Introduction to MATLAB, Linear Algebra and Numerical Analysis for Materials Science

Ralph C. Smith
Department of Mathematics
North Carolina State University

Overview: 9:00-Noon

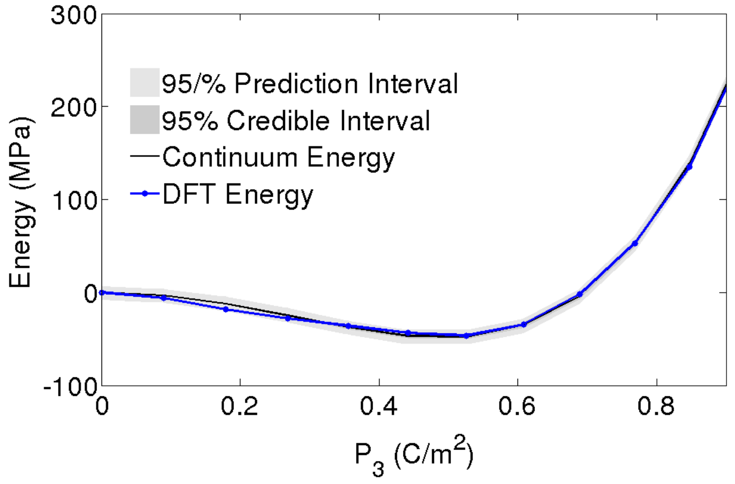
1. Motivating examples from Materials Science and Nuclear Engineering
2. MATLAB examples throughout tutorial
3. Elements of linear algebra
   - Fundamental properties of vectors and matrices
   - Eigenvalues, eigenvectors and singular values
4. Elements of numerical analysis
   - Numerical integration
   - Optimization
   - Numerical differentiation and differential equations

Example 1: Quantum-Informed Continuum Models

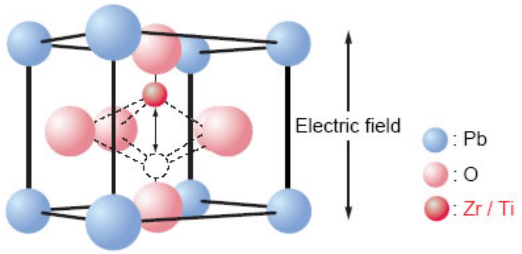
Objectives:

- Employ density function theory (DFT) to construct/calibrate continuum energy relations.
  - e.g., Landau energy

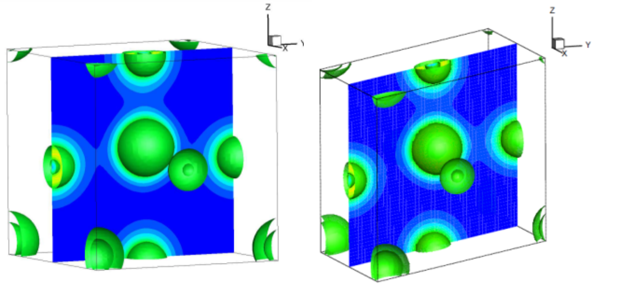
$$\psi(P) = \underline{\alpha_1} P^2 + \underline{\alpha_{11}} P^4 + \underline{\alpha_{111}} P^6$$



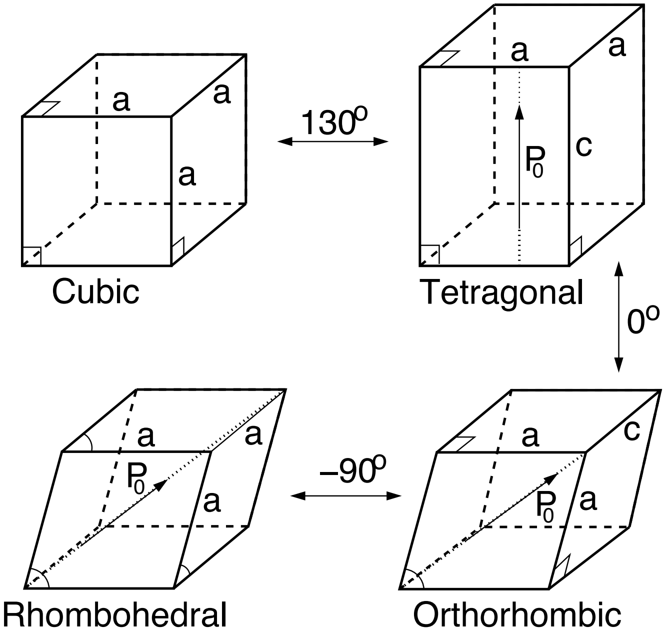
Landau energy



Lead Titanate Zirconate (PZT)



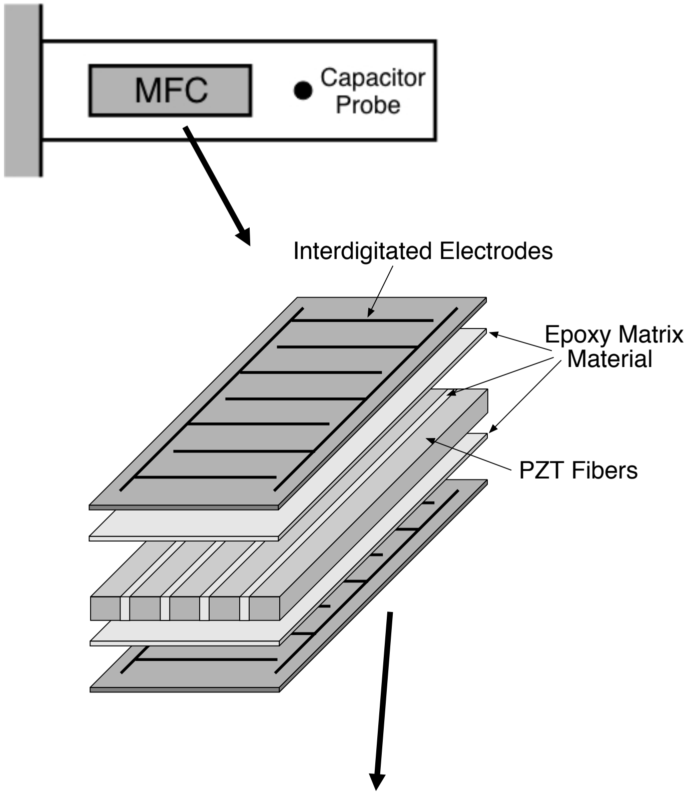
DFT Electronic Structure Simulation



UQ and SA Issues:

- Is 6th order term required to accurately characterize material behavior?
- **Note:** Determines molecular structure

Example 2: PZT-Based MFC and Robobee

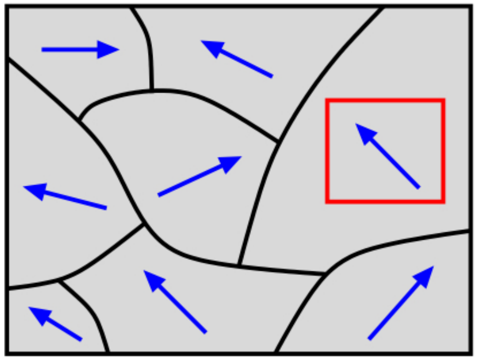


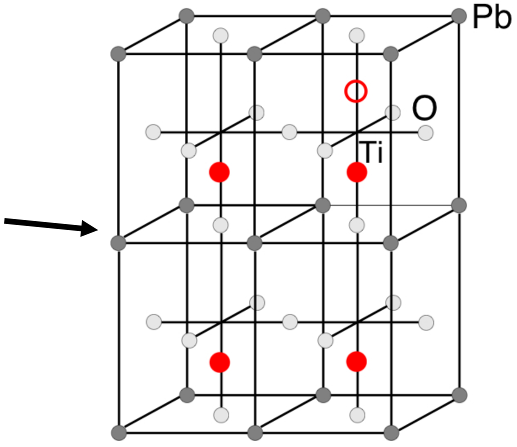
Beam Model: 20 parameters

$$\underline{\rho}\frac{\partial^2 w}{\partial t^2} + \underline{\gamma}\frac{\partial w}{\partial t} - \frac{\partial^2 M}{\partial x^2} = 0$$

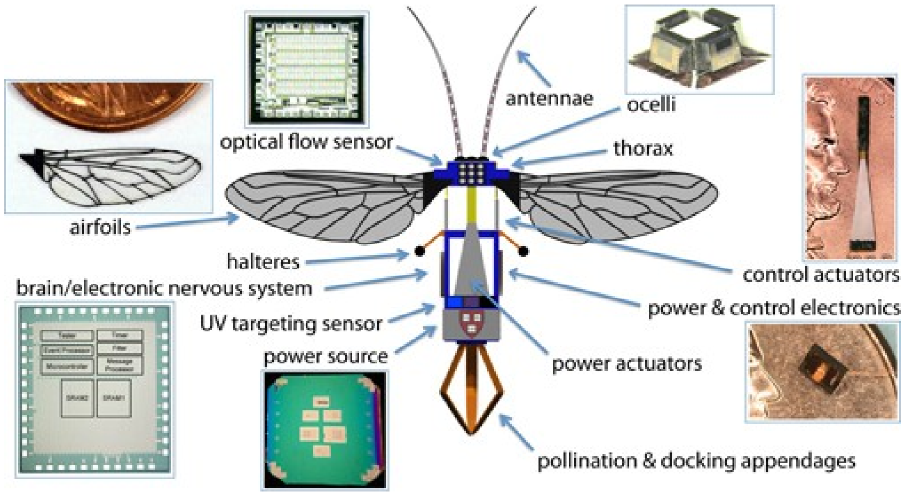
$$M = -\underline{c^E I}\frac{\partial^2 w}{\partial x^2} - \underline{c_D I}\frac{\partial^3 w}{\partial x^2 \partial t} - \left[\underline{k_1} e(E,\sigma_0)E + \underline{k_2}\varepsilon_{irr}(E,\sigma_0)\right]\chi_{MFC}(x)$$

Homogenized Energy Model (HEM)

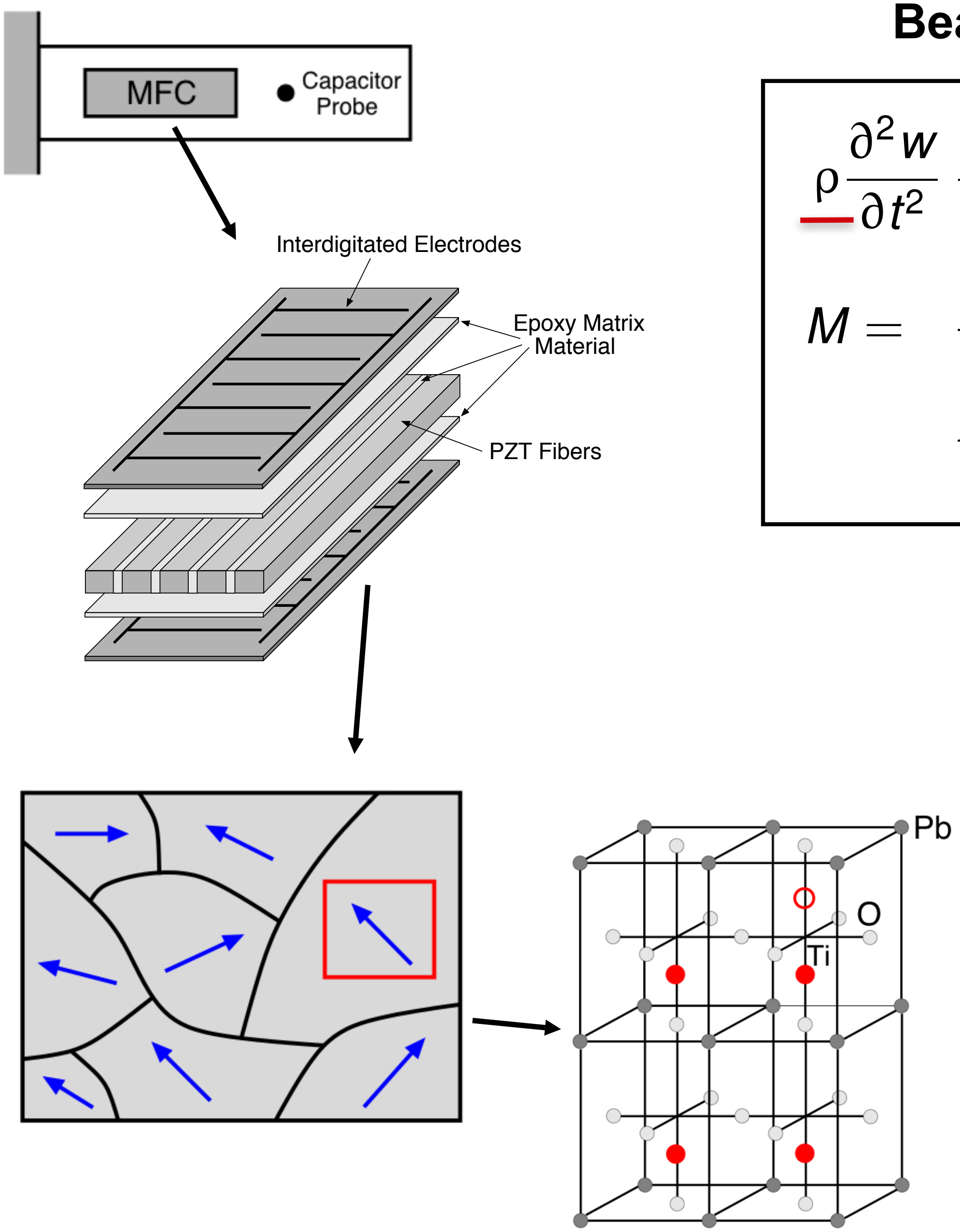




2nd Example: Robobee Drive Mechanism



Example: PZT-Based MFC and Robobee



Beam Model: 20 parameters

$$\boxed{\begin{aligned} &\underline{\rho}\frac{\partial^2 w}{\partial t^2} + \underline{\gamma}\frac{\partial w}{\partial t} - \frac{\partial^2 M}{\partial x^2} = 0 \\ &M = -\underline{c^E I}\frac{\partial^2 w}{\partial x^2} - \underline{c_D I}\frac{\partial^3 w}{\partial x^2 \partial t} \\ &\qquad - [\underline{k_1} e(E,\sigma_0)E + \underline{k_2}\varepsilon_{irr}(E,\sigma_0)]\chi_{MFC}(x) \end{aligned}}$$

UQ and Control Formulation:

$$\frac{dz}{dt} = f(t,z,q) + v_1(t)$$

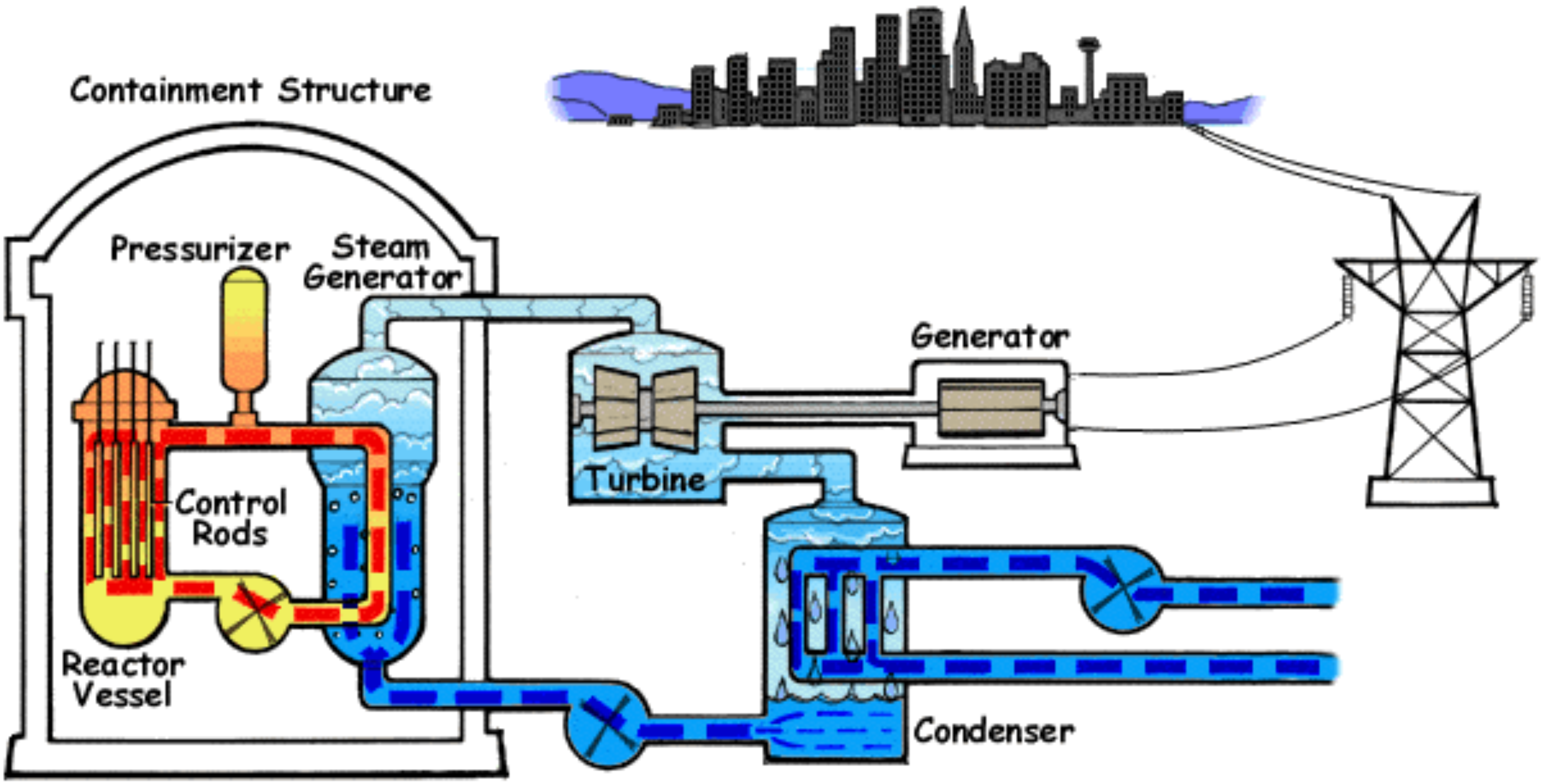
$$y(t) = \int_{\mathbb{R}^{20}} w^N(t,\bar{x},q)\rho(q)dq$$

E.g., Average tip displacement

Numerical Issues:

- Accurately approximate derivatives and high-dimensional integrals.
- Parameter selection: Linear algebra

Example 3: Pressurized Water Reactors (PWR)



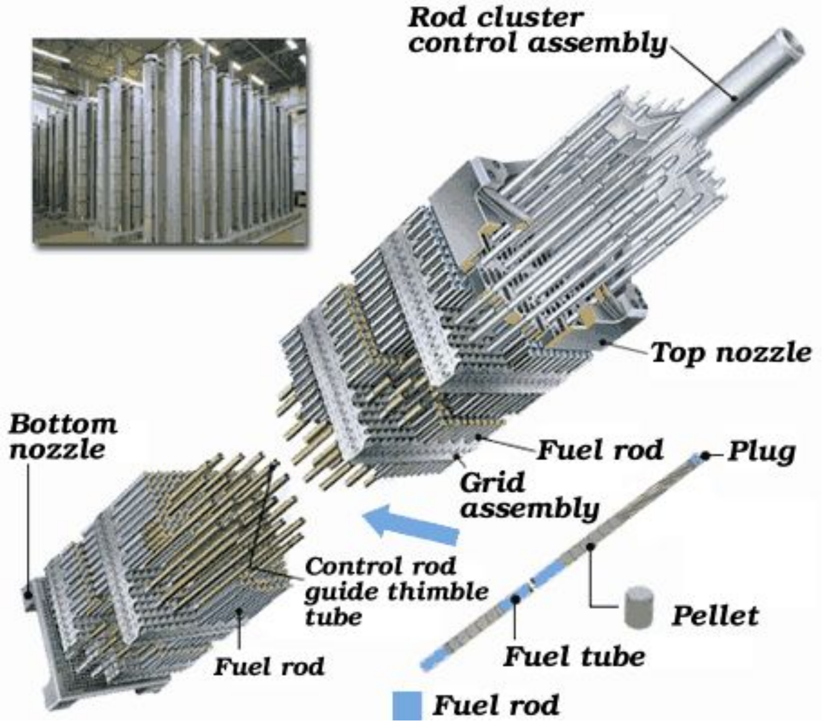
Models:

- Involve neutron transport, thermal-hydraulics, chemistry.
- Inherently multi-scale, multi-physics.

Example: Pressurized Water Reactors (PWR)

3-D Neutron Transport Equations:

$$\begin{aligned}
&\frac{1}{|v|}\frac{\partial\varphi}{\partial t} + \Omega\cdot\nabla\varphi + \Sigma_t(r,E)\varphi(r,E,\Omega,t)\\
&\quad = \int_{4\pi} d\Omega' \int_0^\infty dE'\Sigma_s(E'\to E,\Omega'\to\Omega)\varphi(r,E',\Omega',t)\\
&\quad + \frac{\chi(E)}{\underline{4\pi}}\int_{4\pi} d\Omega' \int_0^\infty dE'\underline{\nu(E')}\,\underline{\Sigma_f(E')}\varphi(r,E',\Omega',t)
\end{aligned}$$



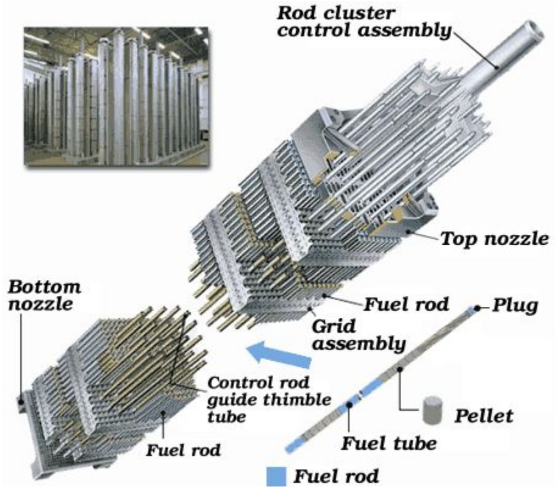
Challenges and Numerical Issues:

- Very large number of inputs; e.g., 100,000; Active subspace construction critical: Requires highly efficient numerical linear algebra.
- Must accurately approximate derivatives and high-dimensional integrals.

