Outline

1. BT 3.4: Solving Nonlinear Systems
   - Univariate Problems
   - Multivariate Problems

2. Chapter 4: Numerical Integration: Deterministic and Monte Carlo Methods
   - BT 4.1: Numerical Integration
   - BT 4.2: Monte Carlo Integration
Outline

1. **BT 3.4: Solving Nonlinear Systems**
   - Univariate Problems
   - Multivariate Problems

2. **Chapter 4: Numerical Integration: Deterministic and Monte Carlo Methods**
   - BT 4.1: Numerical Integration
   - BT 4.2: Monte Carlo Integration
Example

A European call option has strike price $54 on a stock with current price $50, expires in five months, and the risk-free rate is 7%. Its current price is $2.85. What is the implied volatility?

Solution. We have a standard formula for this situation that is stored in the function bseurcall. Get help on it and use bseurcall to set up an anonymous function of $\sigma$ using which equals zero when the correct $\sigma$ is used.
Nonlinear Optimization for Univariate Functions

Basic Problem:
Given a function \( f(x) \), find real number \( x^* \) with \( f(x^*) = \min f(x) \) over a range of \( x \) values. How do we find a solution (if it exists)?

- We could solve the equation \( f'(x) = 0 \) using ideas of root finding above. Why does this help?
- Matlab has a built-in command fminbnd that does not use derivative information, but a “bracketing” procedure.
- Use Matlab to minimize \( f(x) = x - 2 \sin(x) \) on interval \([0, 3]\).

```matlab
> help fminbnd
> fminbnd(myfcn,0,3)
> [x,y,exitflag,output]=fminbnd(@(x) x-2*sin(x),0,3)
> x = 0:.01:3;
> plot(x, x-2*sin(x))
```
Basic Problem:

Given a function \( f(x) \), find real number \( x^* \) with \( f(x^*) = \min f(x) \) over a range of \( x \) values. How do we find a solution (if it exists)?

- We could solve the equation \( f'(x) = 0 \) using ideas of root finding above. Why does this help?
- Matlab has a built-in command fminbnd that does not use derivative information, but a “bracketing” procedure.
- Use Matlab to minimize \( f(x) = x - 2\sin(x) \) on interval \([0, 3]\).

```matlab
> help fminbnd
> fminbnd(myfcn,0,3)
> [x,y,exitflag,output]=fminbnd(@(x) x-2*sin(x),0,3)
> x = 0:.01:3;
> plot(x, x-2*sin(x))
```
Basic Problem:

Given a function \( f(x) \), find real number \( x^* \) with \( f(x^*) = \min f(x) \) over a range of \( x \) values. How do we find a solution (if it exists)?

- We could solve the equation \( f'(x) = 0 \) using ideas of root finding above. Why does this help?
- Matlab has a built-in command fminbnd that does not use derivative information, but a “bracketing” procedure.
- Use Matlab to minimize \( f(x) = x - 2 \sin(x) \) on interval \([0, 3]\).

> help fminbnd
> fminbnd(myfcn,0,3)
> [x,y,exitflag,output]=fminbnd(@(x) x-2*sin(x),0,3)
> x = 0:.01:3;
> plot(x, x-2*sin(x))
Basic Problem:

Given a function \( f(x) \), find real number \( x^* \) with \( f(x^*) = \min f(x) \) over a range of \( x \) values. How do we find a solution (if it exists)?

- We could solve the equation \( f'(x) = 0 \) using ideas of root finding above. Why does this help?
- Matlab has a built-in command fminbnd that does not use derivative information, but a “bracketing” procedure.
- Use Matlab to minimize \( f(x) = x - 2 \sin(x) \) on interval \([0, 3]\).

```matlab
> help fminbnd
> fminbnd(myfcn,0,3)
> [x,y,exitflag,output]=fminbnd(@(x) x-2*sin(x),0,3)
x = 0:.01:3;
> plot(x, x-2*sin(x))
```
Outline

1. BT 3.4: Solving Nonlinear Systems
   - Univariate Problems
   - Multivariate Problems

2. Chapter 4: Numerical Integration: Deterministic and Monte Carlo Methods
   - BT 4.1: Numerical Integration
   - BT 4.2: Monte Carlo Integration
Basic Problem:

Given a scalar valued function $f(x_1, x_2, \ldots, x_n) = f(x)$, find vector $x^*$ with $f(x^*) = \min f(x)$ over a range of $x$ values. How do we find a solution (given that there is one)?

