dot{v} = u - C_{rr}mg - 0.5\rho v^2 C_d A_{ref}$$
$$\dot{x} = v$$

Parameter values: $m = 1000kg$, $g = 9.8\frac{m}{s^2}$, $C_{rr} = 0.01$, $\rho = 1.2\frac{kg}{m^3}$, $C_d = 0.4$, $A_{ref} = 5m^2$

Suppose that our goals are:

- Given an initial position of $x = 0$, achieve a final position of $x = 1600$ (approximately one "metric" mile of travel) after 60 seconds
- Minimize total energy expended over 60 seconds

Set up a nonlinear optimization problem and solve using fmincon

We will start this in class, and you will finish it in the homework.

fmincon – Example 3

Ex: $m\dot{v} = u - C_{rr}mg - 0.5\rho v^2 C_d A_{ref}$

$$\dot{x} = v$$

$$v(k+1) = v(k) + \dot{v}_k T$$

$$= v(k) + \frac{T}{m}\left(u(k) - C_{rr}mg - 0.5\rho v(k)^2 C_d A_{ref}\right)$$

$$x(k+1) = x(k) + \dot{x}_k T$$

$$= x(k) + Tv(k)$$

$$J(u(t)) = \text{work} = \int_0^{60} u(t)v(t)\,dt$$

$$\approx \boxed{\sum_{i=0}^{N-1} u(k)v(k)T}$$

$\Uparrow$

minimize that

fmincon – Example 3

Pseudo-code for cost:

```
function J = cost_fun(u)

  Initialize params., x(1), v(1)

  for i = 2:N
      v(i) = .....
      x(i) = .....
  end

  J = sum(u.*v)*T
```

fmincon – Example 3

Pseudo-code for constraints:

```
function [g, h] = constraint_fun(u)
  Initialize params, x(1), v(1)
  for i = 2 : N+1
      v(i) = ...
      x(i) = ...
  end
  g1 =    1600 - x(N+1);
  g2 = v(1) - v(N+1);
```

Saturation limits can be specified here or as upper and lower bounds in the fmincon call.

Summary and Limitations of SQP

***Reminder* – Summary of LP:**

- Restricted linear objective function and constraints
- **Globally convex…therefore, a global optimum is guaranteed**

***Reminder* – Summary of QP:**

- Restricted to quadratic objective function ($J(\mathbf{u}) = r^T\mathbf{u} + \mathbf{u}^T Q\mathbf{u}$) and linear constraints
- Globally convex whenever Q is positive definite…QP will find a global optimizer in this circumstance

Summary of SQP:

- Objective function and constraints are ***approximated*** by a QP…therefore, a wider variety of optimization problems can be solved using SQP
- A ***locally convex*** quadratic approximation will yield a solution to the QP subproblem…however, SQP is ***not*** guaranteed to converge to a global optimizer

Preview of Upcoming Lectures

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