Applied Mathematics 205

Unit IV: Nonlinear Equations and Optimization

Lecturer: Dr. David Knezevic

Unit IV: Nonlinear Equations and Optimization

Chapter $\mathrm{IV}.1$: Motivation

Nonlinear Equations

Motivation: Nonlinear Equations

So far we have mostly focused on linear phenomena

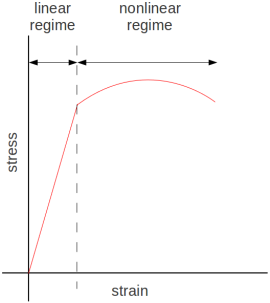
- Interpolation leads to a linear system $Vb = y$ (monomials) or $\mathrm{I}b = y$ (Lagrange polynomials)
- Linear least-squares leads to the normal equations $A^T Ab = A^T y$
- We saw examples of linear physical models (Ohm's Law, Hooke's Law, Leontief equations) $\implies Ax = b$
- F.D. discretization of a linear PDE leads to a linear algebraic system $AU = F$

Motivation: Nonlinear Equations

Of course, nonlinear models also arise all the time

- Nonlinear least-squares, Gauss-Newton/Lev.-Mar. in I.4

- Many nonlinear physical models in nature, e.g. non-Hookean material models[1]



- F.D. discretization of a non-linear PDE leads to a nonlinear algebraic system

[1] Important in modeling large deformations of solids

Motivation: Nonlinear Equations

Another example is computation of Gauss quadrature points/weights

We know this is possible via roots of Legendre polynomials

But we could also try to solve the nonlinear system of equations for $\{(x_1, w_1), (x_2, w_2), \ldots, (x_n, w_n)\}$

Motivation: Nonlinear Equations

e.g. for $n = 2$, we need to find points/weights such that all polynomials of degree 3 are integrated exactly, hence

$$
\begin{aligned}
w_1 + w_2 &= \int_{-1}^{1} 1 \mathrm{d}x = 2 \\
w_1 x_1 + w_2 x_2 &= \int_{-1}^{1} x \mathrm{d}x = 0 \\
w_1 x_1^2 + w_2 x_2^2 &= \int_{-1}^{1} x^2 \mathrm{d}x = 2/3 \\
w_1 x_1^3 + w_2 x_2^3 &= \int_{-1}^{1} x^3 \mathrm{d}x = 0
\end{aligned}
$$

Motivation: Nonlinear Equations

We usually write a nonlinear system of equations as

$$F(x) = 0,$$

where $F : \mathbb{R}^n \to \mathbb{R}^m$

We implicity absorb the "right-hand side" into F and seek a root of F

In this Unit we focus on the case $m = n$, $m > n$ gives nonlinear least-squares

Motivation: Nonlinear Equations

We are very familiar with scalar ($m = 1$) nonlinear equations

Simplest case is a quadratic equation

$$ax^2 + bx + c = 0$$

We can write down a closed-form solution[2], the quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

[2] A closed-form expression involves only a finite number of "well-known" functions, e.g. $+, -\times, \div$, trigonometric functions, logarithms, etc

Motivation: Nonlinear Equations

In fact, there are also closed-form solutions for arbitrary cubic and quartic polynomials, due to Ferrari and Cardano (~ 1540)

Important mathematical result (Galois, Abel) is that there is no closed-form solution for fifth or higher order polynomial equations

Hence, even for the simplest possible type of nonlinear equation (polynomials on $\mathbb{R}$), only hope is to employ an iterative algorithm

An iterative method should converge in the limit $n \to \infty$, and ideally yields an accurate approximation after few iterations

Motivation: Nonlinear Equations

There are many well-known iterative methods for nonlinear equations

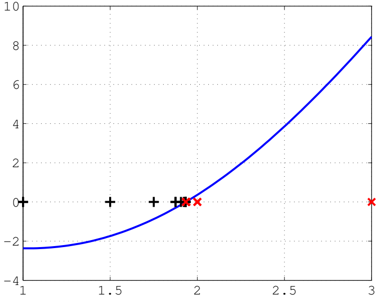
Probably the simplest is the bisection method for a scalar equation $f(x) = 0$, where $f \in C[a, b]$

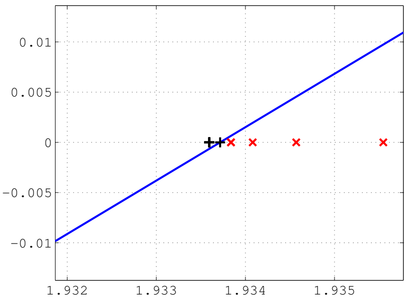
Look for a root in the interval $[a, b]$ by bisecting based on sign of f

Motivation: Nonlinear Equations

```
a = 1; b = 3;
f = @(x)(x.^2 - 4*sin(x));
x = linspace(a,b);
plot(x,f(x),'linewidth',2)
hold on
TOL = 1e-4;
while( (b-a) > TOL )
    plot(a,0,'k+','markersize',10,'linewidth',2)
    plot(b,0,'rx','markersize',10,'linewidth',2)
    m = a + (b-a)/2;
    if sign(f(a)) == sign(f(m))
        a = m
    else
        b = m
    end
end
```

Motivation: Nonlinear Equations





Root in the interval $[1.933716, 1.933777]$

Motivation: Nonlinear Equations

Bisection is a robust root-finding method in 1D, but it does not generalize easily to $\mathbb{R}^n$ for $n > 1$

Also, bisection is a crude method in the sense that it only uses $\text{sign}(f)$, and ignores the magnitude and gradient of f

We will consider methods which generalize to $\mathbb{R}^n$, and which converge faster than bisection:

- Fixed-point iteration
- Newton's method

Optimization

Motivation: Optimization

Another major topic in Scientific Computing is optimization

Very important in science, engineering, industry, finance, economics, logistics,...

