Bellman’s Curse of Dimensionality

- n-dimensional state space
- Number of states grows exponentially in n (for fixed number of discretization levels per coordinate)
- In practice
  - Discretization is considered only computationally feasible up to 5 or 6 dimensional state spaces even when using
    - Variable resolution discretization
    - Highly optimized implementations
Optimization for Optimal Control

- Goal: find a sequence of control inputs (and corresponding sequence of states) that solves:

\[
\min_{u,x} \sum_{t=0}^{H} g(x_t, u_t) \\
\text{subject to} \quad x_{t+1} = f(x_t, u_t) \quad \forall t \\\n\quad u_t \in U_t \quad \forall t \\\n\quad x_t \in X_t \quad \forall t
\]

- Generally hard to do. In this set of slides we will consider convex problems, which means \( g \) is convex, the sets \( U_t \) and \( X_t \) are convex, and \( f \) is linear. Next set of slides will relax these assumptions.

- Note: iteratively applying LQR is one way to solve this problem but can get a bit tricky when there are constraints on the control inputs and state.

- In principle (though not in our examples), \( u \) could be parameters of a control policy rather than the raw control inputs.
Convex Optimization

Pieter Abbeel
UC Berkeley EECS

Many slides and figures adapted from Stephen Boyd

[optional] Boyd and Vandenberghe, Convex Optimization, Chapters 9 – 11
[optional] Betts, Practical Methods for Optimal Control Using Nonlinear Programming
Outline

- Convex optimization problems
- Unconstrained minimization
  - Gradient Descent
  - Newton’s Method
- Equality constrained minimization
- Inequality and equality constrained minimization
Convex Functions

A function $f$ is convex if and only if

$$\forall x_1, x_2 \in \text{Domain}(f), \forall t \in [0, 1]:$$

$$f(tx_1 + (1 - t)x_2) \leq tf(x_1) + (1 - t)f(x_2)$$

Image source: wikipedia
Convex Functions

- Unique minimum
- Set of points for which \( f(x) \leq a \) is convex

Source: Thomas Jungblut’s Blog
Convex Optimization Problems

- Convex optimization problems are a special class of optimization problems, of the following form:

\[
\min_{x \in \mathbb{R}^n} f_0(x) \quad \text{s.t.} \quad f_i(x) \leq 0 \quad i = 1, \ldots, n \\
Ax = b
\]

with \( f_i(x) \) convex for \( i = 0, 1, \ldots, n \)

- A function \( f \) is convex if and only if

\[
\forall x_1, x_2 \in \text{Domain}(f), \forall \lambda \in [0, 1] \\
f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)
\]
Outline

- Convex optimization problems
- **Unconstrained minimization**
  - Gradient Descent
  - Newton’s Method
- Equality constrained minimization
- Inequality and equality constrained minimization
Unconstrained Minimization

\[
\min_{x} f(x) \quad (1)
\]

(Implicitly assumed \( x \) can be chosen from the entire domain of \( f \), often \( \mathbb{R}^n \).)

- \( x^* \) is a local minimum of (differentiable) \( f \) than it has to satisfy:
  \[
  \nabla_x f(x^*) = 0 \quad (2)
  \]
  \[
  \nabla_x^2 f(x^*) \succeq 0 \quad (3)
  \]

- In simple cases we can directly solve the system of \( n \) equations given by (2) to find candidate local minima, and then verify (3) for these candidates.

- In general however, solving (2) is a difficult problem. Going forward we will consider this more general setting and cover numerical solution methods for (1).
Steepest Descent

- Idea:
  - Start somewhere
  - Repeat: Take a step in the steepest descent direction

Figure source: Mathworks
1. Initialize $x$

2. Repeat
   1. Determine the steepest descent direction $\Delta x$
   2. Line search: Choose a step size $t > 0$.
   3. Update: $x := x + t \Delta x$.

