Newton's Fractal Fragility

Fredrik Magnusson

Department of Automatic Control
Faculty of Engineering
Lund University, Sweden

October 21, 2016

Research

Thesis defense in 4 weeks. *Numerical and Symbolic Methods for Dynamic Optimization*. Keywords:

- Modelica
- differential-algebraic equations
- dynamic optimization
- direct collocation
- block-triangular ordering

The problem

- We want to (numerically) solve a nonlinear system of equations

$$f(x) = 0, \quad f : \mathbb{R}^n \to \mathbb{R}^n$$

- Several possible methods. Methods based on Newton's method are most widely used.
- Fixed-point iterations are a common alternative. Properties of Newton compared to fixed-point methods:
  - $+$ fast (second-order) convergence
  - $\pm$ requires differentiability of f, rather than contractivity
  - $-$ expensive iterations (solve a linear system of equations)
- All algorithms of MATLAB's `fsolve` based on Newton

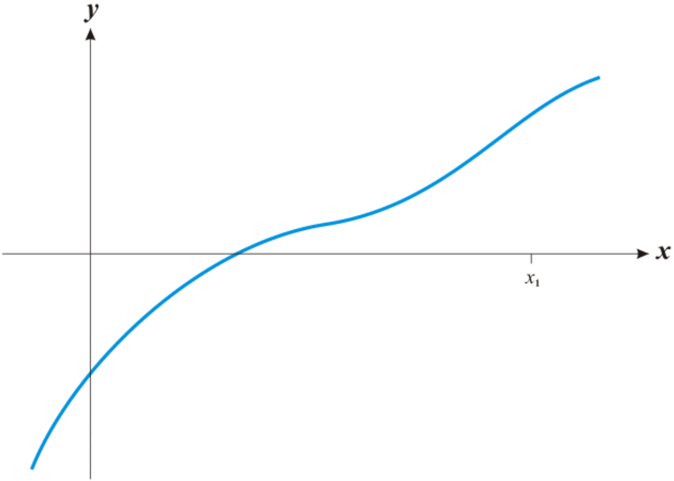
The application

Many applications of this (maybe even more than there are applications not of this). In particular:

- Numerical optimization (KKT conditions can be transformed into nonlinear system of equations)
- Solving nonlinear differential equations (implicit integration methods)