Example: Pressurized Water Reactors (PWR)

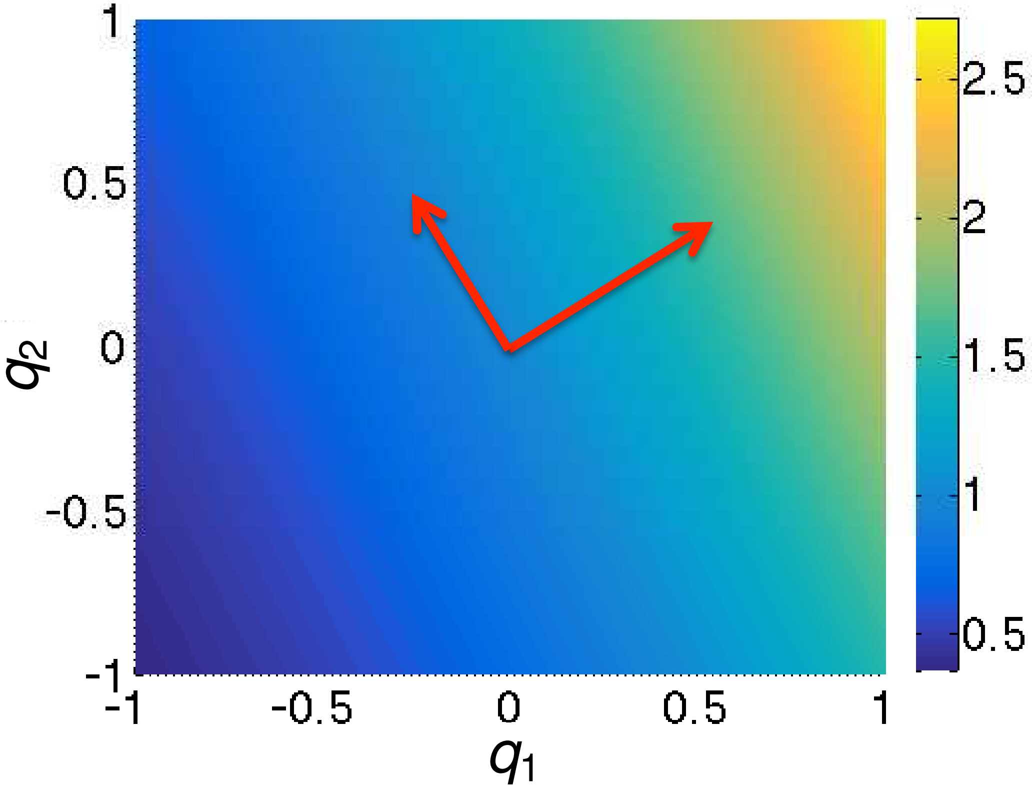
Challenges and Numerical Issues:

- Very large number of inputs; e.g., 100,000; Active subspace construction critical: Requires highly efficient numerical linear algebra.



Example: $y = \exp(0.7q_1 + 0.3q_2)$

- Varies most in [0.7, 0.3] direction
- No variation in orthogonal direction



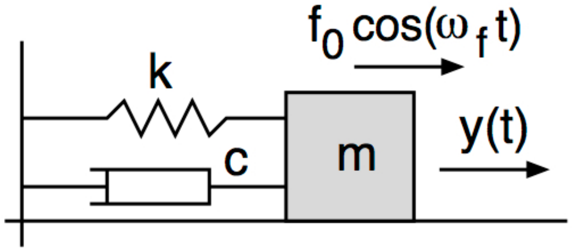
Strategy:

- *Linearly parameterized problems:* Employ SVD or QR decomposition.
- *Nonlinear problems:* Construct approximate gradient matrix and employ SVD or QR.

Parameter Selection Techniques

First Issue: Parameters often *not identifiable* in the sense that they are uniquely determined by the data.

Example: Spring model



$$\underline{m}\frac{d^2z}{dt^2} + \underline{c}\frac{dz}{dt} + \underline{k}z = \underline{f_0}\cos(\omega_F t)$$

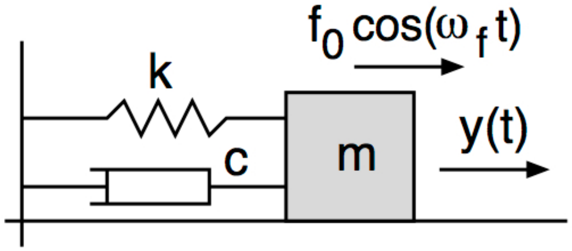
$$z(0) = z_0 \ , \ \frac{dz}{dt}(0) = z_1$$

Problem: Parameters $q = [m, c, k, f_0]$ and $q = \left[1, \frac{c}{m}, \frac{k}{m}, \frac{f_0}{m}\right]$ yield same displacements

Parameter Selection Techniques

First Issue: Parameters often not *identifiable* in the sense that they are uniquely determined by the data.

Example: Spring model



$$\underline{m}\frac{d^2z}{dt^2} + \underline{c}\frac{dz}{dt} + \underline{k}z = \underline{f_0}\cos(\omega_F t)$$

$$z(0) = z_0 \,,\; \frac{dz}{dt}(0) = z_1$$

Problem: Parameters $q = [m, c, k, f_0]$ and $q = \left[1, \frac{c}{m}, \frac{k}{m}, \frac{f_0}{m}\right]$ yield same displacements

Solution: Reformulate problem as

$$\frac{d^2z}{dt^2} + \underline{C}\frac{dz}{dt} + \underline{K}z = \underline{F_0}\cos(\omega_F t)$$

$$z(0) = z_0 \,,\; \frac{dz}{dt}(0) = z_1$$

where $C = \frac{c}{m}$, $K = \frac{k}{m}$ and $F_0 = \frac{f_0}{m}$

Techniques for General Models:

- Linear algebra analysis;
  - e.g., SVD or QR algorithms
- Active subspaces
- Sensitivity analysis
- Parameter subset selection

Sensitivity Analysis

Example: Linear elastic constitute relation

$$\sigma = Ee + c\frac{de}{dt}$$

Nominal Values: $E = 100,\ c = 0.1,\ e = 0.001,\ \frac{de}{dt} = 0.1$

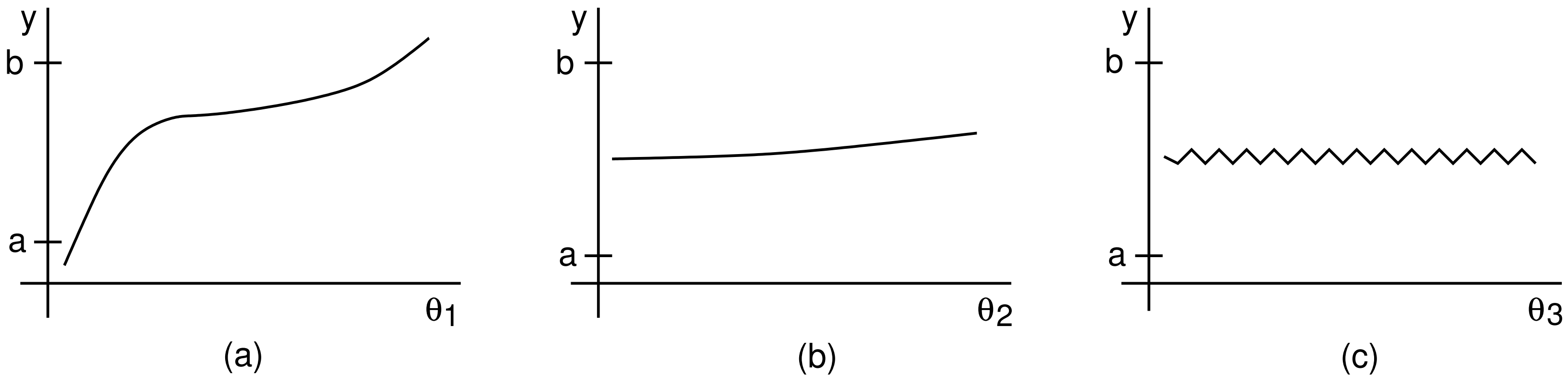
Question: To which parameter E or c is stress most sensitive?

Local Sensitivity Analysis:

$$\frac{\partial\sigma}{\partial E} = e = 0.001 \qquad \boxed{\frac{\partial\sigma}{\partial c} = \frac{de}{dt} = 0.1}$$

Conclusion: Model most sensitive to damping parameter c

Requires Derivatives: Must accommodate noise



Deterministic Model Calibration

Example: Helmholtz energy $\psi(P,q) = \underline{\alpha_1} P^2 + \underline{\alpha_{11}} P^4 + \underline{\alpha_{111}} P^6$

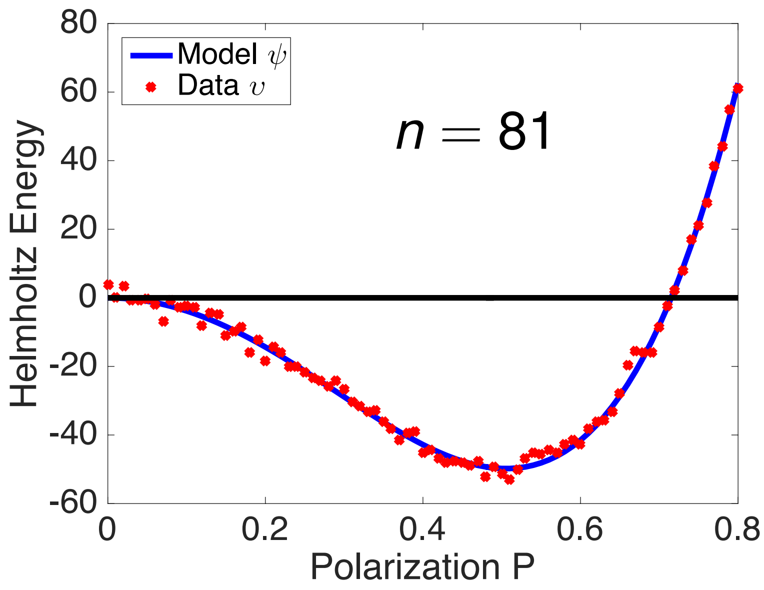
$$q = [\alpha_1, \alpha_{11}, \alpha_{111}]$$

Statistical Model: Describes observation process

$$\upsilon_i = \psi(P_i, q) + \varepsilon_i \quad , \; i = 1, \ldots, n$$

Point Estimates: Ordinary least squares

$$q^0 = \arg\min_q \frac{1}{2} \sum_{j=1}^{n} [\upsilon_i - \psi(P_i, q)]^2$$



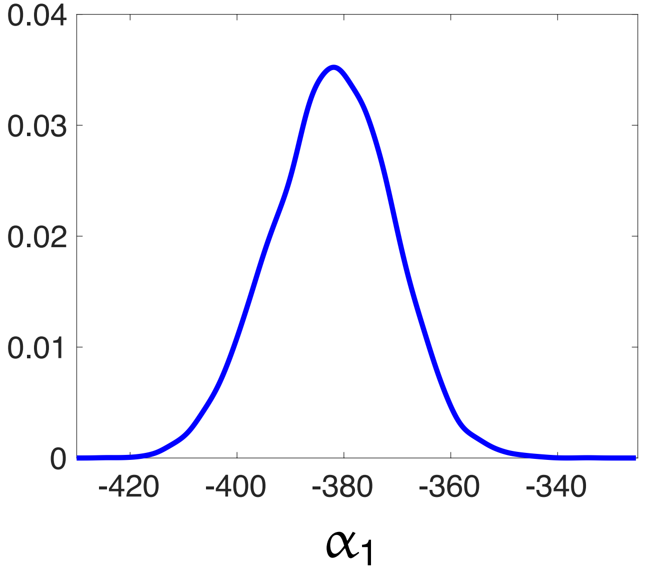
Note: Provides point estimates but no quantification of uncertainty in:

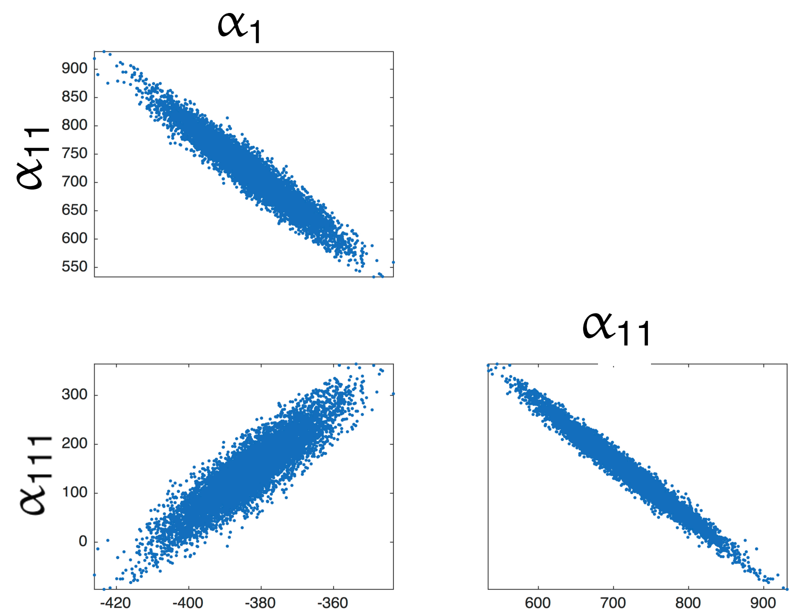
- Model
- Parameters
- Data

Objectives for Uncertainty Quantification

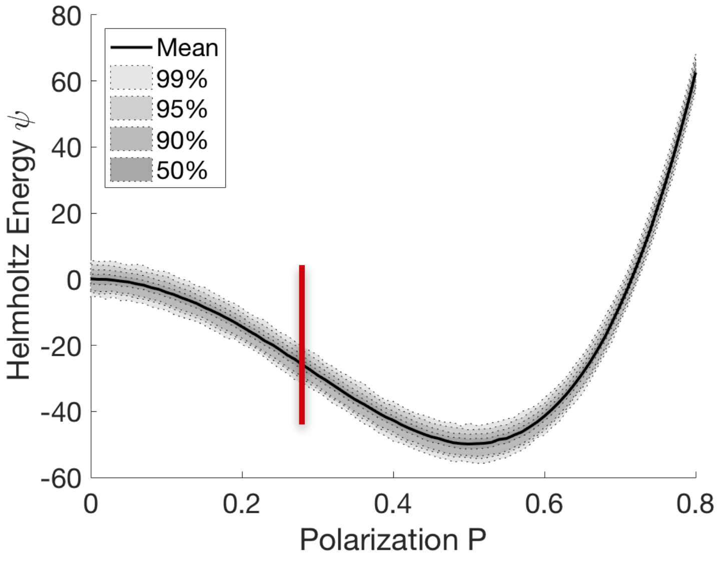
Goal: Replace point estimates with distributions or credible intervals

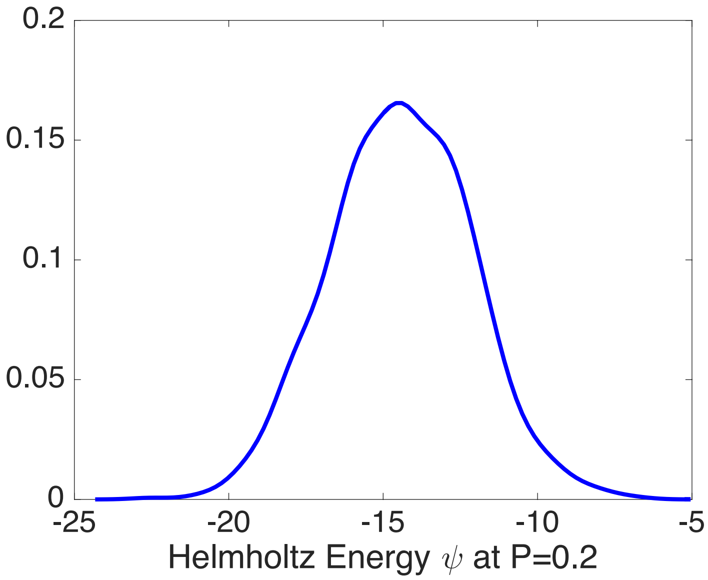
E.g., Parameter Densities





E.g., Response Intervals





Objectives for Uncertainty Quantification

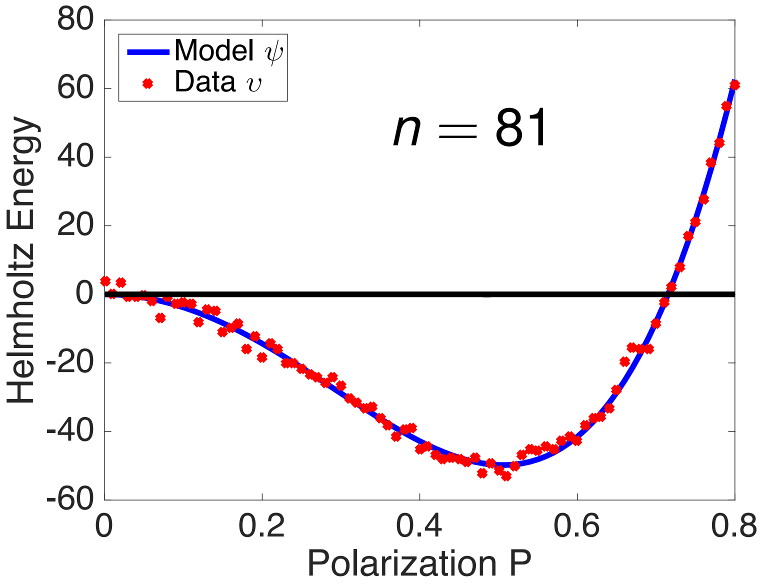
Example: Helmholtz energy $\psi(P,q) = \alpha_1 P^2 + \alpha_{11} P^4$, $q = [\alpha_1, \alpha_{11}]$

Statistical Model: Describes observation process

$$\upsilon_i = \psi(P_i, q) + \varepsilon_i \quad , \; i = 1, \ldots, n$$

Common Assumption: $\varepsilon_i \sim N(0, \sigma^2)$

UQ Goals: Quantify parameter and response uncertainties

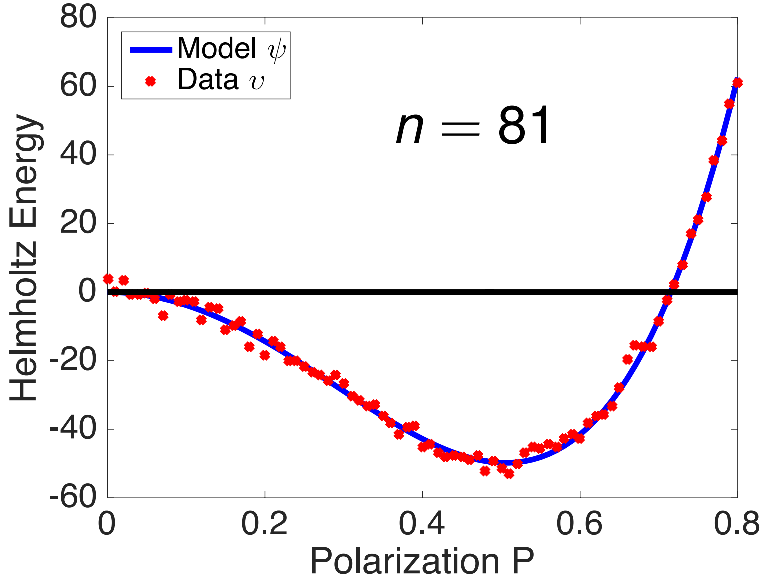


Strategy 1: Perform Experiments

Example: Helmholtz energy $\psi(P,q) = \alpha_1 P^2 + \alpha_{11} P^4$, $q = [\alpha_1, \alpha_{11}]$

Statistical Model: Describes observation process

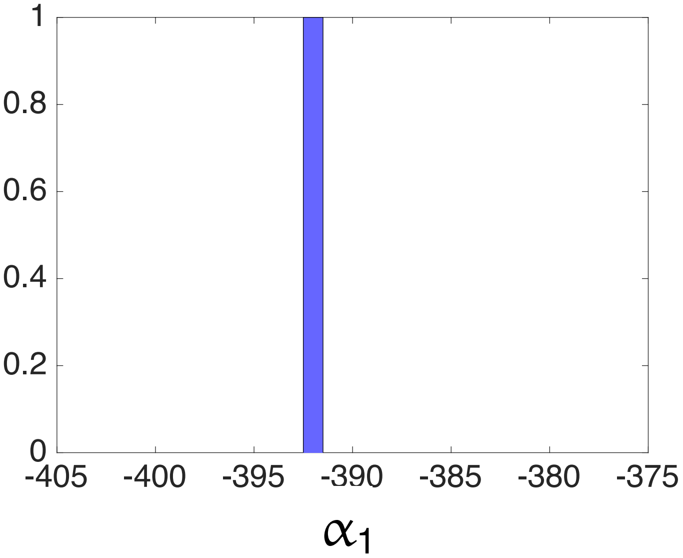
$$\upsilon_i = \psi(P_i, q) + \varepsilon_i \quad , \; i = 1, \dots, n$$

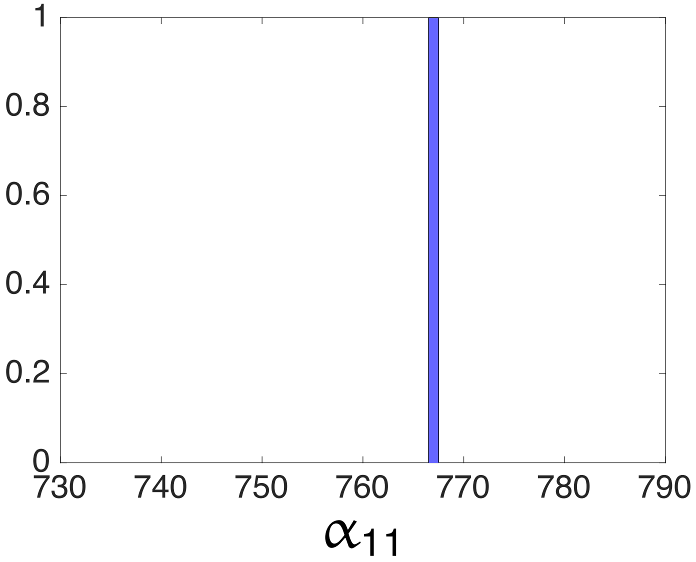


Common Assumption: $\varepsilon_i \sim N(0, \sigma^2)$

UQ Goals: Quantify parameter and response uncertainties

Strategy 1: Perform experiments; e.g., 1





Strategy 1: Perform Experiments

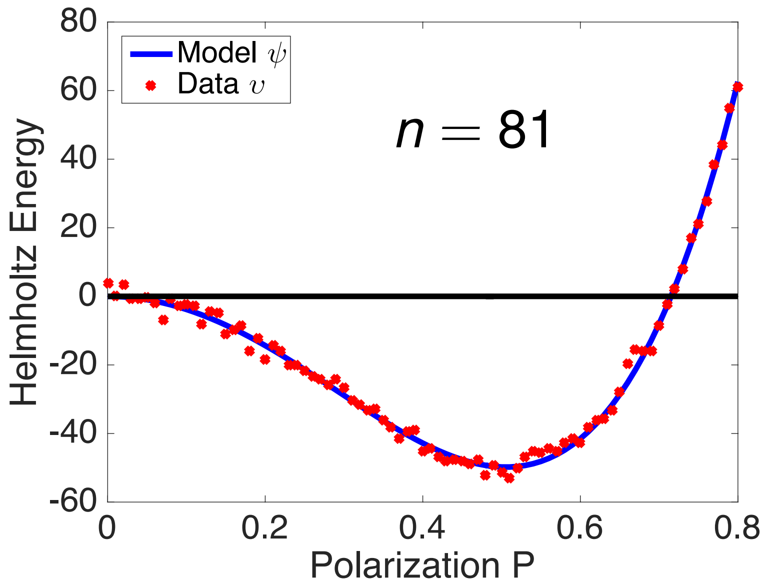
Example: Helmholtz energy $\psi(P,q) = \alpha_1 P^2 + \alpha_{11} P^4$, $q = [\alpha_1, \alpha_{11}]$