3. Until stopping criterion is satisfied
What is the Steepest Descent Direction?

Assuming a smooth function, we have that

$$f(x_0 + \Delta x) \approx f(x_0) + \nabla_x f(x_0) \Delta x$$

The (locally at $x_0$) direction of steepest descent is given by:

$$\Delta x^* = \arg \min_{\Delta x: \|\Delta x\|_2 = 1} f(x_0) + \nabla_x f(x_0) \Delta x$$

$$= \arg \min_{\Delta x: \|\Delta x\|_2 = 1} \nabla_x f(x_0) \Delta x$$

As we have all $a, b \in \mathbb{R}^n$ that $\min_{b: \|b\|_2 = 1} a^T b$ is achieved for $b = -\frac{a}{\|a\|_2}$, we have that the steepest descent direction

$$\Delta x^* = -\nabla_x f(x_0)$$

$\rightarrow$ Steepest Descent = Gradient Descent
Steps: Selection: Exact Line Search

\[ t = \arg \min_{s \geq 0} f(x + s\Delta x) \]

Used when the cost of solving the minimization problem with one variable is low compared to the cost of computing the search direction itself.
Inexact: step length is chose to approximately minimize $f$ along the ray $\{x + t \Delta x \mid t > 0\}$

Backtracking Line Search.
given a descent direction $\Delta x$ for $f$ at $x \in \text{dom}f$, $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.
$t := 1$
while $f(x + t \Delta x) > f(x) + \alpha t \nabla f(x)^\top \Delta x$, $t := \beta t$. 
Figure 9.1 Backtracking line search. The curve shows $f$, restricted to the line over which we search. The lower dashed line shows the linear extrapolation of $f$, and the upper dashed line has a slope a factor of $\alpha$ smaller. The backtracking condition is that $f$ lies below the upper dashed line, i.e., $0 \leq t \leq t_0$. 

Figure source: Boyd and Vandenberghe
Steepest Descent (= Gradient Descent)

Algorithm 9.3  Gradient descent method.

given a starting point $x \in \text{dom } f$.

repeat
1. $\Delta x := -\nabla f(x)$.
2. Line search. Choose step size $t$ via exact or backtracking line search.
3. Update. $x := x + t\Delta x$.

until stopping criterion is satisfied.

The stopping criterion is usually of the form $\|\nabla f(x)\|_2 \leq \eta$, where $\eta$ is small and positive. In most implementations, this condition is checked after step 1, rather than after the update.

Source: Boyd and Vandenberghe
Gradient Descent: Example 1

\[ f(x_1, x_2) = e^{x_1 + 3x_2 - 0.1} + e^{x_1 - 3x_2 - 0.1} + e^{-x_1 - 0.1} \]

Figure source: Boyd and Vandenberghe
Gradient Descent: Example 2

A problem in $\mathbb{R}^{100}$

$$f(x) = c^T x - \sum_{i=1}^{500} \log(b_i - a_i^T x)$$

‘linear’ convergence, \textit{i.e.}, a straight line on a semilog plot

Figure source: Boyd and Vandenberghe
Gradient Descent: Example 3

\[ f(x) = \frac{1}{2} (x_1^2 + \gamma x_2^2) \quad (\gamma > 0) \]

with exact line search, starting at \( x^{(0)} = (\gamma, 1) \):

\[ x_1^{(k)} = \gamma \left( \frac{\gamma - 1}{\gamma + 1} \right)^k, \quad x_2^{(k)} = \left( -\frac{\gamma - 1}{\gamma + 1} \right)^k \]

- very slow if \( \gamma \gg 1 \) or \( \gamma \ll 1 \)
- example for \( \gamma = 10 \):

Figure source: Boyd and Vandenberghe
For quadratic function, convergence speed depends on ratio of highest second derivative over lowest second derivative ("condition number")

In high dimensions, almost guaranteed to have a high (=bad) condition number

Rescaling coordinates (as could happen by simply expressing quantities in different measurement units) results in a different condition number
Outline