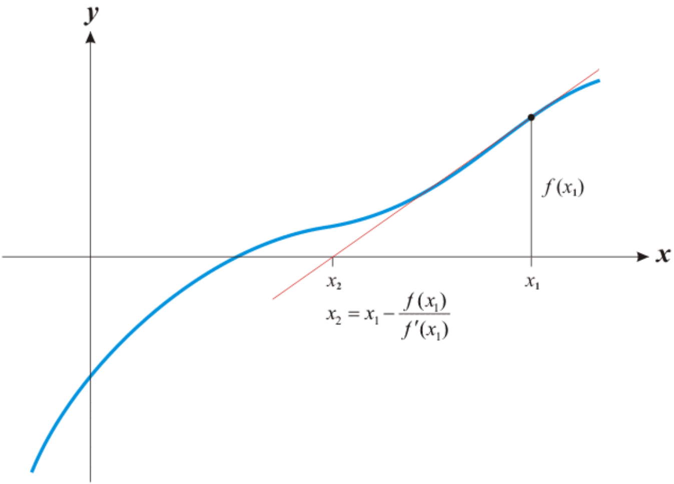
The method

$$x_{k+1} = x_k - J^{-1}(x_k) f(x_k).$$



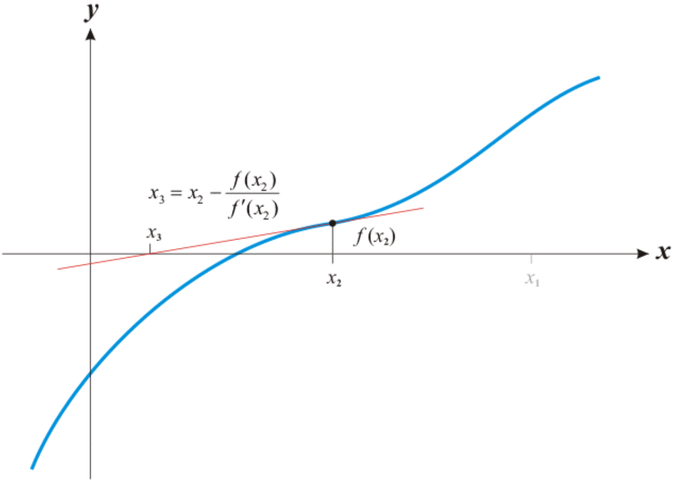
The method

$$x_{k+1} = x_k - J^{-1}(x_k) f(x_k).$$



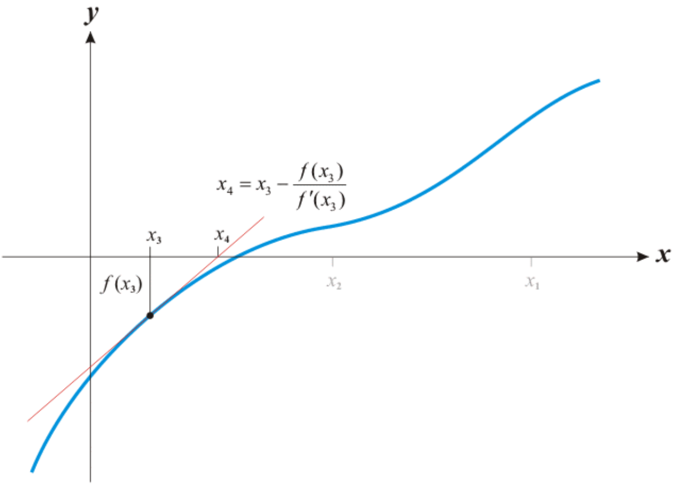
The method

$$x_{k+1} = x_k - J^{-1}(x_k) f(x_k).$$



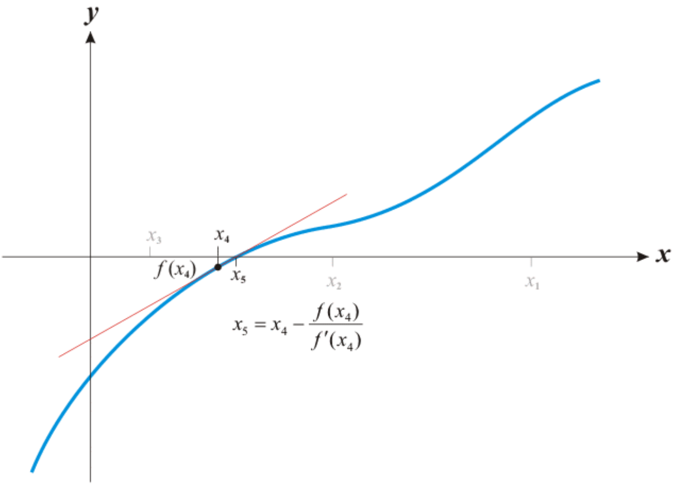
The method

$$x_{k+1} = x_k - J^{-1}(x_k)f(x_k).$$



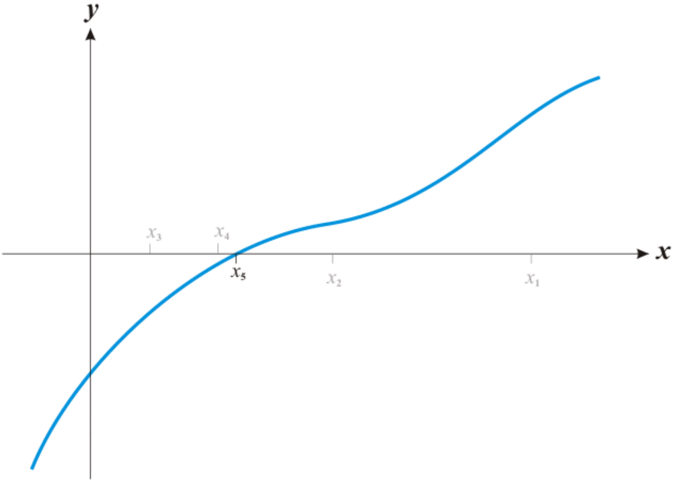
The method

$$x_{k+1} = x_k - J^{-1}(x_k)f(x_k).$$



The method

$$x_{k+1} = x_k - J^{-1}(x_k) f(x_k).$$



The issue

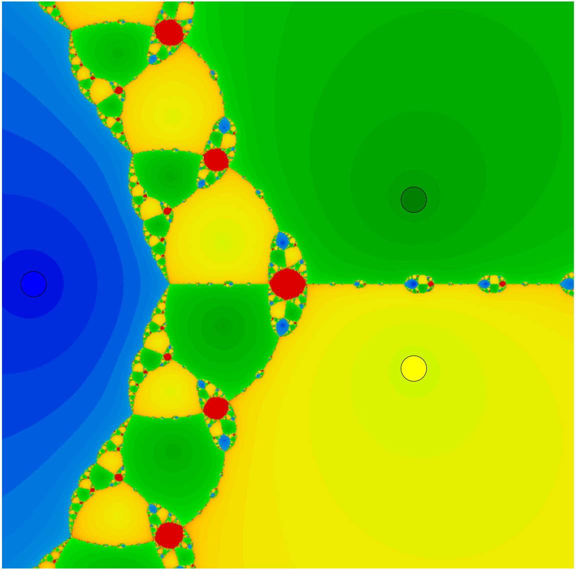
Convergence if all of the following:

- f is sufficiently regular (C^1 or something along those lines)
- The initial guess is sufficiently close to the solution
- Uninteresting technicalities

Second condition is critical for many applications. Consider the very simple example $f(z) = z^3 - 2z + 2 = 0$.