Many engineering challenges can be formulated as optimization problems, e.g.:

- Design car body that maximizes downforce[3]
- Design a bridge with minimum weight

[3] An important design goal in racing car design

Motivation: Optimization

Of course, in practice, it is more realistic to consider optimization problems with constraints, e.g.:

- Design car body that maximizes downforce, subject to a constraint on drag
- Design a bridge with minimum weight, subject to a constraint on stiffness

Motivation: Optimization

Also, (constrained and unconstrained) optimization problems arise naturally in science

Physics:

- Many physical systems will naturally occupy a minimum energy state
- If we can describe the energy of the system mathematically, then we can find minimum energy state via optimization

Motivation: Optimization

Biology:

- Computational optimization of, e.g. fish swimming or insect flight, can reproduce behavior observed in nature
- This fits with the idea that evolution has been "optimizing" organisms for millions of year

Motivation: Optimization

All of these problems can be formulated as: Optimize (max. or min.) an objective function over a set of feasible choices, i.e.

> Given an objective function $f : \mathbb{R}^n \to \mathbb{R}$ and a set $S \subset \mathbb{R}^n$, we seek $x^* \in S$ such that $f(x^*) \leq f(x)$, $\forall x \in S$

(It suffices to consider only minimization, maximization is equivalent to minimizing $-f$)

S is the feasible set, usually defined by a set of equations and/or inequalities, which are the constraints

If $S = \mathbb{R}^n$, then the problem is unconstrained

Motivation: Optimization

The standard way to write an optimization problem for $S \subset \mathbb{R}^n$ is

$$\min_{x \in S} f(x) \text{ subject to } g(x) = 0 \text{ and } h(x) \leq 0,$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^p$

Here m and p are the number of equality and inequality constraints, respectively

Motivation: Optimization

For example, let x_1 and x_2 denote radius and height of a cylinder, respectively

Minimize the surface area of a cylinder subject to a constraint on its volume[4] (we will return to this example later)

$$\min_{x \in \mathbb{R}^2} f(x_1, x_2) = 2\pi x_1(x_1 + x_2)$$

$$\text{subject to } g(x_1, x_2) = \pi x_1^2 x_2 - V = 0$$

[4]Heath Example 6.2

Motivation: Optimization

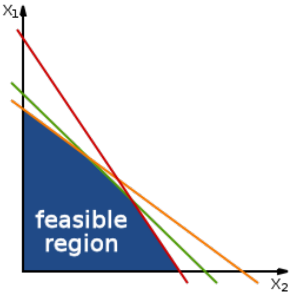
If f, g and h are all affine, then the optimization problem is called a linear program

(Here the term "program" has nothing to do with computer programming; instead it refers to logistics/planning)

Affine if $f(x) = Ax + b$ for a matrix A, i.e. linear plus a constant[5]

Linear programming may already be familiar

Just need to check $f(x)$ on vertices of the feasible region



[5]Recall that "affine" is not the same as "linear", i.e. $f(x+y) = Ax + Ay + b$ and $f(x) + f(y) = Ax + Ay + 2b$

Motivation: Optimization

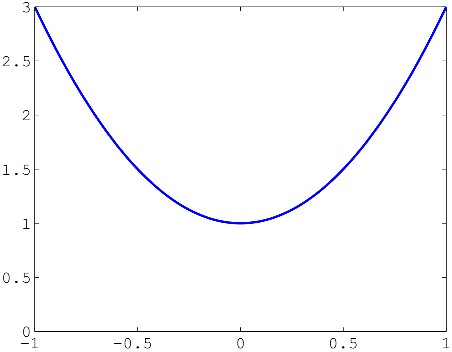
If the objective function or any of the constraints are nonlinear then we have a nonlinear optimization problem or nonlinear program

We will consider several different approaches to nonlinear optimization in this Unit

Optimization routines typically use local information about a function to iteratively approach a local minimum

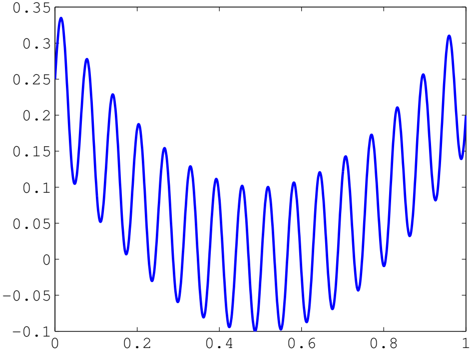
Motivation: Optimization

In some cases this easily gives a global minimum



Motivation: Optimization

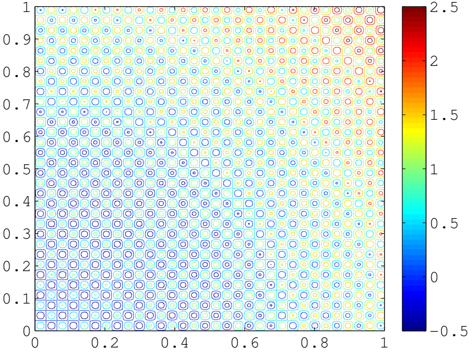
But in general, global optimization can be very difficult



We can get "stuck" in local minima!

Motivation: Optimization

And can get much harder in higher spatial dimensions



Motivation: Optimization

There are robust methods for finding local minimima, and this is what we focus on in AM205

Global optimization is very important in practice, but in general there is no way to guarantee that we will find a global minimum

Global optimization basically relies on heuristics:

- try several different starting guesses ("multistart" methods)
- stochastic methods, e.g. Markov Chain Monte Carlo (MCMC), see AM207
- simulated annealing
- "genetic" methods