- Unconstrained minimization
  - Gradient Descent
  - Newton’s Method
- Equality constrained minimization
- Inequality and equality constrained minimization
Newton’s Method

- 2\textsuperscript{nd} order Taylor Approximation rather than 1\textsuperscript{st} order:

\[
f(x + \Delta x) \approx f(x) + \nabla f(x)^\top \Delta x + \frac{1}{2} \Delta x^\top \nabla^2 f(x) \Delta x
\]

assuming \( \nabla^2 f(x) \succeq 0 \) (which is true for convex \( f \)) the minimum of the 2\textsuperscript{nd} order approximation is achieved at:

\[
\Delta x_{nt} = - (\nabla^2 f(x))^{-1} \nabla f(x)
\]

Figure source: Boyd and Vandenberghe
Newton’s Method

Algorithm 9.5  Newton’s method.

given a starting point $x \in \text{dom } f$, tolerance $\epsilon > 0$.

repeat

1. Compute the Newton step and decrement.
   \[ \Delta x_{nt} := -\nabla^2 f(x)^{-1} \nabla f(x); \quad \lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x). \]
2. Stopping criterion. quit if $\lambda^2 / 2 \leq \epsilon$.
3. Line search. Choose step size $t$ by backtracking line search.
4. Update. $x := x + t \Delta x_{nt}$.

Figure source: Boyd and Vandenberghe
Affine Invariance

- Consider the coordinate transformation $y = A^{-1} x$ \quad (x = Ay)

- If running Newton’s method starting from $x^{(0)}$ on $f(x)$ results in
  $$x^{(0)}, x^{(1)}, x^{(2)}, ...$$

- Then running Newton’s method starting from $y^{(0)} = A^{-1} x^{(0)}$ on $g(y) = f(Ay)$, will result in the sequence
  $$y^{(0)} = A^{-1} x^{(0)}, y^{(1)} = A^{-1} x^{(1)}, y^{(2)} = A^{-1} x^{(2)}, ...$$

Exercise: try to prove this!
Affine Invariance --- Proof

\[ \frac{\partial g}{\partial y_i} = \sum_j \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial y_i} \]

\[ = \sum_j \frac{\partial f}{\partial x_j} A_{ji} \]

\[ \nabla g = A^T \nabla f \]

\[ \frac{\partial^2 g}{\partial y_k \partial y_i} = \frac{\partial}{\partial y_i} \left( \sum_j \frac{\partial f}{\partial x_j} A_{ji} \right) \]

\[ = \sum_j \frac{\partial}{\partial y_k} \left( \frac{\partial f}{\partial x_j} \right) A_{ji} \]

\[ = \sum_j \sum_l \frac{\partial^2 f}{\partial x_l \partial x_j} \frac{\partial x_l}{\partial y_k} A_{ji} \]

\[ = \sum_j \sum_l \frac{\partial^2 f}{\partial x_l \partial x_j} A_{l,k} A_{ji} \]

\[ \nabla^2 g = A^T \nabla^2 f A \]

\[ \Delta y = - (\nabla^2 g)^{-1} \nabla g \]

\[ = - (A^T \nabla^2 f A)^{-1} A^T \nabla f \]

\[ = - A^{-1} (\nabla^2 f)^{-1} A^{-T} A^T \nabla f \]

\[ = - A^{-1} (\nabla^2 f)^{-1} \nabla f \]

\[ = A^{-1} \Delta x \]
Example 1

\[ f(x_1, x_2) = e^{x_1 + 3x_2 - 0.1} + e^{x_1 - 3x_2 - 0.1} + e^{-x_1 - 0.1} \]

Figure source: Boyd and Vandenberghe
Example 2

A problem in $\mathbb{R}^{100}$

$$f(x) = c^T x - \sum_{i=1}^{500} \log(b_i - a_i^T x)$$

**Gradient descent**

**Newton’s method**

Figure source: Boyd and Vandenberghe
Larger Version of Example 2

Example in $\mathbb{R}^{10000}$ (with sparse $a_i$)

$$f(x) = -\sum_{i=1}^{10000} \log(1 - x_i^2) - \sum_{i=1}^{10000} \log(b_i - a_i^T x)$$

- Backtracking parameters $\alpha = 0.01$, $\beta = 0.5$.
- Performance similar as for small examples.