Statistical Model: Describes observation process

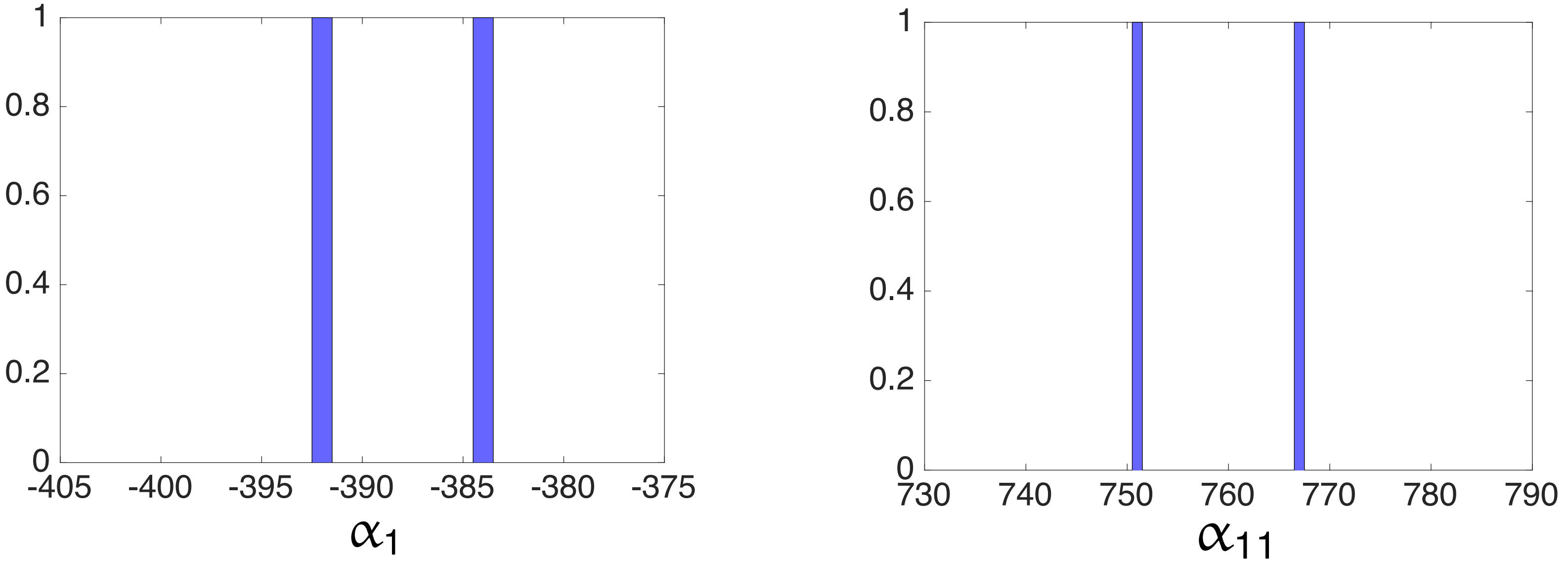
$$\upsilon_i = \psi(P_i, q) + \varepsilon_i \quad , \ i = 1, \dots, n$$

Common Assumption: $\varepsilon_i \sim N(0, \sigma^2)$

UQ Goals: Quantify parameter and response uncertainties



Strategy 1: Perform experiments; e.g., 2

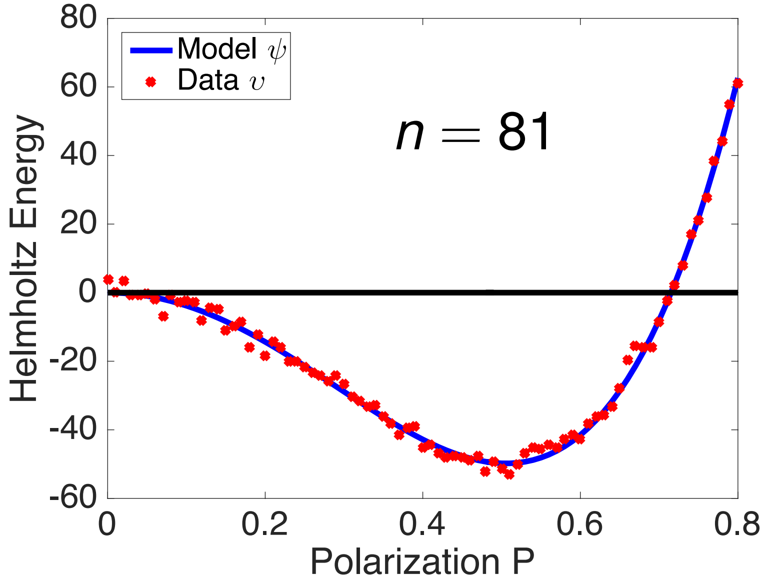


Strategy 1: Perform Experiments

Example: Helmholtz energy $\psi(P,q) = \alpha_1 P^2 + \alpha_{11} P^4$, $q = [\alpha_1, \alpha_{11}]$

Statistical Model: Describes observation process

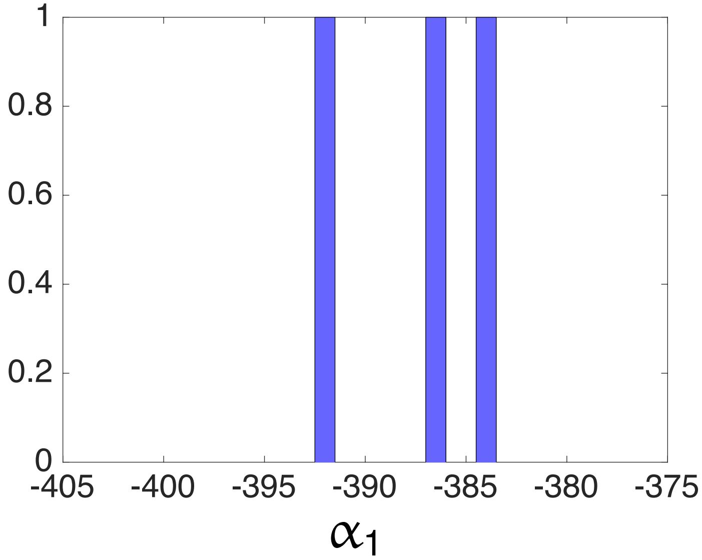
$$\upsilon_i = \psi(P_i, q) + \varepsilon_i \quad , \; i = 1, \ldots, n$$

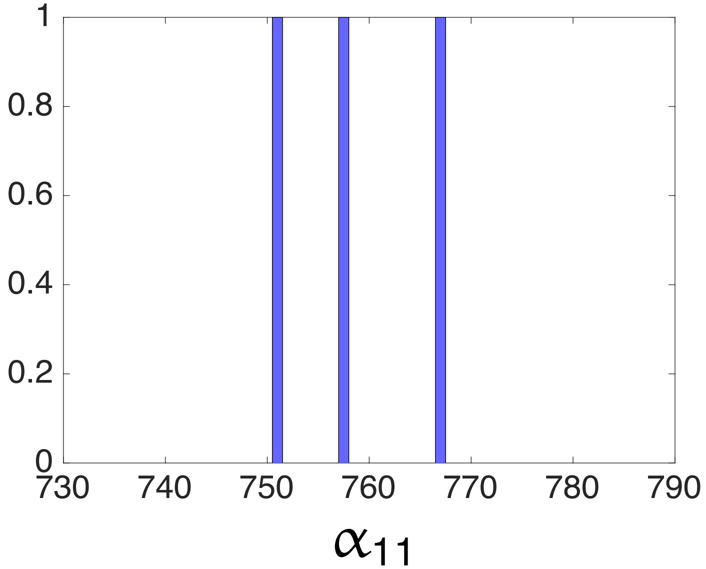


Common Assumption: $\varepsilon_i \sim N(0, \sigma^2)$

UQ Goals: Quantify parameter and response uncertainties

Strategy 1: Perform experiments; e.g., 3





Strategy 1: Perform Experiments

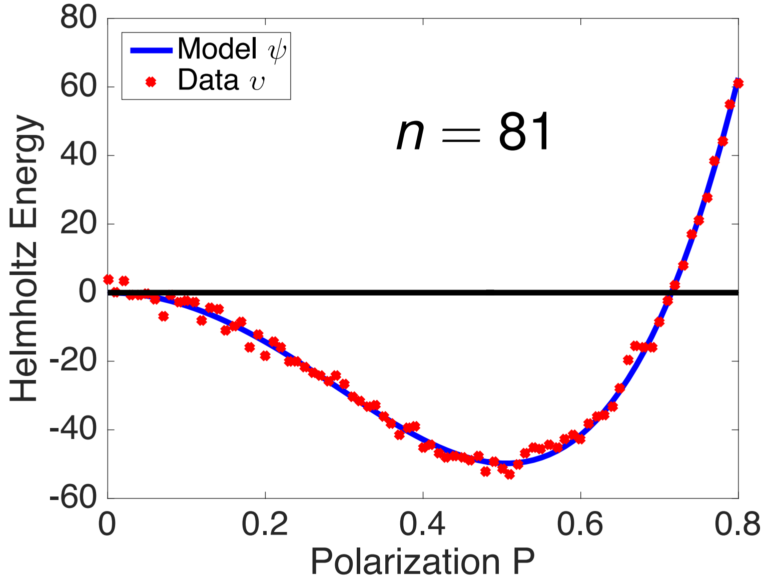
Example: Helmholtz energy $\psi(P, q) = \alpha_1 P^2 + \alpha_{11} P^4$, $q = [\alpha_1, \alpha_{11}]$

Statistical Model: Describes observation process

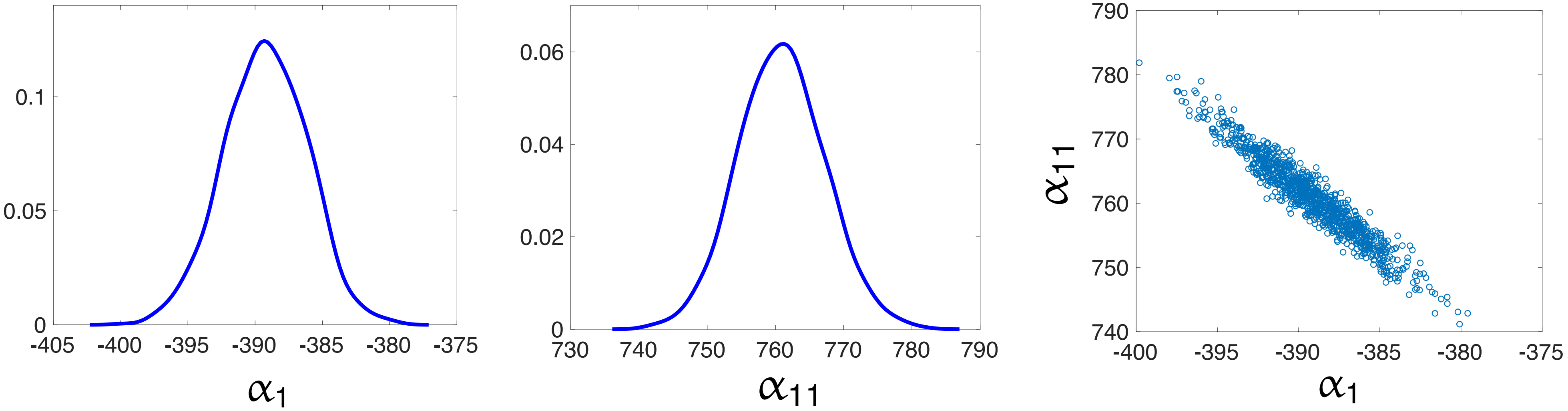
$$\upsilon_i = \psi(P_i, q) + \varepsilon_i \quad , \; i = 1, \dots, n$$

Common Assumption: $\varepsilon_i \sim N(0, \sigma^2)$

UQ Goals: Quantify parameter and response uncertainties



Strategy 1: Perform many experiments; e.g., 1000



Strategy 1: Perform Experiments

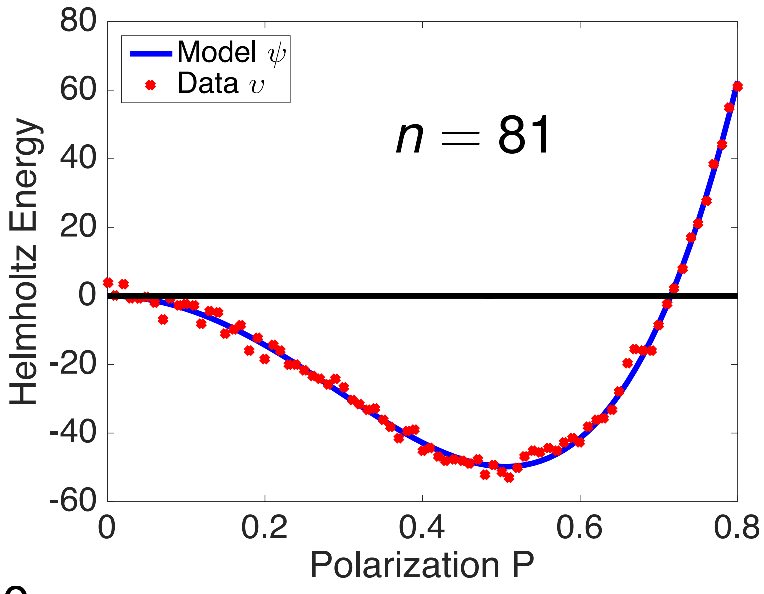
Example: Helmholtz energy $\psi(P,q) = \alpha_1 P^2 + \alpha_{11} P^4$, $q = [\alpha_1, \alpha_{11}]$

Statistical Model: Describes observation process

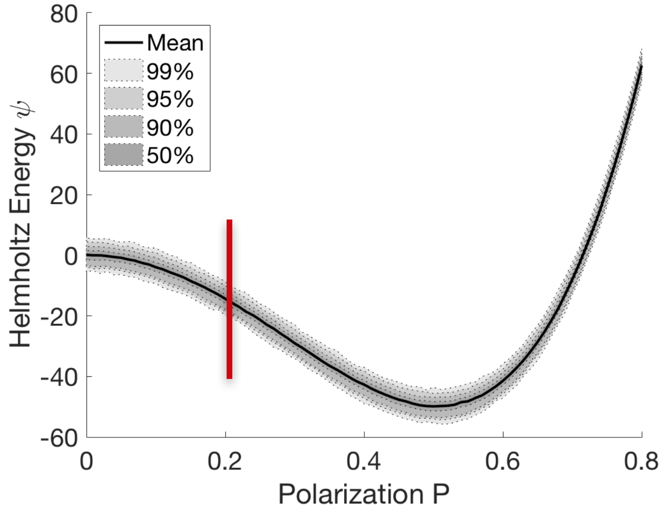
$$\upsilon_i = \psi(P_i, q) + \varepsilon_i \quad , \; i = 1, \ldots, n$$

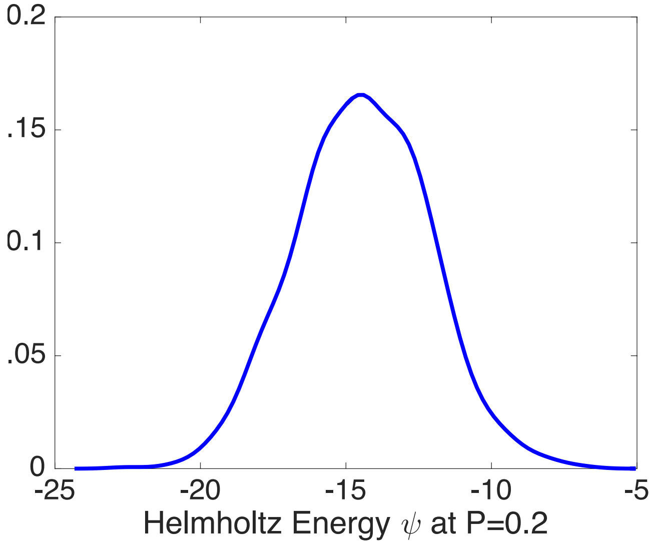
Common Assumption: $\varepsilon_i \sim N(0, \sigma^2)$

UQ Goals: Quantify parameter and response uncertainties



Strategy 1: Perform many experiments; e.g., 1000





Strategy 1: Perform Experiments

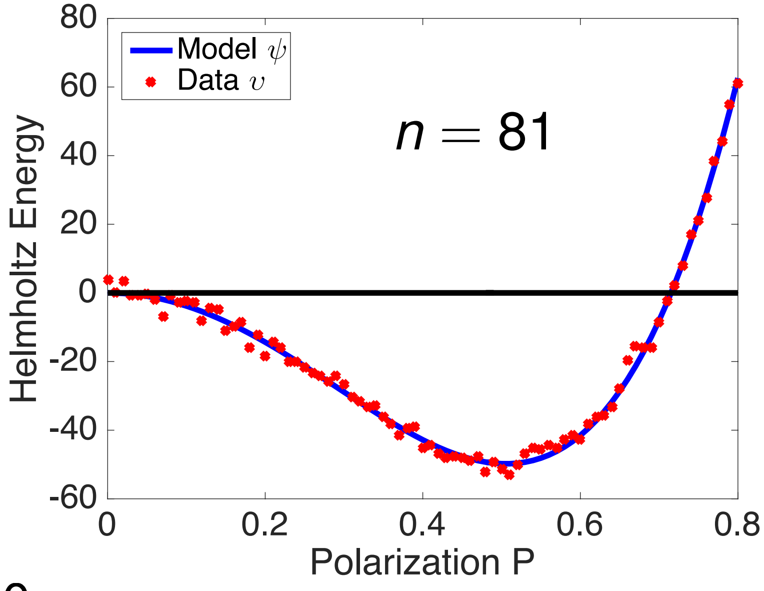
Example: Helmholtz energy $\psi(P,q) = \alpha_1 P^2 + \alpha_{11} P^4$, $q = [\alpha_1, \alpha_{11}]$

Statistical Model: Describes observation process

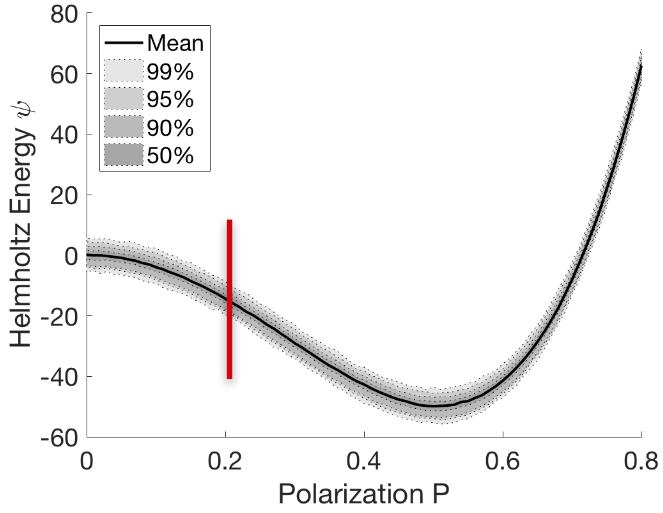
$$\upsilon_i = \psi(P_i, q) + \varepsilon_i \quad , \; i = 1, \ldots, n$$

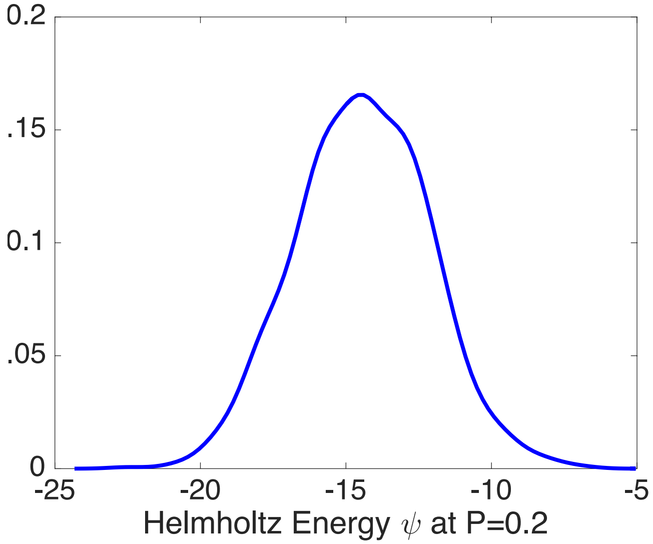
Common Assumption: $\varepsilon_i \sim N(0, \sigma^2)$

UQ Goals: Quantify parameter and response uncertainties



Strategy 1: Perform many experiments; e.g., 1000





Problem: Often cannot perform required number of experiments or high-fidelity simulations.

Solution: Statistical inference

Polarization Example

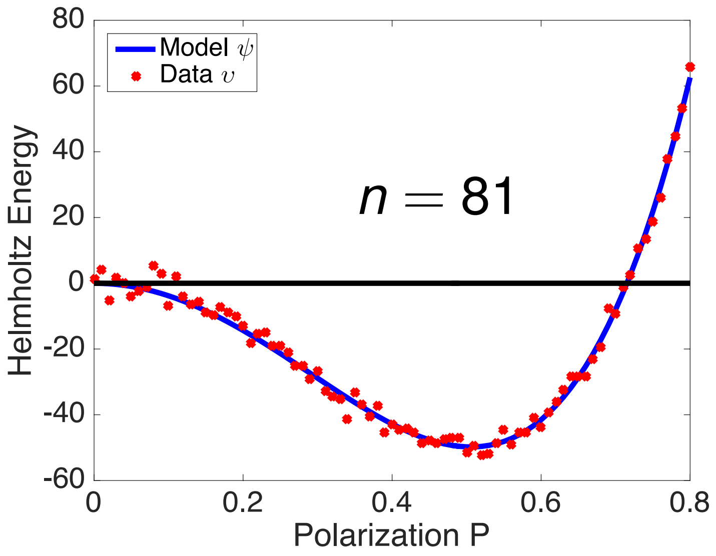
Statistical Model: For i = 1, …, n

$$\upsilon_i = \psi(P_i, q) + \varepsilon_i \longleftarrow \varepsilon_i \sim N(0, \sigma^2)$$

$$= \alpha_1 P_i^2 + \alpha_{11} P_i^4 + \varepsilon_i$$

$$\Rightarrow \begin{bmatrix} \upsilon_i \end{bmatrix} = \begin{bmatrix} P_i^2 & P_i^4 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_{11} \end{bmatrix} + \begin{bmatrix} \varepsilon_i \end{bmatrix}$$

$$\Rightarrow \upsilon = Xq + \varepsilon$$



Statistical Quantities:

$$q = (X^T X)^{-1} X^T \upsilon$$

Note: $\mathbb{E}(q) = \mathbb{E}[(X^T X)^{-1} X^T \upsilon]$

$$= (X^T X)^{-1} X^T \mathbb{E}(\upsilon)$$

$$= q_0$$

$$\upsilon = Xq_0 + \varepsilon$$

And: Let $A = (X^T X)^{-1} X^T$

$$\begin{aligned} V(q) = & \ \mathbb{E}[(q - q_0)(q - q_0)^T] \\ = & \ \mathbb{E}[(q_0 + A\varepsilon - q_0)(q_0 + A\varepsilon - q_0)^T] \ \text{since } q = A\Upsilon = A(Xq_0 + \varepsilon) \\ = & \ A\mathbb{E}(\varepsilon\varepsilon^T)A^T \\ = & \ \sigma^2 (X^T X)^{-1} \end{aligned}$$

