Lecture 4: Miscellaneous Numerical Tools

Fatih Guvenen
University of Minnesota

November 2023

High-Quality Free Source Codes: Where to Find?

- Lots of high-quality free software for the computational tools we will learn in this class.
- Note: Different implementations can differ substantially! So, get it from a reliable, broadly used source.

High-Quality Free Source Codes: Where to Find?

- Lots of high-quality free software for the computational tools we will learn in this class.
- Note: Different implementations can differ substantially! So, get it from a reliable, broadly used source.
- A few good options (there are many more):
  - Numerical Recipes code (Fortran and C)
  - GNU Scientific Library (C): http://www.gnu.org/software/gsl/
  - Netlib repository (Fortran and C): https://www.netlib.org
  - NLOPT: lots of optimization routines in many languages: https://nlopt.readthedocs.io/

High-Quality Free Source Codes: Where to Find?

- Lots of high-quality free software for the computational tools we will learn in this class.
- Note: Different implementations can differ substantially! So, get it from a reliable, broadly used source.
- A few good options (there are many more):
  - Numerical Recipes code (Fortran and C)
  - GNU Scientific Library (C): http://www.gnu.org/software/gsl/
  - Netlib repository (Fortran and C): https://www.netlib.org
  - NLOPT: lots of optimization routines in many languages: https://nlopt.readthedocs.io/
- In general, avoid downloading software from some random researcher's website.
  - You often won't know who originally wrote the code,
  - Whether it was modified for the specific use of that researcher.

Maximizing RHS

$$V(k,z) = \max_{c,k'} \left[u(c) + \beta \int V(k',z') f(z'|z) dz' \right]$$

$$c + k' = (1+r)k + z$$

$$z' = \rho z + \eta.$$

Three Steps:

1. As part of maximizing the RHS, evaluate $\mathbb{E}(V(k',z')|z)$ *repeatedly*. Two components:
    1. Interpolation: Maybe required twice.
        1. Interpolate in the k' direction.
        2. Also interpolate in the z' direction, if $f(z'|z = z_j)$ is continuous.
    2. Integration: Again, if $f(z'|z = z_j)$ is continuous, we need to integrate. One option will be to treat it as discrete (saves time and headaches; not always feasible).

Maximizing RHS

$$V(k,z) = \max_{c,k'} \left[u(c) + \beta \int V(k',z') f(z'|z) dz' \right]$$

$$c + k' = (1+r)k + z$$

$$z' = \rho z + \eta.$$

Three Steps:

1. As part of maximizing the RHS, evaluate $\mathbb{E}(V(k',z')|z)$ *repeatedly*. Two components:
    1. Interpolation: Maybe required twice.
        1. Interpolate in the k' direction.
        2. Also interpolate in the z' direction, if $f(z'|z = z_j)$ is continuous.
    2. Integration: Again, if $f(z'|z = z_j)$ is continuous, we need to integrate. One option will be to treat it as discrete (saves time and headaches; not always feasible).
2. How to perform the maximization in the Bellman objective? (Constrained optimization)

Root Finding

ROOT FINDING

Introduction

- Let $f : \mathbb{R}^N \rightarrow \mathbb{R}^N$. Solve $f(x) = 0$.
  - Depending on f, there may be zero, one, or multiple solutions.
- I will start with the root finding problem in one dimension $(N = 1)$.
- $N = 1$ is a special case where we can guarantee that we are bracketing a zero.

Introduction

- Let $f : \mathbb{R}^N \to \mathbb{R}^N$. Solve $f(x) = 0$.
  - Depending on f, there may be zero, one, or multiple solutions.
- I will start with the root finding problem in one dimension ($N = 1$).
- $N = 1$ is a special case where we can guarantee that we are bracketing a zero.
- In higher-dimensional problems, no method can (generically) guarantee that you are bracketing a zero.
- In higher dimensions, I often convert zero finding into a minimization problem. More on this in a moment.

