310.

Lectra 0: Introduction variant of "Hatha Programming"

0.1. Formulation 

0.1.1. Formulation of Nonlinear Programming

- Numerical Optimization

Q Sometimes called "continuous constrained optimization".

\[
\begin{align*}
\text{(NLP)} & \quad \min f(x) \quad (= \max -f(x)) \quad \text{"objective"} \\
& \quad \text{s.t. } c_i(x) = 0 \quad i \in I \\
& \quad c_i(x) \leq 0 \quad i \in I
\end{align*}
\]

We Assume:

1. \( f: \mathbb{R}^n \rightarrow \mathbb{R} \) and \( c_i: \mathbb{R}^n \rightarrow \mathbb{R}; i \in \mathbb{N} \) are twice continuously differentiable.

2. \( f(x), c_i(x), \frac{\partial f(x)}{\partial x}, \{ \frac{\partial^2 c_i(x)}{\partial x^2} \}; i \in \mathbb{N} \) are computable and sometimes even \( \frac{\partial^2 f(x)}{\partial x^2} \) and \( \{ \frac{\partial^2 c_i(x)}{\partial x^2} \}; i \in \mathbb{N} \).
0.1.2 Abreviation: NLP

Why study it? It either represents or approximates distinguished application problems.
0. Introduction

0.2. Example Problems

0.2.1 A. Exactly Representable

1) Maximum likelihood Problems

\[ \min_{\theta} - L(\mathbf{x}; \theta) = \left( \min_{i} \log(L_i) \right) \]

Here \( L(\mathbf{x}; \theta) = \prod_{i=1}^{N} f(x_i; \theta) \)

2) Constrained Max Likelihood Problems

\[ V = \Theta_{M1} + \sum_{i=1}^{m-1} \theta_i \mathbf{z}_i \mathbf{z}_i^T \]

\( \mathbf{z}_i \) is an \( h \times 2i \) matrix

\( X \): fixed set of predictors

\[ \ell(\theta, \mathbf{x}; y) = -\frac{1}{2} \log \det(V) - \frac{1}{2} (y - \mathbf{x}\alpha)^T V^{-1} (y - \mathbf{x}\alpha) \]
0.2.2 \[ \max_{\theta, x} L(\theta, x; y) \]

\[ \theta_i \geq 0; \ i = 1, 2, \ldots, m \]

Observation: almost certainly \( \theta_i \to 0 \) at solution.

\[ \log(\det(\mathbf{V})) \] acts as a "barrier" for \( \theta_i \)

We will see this again in interior-point methods.

3) Transportation problems: Example 1.3 in book \( \to \) and many other management science problems.

4) Nonlinear least squares (if time)

\[ \min_{\theta} f(\theta) = \sum_{i=1}^{n} (y_i - g(x_i, \theta))^2 \]
B. Converted to NLP by approximation

\[ \min_{u(t) \in \mathcal{U}} J(u) = \int_0^T f(t, x(t), u(t)) \, dt \]
\[ \dot{x}(t) = g(t, x(t), u(t)), \quad x(0) = x_0 \]

and path constraints

\{ e.g. \quad g_i(x(t)) \leq 0 ; \quad i = 1, 2, \ldots \}
\{ e.g. \quad x_j(t) \in [c_{ij}, b_{ij}], \quad \text{etc} \}

\{ U: \text{admissible controls} \}

\{ e.g.: \quad U: \{ u: [0, T] \rightarrow \mathbb{R} \mid \text{measurable} \} \}
\{ U: \{ u: [0, T] \rightarrow \mathbb{R} \times [c_{ij}, b_{ij}] \} \}

Discretize

\[ \min_{\{ u_i(t_i) \}_{i=1}^N} \sum_{i=1}^N f(t_i, x(t_i), u(t_i)) \]
\[ x(t_{n+1}) = x(t_n) + \Delta t \, g(t_n, x(t_n), u(t_n)) \]
\[ x_j(t_n) \in [c_{ij}, b_{ij}] \]
0.2.4 Here \( x'(t_{n+1}) = x(t_{n+1}): \) Forward Euler
\( \tilde{x}(t_n) = x(t_n): \) Backward Euler

Example: Building thermal control
\( f: \) cost; \( g: \) dynamics \( \{ \text{heat equation, furnace} \}
path constraints: thermal comfort constraints

5) Stochastic Programming
If no time: Take it from same reference.

If time:
\[
\min_{x, y(\omega)} \mathbb{E}_{\omega} [f(x, y(\omega), \omega)]
\]
\( g_1(x) \leq 0 \) (vector notation)
\( g_2(x, y(\omega), \omega) \leq 0 \) \( \forall \omega \in \Omega \)
0.2.5.

\[ \text{If } \Omega \text{ is discrete and finite } \rightarrow \text{NLP} \]
\[ \text{If } \Omega \text{ is infinite } \rightarrow \text{SAA: sample average approximation} \]

\[ \min_{x, y(w_i)} \frac{1}{N_s} \sum_{i=1}^{N_s} f(x, y(w_i), w_i) \]
\[ \begin{align*}
  g_1(x) & \leq 0 \\
  g_2(x, y(w_i), w_i) & \leq 0
\end{align*} \]

\[ x^* \rightarrow x^* \] \rightarrow \text{See Shapiro.} \]

\[ \text{It is again an NLP} \]

Example: Optimal control of building system under ambient temperature uncertainty
0.3 Why use Matlab?