Polarization Example

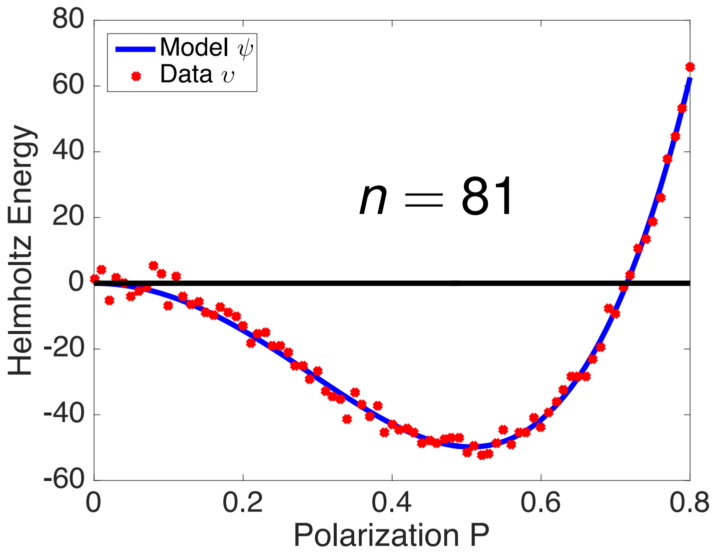
Statistical Model: For i = 1, …, n

$$\upsilon_i = \psi(P_i, q) + \varepsilon_i \longleftarrow \varepsilon_i \sim N(0, \sigma^2)$$

$$= \alpha_1 P_i^2 + \alpha_{11} P_i^4 + \varepsilon_i$$

$$\Rightarrow \begin{bmatrix} \upsilon_i \end{bmatrix} = \begin{bmatrix} P_i^2 & P_i^4 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_{11} \end{bmatrix} + \begin{bmatrix} \varepsilon_i \end{bmatrix}$$

$$\Rightarrow \upsilon = Xq + \varepsilon$$



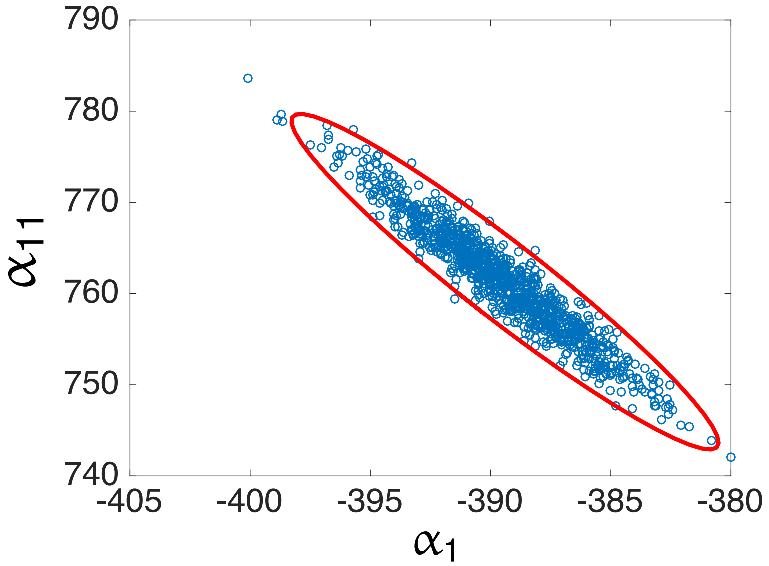
Statistical Quantities:

$$q = (X^T X)^{-1} X^T \upsilon$$

var(α_1)

$$V = \underline{\sigma^2}(X^T X)^{-1} = \begin{bmatrix} 8.8 & -17.4 \\ -17.4 & 37.6 \end{bmatrix}$$

cov(α_1, α_{11}) var(α_{11})



Note: Covariance matrix incorporates "geometry"

- May require inversion of very large matrices; million by million for neutronics

Mathematical Topics

Linear Algebra and Numerical Matrix Theory:

- Vector properties including orthogonality
- Matrix analysis, inversion and solving Ax = b for very large systems
- Eigen- and singular value decompositions

Numerical Analysis:

- Approximate integration including high-dimensional spaces; e.g., p= 100
- Approximate differentiation including in the presence of noise
- Polynomial interpolation and regression

Linear Algebra and Numerical Matrix Theory

Topics: Illustrate with MATLAB as topics are introduced

- Basic concepts
- Linear transformations
- Linear independence, basis vectors, and span of a vector space
- Fundamental Theorem of Linear Algebra
- Determinants and matrix rank
- Eigenvalues and eigenvectors
- Solving Ax = b and condition numbers
- Singular value decomposition (SVD)
- Cholesky and QR decompositions

Linear Algebra has become as basic and as applicable
as calculus, and fortunately it is easier.
--Gilbert Strang, MIT

Vectors and Matrices

Note:

- *Vector* in R^n is an ordered set of n real numbers.
  - e.g. v = (1,6,3,4) is in R^4
  - "(1,6,3,4)" is a column vector:

    $$\begin{pmatrix} 1 \\ 6 \\ 3 \\ 4 \end{pmatrix}$$
  - as opposed to a row vector:

    $$\begin{pmatrix} 1 & 6 & 3 & 4 \end{pmatrix}$$
- m-by-n *matrix* is an object with m rows and n columns, each entry fill with a real number:

  $$\begin{pmatrix} 1 & 2 & 8 \\ 4 & 78 & 6 \\ 9 & 3 & 2 \end{pmatrix}$$

Vector Norms

Vector Norms: The norm ||x|| quantifies the ``length’’

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

Common Norms:

$$||x||_1 = \sum_{i=1}^{n} |x_i|$$

$$||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Euclidean: Vector length

$$||x||_\infty = \mathrm{max}_i \, |x_i|$$

Vector and Matrix Products

Vector products:

- Dot product: $u \bullet v = u^T v = \begin{pmatrix} u_1 & u_2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2$

Note: $A \cdot B = \|A\| \, \|B\| \cos(\theta)$

If u•v=0, $||u||_2$!= 0, $||v||_2$!= 0 → *u and v are orthogonal*

If u•v=0, $||u||_2$ = 1, $||v||_2$ = 1 → *u and v are orthonormal*

- Outer product: $uv^T = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} v_1 & v_2 \end{pmatrix} = \begin{pmatrix} u_1 v_1 & u_1 v_2 \\ u_2 v_1 & u_2 v_2 \end{pmatrix}$

Matrix Product:

$$A \in \mathbb{R}^{m \times n} \quad B \in \mathbb{R}^{n \times p}$$

$$C_{ij} = \sum_{k=1} A_{ik} B_{kj} \quad C = AB \in \mathbb{R}^{m \times p}$$

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$AB = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

Special Matrices

$$\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}$$ diagonal

$$\begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix}$$ upper-triangular

$$\begin{pmatrix} a & b & 0 & 0 \\ c & d & e & 0 \\ 0 & f & g & h \\ 0 & 0 & i & j \end{pmatrix}$$ tri-diagonal

$$\begin{pmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{pmatrix}$$ lower-triangular

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$ I (identity matrix)

Special Matrices

- Matrix A is *symmetric* if A = A^T
- A is *positive definite* if $x^TAx>0$ for all non-zero x (*positive semi-definite* if inequality is not strict)

$$\begin{pmatrix} a & b & c \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} a \\ b \\ c \end{pmatrix} = a^2 + b^2 + c^2 \quad \begin{pmatrix} a & b & c \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} a \\ b \\ c \end{pmatrix} = a^2 - b^2 + c^2$$

- Useful fact: Any matrix of form A^TA is positive semi-definite.

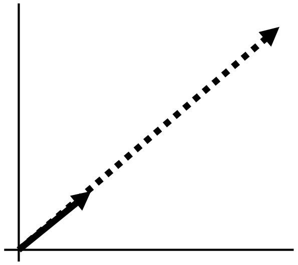
To see this, $x^T(A^TA)x = (x^TA^T)(Ax) = (Ax)^T(Ax) \geq 0$

Recall: Covariance matrix

$$V = \sigma^2(X^TX)^{-1}$$

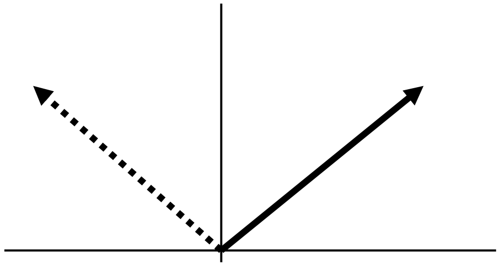
Matrices as Linear Transformations

$$\begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$$



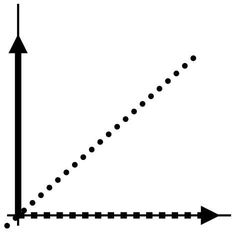
(stretching)

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$



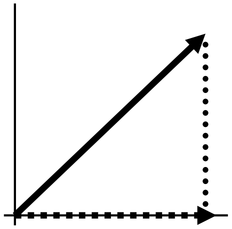
(rotation)

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



(reflection)

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



(projection)

Linear Independence

- A set of vectors is linearly independent if none of them can be written as a linear combination of the others.
- Vectors $v_1,\ldots,v_k$ are linearly independent if $c_1v_1+\ldots+c_kv_k = 0$ implies $c_1=\ldots=c_k=0$

$$\begin{pmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

e.g.

$$\begin{pmatrix} 1 & 0 \\ 2 & 3 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

(u,v)=(0,0), i.e. the columns are linearly independent.

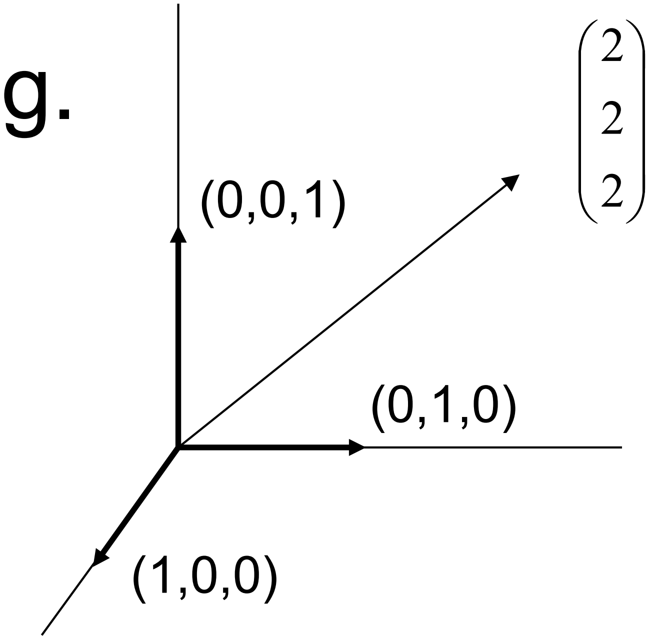
$$x_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad x_2 = \begin{bmatrix} 4 \\ 1 \\ 5 \end{bmatrix} \quad x_3 = \begin{bmatrix} 2 \\ -3 \\ -1 \end{bmatrix}$$

x3 = −2x1 + x2

Basis Vectors

- If all vectors in a vector space may be expressed as linear combinations of a set of vectors $v_1,\ldots,v_k$, then $v_1,\ldots,v_k$ spans the space.
- The cardinality of this set is the dimension of the vector space.

e.g.



$$\begin{pmatrix}2\\2\\2\end{pmatrix} = 2\begin{pmatrix}1\\0\\0\end{pmatrix} + 2\begin{pmatrix}0\\1\\0\end{pmatrix} + 2\begin{pmatrix}0\\0\\1\end{pmatrix}$$

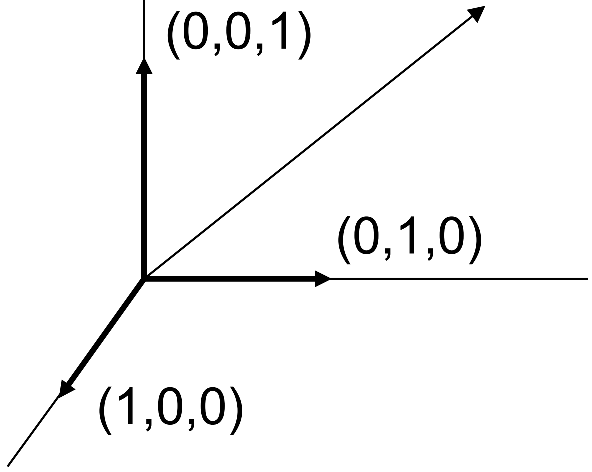
- A basis is a maximal set of linearly independent vectors and a minimal set of spanning vectors of a vector space

Basis Vectors

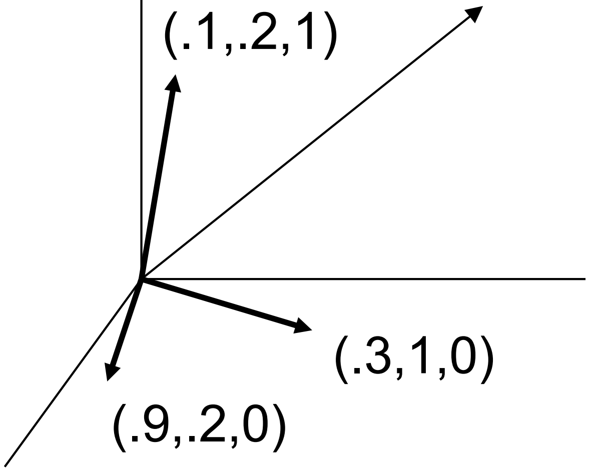
Note:

- An *orthonormal basis* consists of orthogonal vectors of unit length.

$$\begin{pmatrix}2\\2\\2\end{pmatrix} = 2\begin{pmatrix}1\\0\\0\end{pmatrix} + 2\begin{pmatrix}0\\1\\0\end{pmatrix} + 2\begin{pmatrix}0\\0\\1\end{pmatrix}$$



$$\begin{pmatrix}2\\2\\2\end{pmatrix} = 1.57\begin{pmatrix}.9\\.2\\0\end{pmatrix} + 1.29\begin{pmatrix}.3\\1\\0\end{pmatrix} + 2\begin{pmatrix}.1\\.2\\1\end{pmatrix}$$



Rank of a Matrix

- The *rank* of A is the dimension of the column space of A.
- It also equals the dimension of the *row space* of A (the subspace of vectors which may be written as linear combinations of the rows of A).

$$\begin{pmatrix} 1 & 0 \\ 2 & 3 \\ 1 & 3 \end{pmatrix}$$

(1,3) = (2,3) – (1,0)

Only 2 linearly independent rows, so rank = 2.

Fundamental Theorem of Linear Algebra:

If A is m x n with rank r,

Column space(A) has dimension r

Nullspace(A) has dimension n-r (= *nullity* of A)

Row space(A) = Column space(A^T) has dimension r

Left nullspace(A) = Nullspace(A^T) has dimension m - r

Rank-Nullity Theorem: rank + nullity = n

Matrix Inverse

Note:

- To solve $Ax=b$, we can write a closed-form solution if we can find a matrix A^{-1} such that $AA^{-1} = A^{-1}A = I$ (Identity matrix)
- Then $Ax=b$ iff $x=A^{-1}b$:
  $x = Ix = A^{-1}Ax = A^{-1}b$
- A is *non-singular* iff A^{-1} exists iff $Ax=b$ has a unique solution.
- Note: If A^{-1},B^{-1} exist, then $(AB)^{-1} = B^{-1}A^{-1}$, and $(A^T)^{-1} = (A^{-1})^T$

- For orthonormal matrices

$$A^{-1} = A^T$$

Matrix Determinants

Note:

- If det(A) = 0, then A is singular.
- If det(A) ≠ 0, then A is invertible.
- To compute:
  - Simple example:

$$\det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

  - Matlab: det(A)

MATLAB Interlude

Special Variables:

- Special variables:
  - ans : default variable name for the result
  - pi: π = 3.1415926…………
  - eps: $\in$ = 2.2204e-016, smallest amount by which 2 numbers can differ.
  - Inf or inf : ∞, infinity
  - NaN or nan: not-a-number

Vectors: Example:

```
>> x = [ 0   0.25*pi   0.5*pi   0.75*pi   pi ]
x =
          0    0.7854    1.5708    2.3562    3.1416
>> y = [ 0; 0.25*pi; 0.5*pi; 0.75*pi; pi ]
y =
          0
     0.7854
     1.5708
     2.3562
     3.1416
```

x is a row vector.

y is a column vector.

Vectors

- Vector Addressing – A vector element is addressed in MATLAB with an integer index enclosed in parentheses.
- Example:

```
>> x(3)
ans =
    1.5708   ← 3rd element of vector x
```

- The colon notation may be used to address a block of elements.

(start : increment : end)

start is the starting index, increment is the amount to add to each successive index, and end is the ending index. A shortened format (start : end) may be used if increment is 1.

- Example:

```
>> x(1:3)
ans =
          0    0.7854    1.5708   ← 1st to 3rd elements of vector x
```

NOTE: MATLAB index starts at 1.

Vectors

Some useful commands:

x = start:end	create row vector x starting with start, counting by one, ending at end
x = start:increment:end	create row vector x starting with start, counting by increment, ending at or before end
linspace(start,end,number)	create row vector x starting with start, ending at end, having number elements
length(x)	returns the length of vector x
y = x’	transpose of vector x
dot (x, y)	returns the scalar dot product of the vector x and y.

Matrices

- A Matrix array is two-dimensional, having both multiple rows and multiple columns, similar to vector arrays:
  - it begins with [, and end with]
  - spaces or commas are used to separate elements in a row
  - semicolon or enter is used to separate rows.

- Matrix Addressing:

  -- *matrixname(row, column)*

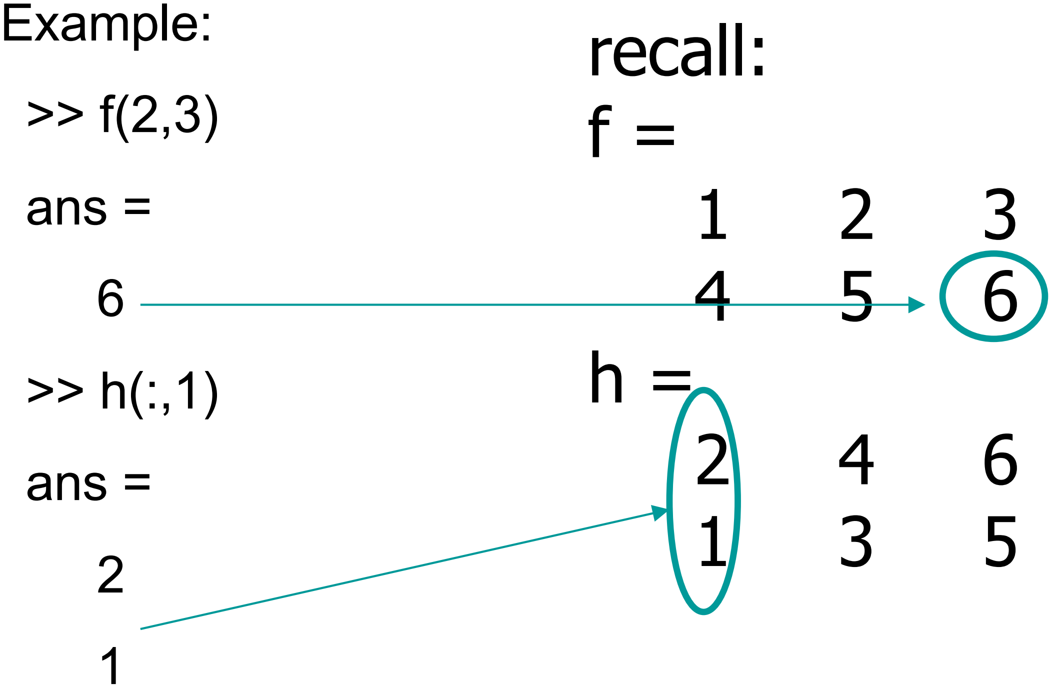
  -- **colon** may be used in place of a row or column reference to select the entire row or column.

•Example:

•>> f = [1 2 3; 4 5 6]

```
      f =
          1    2    3
          4    5    6
```

- Example:



```
>> f(2,3)

ans =

     6

>> h(:,1)

ans =

     2

     1
```

recall:

```
f =
     1    2    3
     4    5    6
h =
     2    4    6
     1    3    5
```

Matrices

more commands

Transpose	B = A′
Identity Matrix	eye(n) ➔ returns an n x n identity matrix eye(m,n) ➔ returns an m x n matrix with ones on the main diagonal and zeros elsewhere.

Addition and subtraction	C = A + B
	C = A – B

Scalar Multiplication	B = αA, where α is a scalar.
Matrix Multiplication	C = A*B
Matrix Inverse	B = inv(A), A must be a square matrix in this case. rank (A) ➔ returns the rank of the matrix A.
Matrix Powers	B = A.^2 ➔ squares each element in the matrix C = A * A ➔ computes A*A, and A must be a square matrix.
Determinant	det (A), and A must be a square matrix.

A, B, C are matrices, and m, n, α are scalars.

Matrices

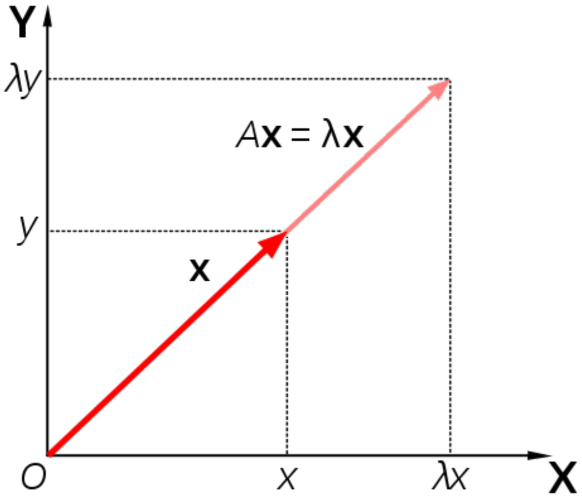
```
>> A = [1 2;3 4]
 A =
           1     2
           3     4

>> B = [2 0;2 1]
B =
          2     0
          2     1

>> A*B
ans =
          6     2
         14     4

>> A.*B
ans =
         2     0
         6     4
```

```
>> A = [1 2 3;0 2 0]
A =
      1     2     3
      0     2     0

>> B = [1;-1;0]
B =
      1
     −1
      0

>> A*B
ans =
     −1
     −2
```

Eigenvalues and Eigenvectors

Note:

- How can we characterize matrices?
- The solutions to Ax = λx in the form of eigenpairs (λ,x) = (eigenvalue,eigenvector) where x is non-zero
- To solve this, (A – λI)x = 0
- λ is an eigenvalue iff det(A – λI) = 0



Example:

$$A = \begin{pmatrix} 1 & 4 & 5 \\ 0 & 3/4 & 6 \\ 0 & 0 & 1/2 \end{pmatrix}$$

$$\det(A - \lambda I) = \begin{pmatrix} 1-\lambda & 4 & 5 \\ 0 & 3/4-\lambda & 6 \\ 0 & 0 & 1/2-\lambda \end{pmatrix} = (1-\lambda)(3/4-\lambda)(1/2-\lambda)$$

$$\lambda = 1, \lambda = 3/4, \lambda = 1/2$$

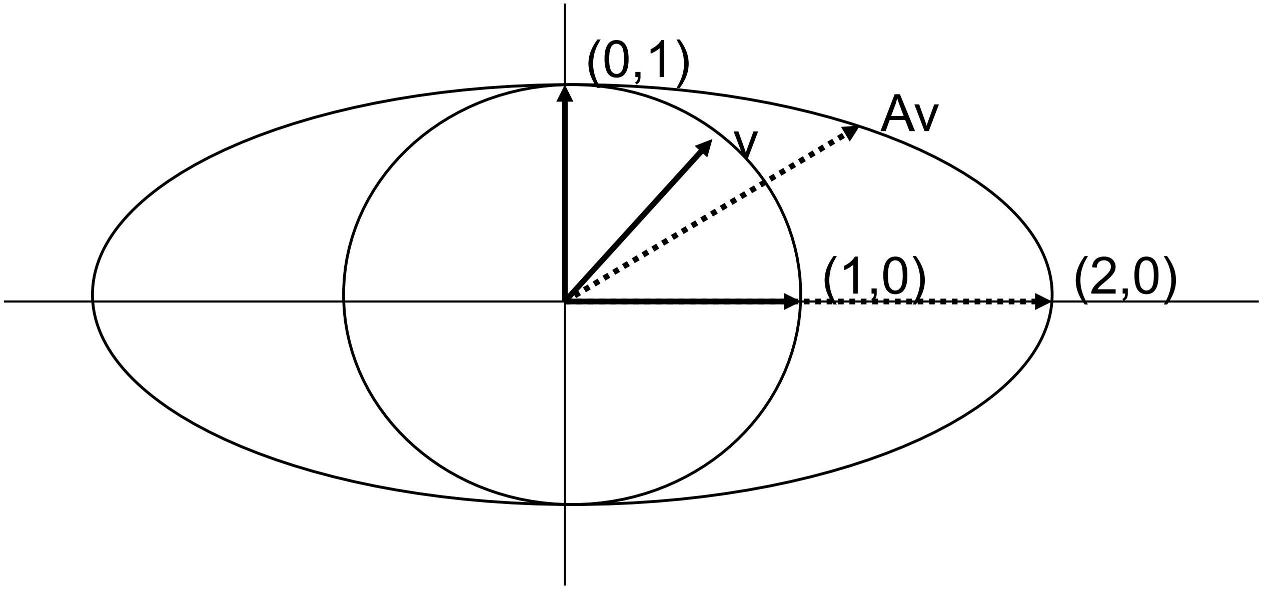
Eigenvalues and Eigenvectors

Example:

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

Eigenvalues λ = 2, 1 with eigenvectors (1,0), (0,1)

Eigenvectors of a linear transformation A are not rotated (but will be scaled by the corresponding eigenvalue) when A is applied.