Bisection, Secant and False Position Methods

Many zero finding methods begin with two points that bracket a zero:
$f(a) \times f(b) < 0$

Bisection, Secant and False Position Methods

Many zero finding methods begin with two points that bracket a zero:
$f(a) \times f(b) < 0$

- **Bisection:**

Bisection, Secant and False Position Methods

Many zero finding methods begin with two points that bracket a zero:
$f(a) \times f(b) < 0$

- **Bisection:**
  1. First, evaluate $f(\frac{a+b}{2})$. For $x = a, b$

Many zero finding methods begin with two points that bracket a zero: $f(a) \times f(b) < 0$

- **Bisection:**
  1. First, evaluate $f(\frac{a+b}{2})$. For $x = a, b$
  2. if $f(\frac{a+b}{2}) \times f(x) > 0$, replace x with $\frac{a+b}{2}$. Go back to step 1.

Bisection, Secant and False Position Methods

Many zero finding methods begin with two points that bracket a zero:
$f(a) \times f(b) < 0$

- **Bisection:**

  1. First, evaluate $f(\frac{a+b}{2})$. For $x = a, b$
  2. if $f(\frac{a+b}{2}) \times f(x) > 0$, replace x with $\frac{a+b}{2}$. Go back to step 1.

  - Relatively slow (linear convergence) but converges **for sure.**
  - In some challenging problems (will see in later lectures) where other methods fail, this is the best fallback

Bisection, Secant and False Position Methods

Many zero finding methods begin with two points that bracket a zero:
$f(a) \times f(b) < 0$

- **Bisection:**
  1. First, evaluate $f(\frac{a+b}{2})$. For $x = a, b$
  2. if $f(\frac{a+b}{2}) \times f(x) > 0$, replace x with $\frac{a+b}{2}$. Go back to step 1.
  - Relatively slow (linear convergence) but converges **for sure.**
  - In some challenging problems (will see in later lectures) where other methods fail, this is the best fallback
- Secant and False position methods:

Bisection, Secant and False Position Methods

Many zero finding methods begin with two points that bracket a zero: $f(a) \times f(b) < 0$

- **Bisection:**
  1. First, evaluate $f(\frac{a+b}{2})$. For $x = a, b$
  2. if $f(\frac{a+b}{2}) \times f(x) > 0$, replace x with $\frac{a+b}{2}$. Go back to step 1.
  - Relatively slow (linear convergence) but converges **for sure.**
  - In some challenging problems (will see in later lectures) where other methods fail, this is the best fallback

- Secant and False position methods:
  - Idea: Treat $f(\bullet)$ as if it is linear near the zero. Given $f(a)$ and $f(b)$, solve for the zero of the line that connects these two points.

Secant and False Position Methods

The two methods are very similar but differ in whether they lean towards **speed** (secant) or **robustness** (false position)

(see fig 9.2 in Numerical Recipes 77 book for a comparison)

Secant and False Position Methods

The two methods are very similar but differ in whether they lean towards **speed** (secant) or **robustness** (false position)

(see fig 9.2 in Numerical Recipes 77 book for a comparison)

- **Secant:** take the two most recently evaluated points.

Secant and False Position Methods

The two methods are very similar but differ in whether they lean towards **speed** (secant) or **robustness** (false position)

(see fig 9.2 in Numerical Recipes 77 book for a comparison)

- **Secant:** take the two most recently evaluated points.
  - Secant method is very aggressive because it always throws out the oldest evaluation, which yields superlinear convergence: $\lim_{k\to\infty} |\epsilon_{k+1}| \approx K \times |\epsilon|^{1.618}$.

Secant and False Position Methods

The two methods are very similar but differ in whether they lean towards **speed** (secant) or **robustness** (false position)

(see fig 9.2 in Numerical Recipes 77 book for a comparison)

- **Secant:** take the two most recently evaluated points.
  - Secant method is very aggressive because it always throws out the oldest evaluation, which yields superlinear convergence: $\lim_{k\to\infty} |\epsilon_{k+1}| \approx K \times |\epsilon|^{1.618}$.
- **False position**: take two most recent points that bracket zero.