- Many techniques exist (all of Chapter 6!)
- Although this is not efficient, *theoretically* one can turn every root finding problem into an optimization problem: to solve the vector equation $f(x) = 0$ for $x$, simply find the $x^*$ that minimizes the scalar function $g(x) = \|f(x)\|^2$. If $g(x^*) = 0$, then $f(x^*) = 0$. So optimization is a more general problem than rootfinding.
- Matlab does provide a multivariate solver called `fminsearch`. Get help on it and use it to solve the example on the next slide. Try different starting points.
**Basic Problem:**

Given a scalar valued function $f(x_1, x_2, \ldots, x_n) = f(x)$, find vector $x^*$ with $f(x^*) = \min f(x)$ over a range of $x$ values. How do we find a solution (given that there is one)?

- Many techniques exist (all of Chapter 6!)
- Although this is not efficient, *theoretically* one can turn every root finding problem into an optimization problem: to solve the vector equation $f(x) = 0$ for $x$, simply find the $x^*$ that minimizes the scalar function $g(x) = \|f(x)\|^2$. If $g(x^*) = 0$, then $f(x^*) = 0$. So optimization is a more general problem than rootfinding.
- Matlab does provide a multivariate solver called fminsearch. Get help on it and use it to solve the example on the next slide. Try different starting points.
Basic Problem:

Given a scalar valued function $f(x_1, x_2, \ldots, x_n) = f(x)$, find vector $x^*$ with $f(x^*) = \min f(x)$ over a range of $x$ values. How do we find a solution (given that there is one)?

- Many techniques exist (all of Chapter 6!)
- Although this is not efficient, *theoretically* one can turn every root finding problem into an optimization problem: to solve the vector equation $f(x) = 0$ for $x$, simply find the $x^*$ that minimizes the scalar function $g(x) = \|f(x)\|^2$. If $g(x^*) = 0$, then $f(x^*) = 0$. So optimization is a more general problem than root finding.

- Matlab does provide a multivariate solver called `fminsearch`. Get help on it and use it to solve the example on the next slide. Try different starting points.
Basic Problem:

Given a scalar valued function $f(x_1, x_2, \ldots, x_n) = f(x)$, find vector $x^*$ with $f(x^*) = \min f(x)$ over a range of $x$ values. How do we find a solution (given that there is one)?

- Many techniques exist (all of Chapter 6!)

- Although this is not efficient, *theoretically* one can turn every root finding problem into an optimization problem: to solve the vector equation $f(x) = 0$ for $x$, simply find the $x^*$ that minimizes the scalar function $g(x) = \|f(x)\|^2$. If $g(x^*) = 0$, then $f(x^*) = 0$. So optimization is a more general problem than rootfinding.

- Matlab does provide a multivariate solver called fminsearch. Get help on it and use it to solve the example on the next slide. Try different starting points.
Example Calculations

Start by writing out what the function $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined below actually represents.

```matlab
> f = @(x) [x(1)^2 - 10*x(1) + x(2)^3 + 8;
> x(1)*x(2)^2 + x(1) - 10*x(2) + 8]
> g = @(x) norm(f(x))^2
> help fminsearch
> fminsearch(g,[0;0])
> f(ans)
> fminsearch(g,[2;3])
> f(ans)
```
# Outline

1. BT 3.4: Solving Nonlinear Systems
   - Univariate Problems
   - Multivariate Problems

2. Chapter 4: Numerical Integration: Deterministic and Monte Carlo Methods
   - BT 4.1: Numerical Integration
   - BT 4.2: Monte Carlo Integration
Basic Problem:

To calculate the definite integral \( I = \int_{a}^{b} f(x) \, dx \) approximately when analytical methods fail us. Divide interval \([a, b]\) into \(N\) equal subintervals by nodes \(x_0, x_1, \ldots, x_N\) and width \(dx = (b - a) / N\):

- **Left Riemann sums:** 
  \[
  I \approx dx \sum_{j=0}^{N-1} f(x_j)
  \]

- **Right Riemann sums:** 
  \[
  I \approx dx \sum_{j=1}^{N} f(x_j).
  \]

- **Trapezoidal:** 
  \[
  I \approx \frac{dx}{2} \left( f(x_0) + \sum_{j=1}^{N-1} f(x_j) + f(x_N) \right)
  \]
Basic Problem:

To calculate the definite integral \( I = \int_{a}^{b} f(x) \, dx \) approximately when analytical methods fail us. Divide interval \([a, b]\) into \(N\) equal subintervals by nodes \(x_0, x_1, \ldots, x_N\) and width \(dx = (b - a) / N\):