- Two-dimensional if we let $z \in \mathbb{C}$
- Three solutions

Convergence regions



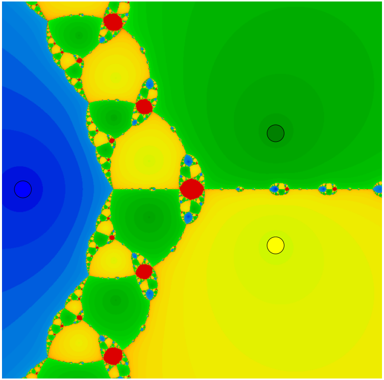
Fractal

"Beautiful, damn hard, increasingly useful. That's fractals."

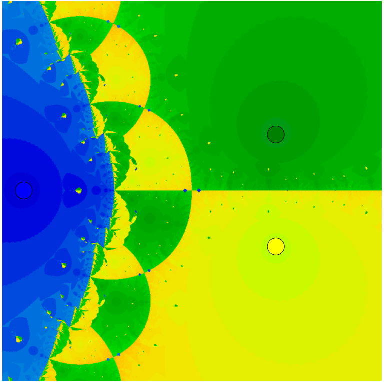
— Mandelbrot

- Newton's method generates fractals for large classes of root-finding problems
- Consequence: Exist points such that small perturbations yield unpredictable behavior
- Newton iterations can be viewed as a chaotic dynamic system
- Vanilla Newton is never used in practice. What about IPOPT?

IPOPT convergence



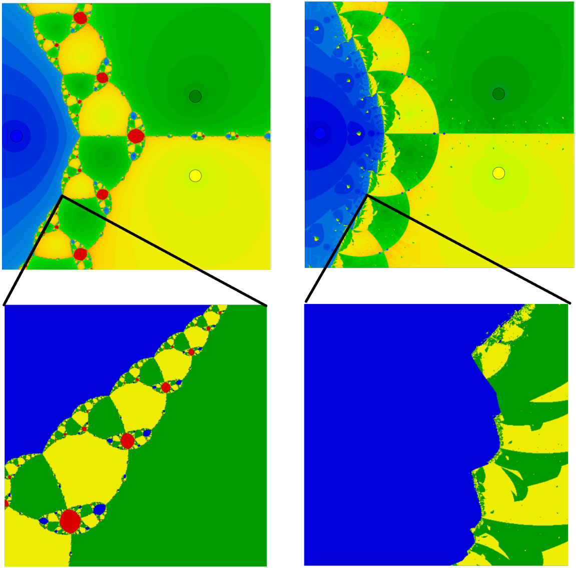
Newton



IPOPT

- $+$ Divergence is gone
- $\pm$ Fractal?
- $-$ Smaller region of convergence

Fractal zoom



- Fractality of Newton is clear
- IPOPT fractal? Probably not. But probably chaotic.

Pendulum on a cart

```
optimization pendulum(
    objective=finalTime, startTime=0,
    finalTime(free=true, min=0.1, max=10,
              initialGuess=3.371260290708))

    parameter Real l = 0.345;
    parameter Real lact = 0.4;
    parameter Real g = 9.81;
    Real p(start=-0.8,fixed=true);
    Real p_dot(start=0,fixed=true);
    Real theta(start=0,fixed=true);
    Real theta_dot(start=0,fixed=true);
    Real x_p;
    Real y_p;
    input Real a_ref_dot(free=true);
    Real a_ref(start=0,fixed=true);
```

Perturbations

- Perturbed initial guess are rarely a problem in practice
- However, problems themselves are often “perturbed”, with the same consequence!
  - Add a variable and trivial equation, e.g., $y = 3$
  - Reorder the equations/variables
  - Leads to different computations on the order of $\epsilon_{\mathrm{mach}} \implies$ can knock Newton off course

Conclusions

- Newton's method is not robust
- Professional implementations only partially resolve the issue
- Practical concern for many applications. Insufficient awareness!
- But at least you can make pretty pictures

The end