Secant and False Position Methods

The two methods are very similar but differ in whether they lean towards **speed** (secant) or **robustness** (false position)

(see fig 9.2 in Numerical Recipes 77 book for a comparison)

- **Secant:** take the two most recently evaluated points.
  - Secant method is very aggressive because it always throws out the oldest evaluation, which yields superlinear convergence: $\lim_{k\to\infty} |\epsilon_{k+1}| \approx K \times |\epsilon|^{1.618}$.
- **False position**: take two most recent points that bracket zero.
  - Typically slower than Secant because it sometimes keeps an older evaluation in the interest of bracketing a zero (hard to determine exact convergence rate).

What to Use in Practice?

- None of the above! These simple methods are often building blocks of more complex methods that you should use in practice.

What to Use in Practice?

- None of the above! These simple methods are often building blocks of more complex methods that you should use in practice.
  - But they are useful for understanding more complex methods.

What to Use in Practice?

- None of the above! These simple methods are often building blocks of more complex methods that you should use in practice.
  - But they are useful for understanding more complex methods.

1. If derivatives are (i) painfully expensive to compute or (ii) unreliable (we will see examples) then use Brent's method.

What to Use in Practice?

- None of the above! These simple methods are often building blocks of more complex methods that you should use in practice.
  - But they are useful for understanding more complex methods.

1. If derivatives are (i) painfully expensive to compute or (ii) unreliable (we will see examples) then use Brent's method.
2. Else, use the Newton-Raphson method—but only when you know you are near a zero!

What to Use in Practice?

- None of the above! These simple methods are often building blocks of more complex methods that you should use in practice.
  - But they are useful for understanding more complex methods.

1. If derivatives are (i) painfully expensive to compute or (ii) unreliable (we will see examples) then use Brent's method.
2. Else, use the Newton-Raphson method—but only when you know you are near a zero!

- **TIP:** Convert "zero finding" into a minimization problem: if $f(a) \times f(b) < 0$ then the squared function $f(\bullet)^2$ will have a local minimum at the zero of $f(\bullet)$.

What to Use in Practice?

- None of the above! These simple methods are often building blocks of more complex methods that you should use in practice.
  - But they are useful for understanding more complex methods.

1. If derivatives are (i) painfully expensive to compute or (ii) unreliable (we will see examples) then use Brent's method.
2. Else, use the Newton-Raphson method—but only when you know you are near a zero!

- **TIP:** Convert "zero finding" into a minimization problem: if $f(a) \times f(b) < 0$ then the squared function $f(\bullet)^2$ will have a local minimum at the zero of $f(\bullet)$.
- Then one can use various methods for minimization (not surprisingly Brent and Newton's methods also have versions for optimization).

Brent's method

TIP: **Brent's method is your best bet** if you really want to avoid using derivatives.

Brent's method

TIP: Brent's method is your best bet if you really want to avoid using derivatives.

- It combines the speedier version of the Secant method with the reliability of bisection.

Brent's method

TIP: Brent's method is your best bet if you really want to avoid using derivatives.

- It combines the speedier version of the Secant method with the reliability of bisection.
- Basically, it takes two points (a, b) that bracket a zero and a third one in between (e.g., $\frac{a+b}{2}$)

Brent's method

TIP: Brent's method is your best bet if you really want to avoid using derivatives.

- It combines the speedier version of the Secant method with the reliability of bisection.
- Basically, it takes two points (a, b) that bracket a zero and a third one in between (e.g., $\frac{a+b}{2}$)
- It fits a quadratic (instead of the linear function in Secant) to these three points. This yields faster convergence.

Brent's method

TIP: Brent's method is your best bet if you really want to avoid using derivatives.

