AMSC 607 / CMSC 764 Advanced Numerical Optimization

Fall 2008

UNIT 1: Introduction

Dianne P. O'Leary

Numerical Optimization

Topic of the course: Given a problem of the form

$$\min_{\mathbf{x}} f(\mathbf{x})$$

where $\mathbf{x} \in \mathcal{R}^n$, and $f : \mathcal{R}^n \rightarrow \mathcal{R}$,

- We want to be able to recognize a solution $\mathbf{x}^*$.
- We want to be able to compute a solution.

Sometimes the problem has constraints: for example, we might only be interested in vectors $\mathbf{x}$ that satisfy

$$\mathbf{c}(\mathbf{x}) \geq \mathbf{0}$$

where $\mathbf{c} : \mathcal{R}^n \rightarrow \mathcal{R}^m$.

We can also write the constraints as

$$c_i(x) \geq 0\,, \quad i = 1, \ldots, m\,.$$

Notes:

- f and c_i can be linear functions or nonlinear functions, but we'll assume that they are continuous.
- The vector $\mathbf{x}$ is not restricted to a discrete set of values (integers, for example). That is the topic of a different course!
- The number of variables n and the number of constraints m may be small (1 or 2) or quite large (thousands).

The plan

Goal: Give you a view of the state-of-the-art in numerical optimization for this kind of problem.

Big difficulty: This is a moving target!

First: Some background material implicit to the rest of our discussions.

- The course organization.
- Some sample optimization problems.
- Why this isn't a math course.
  - Floating point arithmetic.
  - Algorithmic design.

First, the course organization

Handouts:

- Course information: text, grading, etc.
- Course syllabus.
- Information form: please fill out at end of class.

Some motivating examples

Example 1: Financial and other planning

The Prost Company has the equipment to manufacture n different products: gizmos, widgets, etc.

The manufacture uses m different resources: steel, paper, labor time on the punch machine, etc.

To produce 1 unit of Product 1, the gizmo, we use

- a_{11} units of steel
- a_{21} units of paper
- a_{31} units on the punch machine
- etc.

and we make a profit of c_1 dollars.

To produce one unit of Product 2, the widget, we use

- a_{12} units of steel
- a_{22} units of paper
- a_{32} units on the punch machine

- etc.

and we make a profit of c_2 dollars.

We have b_1 units of steel available, b_2 units of paper, b_3 units on the punch machine, etc.

Problem: Maximize the profit, using only the resources that are available:

$$\max c_1x_1 + c_2x_2 + \ldots c_nx_n = \mathbf{c}^T\mathbf{x}$$

subject to

$$\mathbf{x} \geq \mathbf{0}$$

and

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &\leq b_2 \\ &\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &\leq b_m \end{aligned}$$

or

$$\mathbf{Ax} \leq \mathbf{b}$$

This is called a linear programming problem because the objective function $\mathbf{c}^T\mathbf{x}$ and the constraints $\mathbf{x} \geq \mathbf{0}$ and $\mathbf{Ax} \leq \mathbf{b}$ are all linear.

Example 2: Engineering

Suppose we have a bar, attached to a wall at a 90^o angle.

We want to know how the bar will behave as we twist it.

Let θ measure the twist per unit length, and let ϕ be the stress function, with G equal to the shear modulus.

The usual mathematical models:

- If the material behaves elastically:

  $$\bigtriangledown^2\phi = -2G\theta$$

  in a cross-section D of the bar, with

  $$\phi = 0$$

  on the boundary.

- If we pass the plastic boundary, then ϕ minimizes

$$J(\phi) = \frac{1}{2} \int \int_D | \bigtriangledown \phi|^2 dA - 2G\theta \int \int_D \phi(x, y) dA$$

with

$$\phi = 0$$

on the boundary, but with the extra condition that the stress remains less than some given yield stress value σ:

$$| \bigtriangledown \phi| \leq \sigma$$

We can't solve the problem without some discretization, either by finite differences or finite elements. Using either of these techniques, we get a minimization problem with

- a quadratic objective function.
- linear and quadratic constraints.