Figure source: Boyd and Vandenberghe
Gradient Descent: Example 3

\[ f(x) = \frac{1}{2}(x_1^2 + \gamma x_2^2) \quad (\gamma > 0) \]

with exact line search, starting at \( x^{(0)} = (\gamma, 1) \):

\[ x_1^{(k)} = \gamma \left( \frac{\gamma - 1}{\gamma + 1} \right)^k, \quad x_2^{(k)} = \left( \frac{-\gamma - 1}{\gamma + 1} \right)^k \]

- very slow if \( \gamma \gg 1 \) or \( \gamma \ll 1 \)
- example for \( \gamma = 10 \):

Figure source: Boyd and Vandenberghe
Example 3

- Gradient descent
- Newton’s method (converges in one step if \( f \) convex quadratic)
Quasi-Newton Methods

- Quasi-Newton methods use an approximation of the Hessian

  - Example 1: Only compute diagonal entries of Hessian, set others equal to zero. Note this also simplifies computations done with the Hessian.

  - Example 2: natural gradient --- see next slide
Consider a standard maximum likelihood problem:

$$
\max_{\theta} f(\theta) = \max_{\theta} \sum_i \log p(x^{(i)}; \theta)
$$

**Gradient:**

$$
\frac{\partial f(\theta)}{\partial \theta_p} = \sum_i \frac{\partial \log p(x^{(i)}; \theta)}{\partial \theta_p}
= \sum_i \frac{\partial p(x^{(i)}; \theta)}{\partial \theta_p} \frac{1}{p(x^{(i)}; \theta)}
$$

**Hessian:**

$$
\frac{\partial^2 f(\theta)}{\partial \theta_q \partial \theta_p} = \sum_i \frac{\partial^2 p(x^{(i)}; \theta)}{\partial \theta_q \partial \theta_p} \frac{1}{p(x^{(i)}; \theta)}
- \frac{\partial p(x^{(i)}; \theta)}{\partial \theta_q} \frac{1}{p(x^{(i)}; \theta)} \frac{\partial p(x^{(i)}; \theta)}{\partial \theta_p} \frac{1}{p(x^{(i)}; \theta)}
$$

$$
\nabla^2 f(\theta) = \sum_i \frac{\nabla^2 p(x^{(i)}; \theta)}{p(x^{(i)}; \theta)}
- \left( \nabla \log p(x^{(i)}; \theta) \right) \left( \nabla \log p(x^{(i)}; \theta) \right)^T
$$

**Natural gradient:**

$$
= \left( \sum_i \left( \nabla \log p(x^{(i)}; \theta) \right) \left( \nabla \log p(x^{(i)}; \theta) \right)^T \right)^{-1}
\left( \sum_i \nabla \log p(x^{(i)}; \theta) \right)
$$

only keeps the $2^{nd}$ term in the Hessian. Benefits: (1) faster to compute (only gradients needed); (2) guaranteed to be negative definite; (3) found to be superior in some experiments; (4) invariant to re-parametrization.
Property: Natural gradient is invariant to parameterization of the family of probability distributions $p(x; \theta)$

Hence the name.

Note this property is stronger than the property of Newton’s method, which is invariant to affine re-parameterizations only.