- It combines the speedier version of the Secant method with the reliability of bisection.
- Basically, it takes two points (a, b) that bracket a zero and a third one in between (e.g., $\frac{a+b}{2}$)
- It fits a quadratic (instead of the linear function in Secant) to these three points. This yields faster convergence.
- However, it carefully checks that the points always bracket a zero and the algorithm converges nicely. If not, it reverts back to bisection for a while.

Newton-Raphson Method

- First-order Taylor approximation:

$$f(x+\delta) = f(x) + \delta f'(x) + \text{higher order terms we will ignore...}$$

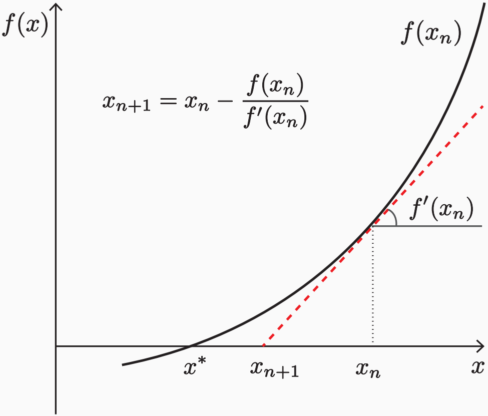
Setting $f(x+\delta) = 0$ yields $\delta = -\frac{f(x)}{f'(x)}$.

- Therefore, begin with x_0 and update:

$$\delta = x_{t+1} - x_t \Rightarrow x_{t+1} = x_t - \frac{f(x)}{f'(x)}$$

until convergence (if it does!)

Newton-Raphson method



Newton-Raphson method

- Pros:
  - Quadratic convergence with each step! (That is, the number of significant digits doubles in every step!)
  - Extends easily to multi-dimensional problems.

Newton-Raphson method

- Pros:
  - Quadratic convergence with each step! (That is, the number of significant digits doubles in every step!)
  - Extends easily to multi-dimensional problems.

- Cons:
  - computation of derivatives can be very slow if done numerically (as opposed to having an analytical formula).

Newton-Raphson method

- Pros:
  - Quadratic convergence with each step! (That is, the number of significant digits doubles in every step!)
  - Extends easily to multi-dimensional problems.

- Cons:
  - computation of derivatives can be very slow if done numerically (as opposed to having an analytical formula).
  - Poor global convergence properties.

Newton-Raphson method

- Pros:
  - Quadratic convergence with each step! (That is, the number of significant digits doubles in every step!)
  - Extends easily to multi-dimensional problems.
- Cons:
  - computation of derivatives can be very slow if done numerically (as opposed to having an analytical formula).
  - Poor global convergence properties.
  - If there is a local minimum near the zero, updating can overshoot to infinity.
  - Use with maximum care!!

Newton-Raphson method

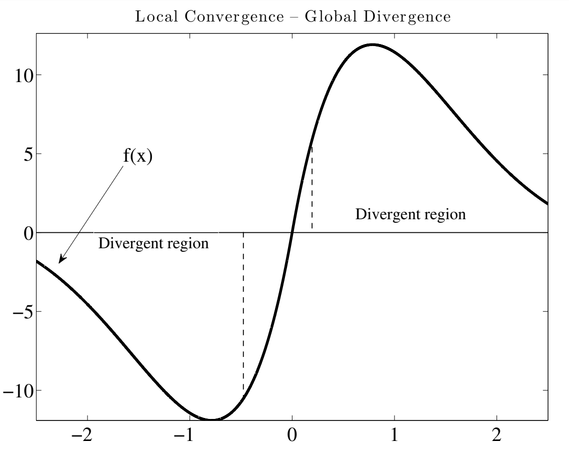
- Pros:
  - Quadratic convergence with each step! (That is, the number of significant digits doubles in every step!)
  - Extends easily to multi-dimensional problems.

- Cons:
  - computation of derivatives can be very slow if done numerically (as opposed to having an analytical formula).
  - Poor global convergence properties.
  - If there is a local minimum near the zero, updating can overshoot to infinity.
  - Use with maximum care!!