- **Left Riemann sums**: \( I \approx dx \sum_{j=0}^{N-1} f(x_j) \)

- **Right Riemann sums**: \( I \approx dx \sum_{j=1}^{N} f(x_j) \). Average left/right:

- **Trapezoidal**: \( I \approx \frac{dx}{2} \left\{ f(x_0) + \sum_{j=1}^{N-1} f(x_j) + f(x_N) \right\} \)
Numerical Integration

Basic Problem:

To calculate the definite integral \( I = \int_{a}^{b} f(x) \, dx \) approximately when analytical methods fail us. Divide interval \([a, b]\) into \( N\) equal subintervals by nodes \( x_0, x_1, \ldots, x_N \) and width \( dx = (b - a) / N\):

- Left Riemann sums: \( I \approx dx \sum_{j=0}^{N-1} f(x_j) \)

- Right Riemann sums: \( I \approx dx \sum_{j=1}^{N} f(x_j) \). Average left/right:

- Trapezoidal: \( I \approx \frac{dx}{2} \left\{ f(x_0) + \sum_{j=1}^{N-1} f(x_j) + f(x_N) \right\} \)
Basic Problem:

To calculate the definite integral \( I = \int_{a}^{b} f(x) \, dx \) approximately when analytical methods fail us. Divide interval \([a, b]\) into \(N\) equal subintervals by nodes \(x_0, x_1, \ldots, x_N\) and width \(dx = (b - a) / N\):

- **Left Riemann sums:** \( I \approx dx \sum_{j=0}^{N-1} f(x_j) \)

- **Right Riemann sums:** \( I \approx dx \sum_{j=1}^{N} f(x_j) \). Average left/right:

- **Trapezoidal:** \( I \approx \frac{dx}{2} \left\{ f(x_0) + \sum_{j=1}^{N-1} f(x_j) + f(x_N) \right\} \)
Basic Problem:

To calculate the definite integral \( I = \int_a^b w(x)f(x) \, dx \) approximately when analytical methods fail us. Here \( w(x) \) is a nonnegative “weight” function and either \( a \) or \( b \) could be infinite.

- Motivating formula: \( I \approx \sum_{j=1}^{N} w_j f(x_j) \), where \( x_1, \ldots, x_N \) are certain nodes on a fixed reference interval and \( w_1, \ldots, w_N \) are “weights”, both of which are computed for once and for all.
- Any other integral can be mapped to the reference interval by a simple change of variables.
- A classical example (Gaussian quadrature): \( I = \int_{-1}^{1} f(x) \, dx \)
- Another classic (Gauss-Hermite quadrature, text, p. 216): \( I = \int_{-\infty}^{\infty} e^{-x^2} f(x) \, dx \).
Another Numerical Integration

Basic Problem:

To calculate the definite integral \( I = \int_{a}^{b} w(x)f(x) \, dx \) approximately when analytical methods fail us. Here \( w(x) \) is a nonnegative “weight” function and either \( a \) or \( b \) could be infinite.

- Motivating formula: \( I \approx \sum_{j=1}^{N} w_j f(x_j) \), where \( x_1, \ldots, x_N \) are certain nodes on a fixed reference interval and \( w_1, \ldots, w_N \) are “weights”, both of which are computed for once and for all.

- Any other integral can be mapped to the reference interval by a simple change of variables.

- A classical example (Gaussian quadrature): \( I = \int_{-1}^{1} 1 \, f(x) \, dx \)

- Another classic (Gauss-Hermite quadrature, text, p. 216): \( I = \int_{-\infty}^{\infty} e^{-x^2} f(x) \, dx \).
Another Numerical Integration

Basic Problem:

To calculate the definite integral \( I = \int_a^b w(x)f(x) \, dx \)
approximately when analytical methods fail us. Here \( w(x) \) is a nonnegative “weight” function and either \( a \) or \( b \) could be infinite.

- Motivating formula: \( I \approx \sum_{j=1}^{N} w_j f(x_j) \), where \( x_1, \ldots, x_N \) are certain nodes on a fixed reference interval and \( w_1, \ldots, w_N \) are “weights”, both of which are computed for once and for all.
- Any other integral can be mapped to the reference interval by a simple change of variables.
- A classical example (Gaussian quadrature): \( I = \int_{-1}^{1} 1 f(x) \, dx \)
- Another classic (Gauss-Hermite quadrature, text, p. 216):
  \[ I = \int_{-\infty}^{\infty} e^{-x^2} f(x) \, dx \]
Another Numerical Integration

Basic Problem:

To calculate the definite integral \( I = \int_{a}^{b} w(x)f(x) \, dx \) approximately when analytical methods fail us. Here \( w(x) \) is a nonnegative “weight” function and either \( a \) or \( b \) could be infinite.