Important Results:

- Eigenvalues of nxn symmetric matrix are real but may not be distinct. There are n orthogonal eigenvectors.

Eigenvalues and Eigenvectors

Important Results:

- Eigenvalues of nxn symmetric matrix are real but may not be distinct. There are n orthogonal eigenvectors; e.g., Identity matrix
- Eigenvalues of a positive definite matrix are positive.
- MATLAB Command: [V,D] = eig(A)
- Suppose A has n linearly independent eigenvectors. Then

$$Q = [v_1 \quad v_2 \ \cdots \ v_n]$$

$$\Rightarrow AQ = [\lambda_1 v_1 \quad \lambda_2 v_2 \ \cdots \ \lambda_n v_n]$$

$$\Rightarrow AQ = Q\Lambda$$

$$\Rightarrow A = Q\Lambda Q^{-1}$$

$$\Rightarrow Q^{-1}AQ = \Lambda$$

Note:

- The matrix Q thus provides a new basis for A.

Eigenvalues and Eigenvectors

Example:

```
>> A = [2 1 3;-2 0 -2;5 5 2]
A =
      2     1     3
     −2     0    −2
      5     5     2


>> X = A'*A
X =
      33    27    20
       27    26    13
       20    13    17


>> [Q,D] = eig(X)
Q =
   −0.7178    0.0398    0.6952
    0.5287   −0.6185    0.5813
    0.4531    0.7848   0.4229


D =
    0.4864          0          0
         0        7.7683       0
           0          0      67.7452
```

```
>>Q*D*inv(Q)
ans =
          33.0000    27.0000    20.0000
          27.0000    26.0000    13.0000
          20.0000    13.0000    17.0000


>> Q’
ans =
          −0.7178     0.5287     0.4531
            0.0398   −0.6185     0.7848
            0.6952     0.5813     0.4229


>> inv(Q)
ans =
          −0.7178     0.5287     0.4531
           0.0398   −0.6185     0.7848
           0.6952     0.5813     0.4229
```

Eigenvalues and Eigenvectors

Cholesky Decomposition: Matrix must be positive definite

```
>> A = [2 1 3;-2 0 -2;5 5 2]
A =
     2     1     3
    −2     0    −2
     5     5     2

>> X = A'*A
X =
     33    27    20
     27    26    13
     20    13    17

>> R = chol(X)
R =
   5.7446    4.7001    3.4816
        0    1.9771   −1.7013
        0         0    1.4087

>> R'*R
ans =
   33    27    20
   27    26    13
   20    13    17
```

Solving Ax = b

Notes:

- MATLAB: x = A\b;
  - How stable is the solution?
  - If A or b are changed slightly, how much does it effect x?
  - The *condition number* c of A measures this:

    $$c = \lambda_{max}/ \lambda_{min}$$

  - Values of c near 1 are good.

```
>> A = [1 0;0 1e-10]
A =
     1.0000         0
      0          0.0000

>> cond(A)
ans =
     1.0000e+10>>
```

MATLAB

- Example: a system of 3 linear equations with 3 unknowns (x_1, x_2, x_3):

$$\begin{aligned} 3x_1 + 2x_2 - x_3 &= 10 \\ -x_1 + 3x_2 + 2x_3 &= 5 \\ x_1 - x_2 - x_3 &= -1 \end{aligned}$$

Let :

$$A = \begin{bmatrix} 3 & 2 & 1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad b = \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}$$

Then, the system can be described as:

$$Ax = b$$

MATLAB

- <u>Solution by Matrix Inverse:</u>

  $Ax = b$

  $A^{-1}Ax = A^{-1}b$

  $x = A^{-1}b$

- MATLAB:

```
>> A = [ 3 2 -1; -1 3 2; 1 -1 -1];
>> b = [ 10; 5; -1];
>> x = inv(A)*b
x =
   -2.0000
    5.0000
   -6.0000
```

<u>Answer</u>:

x1 = -2, x2 = 5, x3 = -6

- <u>Solution by Matrix Division:</u>

The solution to the equation

$$Ax = b$$

can be computed using left division.

- MATLAB:

```
>> A = [ 3 2 -1; -1 3 2; 1 -1 -1];
>> b = [ 10; 5; -1];
>> x = A\b
x =
   -2.0000
    5.0000
   -6.0000
```

<u>Answer</u>:

x1 = -2, x2 = 5, x3 = -6

MATLAB: Flow Control

For Loops:

```
for j=1:5                     % use for-loops to execute iterations / repetitions
    for i=1:3
        a(i, j) = i + j ;
    end
end
```

If Conditional:

```
a = zeros(3);  b = zeros(3);
for j=1:3
   for i=1:3
      a(i,j) = rand;          % use rand to generate a random number
      if a(i,j) > 0.5
         b(i,j) = 1;
      end
  end
end
```

MATLAB

Cell Arrays:

A cell array is a special array of arrays. Each element of the cell array may point to a scalar, an array, or another cell array.

```
>> C = cell(2, 3);   % create 2x3 empty cell array
>> M = magic(2);
>> a = 1:3; b = [4;5;6]; s = 'This is a string.';
>> C{1,1} = M; C{1,2} = a; C{2,1} = b; C{2,2} = s; C{1,3} = {1};
C =
    [2x2 double]     [1x3 double]          {1x1 cell}
    [2x1 double]     ‘This is a string.‘    []
>> C{1,1}     % prints contents of a specific cell element
ans =
     1     3
     4     2
>> C(1,:)     % prints first row of cell array C; not its content
```

MATLAB

Structures:

Ideal layout for grouping arrays that are related.

```
>> name(1).last = ‘Smith’;  name(2).last  = ‘Hess’;
>> name(1).first = ‘Mary’;   name(2).first = ‘Robert’;
>> name(1).sex = ‘female’; name(2).sex = ‘male’;
>> name(1).age = 45;         name(2).age = 50;
>> name(2)
ans =
     last: 'Hess'
    first: 'Robert'
      sex: 'male'
      age: 50
```

MATLAB

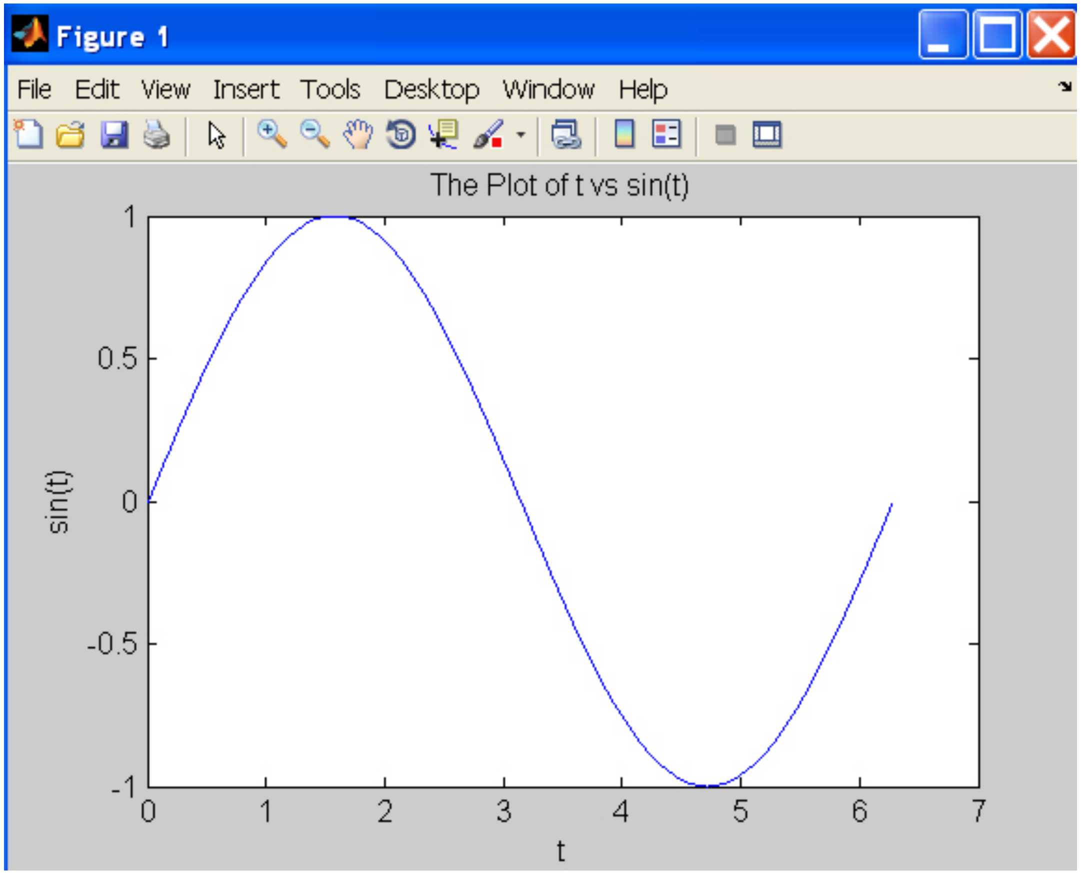
Frequently Used Functions:

```
>> magic(n)       % creates a special n x n  matrix; handy for testing
>> zeros(n,m)     % creates n x m matrix of zeroes (0)
>> ones(n,m)      % creates n x m matrix of ones (1)
>> rand(n,m)      % creates n x m matrix of random numbers
>> repmat(a,n,m)     % replicates a by n rows and m columns
>> diag(M)        % extracts the diagonals of a matrix M
>> help elmat     % list all elementary matrix operations ( or elfun)
>> abs(x);        % absolute value of x
>> exp(x);        % e to the x-th power
>> fix(x);        % rounds x to integer towards 0
>> log10(x);      % common logarithm of x to the base 10
>> rem(x,y);      % remainder of x/y
>> mod(x, y);     % modulus after division – unsigned rem
>> sqrt(x);       % square root of x
>> sin(x);        % sine of x; x in radians
>> acoth(x)       % inversion hyperbolic cotangent of x
```

Plotting

Line Plot:

```
>> t = 0:pi/100:2*pi;
>> y = sin(t);
>> plot(t,y)
>> xlabel('t');
>> ylabel('sin(t)');
>> title('The plot of t vs sin(t)');
```



Plotting

Customizing Graphical Effects

Generally, MATLAB’s default graphical settings are adequate which make plotting fairly effortless. For more customized effects, use the *get* and *set* commands to change the behavior of specific rendering properties.

```
>> hp1 = plot(1:5)           % returns the handle of this line plot
>> get(hp1)                  % to view line plot’s properties and their values
>> set(hp1, ‘lineWidth’)     % show possible values for lineWidth
>> set(hp1, ‘lineWidth’, 2)  % change line width of plot to 2
>> gcf            % returns current figure handle
>> gca            % returns current axes handle
>> get(gcf)       % gets current figure’s property settings
>> set(gcf, ‘Name’, ‘My First Plot’)     % Figure 1 => Figure 1: My First Plot
>> get(gca)       % gets the current axes’  property settings
>> figure(1)      % create/switch to Figure 1 or pop Figure 1 to the front
>> clf            % clears current figure
>> close          % close current figure; “close 3” closes Figure 3
>> close all      % close all figures
```

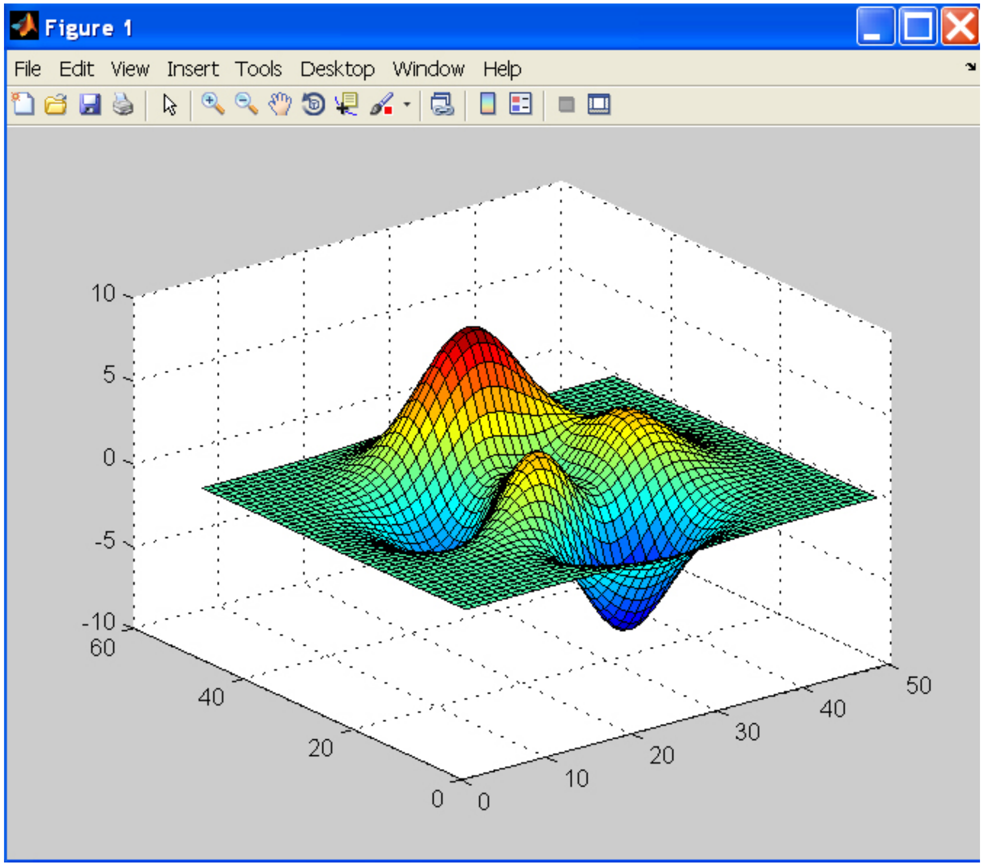
Plotting

Surface Plot

```
>> Z = peaks;     % generate data for plot;   peaks returns  function values
>> surf(Z)          % surface plot of Z
```

Try these commands also:

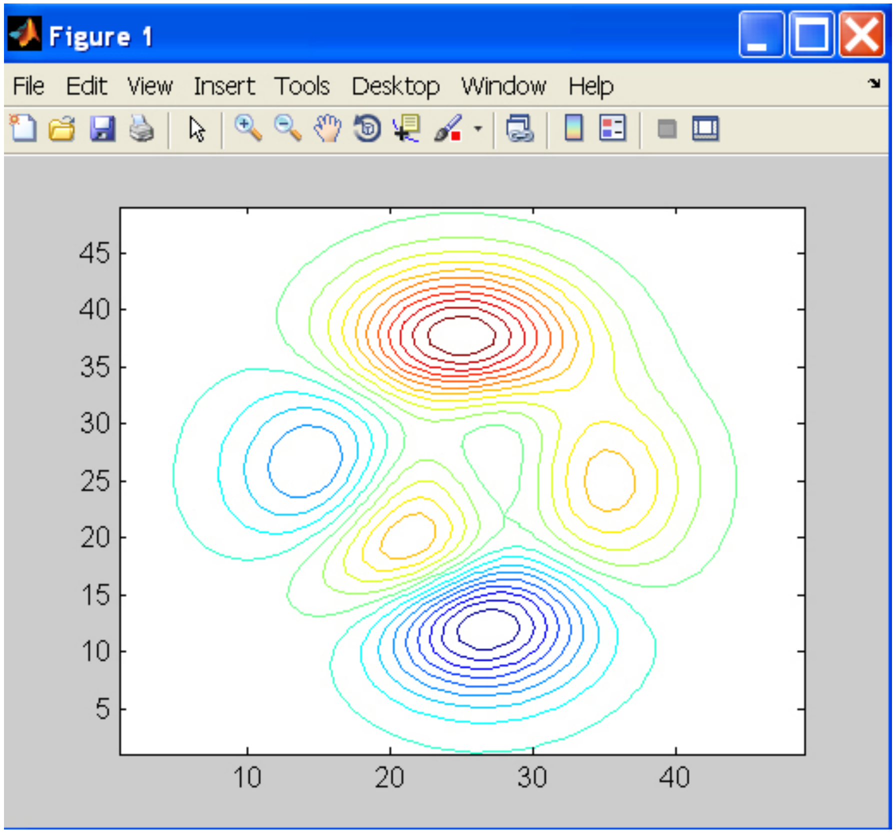
```
>> shading flat
>> shading interp
>> shading faceted
>> grid off
>> axis off
>> colorbar
>> colormap(‘winter’)
>> colormap(‘jet’)
```



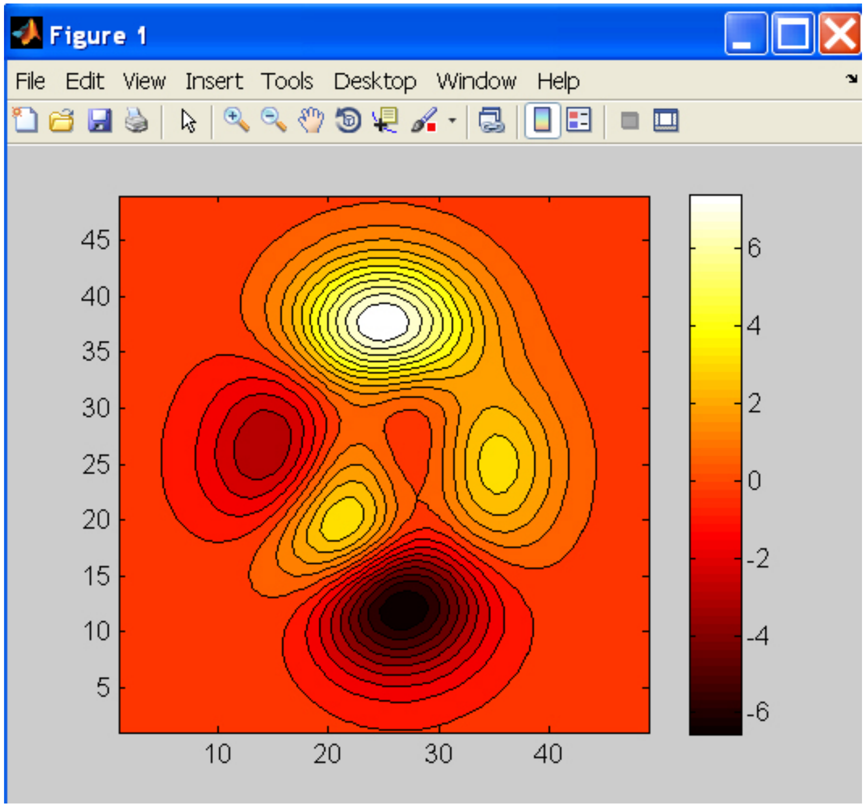
Plotting

Contour Plots

```
>> Z = peaks;
>> contour(Z, 20)    % contour plot of Z with 20 contours
```



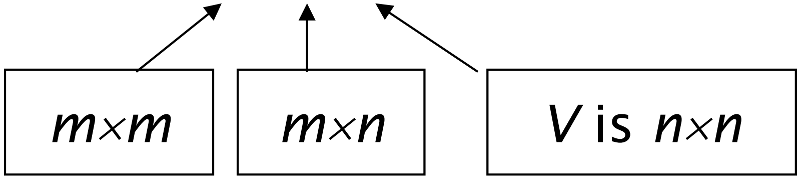
```
>> contourf(Z, 20);  % with color fill
>> colormap('hot')   % map option
>> colorbar    % make color bar
```



Singular Value Decomposition

For an $m\times n$ matrix $\mathbf{A}$ of rank r there exists a factorization (Singular Value Decomposition = **SVD**) as follows:

$$A = U\Sigma V^T$$



$m\times m$

$m\times n$

V is $n\times n$

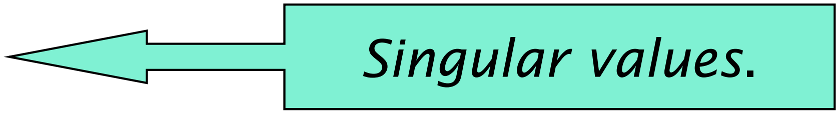
The columns of ***U*** are orthogonal eigenvectors of $\boldsymbol{AA^T}$.

The columns of ***V*** are orthogonal eigenvectors of $\boldsymbol{A^TA}$.

Eigenvalues λ1 … λr of AA^T are the eigenvalues of A^TA.

$$\sigma_i = \sqrt{\lambda_i}$$

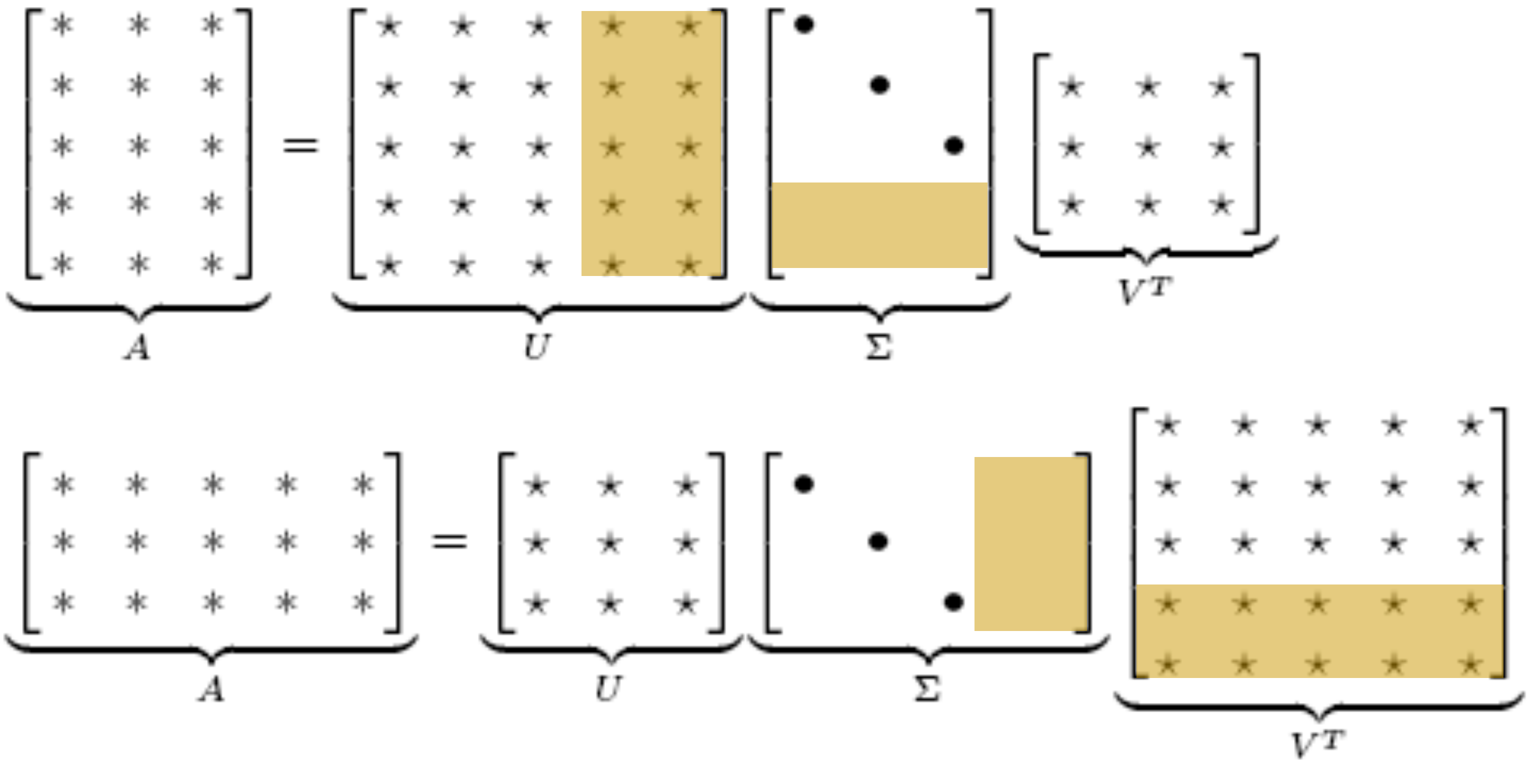
$$\Sigma = diag(\sigma_1 \ldots \sigma_r)$$



Singular values.