- **What to do?** Use a slow but sure method early on (like bisection) but then once you are “close” to the root switch to Newton-Raphson.

Poor Global Convergence

Figure 1: Global Divergence, Local Convergence



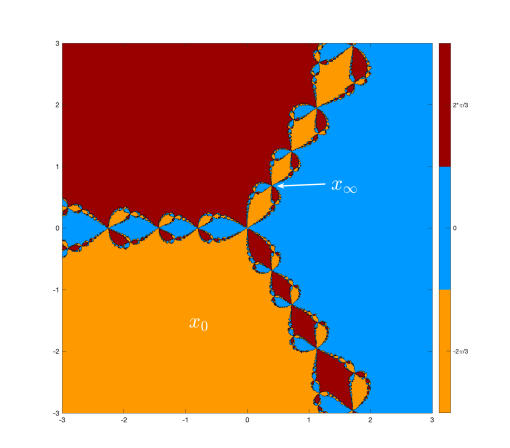
Newton's Fractals

- Consider the equation: $z^3 - 1 = 0$. It has three roots: $z_1 = 1$, $z_2 = -0.5 + i\sqrt{3}/2$, $z_2 = -0.5 - i\sqrt{3}/2$.
- The three points line up on a unit circle in the complex plane, separated by 120 degrees.
- Depending on the starting point, Newton's method can converge to one of three points or can bounce around with chaotic dynamics.
- The whole set feature fractals (near the boundaries).

Newton's Fractals

- Starting Newton's iteration to find the root of $z^3 - 1 = 0$ from an x_0 in a region with a given color, converges to one of the three roots of the same color.

Figure 2: Original set



Poor Global Convergence: Another Example

- Finding the zeroes of $p(x) = (x+3)(x-1)(x-4)$ using Newton's method starting from x_0 :

Table 1: Newton's method's limit

Initial point x_0	$\lim_{n\to\infty} x_n$
2.35287527	4
2.3528**4172**	−3
2.3528**3735**	4
2.35283**6327**	−3
2.35283632**3**	1

NUMERICAL DERIVATIVE

Numerical Differentiation

- **TIP:** Underestimate numerical differentiation at your own risk!

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

Numerical Differentiation

- **TIP:** Underestimate numerical differentiation at your own risk!

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

- Take a small h, say 0.0001 and evaluate (1). Looks pretty straightforward, no? (Hint: No!)

Numerical Differentiation

- **TIP:** Underestimate numerical differentiation at your own risk!

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

- Take a small h, say 0.0001 and evaluate (1). Looks pretty straightforward, no? (Hint: No!)
- Computing the equilibrium of an economy requires satisfying a set of conditions (market clearing, optimality of choices, etc.) that can be expressed as root finding problems.

Numerical Differentiation

- **TIP:** Underestimate numerical differentiation at your own risk!

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \quad (1)$$

- Take a small h, say 0.0001 and evaluate (1). Looks pretty straightforward, no? (Hint: No!)
- Computing the equilibrium of an economy requires satisfying a set of conditions (market clearing, optimality of choices, etc.) that can be expressed as root finding problems.
- Once you get near a zero, you often want to use a Newton based approach $\rightarrow$ requires numerical derivatives.

Numerical Differentiation

- **TIP:** Underestimate numerical differentiation at your own risk!

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

- Take a small h, say 0.0001 and evaluate (1). Looks pretty straightforward, no? (Hint: No!)
- Computing the equilibrium of an economy requires satisfying a set of conditions (market clearing, optimality of choices, etc.) that can be expressed as root finding problems.
- Once you get near a zero, you often want to use a Newton based approach $\rightarrow$ requires numerical derivatives.
- If $f(x)$ is excess demand and x is the price, calculating $f(x+h)$ and $f(x)$ requires solving the entire model twice!

Numerical Differentiation

- **TIP:** Underestimate numerical differentiation at your own risk!