Example 3: Data fitting

Suppose we measure some data

$$(t_i, y_i)$$

$i = 1, \ldots, m$, and we want to fit a mathematical model to the data. Perhaps the model is

$$y(t) \approx \alpha e^{\beta t} + \gamma e^{\delta t} .$$

Then we might choose to determine the parameters by a least squares criterion:

$$\min_{\alpha,\beta,\gamma,\delta} \sum_{i=1}^{m} (y_i - (\alpha e^{\beta t_i} + \gamma e^{\delta t_i}))^2$$

There may also be some constraints to the problem; for example, we might want a model in which α and γ are positive, or in which β and δ are negative.

Other examples

Many sources of optimization problems:

- Economics
- Models of fluid flow
- Optimal control

- etc.

Why this isn't a math course

There are three main obstacles to looking at our problem as an abstract math problem:

- We will need to solve it on a computer, which means we will need to deal with round-off error.
- We have limited resources, so we need algorithms that are efficient in time and storage.
- The origins of the problem often give good clues about how it might be solved efficiently, so it is important to know something about the application areas, too, or to team up with someone who does.

Dealing with round-off error

A motivating example about error

Example:

$$\begin{bmatrix} 10^{-6} & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 + 10^{-6} \\ 3 \end{bmatrix}$$

with solution $\mathbf{x} = [1, 2]^T$.

After one step of Gauss Elimination, on a machine that carries 5 decimal digits, we have transformed the problem to

$$\begin{bmatrix} 10^{-6} & 1 \\ 0 & -10^{6} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \times 10^{6} \end{bmatrix}$$

giving a solution of $x_2 = 2$, so $x_1 = 0$.

Fatal flaw: Although the problem has a well-defined solution, our algorithm was unstable.

We could avoid this problem by using a stable algorithm, performing the elimination after reordering the equations to bring a large element to the (1,1) position. In 6-digit arithmetic, this yields

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

so the computed solution becomes $x_2 = 2$, $x_1 = 1$.

What does this have to do with optimization?

We will see that solving linear systems is the easiest ingredient in our optimization algorithms.

It is important to be aware that rounding errors can affect all of our computations.

Although we won't always emphasize it explicitly, be aware that we need to be careful about how we design algorithms to

- solve linear systems.
- determine parameters like step lengths.
- evaluate the objective function.
- evaluate the constraints.
- evaluate any derivatives we use.

Failure to get accuracy in any one of these components can ruin the entire algorithm.

(If machine arithmetic were exact, this course would be 1 hour instead of 3.)

In addition, we need to make sure that our data, the coefficients that were measured, and our model are good enough.

If not, garbage in, garbage out.

For more information about error analysis:

See the (optional) notes on error posted on the homepage.

Design of Algorithms

Three types of computer programs:

- quick-and-dirty: We need to do a job only once or so, and expect the code to have a short lifespan.
- production code:
  - Longer lifespan.
  - Program may evolve.
  - The user did not write the program.
- library code: must be reliable enough that the programmer doesn't need to be consulted.

The principles of algorithm design are different for each, but only in degree.

For concreteness, in this course we will play the role of program librarian.

We seek codes that are:

- reliable. (In particular, we must use stable algorithms.)
- well documented.
- modular, so that they share pieces.
- convenient for the user, so we reduce the temptation to tinker.
- efficient.
- easy to modify.
- portable.

We want a good general purpose algorithm, and then may choose some special purpose codes to solve certain types of problem that arise frequently in our domain.

My perspective

For a given problem:

- Is the problem well-posed?
- How will I recognize a solution?
- What properties of this particular problem help me solve it?
- What algorithm should I use?
- How sensitive is the solution to small changes in the data?

Final words

- For each algorithm we study, have in mind a model problem that you could apply the algorithm to.
- Keep in mind the necessity to develop numerically stable algorithms. Review error analysis and floating point arithmetic, if necessary.
- In your programming assignments, aim to write library quality codes, at least in the documentation, modularity, efficiency, and reliability.

About the notes:

The lecture notes are a work-in-progress. There are lots of typos in them. I'll appreciate your help during and after class to try to find all of the errors.