Singular Value Decomposition

- Illustration of SVD dimensions and sparseness



Singular Value Decomposition

Let $A = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$

Thus *m*=3, *n*=2. Its SVD is

$$\begin{bmatrix} 0 & 2/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Typically, the singular values arranged in decreasing order.

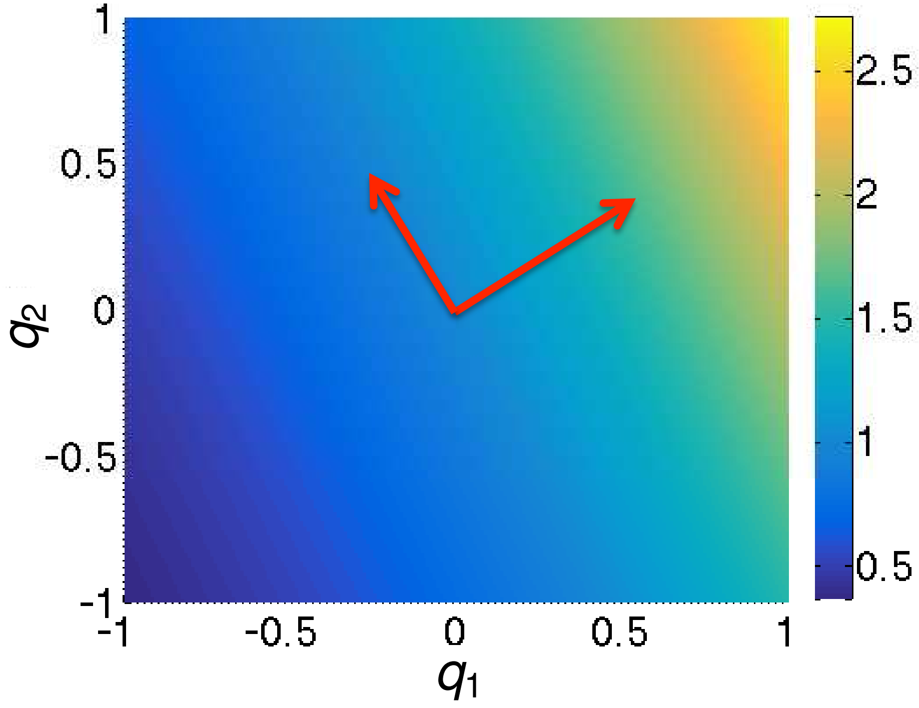
Active Subspaces

Note:

- Functions may vary significantly in only a few directions
- “Active” directions may be linear combination of inputs

Example: $y = \exp(0.7q_1 + 0.3q_2)$

- Varies most in [0.7, 0.3] direction
- No variation in orthogonal direction



A Bit of History:

- Often attributed to Russi (2010).
- Concept same as *identifiable subspaces* from systems and control; e.g., Reid (1977).
- For linearly parameterized problems, active subspace given by SVD or QR; Beltrami (1873), Jordan (1874), Sylvester (1889), Schmidt (1907), Weyl (1912). See 1993 *SIAM Review* paper by Stewart.

Parameter Space Reduction Techniques: Linear Problems

Second Issue: Models depends on very large number of parameters – e.g., millions – but only a few are "significant".

Linear Algebra Techniques: Linearly parameterized problems

$$y = Aq \ , \ q \in \mathbb{R}^p \ , \ y \in \mathbb{R}^m$$

Singular Value Decomposition (SVD):

$$A = U\Sigma V^T \ , \ \Sigma = [S \quad 0]$$

$$S = \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_r & \\ & & & 0 \end{bmatrix} \quad , \ \sigma_1 \geq \sigma_2 \geq \cdots \ \geq \sigma_r \geq \varepsilon$$

Rank Revealing QR Decomposition: $A^T P = QR$

Problem: Neither is directly applicable when m or p are very large; e.g., millions.

Solution: Random range finding algorithms.

Random Range Finding Algorithms: Linear Problems

Algorithm: Halko, Martinsson and Tropp, SIAM Review, 2011

1. Choose ℓ random inputs q^i and compute outputs $y^i = Aq^i$ which are compiled in the $m \times \ell$ matrix Y.

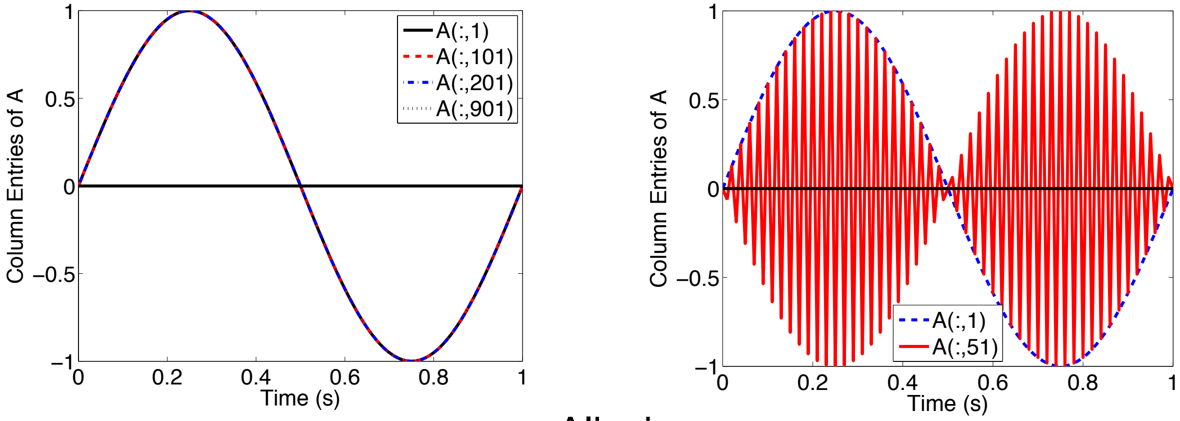
2. Take a pivoted QR factorization $Y = QR$ to construct a matrix Q whose columns form an orthonormal basis for the range of Y.

Example: $y_i = \sum_{k=1}^{p} q_k \sin(2\pi k t_i) \ , \ i = 1, \cdots, m$

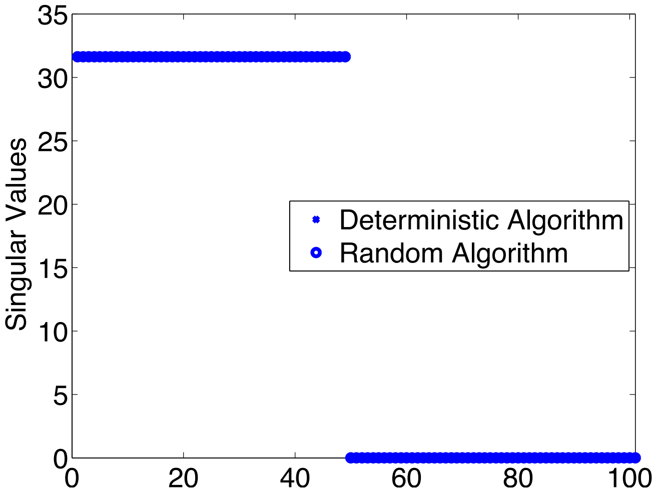
$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} \sin(2\pi t_1) & \cdots & \sin(2\pi p t_1) \\ \vdots & & \vdots \\ \sin(2\pi t_m) & \cdots & \sin(2\pi p t_m) \end{bmatrix} \begin{bmatrix} q_1 \\ \vdots \\ q_p \end{bmatrix}$$

Random Range Finding Algorithms: Linear Problems

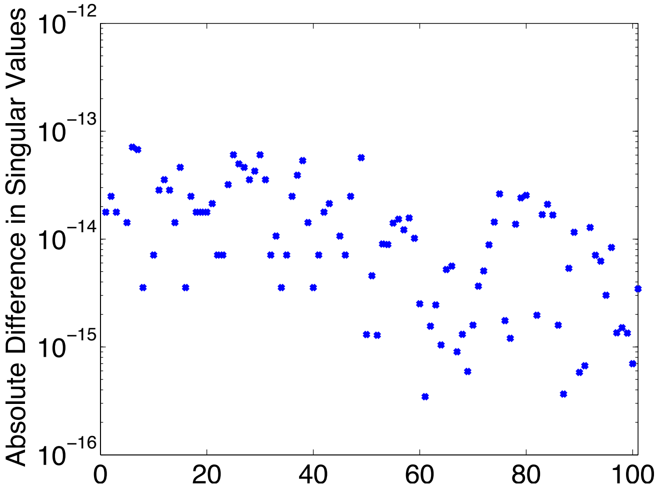
Example: m = 101, p = 1000: Analytic value for rank is 49



Aliasing



Singular Values

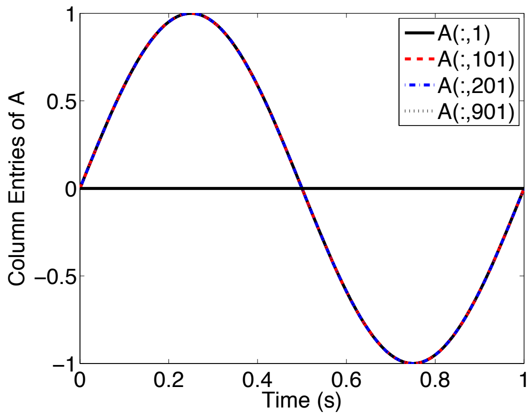


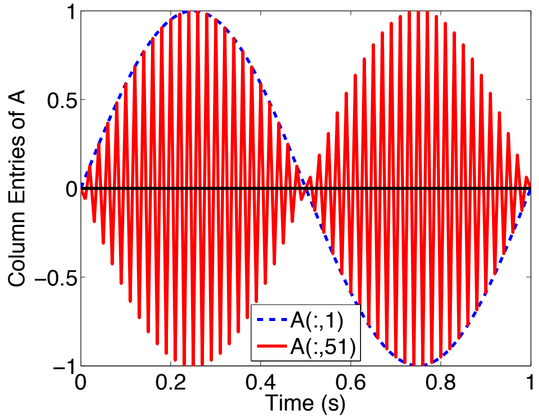
Absolute Difference Between Singular Values

Example: m = 101, p = 1,000,000: Random algorithm still viable

Random Range Finding Algorithms: Linear Problems

Example: m = 101, p = 1000

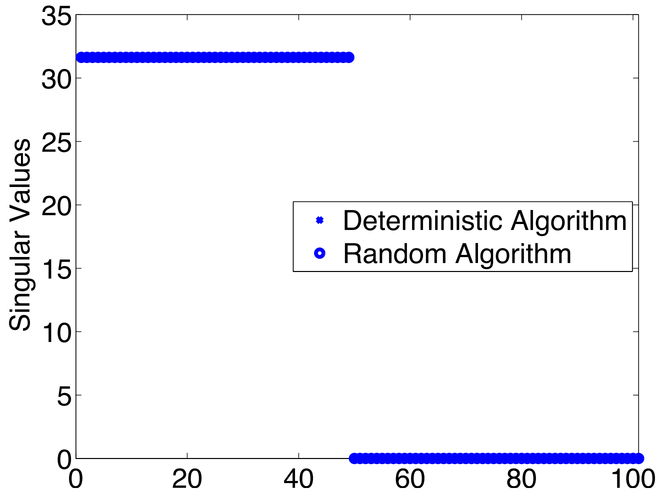




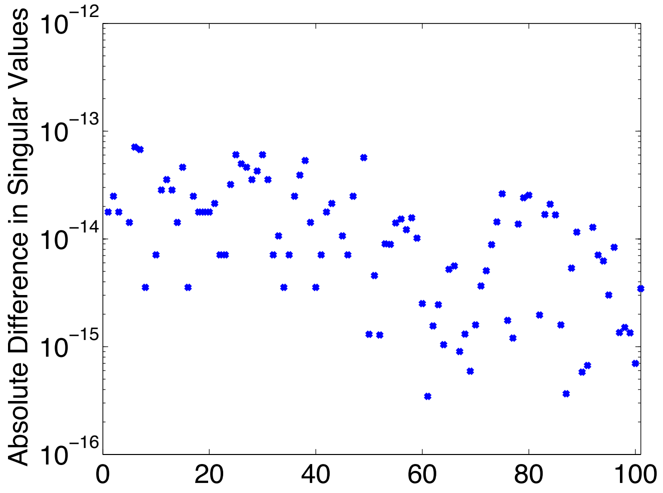
Aliasing

Hands-On:

- Reproduce these results using the code alias.m. You can experiment with the size of the random sample.
- See how far you can increase the dimension.

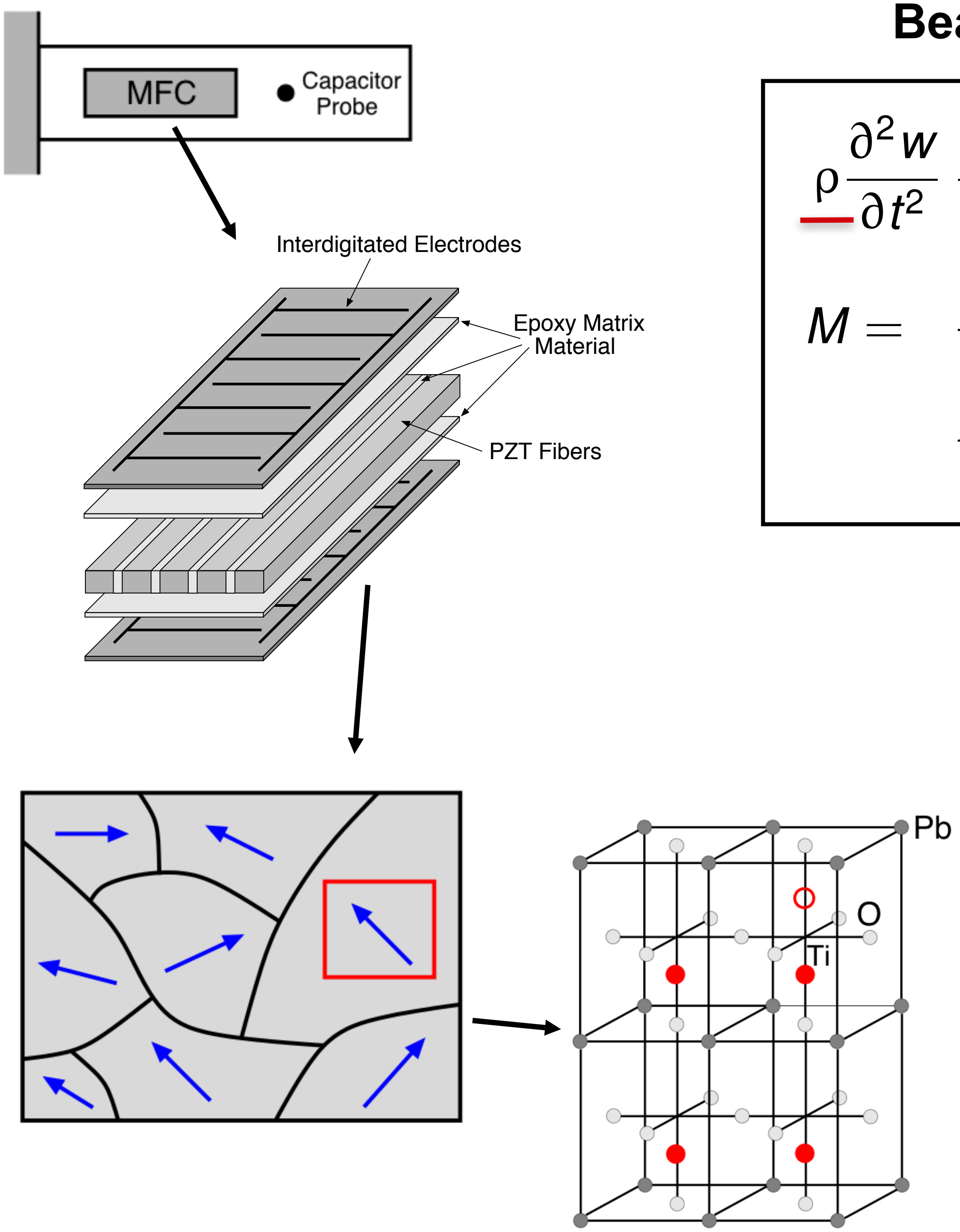


Singular Values



Absolute Difference Between Singular Values

Numerical Integration



Beam Model: 20 parameters

$$\underline{\rho}\frac{\partial^2 w}{\partial t^2} + \underline{\gamma}\frac{\partial w}{\partial t} - \frac{\partial^2 M}{\partial x^2} = 0$$

$$\begin{aligned} M = \; & -\underline{c^E} I \frac{\partial^2 w}{\partial x^2} - \underline{c_D} I \frac{\partial^3 w}{\partial x^2 \partial t} \\ & - [\underline{k_1} e(E, \sigma_0) E + \underline{k_2} \varepsilon_{irr}(E, \sigma_0)] \chi_{MFC}(x) \end{aligned}$$

UQ and Control Formulation:

$$\frac{dz}{dt} = f(t, z, q) + v_1(t)$$

$$y(t) = \int_{\mathbb{R}^{20}} w^N(t, \bar{x}, q) \rho(q) dq$$

E.g., Average tip displacement

Numerical Issues:

- Accurately approximate derivatives and high-dimensional integrals.

Motivation

What does an integral represent?

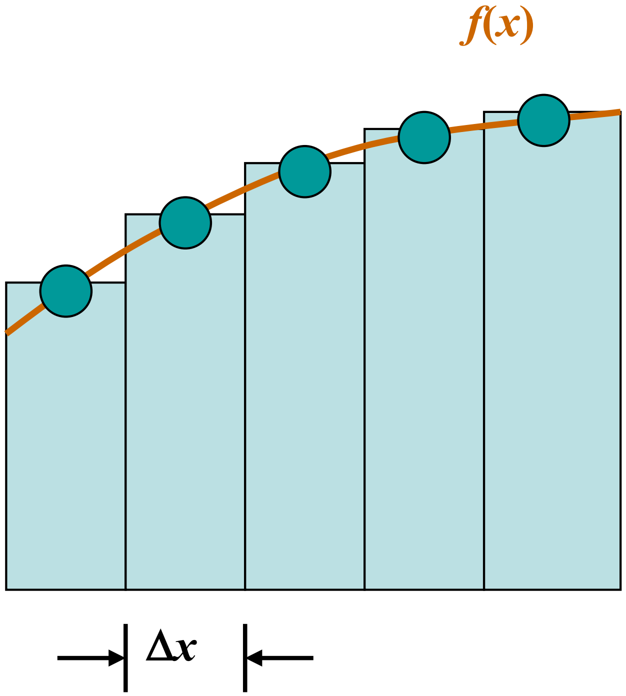
$$\int_a^b f(x)dx = \text{area} \qquad \int_c^d \int_a^b f(x)dxdy = \text{volume}$$

Basic definition of an integral:

$$\int_a^b f(x)dx = \lim_{n\to\infty}\sum_{k=1}^{n} f(x_k)\Delta x$$

where $$\Delta x = \frac{b-a}{n}$$

sum of height × width



Numerical Quadrature

Motivation: Computation of expected values requires approximation of integrals

$$\mathbb{E}[u(t,x)] = \int_{\mathbb{R}^p} u(t,x,q)\rho(q)dq$$

Example: HIV model

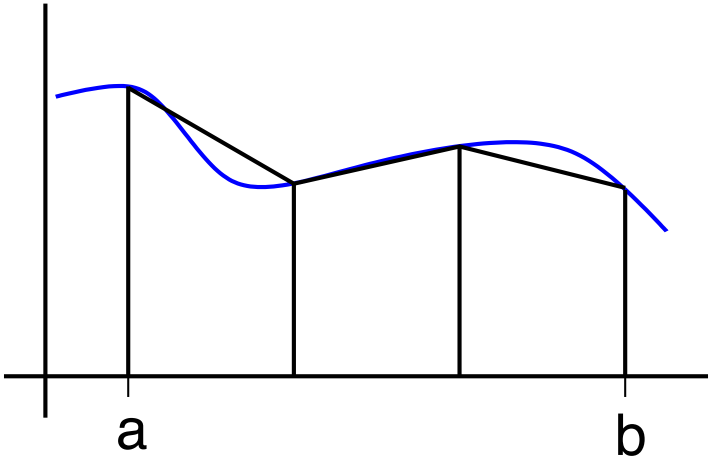
$$\mathbb{E}[V(t)] = \int_{\mathbb{R}^6} V(t,q)\rho(q)dq$$

Numerical Quadrature:

$$\int_{\mathbb{R}^p} f(q)\rho(q)dq \approx \sum_{r=1}^{R} f(q^r)w^r$$

Questions:

- How do we choose the quadrature points and weights?
    - E.g., Newton-Cotes; e.g., trapezoid rule



$$\int_a^b f(q)dq \approx \frac{h}{2}\left[f(a) + f(b) + 2\sum_{r=1}^{R-2} f(q^r)\right]$$

$$q^r = a + hr\ ,\ h = \frac{b-a}{R-1}$$

Error: $\mathcal{O}(h^2)$

Numerical Quadrature

Motivation: Computation of expected values requires approximation of integrals

$$\mathbb{E}[u(t,x)] = \int_{\mathbb{R}^p} u(t,x,q)\rho(q)dq$$

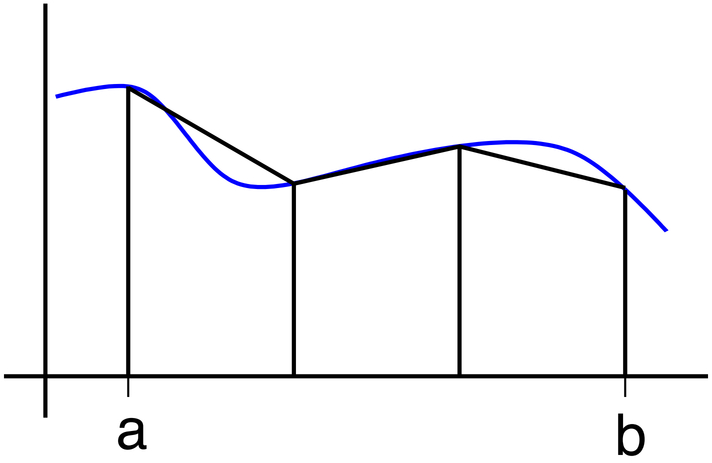
Numerical Quadrature:

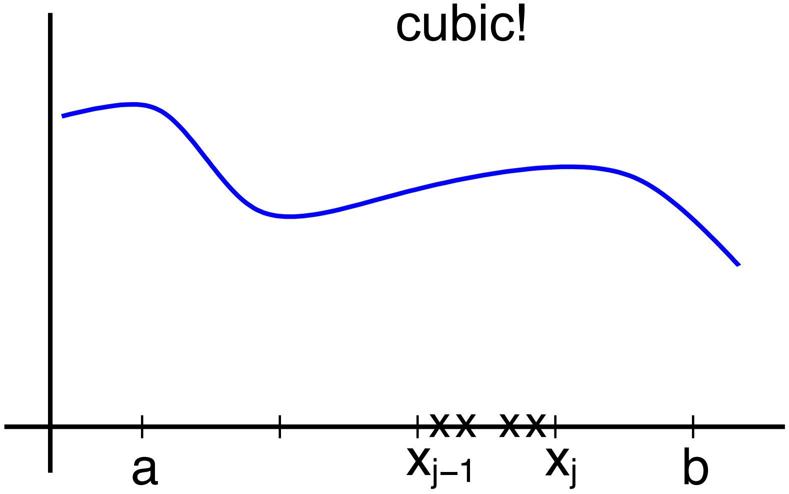
$$\int_{\mathbb{R}^p} f(q)\rho(q)dq \approx \sum_{r=1}^{R} f(q^r)w^r$$

Questions:

- How do we choose the quadrature points and weights?
  - E.g., Newton-Cotes, Gaussian algorithms

Error: Exact for polynomials up to cubic!