- Motivating formula: \( I \approx \sum_{j=1}^{N} w_j f(x_j) \), where \( x_1, \ldots, x_N \) are certain nodes on a fixed reference interval and \( w_1, \ldots, w_N \) are “weights”, both of which are computed for once and for all.
- Any other integral can be mapped to the reference interval by a simple change of variables.
- A classical example (Gaussian quadrature): \( I = \int_{-1}^{1} f(x) \, dx \)
- Another classic (Gauss-Hermite quadrature, text, p. 216): \( I = \int_{-\infty}^{\infty} e^{-x^2} f(x) \, dx \).
Another Numerical Integration

Basic Problem:

To calculate the definite integral $I = \int_a^b \omega(x)f(x) \, dx$ approximately when analytical methods fail us. Here $\omega(x)$ is a nonnegative “weight” function and either $a$ or $b$ could be infinite.

- Motivating formula: $I \approx \sum_{j=1}^N \omega_j f(x_j)$, where $x_1, \ldots, x_N$ are certain nodes on a fixed reference interval and $\omega_1, \ldots, \omega_N$ are “weights”, both of which are computed for once and for all.

- Any other integral can be mapped to the reference interval by a simple change of variables.

- A classical example (Gaussian quadrature): $I = \int_{-1}^1 1 \, f(x) \, dx$

- Another classic (Gauss–Hermite quadrature, text, p. 216): $I = \int_{-\infty}^{\infty} e^{-x^2} \, f(x) \, dx$. 
Matlab uses an adaptive Simpson rule, which involves estimating the function as a quadratic over two subintervals, and using error estimates to determine if the current approximation is good enough. If not, subintervals are further subdivided.

```matlab
> f = @(x) chis_pdf(x,8)
> format long
> N = 40
> dx = (4-0)/N
> x = linspace(0,4,N+1);
> y = f(x);
> Itrue = chis_cdf(4,8)
> IMatlab = quad(f,0,4)
> Irl = dx*sum(f(x(1:N)))
> Irr = dx*sum(f(x(2:N+1)))
> Itrap = 0.5*(Irl+Irr)
> edit GaussInt % don’t change, just look under the hood
> IGquad = GaussInt(f,[0,4],3) % try more nodes, up to 8
```
Outline

1. BT 3.4: Solving Nonlinear Systems
   - Univariate Problems
   - Multivariate Problems

2. Chapter 4: Numerical Integration: Deterministic and Monte Carlo Methods
   - BT 4.1: Numerical Integration
   - BT 4.2: Monte Carlo Integration
The Basic Idea

Monte Carlo Simulation:
- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:
We want to approximate $\int_a^b g(x) \, dx$. For convenience, assume $g(x) \geq 0$, so that this integral represents (positive) area. Let’s use $\int_0^1 e^x \, dx = e - 1 \approx 1.7183$ as a test case.
The Basic Idea

Monte Carlo Simulation:

- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:

We want to approximate \( \int_a^b g(x) \, dx \). For convenience, assume \( g(x) \geq 0 \), so that this integral represents (positive) area. Let’s use \( \int_0^1 e^x \, dx = e - 1 \approx 1.7183 \) as a test case.
The Basic Idea

Monte Carlo Simulation:

- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:

We want to approximate $\int_a^b g(x) \, dx$. For convenience, assume $g(x) \geq 0$, so that this integral represents (positive) area. Let’s use $\int_0^1 e^x \, dx = e - 1 \approx 1.7183$ as a test case.
The Basic Idea

Monte Carlo Simulation:
- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:
We want to approximate \( \int_a^b g(x) \, dx \). For convenience, assume \( g(x) \geq 0 \), so that this integral represents (positive) area. Let’s use \( \int_0^1 e^x \, dx = e - 1 \approx 1.7183 \) as a test case.
The Basic Idea

Monte Carlo Simulation:

- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:

We want to approximate \( \int_{a}^{b} g(x) \, dx \). For convenience, assume \( g(x) \geq 0 \), so that this integral represents (positive) area. Let’s use \( \int_{0}^{1} e^x \, dx = e - 1 \approx 1.7183 \) as a test case.
The Basic Idea

Monte Carlo Simulation:
- Create a quantitative model of a process.
- Treat the events that constitute the process as random.
- Generate random variables to simulate the events.
- Use these values to compute the outcome of the process.