Exercise: Try to prove this property!
Natural Gradient Invariant to Reparametrization --- Proof

- Natural gradient for parametrization with $\theta$:

$$
\bar{g}_\theta = \left( \sum_i \left( \nabla_\theta \log p(x^{(i)}; \theta) \right) \left( \nabla_\theta \log p(x^{(i)}; \theta) \right)^\top \right)^{-1} \left( \sum_i \nabla_\theta \log p(x^{(i)}; \theta) \right)
$$

- Let $\Phi = f(\theta)$, and let $J = \frac{\partial \theta}{\partial \phi}$ i.e., $J_{i,j} = \frac{\partial \theta_i}{\partial \phi_j}$

$$
\bar{g}_\phi = \left( \sum_i \left( \nabla_\phi \log p(x^{(i)}; \phi) \right) \left( \nabla_\phi \log p(x^{(i)}; \phi) \right)^\top \right)^{-1} \left( \sum_i \nabla_\phi \log p(x^{(i)}; \phi) \right)
$$

$$
= \left( \sum_i \left( J^\top \nabla_\theta \log p(x^{(i)}; \phi) \right) \left( J^\top \nabla_\theta \log p(x^{(i)}; \phi) \right)^\top \right)^{-1} \left( J^\top \sum_i \nabla_\theta \log p(x^{(i)}; \phi) \right)
$$

$$
= J^\top \bar{g}_\theta
$$

$\rightarrow$ the natural gradient direction is the same independent of the (invertible, but otherwise not constrained) reparametrization $f$
Outline

- Unconstrained minimization
  - Gradient Descent
  - Newton’s Method

- Equality constrained minimization

- Inequality and equality constrained minimization
Equality Constrained Minimization

- Problem to be solved:

\[
\min_x f(x) \\
\text{s.t. } Ax = b
\]

- We will cover three solution methods:
  - Elimination
  - Newton’s method
  - Infeasible start Newton method
From linear algebra we know that there exist a matrix F (in fact infinitely many) such that:

\[ \{x | Ax = b\} = \{x | x = \hat{x} + Fz\} \]

\(\hat{x}\): any solution to \(Ax = b\)

F: spans the null-space of A

A way to find an F: compute SVD of A, \(A = U S V^T\), for A having k nonzero singular values, set \(F = U(:, k+1:end)\)

So we can solve the equality constrained minimization problem by solving an _unconstrained minimization problem over a new variable z_

\[
\min_z f(\hat{x} + Fz)
\]

Potential cons: (i) need to first find a solution to \(Ax=b\), (ii) need to find F, (iii) elimination might destroy sparsity in original problem structure
Recall problem to be solved:

\[ \min_x f(x) \]
\[ \text{s.t. } Ax = b \]

\[ x^* \text{ with } Ax^* = b \text{ is (local) optimum if and only if: } \forall \Delta x \text{ if } A\Delta x = 0 \text{ then } \nabla f(x^*)^\top \Delta x = 0. \]

Equivalently:

\[ \nabla f(x^*)^\top = \nu^\top A \]
Recall problem to be solved:

\[ \min_x f(x) \]
\[ \text{s.t. } Ax = b \]

Optimality Condition: \( Ax^* = b \) and \( \nabla f(x^*) + A^\top \nu = 0 \)
Method 2: Newton’s Method

- Problem to be solved:
  \[ \min_x f(x) \]
  \[ \text{s.t. } Ax = b \]

- Optimality Condition: \( Ax^* = b \) and \( \nabla f(x^*) + A^\top \nu = 0 \)

- Assume \( x \) is feasible, i.e., satisfies \( Ax = b \), now use 2\(^{\text{nd}}\) order approximation of \( f \):
  \[
  \min_{\Delta x} \quad f(x) + \nabla f(x) \Delta x + \frac{1}{2} \Delta x \nabla^2 f(x) \Delta x \\
  \text{s.t. } \quad A(x + \Delta x) = b
  \]

- Optimality condition for 2\(^{\text{nd}}\) order approximation:
  \[
  \begin{bmatrix}
  \nabla^2 f(x) & A^\top \\
  A & 0
  \end{bmatrix}
  \begin{bmatrix}
  \Delta x \\
  \nu
  \end{bmatrix} =
  \begin{bmatrix}
  -\nabla f(x) \\
  0
  \end{bmatrix}
  \]
Method 2: Newton’s Method

\begin{equation*}
\text{given} \text{ starting point } x \in \text{dom } f \text{ with } Ax = b, \text{ tolerance } \epsilon > 0.
\end{equation*}