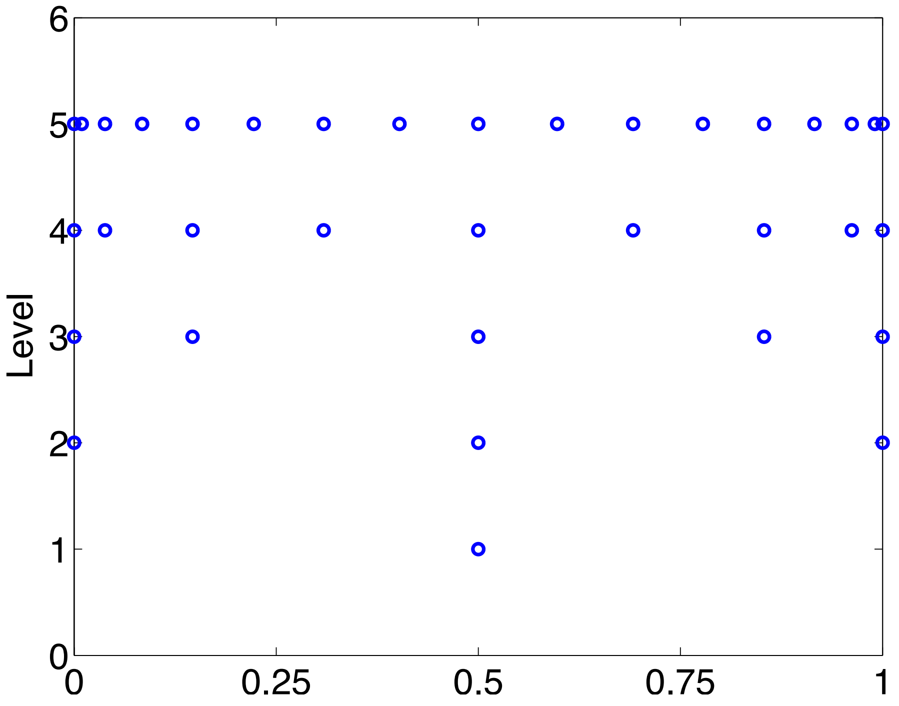
Numerical Quadrature

Numerical Quadrature:

$$\int_{\mathbb{R}^p} f(q)\rho(q)dq \approx \sum_{r=1}^{R} f(q^r)w^r$$

Questions:

- Can we construct nested algorithms to improve efficiency?
  - E.g., employ Clenshaw-Curtis points

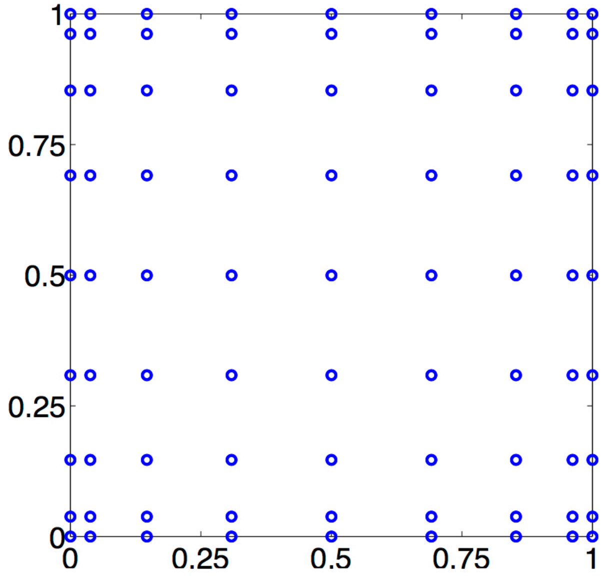


Numerical Quadrature

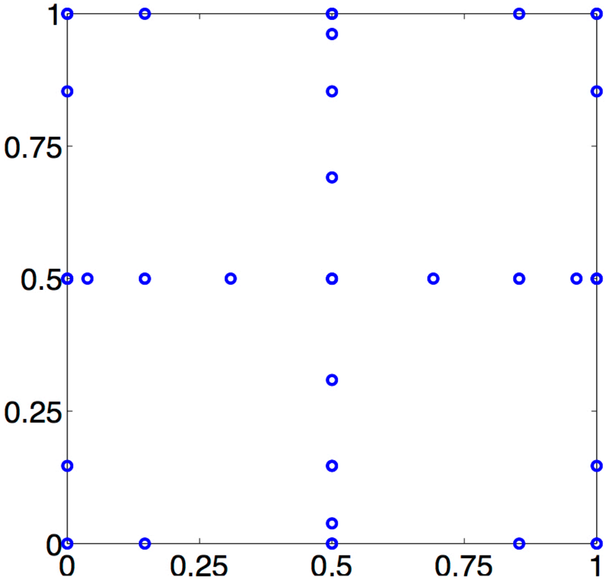
Questions:

- How do we reduce required number of points while maintaining accuracy?

Tensored Grids: Exponential growth



Sparse Grids: Same accuracy



p	R_ℓ	Sparse Grid $\mathcal{R}$	Tensored Grid $R = (R_\ell)^p$
2	9	29	81
5	9	241	59,049
10	9	1581	$> 3 \times 10^9$
50	9	171,901	$> 5 \times 10^{47}$
100	9	1,353,801	$> 2 \times 10^{95}$

Numerical Quadrature

Problem:

- Accuracy of methods diminishes as parameter dimension p increases
- Suppose $f \in C^{\alpha}([0,1]^p)$
- Tensor products: Take R_ℓ points in each dimension so $R = (R_\ell)^p$ total points
- Quadrature errors:

Newton-Cotes: $E \sim \mathcal{O}(R_\ell^{-\alpha}) = \mathcal{O}(R^{-\underline{\alpha/p}})$

Gaussian: $E \sim \mathcal{O}(e^{-\beta R_\ell}) = \mathcal{O}\left(e^{-\beta \underline{\sqrt[p]{R}}}\right)$

Sparse Grid: $E \sim \mathcal{O}\left(\mathcal{R}^{-\alpha} \log{(\mathcal{R})}^{\underline{(p-1)(\alpha+1)}}\right)$

Numerical Quadrature

Problem:

- Accuracy of methods diminishes as parameter dimension p increases
- Suppose $f \in C^\alpha([0,1]^p)$
- Tensor products: Take R_ℓ points in each dimension so $R = (R_\ell)^p$ total points
- Quadrature errors:

Newton-Cotes: $E \sim \mathcal{O}(R_\ell^{-\alpha}) = \mathcal{O}(R^{-\underline{\alpha/p}})$

Gaussian: $E \sim \mathcal{O}(e^{-\beta R_\ell}) = \mathcal{O}\left(e^{-\beta \underline{\sqrt[p]{R}}}\right)$

Sparse Grid: $E \sim \mathcal{O}\left(\mathcal{R}^{-\alpha} \log(\mathcal{R})^{\underline{(p-1)(\alpha+1)}}\right)$

- Alternative: Monte Carlo quadrature

$$\int_{\mathbb{R}^p} f(q)\rho(q)dq \approx \frac{1}{R}\sum_{r=1}^{R} f(q^r) \quad , \quad E \sim \left(\frac{1}{\sqrt{R}}\right)$$

- Advantage: Errors independent of dimension p
- Disadvantage: Convergence is very slow!

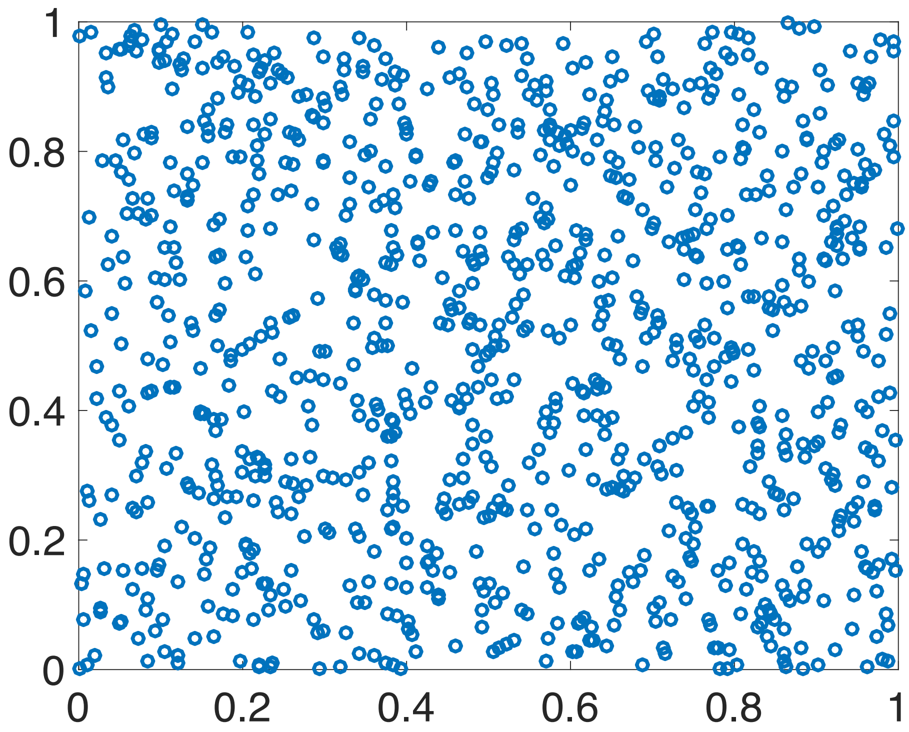
Conclusion: For high enough dimension p, monkeys throwing darts will beat Gaussian and sparse grid techniques!

Monte Carlo Sampling Techniques

Issues:

- Very low accuracy and slow convergence
- Random sampling may not "randomly" cover space …

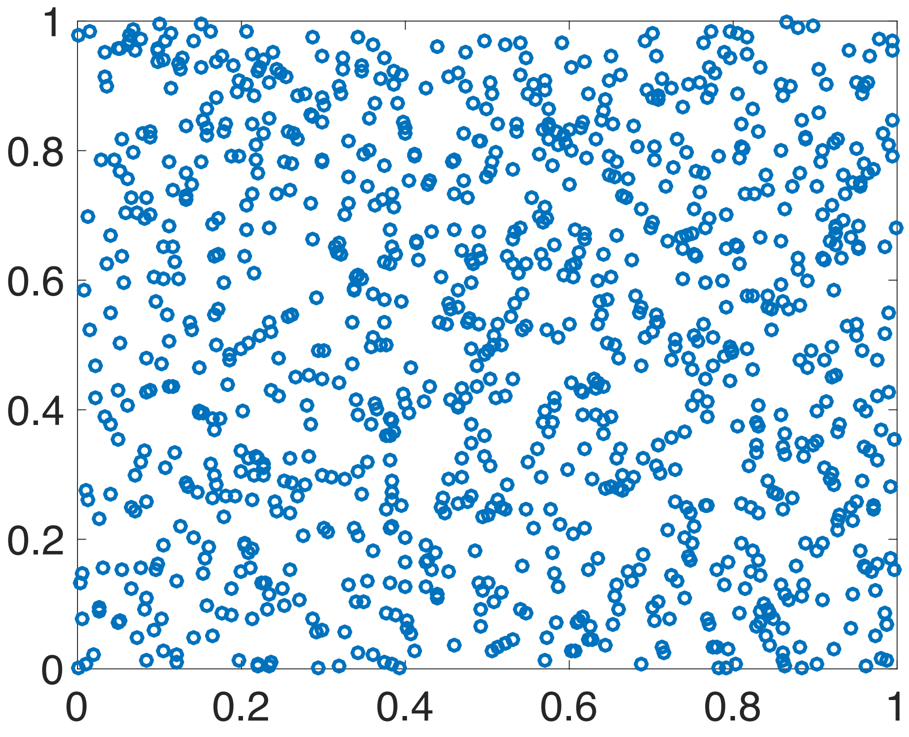
Samples from Uniform Distribution



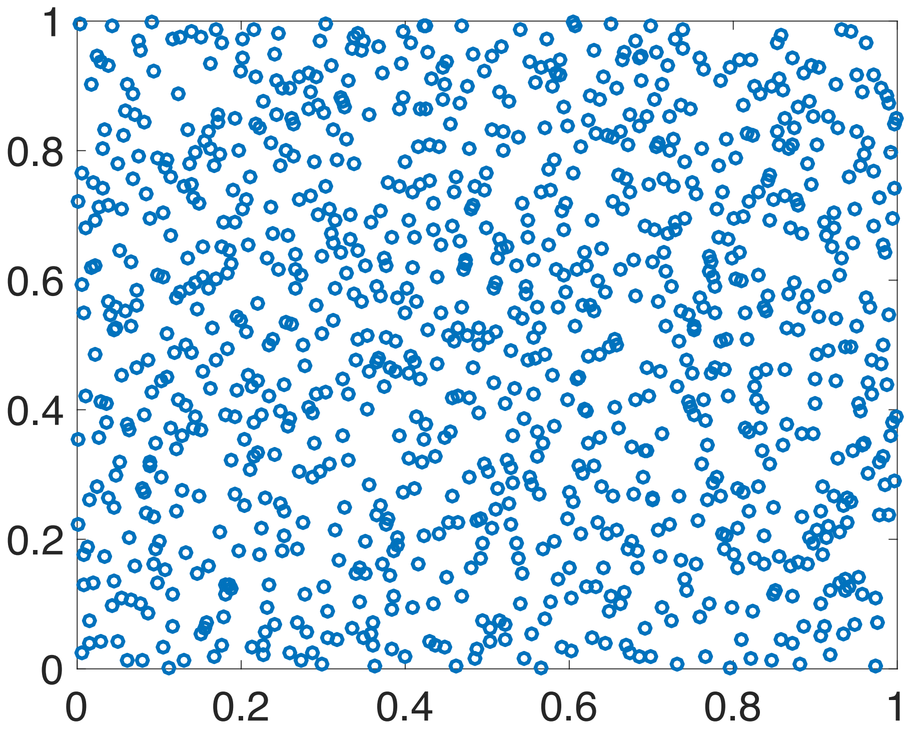
Monte Carlo Sampling Techniques

Issues:

- Very low accuracy and slow convergence
- Random sampling may not “randomly” cover space …



Samples from Uniform Distribution



Sobol’ Points

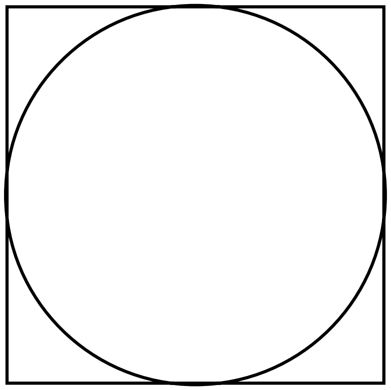
Sobol’ Sequence: Use a base of two to form successively finer uniform partitions of unit interval and reorder coordinates in each dimension.

Monte Carlo Sampling Techniques

Example: Use Monte Carlo sampling to approximate area of circle

$$\frac{A_c}{A_s} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

$$\Rightarrow A_c = \frac{\pi}{4} A_s$$



Strategy:

- Randomly sample N points in square $\Rightarrow$ approximately $N\frac{\pi}{4}$ in circle
- Count M points in circle

$$\Rightarrow \pi \approx \frac{4M}{N}$$

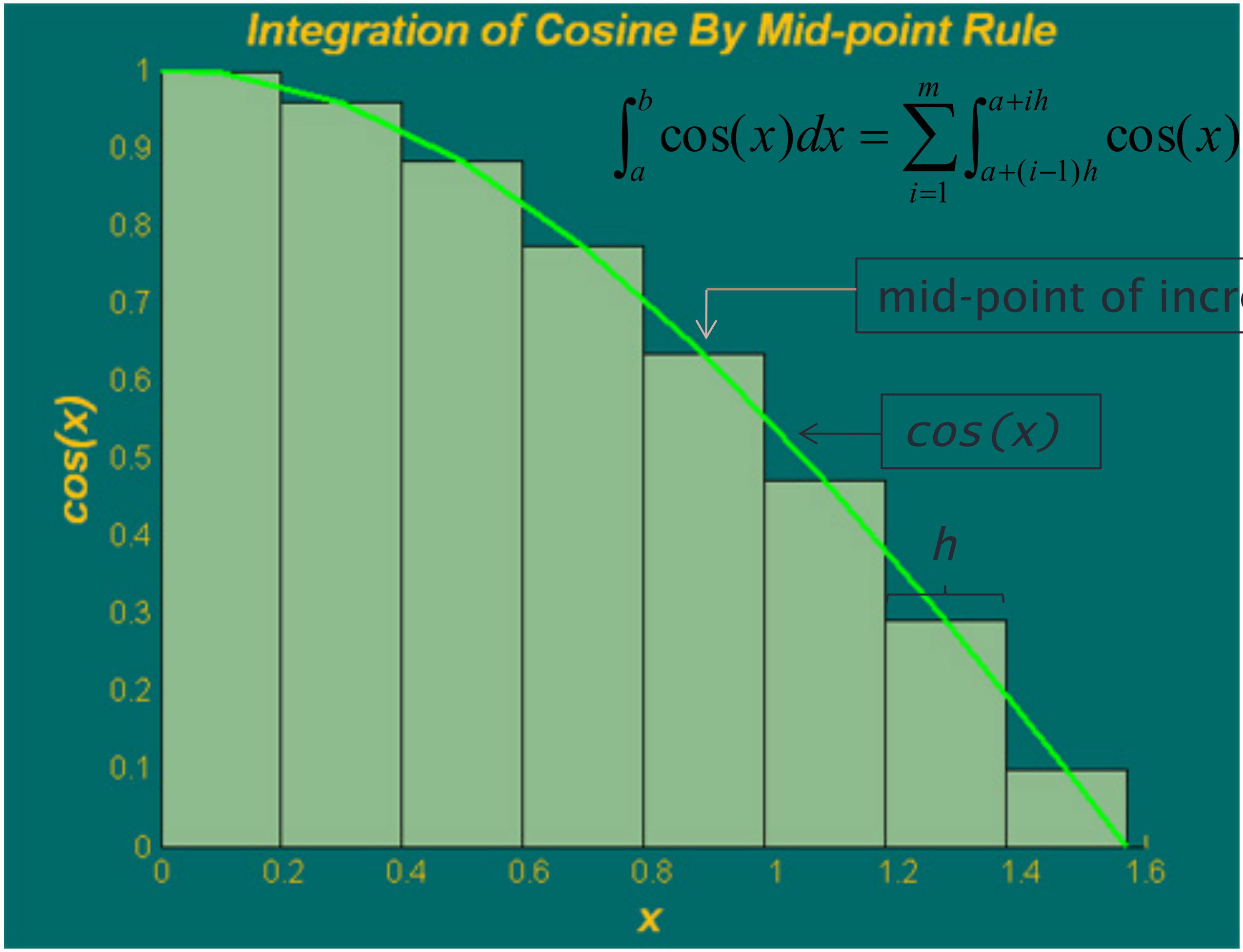
MATLAB Example

Monte Carlo Quadrature:

- Run rand_points.m to observe uniformly sampled and Sobol' points.
- Run pi_approx.m with different values of N to see if you observe convergence rate of $1/\sqrt{N}$

Numerical Integration: Example

- Integration of cosine from 0 to π/2.
- Use mid-point rule for simplicity.