Matlab is a language environment for computational

0.3.1 Why use Matlab?

- Alternatives: Fortran, Java

This is not a complete test set, but results very hard to come by.

I will "paraphrase" Funk

\[
\text{Productivity} = \frac{1}{\text{Relative effort} \times \text{Time to solution}}
\]

\[
\text{relative effort} = \frac{\text{Speedup}}{\text{Time to solution}} = \text{Speedup}
\]

Relative effort = SLOC / Source line of code?
0.3.2

The term: Productivity = \frac{Speedup}{Relative SLOC}

For that test: about even...

So Matlab provides a slow solution but with steady developers trust.

If run twice \rightarrow Productivity from C doubles!! (SLOC is same)
0.3.3 When you need a solution for a problem you do not solve too often in a short amount of time → MATLAB.
0.1. INTRODUCTION

0.4. When and why use modeling language

0.4.1. AMPL: A modeling language for Mathematical Programming

AMPL approaches natural language for optimization.

\[ \text{AMPL} \]

but not

Several !!

\[ \text{No PDE support} \]

\[ \text{Does not} \]

Advantages \( \rightarrow \) Derives an executable

SLOC is small; speedy!

\[ \text{No PDE support} \]
0.4.2 Disadvantages

a) Not general. No PDE/ODE support
b) Only fundamental functions gradient

So we are likely to have trouble.
Cannot compute $\nabla \log(\det V(x))$!!
0.5 Special cases of nonlinear programming

0.5.1 A) When $\Sigma V \cap I = \emptyset$: unconstrained optimization

b) $I = \emptyset$; $f = 0 \Rightarrow$ Nonlinear Equation

c) $f, c$ linear: Linear Programming

D) $f$, $c_i$, convex; $f$, $c_i$, $i \in I$ convex programming

E) $f$: convex quadratic; $c_i$ linear convex quadratic programming
0. INTRODUCTION.

0.6. What does it mean "I solve NLP"?

Context. My aim is in local, iterative algorithms. What can I hope for?

Define "feasible set"

\[ F = \{ x \in \mathbb{R}^n \mid c_i(x) \leq 0 \text{ i.e. } I, c_i(x) = 0 \text{ i.e. } E \} \]

Note: \( I \neq \emptyset \), \( I \cup E = \emptyset \) \( \Rightarrow F = \mathbb{R}^n \)

Assume: \( F \neq \emptyset \).

A) Global Minimizer

\( \exists x^* \text{ is GM } \Leftrightarrow f(x) \leq f(x^*) \text{ } \forall x \in \mathbb{R}^n \)

B) Local Minimizer

\( \exists x^* \text{ is LM } \Leftrightarrow \exists N(x^*) \text{ s.t. } f(x^*) \leq f(x), \forall x \in N(x^*) \cup \mathbb{R}^n \)

\*
0.6.2 c) A point $x^*$ Isolated Local Minimizer

$\exists x^* \text{ is } ILM \, \text{if } \exists N(x^*) \, s.t. \, x^* \\
\text{is only LM in } N(x^*) \cap F$

D) $x^*$ Strict Local Minimizer

$\exists x^* \text{ is } SLM \, \text{if } \exists N(x^*) \, s.t. \, \\
f(x^*) < f(x) \, \forall x \in N(x^*) \cap F$

\[
\begin{align*}
\text{GM } &\Rightarrow \text{ LM, ILM, SLM} \\
\text{ILM} &\Rightarrow \text{ SLM} \\
\text{GM } &\not\Rightarrow \text{ ILM, SLM} \quad \text{e.g. } \min \frac{x+y}{2} \\
\text{s.t. } x+y &= 1 \\
x \geq 0; y \geq 0
\end{align*}
\]

$\frac{1}{2}, \frac{1}{2}$ is neither strict nor isolated

$\text{SLM } \not\Rightarrow \text{ ILM} \quad (\text{see example above}) \quad x^4 \cos x + 2x^4$

$\text{ILM } \Rightarrow \text{ SLM} \quad (I \text{ think, prove!})
0. INTRODUCTION

0.7 GOALS AND SCOPE

0.7.1 SCOPE: Iterative algorithms for NLP

CONstrained OPT

"Saddle point"

UNConstrained OPT

NON Convexity

LINEar EQUATIONS

Goals: Cannot hope to reach A, in general.

Goal A: If started from a vicinity $\epsilon x^*$, converge to it "Local Convergence" LM, ILM, SLM

Goal B:

From wherever I start my algorithm, I converge to a LM "Global Convergence"
0.7.2

For unconstrained optimization, I will get close.

For constrained optimization, constraint

must "help".

It is weak, but it is perhaps actually what good results one gets...