\text{repeat}

1. Compute the Newton step and decrement \( \Delta x_{nt}, \lambda(x) \).
2. \textit{Stopping criterion.} \textbf{quit} if \( \lambda^2/2 \leq \epsilon \).
3. \textit{Line search.} Choose step size \( t \) by backtracking line search.
4. Update. \( x := x + t\Delta x_{nt} \).

\begin{equation*}
\begin{bmatrix}
\nabla^2 f(x) & A^T \\
A & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x_{nt} \\
nu
\end{bmatrix}
= 
\begin{bmatrix}
-\nabla f(x) \\
0
\end{bmatrix}
\end{equation*}

\text{Feasible descent method:} \quad x^{(k)} \text{ feasible and } f(x^{(k+1)}) \leq f(x^{(k)})
Method 3: Infeasible Start Newton Method

- Problem to be solved: \[
\min_x f(x) \\
\text{s.t. } Ax = b
\]

- Optimality Condition: \( Ax^* = b \) and \( \nabla f(x^*) + A^\top \nu = 0 \)

- Use 1\textsuperscript{st} order approximation of the optimality conditions at current \( x \):

\[
A(x + \Delta x) = b \\
\nabla f(x) + \nabla^2 f(x) \Delta x + A^\top \nu = 0
\]

- Equivalently:

\[
\begin{bmatrix}
\nabla^2 f(x) & A^\top \\
A & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\nu
\end{bmatrix} =
\begin{bmatrix}
-\nabla f(x) \\
b - Ax
\end{bmatrix}
\]
Outline

- Unconstrained minimization
- Equality constrained minimization
- Inequality and equality constrained minimization
Equality and Inequality Constrained Minimization

Recall the problem to be solved:

\[
\begin{align*}
\min_x & \quad f_0(x) \\
\text{s.t.} & \quad f_i(x) \leq 0, \quad i = 1, \ldots, m \\
& \quad Ax = b
\end{align*}
\]
Equality and Inequality Constrained Minimization

- Problem to be solved:
  \[
  \begin{align*}
  \min_x & \quad f_0(x) \\
  \text{s.t.} & \quad f_i(x) \leq 0, \quad i = 1, \ldots, m \\
  & \quad Ax = b
  \end{align*}
  \]

- Reformulation via indicator function
  \[
  \begin{align*}
  \min_x & \quad f_0(x) + \sum_{i=1}^{m} I_{-}(f_i(x)) \\
  & \quad Ax = b
  \end{align*}
  \]
  → No inequality constraints anymore, but very poorly conditioned objective function

- Approximation via logarithmic barrier:
  \[
  \begin{align*}
  \min_x & \quad f_0(x) - \left(\frac{1}{t}\right) \sum_{i=1}^{m} \log(-f_i(x)) \\
  \text{s.t.} & \quad Ax = b
  \end{align*}
  \]
  * for \( t > 0 \), \(-1/t \log(-u)\) is a smooth approximation of \( I_{-}(u) \)
  * approximation improves for \( t \to 1 \)
  * better conditioned for smaller \( t \)
Equality and Inequality Constrained Minimization

\[-(1/t)\log(-u)\]
Barrier Method

- Given: strictly feasible $x$, $t^{(0)} > 0$, $\mu > 1$, tolerance $\varepsilon > 0$

- Repeat
  1. **Centering Step.** Compute $x^*(t)$ by solving

$$
\min_x \quad f_0(x) - \frac{1}{t} \sum_{i=1}^{m} \log(-f_i(x))
$$

  s.t. $Ax = b$

starting from $x$

  2. **Update.** $x := x^*(t)$.