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

- Take a small h, say 0.0001 and evaluate (1). Looks pretty straightforward, no? (Hint: No!)
- Computing the equilibrium of an economy requires satisfying a set of conditions (market clearing, optimality of choices, etc.) that can be expressed as root finding problems.
- Once you get near a zero, you often want to use a Newton based approach $\rightarrow$ requires numerical derivatives.
- If $f(x)$ is excess demand and x is the price, calculating $f(x+h)$ and $f(x)$ requires solving the entire model twice!
- If you have a complicated model this can be very time consuming.

Numerical Differentiation

How to choose h? Two issues:

1. Rounding error: If you use double precision variables, not a major issue.

Numerical Differentiation

How to choose h? Two issues:

1. Rounding error: If you use double precision variables, not a major issue.
2. Truncation error: Crucial!!

Numerical Differentiation

How to choose h? Two issues:

1. Rounding error: If you use double precision variables, not a major issue.
2. Truncation error: Crucial!!
   - If $f(x)$ is excess demand, calculated with truncation error ϵ_f, f' will have error $\sim \sqrt{\epsilon_f}$.
   - Because $\epsilon_f \geq \epsilon_m$ (machine precision: typically ~10^{-15} for DP), the best lower bound is $\sim \sqrt{\epsilon_m}$.
   - Actual error will often be much larger, because $\epsilon_f \gg \epsilon_m$.

Numerical Differentiation

How to choose h? Two issues:

1. Rounding error: If you use double precision variables, not a major issue.
2. Truncation error: Crucial!!
   - If $f(x)$ is excess demand, calculated with truncation error ϵ_f, f' will have error $\sim \sqrt{\epsilon_f}$.
   - Because $\epsilon_f \geq \epsilon_m$ (machine precision: typically ~10^{-15} for DP), the best lower bound is $\sim \sqrt{\epsilon_m}$.
   - Actual error will often be much larger, because $\epsilon_f \gg \epsilon_m$.
   - So, to successfully clear mkts, you need to solve the decision rules precisely. But this is costly (and somewhat pointless) when your x is far away from x^*.
   - Notice that in some cases taking a small h may amplify the truncation error leading to **less** precision in f'.

Numerical Differentiation

How to choose h? Two issues:

1. Rounding error: If you use double precision variables, not a major issue.
2. Truncation error: Crucial!!
   - If $f(x)$ is excess demand, calculated with truncation error ϵ_f, f' will have error $\sim \sqrt{\epsilon_f}$.
   - Because $\epsilon_f \geq \epsilon_m$ (machine precision: typically ~10^{-15} for DP), the best lower bound is $\sim \sqrt{\epsilon_m}$.
   - Actual error will often be much larger, because $\epsilon_f \gg \epsilon_m$.
   - So, to successfully clear mkts, you need to solve the decision rules precisely. But this is costly (and somewhat pointless) when your x is far away from x^*.
   - Notice that in some cases taking a small h may amplify the truncation error leading to **less** precision in f'.
   - If you choose a termination condition for your program that is too small (e.g., $f' < \epsilon_f/2$), it may never converge!

Numerical Differentiation

- Two-Sided Derivatives: A better but slower approach:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

- This also requires two function evaluations. So why is it slower?
  - Because often you need $f(x)$ for other reasons, so you are already going to compute it.
  - So two-sided derivatives require 2 additional evaluations compared to 1 for one-sided.

Numerical Differentiation

- Two-Sided Derivatives: A better but slower approach:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

- This also requires two function evaluations. So why is it slower?
  - Because often you need $f(x)$ for other reasons, so you are already going to compute it.
  - So two-sided derivatives require 2 additional evaluations compared to 1 for one-sided.
- What's the benefit?
  - For one-sided derivatives, truncation error is $\sim h$. For two sided, it is $\sim h^2$!
  - Choose a smaller h than in one-sided case though: assuming you know the error in evaluating $f(x)$, choose $h \sim \epsilon_f^{1/3}$.