A Guiding Example is Monte Carlo Integration:
We want to approximate $\int_{a}^{b} g(x) \, dx$. For convenience, assume $g(x) \geq 0$, so that this integral represents (positive) area. Let’s use $\int_{0}^{1} e^x \, dx = e - 1 \approx 1.7183$ as a test case.
Monte Carlo Integration

Hit or Miss Monte Carlo Method:

- Enclose the graph in a box of known area $A$ (in our test case, $0 \leq x \leq 1, 0 \leq y \leq 3$, so $A = 3$.)
- Throw $N$ random darts at the area, uniformly distributed in $x$ and $y$ directions. Note: the event of a dart throw is represented by a random pair $(X_i, Y_i)$ of independent r.v.’s.
- Count up the number $N_H$ of darts that fall in the area, i.e., for which $Y_i \leq g(X_i)$.
- Proportionately, $\frac{\int_a^b g(x) \, dx}{A} \approx \frac{N_H}{N}$, so we have $\int_a^b g(x) \, dx \approx \frac{N_H}{N} A$. 

Monte Carlo Integration

**Hit or Miss Monte Carlo Method:**

- Enclose the graph in a box of known area $A$ (in our test case, $0 \leq x \leq 1$, $0 \leq y \leq 3$, so $A = 3$.)
- Throw $N$ random darts at the area, uniformly distributed in $x$ and $y$ directions. Note: the event of a dart throw is represented by a random pair $(X_i, Y_i)$ of independent r.v.’s.
- Count up the number $N_H$ of darts that fall in the area, i.e., for which $Y_i \leq g(X_i)$.
- Proportionately, \[ \frac{\int_a^b g(x) \, dx}{A} \approx \frac{N_H}{N}, \] so we have \[ \int_a^b g(x) \, dx \approx \frac{N_H}{N} A. \]
Monte Carlo Integration

Hit or Miss Monte Carlo Method:

- Enclose the graph in a box of known area $A$ (in our test case, $0 \leq x \leq 1$, $0 \leq y \leq 3$, so $A = 3$.)
- Throw $N$ random darts at the area, uniformly distributed in $x$ and $y$ directions. Note: the event of a dart throw is represented by a random pair $(X_i, Y_i)$ of independent r.v.’s.
- Count up the number $N_H$ of darts that fall in the area, i.e., for which $Y_i \leq g(X_i)$.
- Proportionately,
  \[
  \frac{\int_a^b g(x) \, dx}{A} \approx \frac{N_H}{N},
  \]
  so we have
  \[
  \int_a^b g(x) \, dx \approx \frac{N_H}{N} A.
  \]
Monte Carlo Integration

**Hit or Miss Monte Carlo Method:**
- Enclose the graph in a box of known area $A$ (in our test case, $0 \leq x \leq 1$, $0 \leq y \leq 3$, so $A = 3$.)
- Throw $N$ random darts at the area, uniformly distributed in $x$ and $y$ directions. Note: the event of a dart throw is represented by a random pair $(X_i, Y_i)$ of independent r.v.’s.
- Count up the number $N_H$ of darts that fall in the area, i.e., for which $Y_i \leq g(X_i)$.
- Proportionately, $\frac{\int_a^b g(x) \, dx}{A} \approx \frac{N_H}{N}$, so we have $\int_a^b g(x) \, dx \approx \frac{N_H}{N} A$. 
Monte Carlo Integration

Hit or Miss Monte Carlo Method:

- Enclose the graph in a box of known area \( A \) (in our test case, \( 0 \leq x \leq 1, 0 \leq y \leq 3 \), so \( A = 3 \)).
- Throw \( N \) random darts at the area, uniformly distributed in \( x \) and \( y \) directions. Note: the event of a dart throw is represented by a random pair \((X_i, Y_i)\) of independent r.v.'s.
- Count up the number \( N_H \) of darts that fall in the area, i.e., for which \( Y_i \leq g(X_i) \).
- Proportionately, \( \frac{\int_a^b g(x) \, dx}{A} \approx \frac{N_H}{N} \), so we have
  \[
  \int_a^b g(x) \, dx \approx \frac{N_H}{N} A.
  \]
Example Calculation

Carry out the following steps in Matlab

```matlab
> help rand
> format
> rand('seed',0)
> A = 3
> N = 10
> X = rand(N,1);
> Y = (3-0)*rand(N,1);
> hits = sum(Y <= exp(X))
> area = A*(hits/N)
> Itrue = exp(1)-1 % now try to improve accuracy
```