  3. **Stopping Criterion.** Quit if $m/t < \varepsilon$

  4. **Increase $t$.** $t := \mu t$
Example 1: Inequality Form LP

Inequality form LP \((m = 100\) inequalities, \(n = 50\) variables)\n
- starts with \(x\) on central path \((t^{(0)} = 1\), duality gap 100\)
- terminates when \(t = 10^8\) \((\text{gap} 10^{-6})\)
- centering uses Newton’s method with backtracking
- total number of Newton iterations not very sensitive for \(\mu \geq 10\)
Example 2: Geometric Program

**geometric program** \((m = 100\) inequalities and \(n = 50\) variables)

minimize \(\log \left( \sum_{k=1}^{5} \exp(a_{0k}^T x + b_{0k}) \right)\)

subject to \(\log \left( \sum_{k=1}^{5} \exp(a_{ik}^T x + b_{ik}) \right) \leq 0, \quad i = 1, \ldots, m\)

---

![Graph showing duality gap vs. Newton iterations for different \(\mu\) values]
Example 3: Standard LPs

family of standard LPs \((A \in \mathbb{R}^{m \times 2m})\)

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax = b, \quad x \geq 0
\end{align*}
\]

\(m = 10, \ldots, 1000; \) for each \(m,\) solve 100 randomly generated instances

number of iterations grows very slowly as \(m\) ranges over a 100 : 1 ratio
Initialization

- Basic phase I method:

  Initialize by first solving:

  $\min_{x,s} \ s$
  s.t.  $f_i(x) \leq s, \ i = 1, \ldots, m$
  $Ax = b$

- Easy to initialize above problem, pick some $x$ such that $Ax = b$, and then simply set $s = \max_i f_i(x)$

- Can stop early---whenever $s < 0$
Initialization

- Sum of infeasibilities phase I method:

- Initialize by first solving:

\[
\min_{x, s} \sum_{i=1}^{m} s_i \\
\text{s.t. } f_i(x) \leq s_i, \quad i = 1, \ldots, m \\
\quad s_i \geq 0, \quad i = 1, \ldots, m \\
\quad Ax = b
\]

- Easy to initialize above problem, pick some \( x \) such that \( Ax = b \), and then simply set \( S_i = \max(0, f_i(x)) \)

- For infeasible problems, produces a solution that satisfies many more inequalities than basic phase I method
Other methods

- We have covered a primal interior point method
  - one of several optimization approaches

- Examples of others:
  - Primal-dual interior point methods
  - Primal-dual infeasible interior point methods
Optimal Control (Open Loop)

- For convex $g_t$ and $f$, we can now solve:

$$\min_{x,u} \sum_{t=0}^{T} g_t(x_t, u_t)$$

s.t.  
$$x_{t+1} = A_t x_t + B_t u_t \quad \forall t$$
$$f_i(x, u) \leq 0, \quad i = 1, \ldots, m$$

Which gives an open-loop sequence of controls
Optimal Control (Closed Loop)

Given: $\bar{x}_0$

For $k=0, 1, 2, \ldots, T$

- **Solve**
  
  \[
  \min_{x, u} \sum_{t=k}^{T} g_t(x_t, u_t) \\
  \text{s.t.} \quad x_{t+1} = A_t x_t + B_t u_t \quad \forall t \in \{k, k+1, \ldots, T - 1\} \\
  f_i(x, u) \leq 0, \quad \forall i \in \{1, \ldots, m\} \\
  x_k = \bar{x}_k
  \]

- **Execute** $u_k$

- **Observe** resulting state, $\bar{x}_{k+1}$

= “Model Predictive Control”

Initialization with solution from iteration $k-1$ can make solver very fast (and would be done most conveniently with infeasible start Newton method)
Disciplined convex programming

= convex optimization problems of forms easily programmatically verified to be convex

Convenient high-level expressions

Excellent for fast implementation

Designed by Michael Grant and Stephen Boyd, with input from Yinyu Ye.

Current webpage: http://cvxr.com/cvx/
CVX

- Matlab Example for Optimal Control, see course webpage