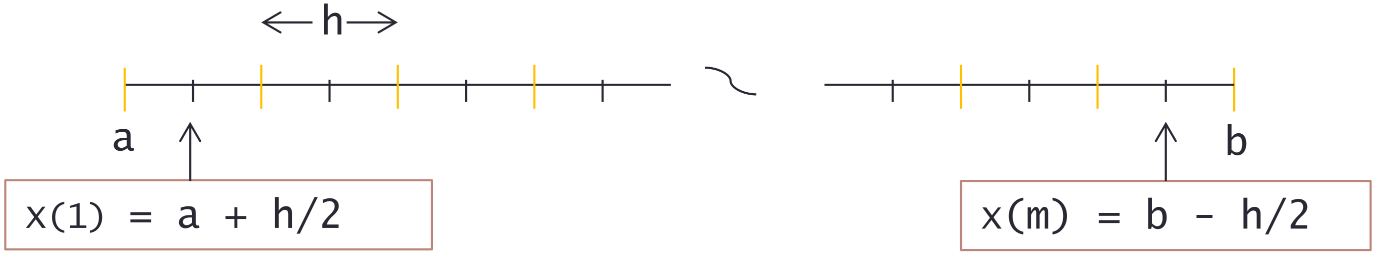


$$\int_a^b \cos(x)dx = \sum_{i=1}^{m} \int_{a+(i-1)h}^{a+ih} \cos(x)dx \approx \sum_{i=1}^{m} \cos(a+(i-\tfrac{1}{2})h)h$$

```
a = 0; b = pi/2;  % range
m = 8;  % # of increments
h = (b-a)/m;  % increment
```

Numerical Integration

```
% integration with for-loop
tic
   m = 100;
   a = 0;                  % lower limit of integration
   b = pi/2;               % upper limit of integration
   h = (b-a)/m;            % increment length
   integral = 0;           % initialize integral
   for i=1:m
     x = a+(i-0.5)*h;      % mid-point of increment i
     integral = integral + cos(x)*h;
   end
toc
```

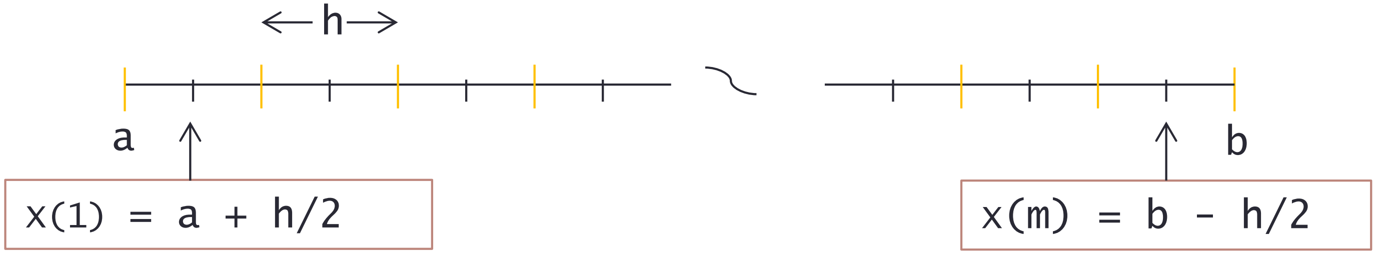


x(1) = a + h/2

x(m) = b - h/2

Numerical Integration

```
% integration with vector form
tic
   m = 100;
   a = 0;                  % lower limit of integration
   b = pi/2;               % upper limit of integration
   h = (b-a)/m;            % increment length
   x = a+h/2:h:b-h/2;      % mid-point of m increments
   integral = sum(cos(x))*h;
toc
```



Optimization

Example: Helmholtz energy $\psi(P,q) = \underline{\alpha_1} P^2 + \underline{\alpha_{11}} P^4 + \underline{\alpha_{111}} P^6$

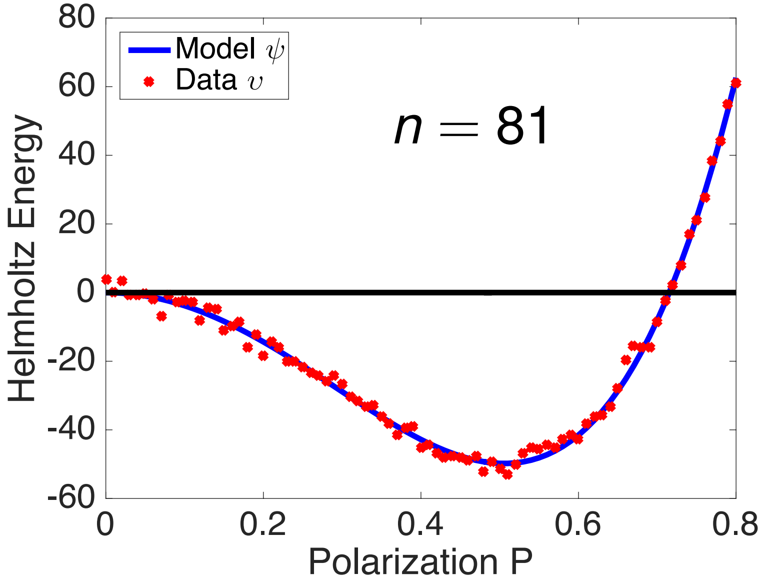
$$q = [\alpha_1, \alpha_{11}, \alpha_{111}]$$

Statistical Model: Describes observation process

$$\upsilon_i = \psi(P_i, q) + \varepsilon_i \quad , \; i = 1, ... , n$$

Point Estimates: Ordinary least squares

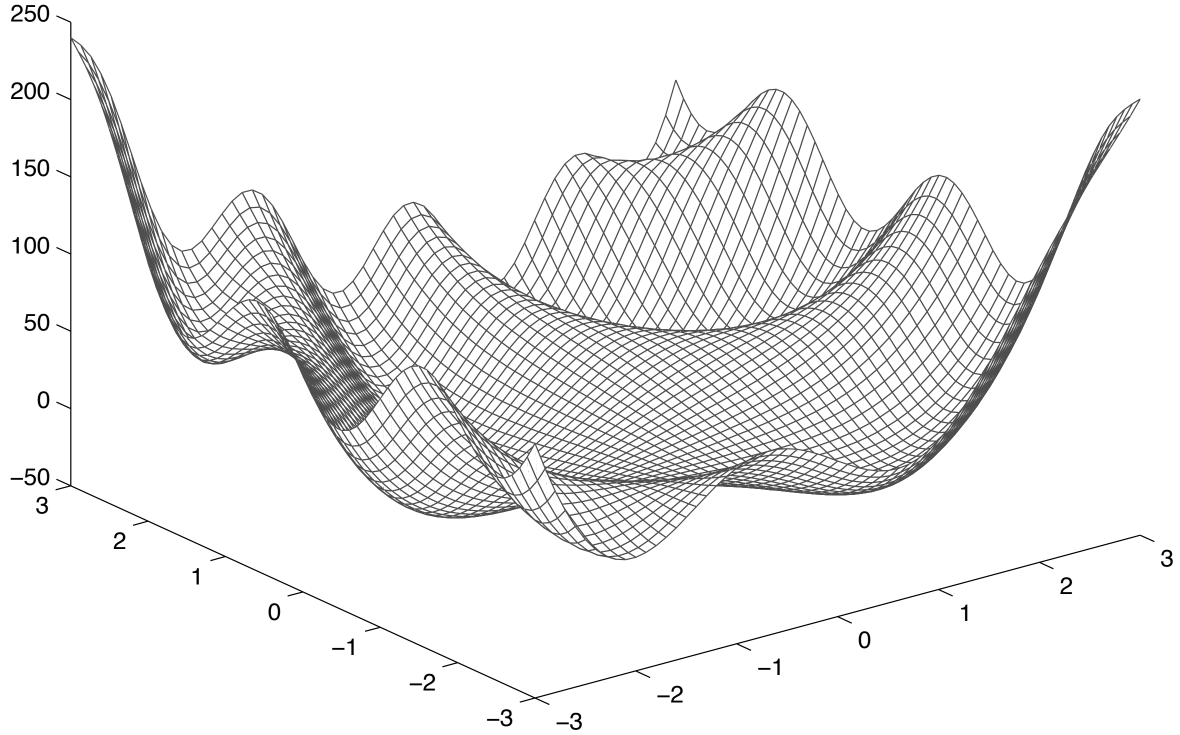
$$q^0 = \arg\min_q \frac{1}{2} \sum_{j=1}^{n} [\upsilon_i - \psi(P_i, q)]^2$$



Note: Optimization is critical for model calibration and design

Optimization

Issues: Nonconvex problems having numerous local minima



Tie between Optimization and Root Finding

Problem 1: minimize $f(x)$, $f : \mathbb{R}^n \to \mathbb{R}$

Problem 2: solve $F(x) = 0$ where $F : \mathbb{R}^n \to \mathbb{R}^n$

Note:

- If x^* solves (1), it also solves (2) with $F(x) = \nabla f(x)$
- If x^* solves (2), it solves (1) with $f(x) = \|F(x)\|^2 = F(x)^T F(x)$

Optimization

Method 1: Gradient descent

Goal: $\min_{x} f(x)$

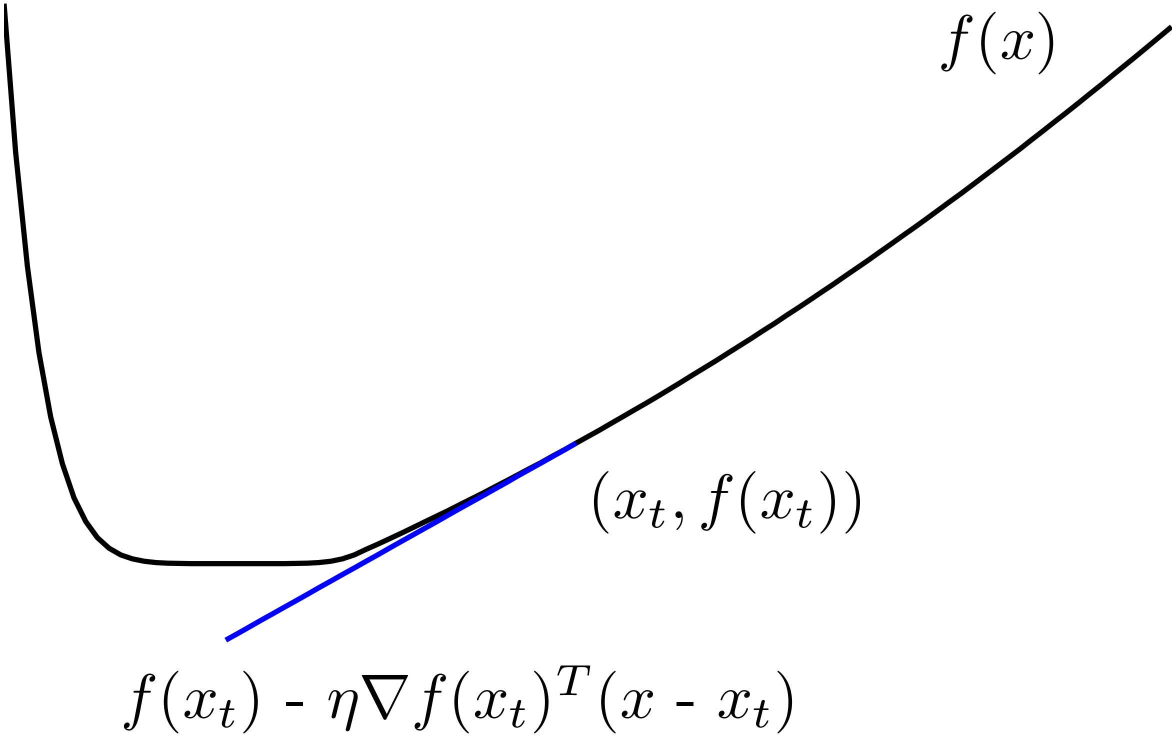
Strategy: Employ iteration

$$x_{t+1} = x_t - \eta_t \nabla f(x_t)$$

where η_t is a stepsize

Strategy: Employ iteration

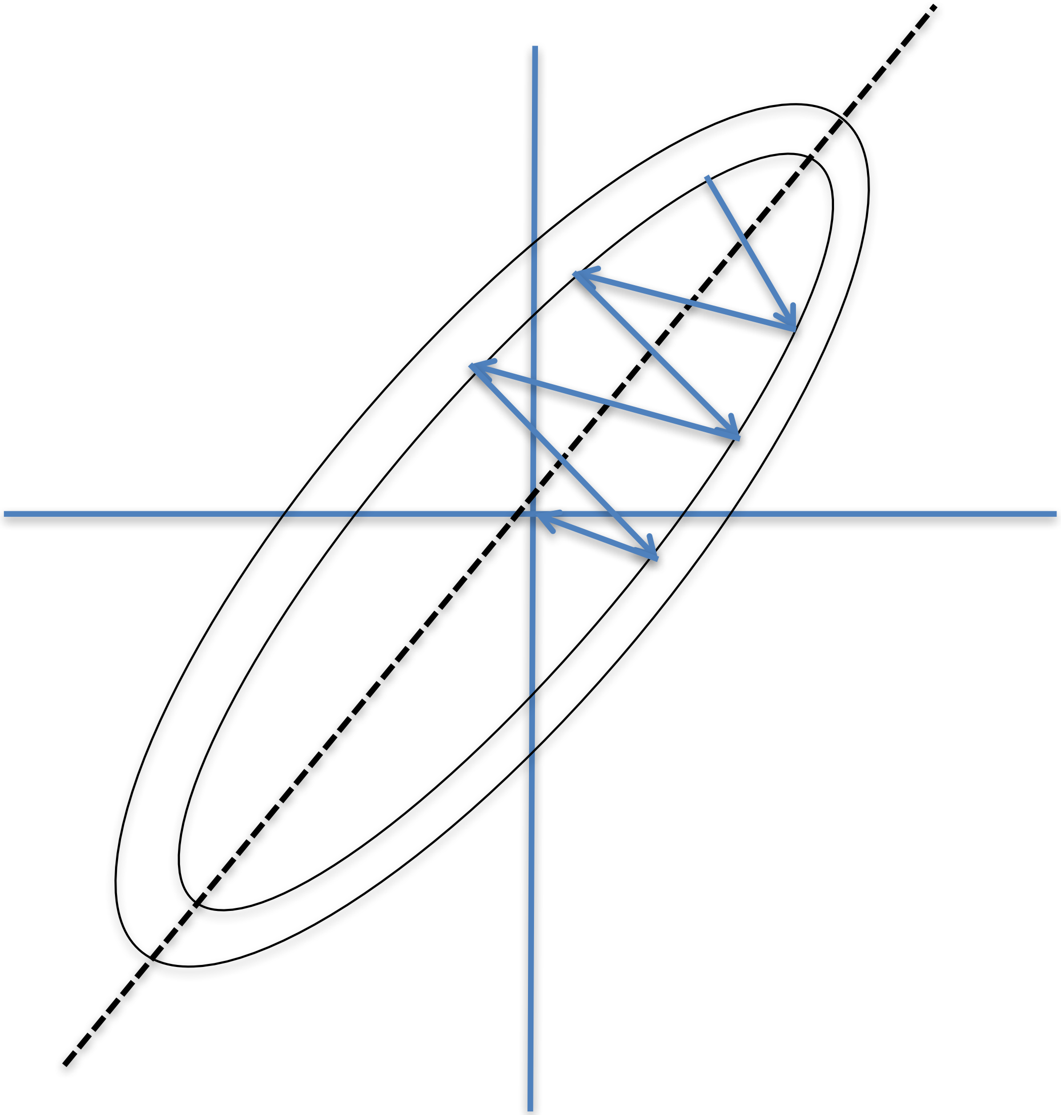
Note: Stochastic gradient descent employed in machine learning including artificial neural nets.



Optimization

Method 1: Gradient descent

Potential issue:



Newton’s Method

Problem 1: minimize $f(x)$, $f : \mathbb{R}^n \to \mathbb{R}$

Problem 2: solve $F(x) = 0$ where $F : \mathbb{R}^n \to \mathbb{R}^n$

Note:

- If x^* solves (1), it also solves (2) with $F(x) = \nabla f(x)$
- If x^* solves (2), it solves (1) with $f(x) = \|F(x)\|^2 = F(x)^T F(x)$

Newton’s Method (n=1): Let x_j approximate the root p with $F'(x_j) \neq 0$. Then

$$F(x) = F(x_j) + F'(x_j)(x - x_j) + \frac{(x - x_j)^2}{2} F''(\xi)$$

$$\Rightarrow 0 \approx F(x_j) + F'(x_j)(p - x_j)$$

$$\Rightarrow p \approx x_j - \frac{F(x_j)}{F'(x_j)}$$

Iteration: $x_{j+1} = x_j - \frac{F(x_j)}{F'(x_j)}$

Note: Quadratic convergence if function is sufficiently smooth and ‘reasonable’ initial value

Newton’s Method

Newton’s Method (n>1): Consider $F(x) = \nabla f(x) = 0$

Iteration: $x_{j+1} = x_j + s_j$ where s_j solves

$$F(x_j) + F'(x_j)s_j = 0$$

$$\Rightarrow x_{j+1} = x_j - H(x_j)^{-1}\nabla f(x_j)$$

Hessian:

$$F'(x) = H(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$$

Note: Hessian computation is expensive so several techniques to approximate its action; e.g., Limited-memory Broyden –Fletcher-Goldfarb-Shanno (L-BFGS) employed in machine learning.

MATLAB Optimization Routines

Note: There is significant documentation for the Optimization Toolbox

Minimization:

- fmincon: Constrained nonlinear minimization
- fminsearch: Unconstrained nonlinear minimization (Nelder-Mead)
- fminunc: Unconstrained nonlinear minimization (gradient-based trust region)
- quadprog: Quadratic programming

Equation Solving:

- fsolve: Nonlinear equation solving
- fzero: scalar nonlinear equation solving

Least Squares:

- lsqlin: Constrained linear least squares
- lsqnonlin: Nonlinear least squares
- lsqnonneg: Nonnegative linear least squares

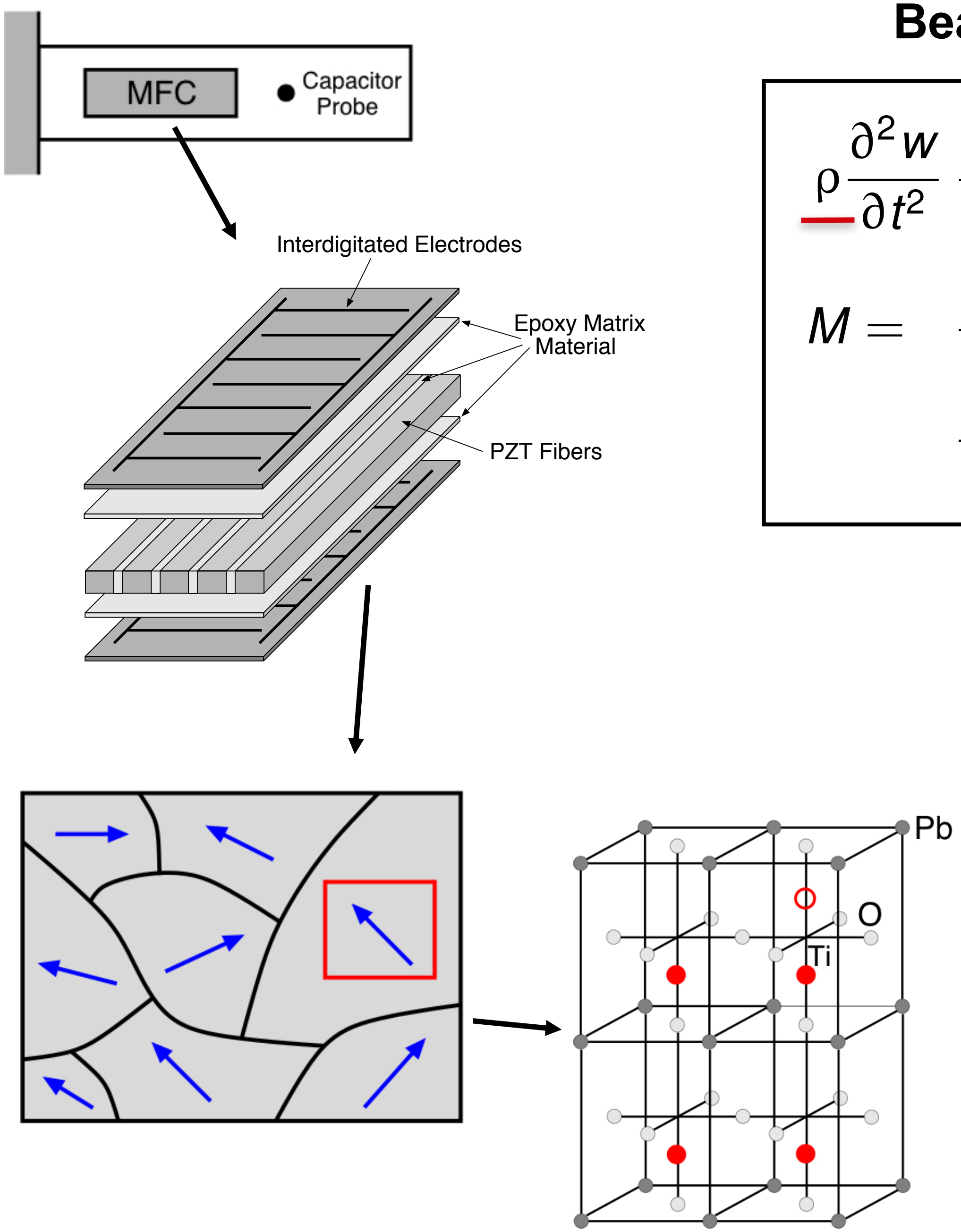
Kelley's Routines: Available at the webpage http://www4.ncsu.edu/~ctk/

MATLAB Example

Optimization:

- Run Helmholtz.m to learn about fminsearch.m

Numerical Differentiation and Differential Equations



Beam Model: 20 parameters

$$\underline{\rho}\frac{\partial^2 w}{\partial t^2} + \underline{\gamma}\frac{\partial w}{\partial t} - \frac{\partial^2 M}{\partial x^2} = 0$$

$$M = -\underline{c^E I}\frac{\partial^2 w}{\partial x^2} - \underline{c_D I}\frac{\partial^3 w}{\partial x^2 \partial t} - [\underline{k_1} e(E,\sigma_0)E + \underline{k_2}\varepsilon_{irr}(E,\sigma_0)]\chi_{MFC}(x)$$

UQ and Control Formulation:

$$\frac{dz}{dt} = f(t,z,q) + v_1(t)$$

$$y(t) = \int_{\mathbb{R}^{20}} w^N(t,\bar{x},q)\rho(q)dq$$

E.g., Average tip displacement

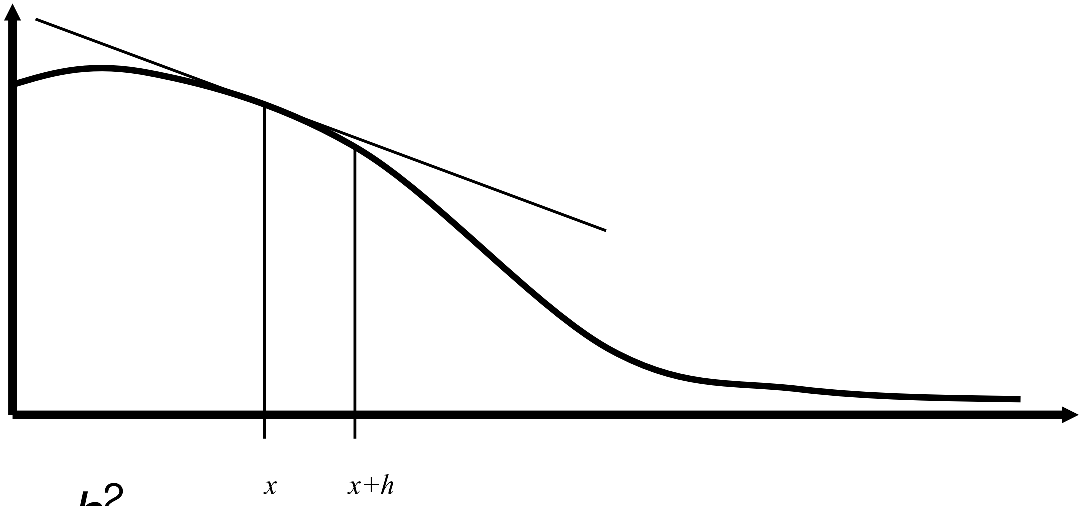
Numerical Issues:

- Accurately approximate derivatives and differential equations

Numerical Differentiation

Derivative:

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

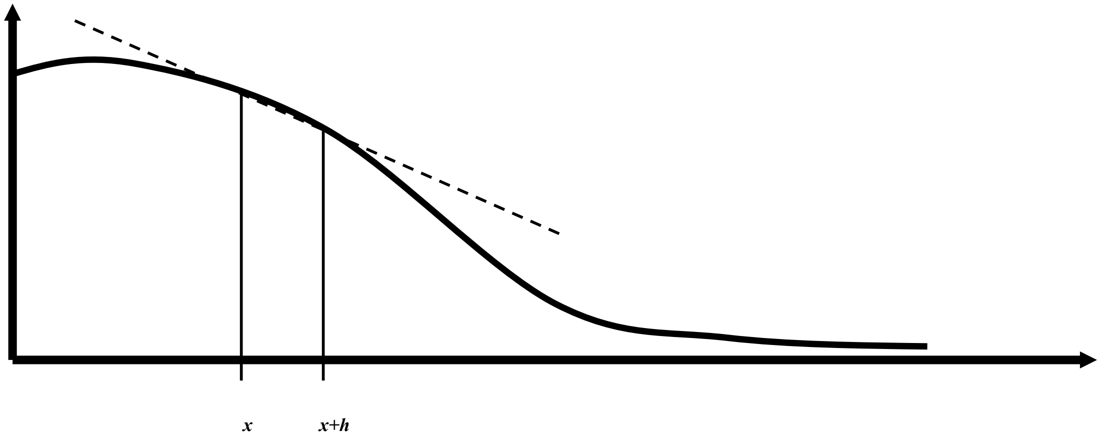


Note:

$$f(x+h) = f(x) + f'(x)h + f''(\xi)\frac{h^2}{2!}$$

$$\Rightarrow f'(x) = \frac{f(x+h)-f(x)}{h} - f''(\xi)\frac{h}{2!}$$

Forward Difference: $f'(x) = \dfrac{f(x+h)-f(x)}{h} + \mathcal{O}(h)$



Numerical Differentiation

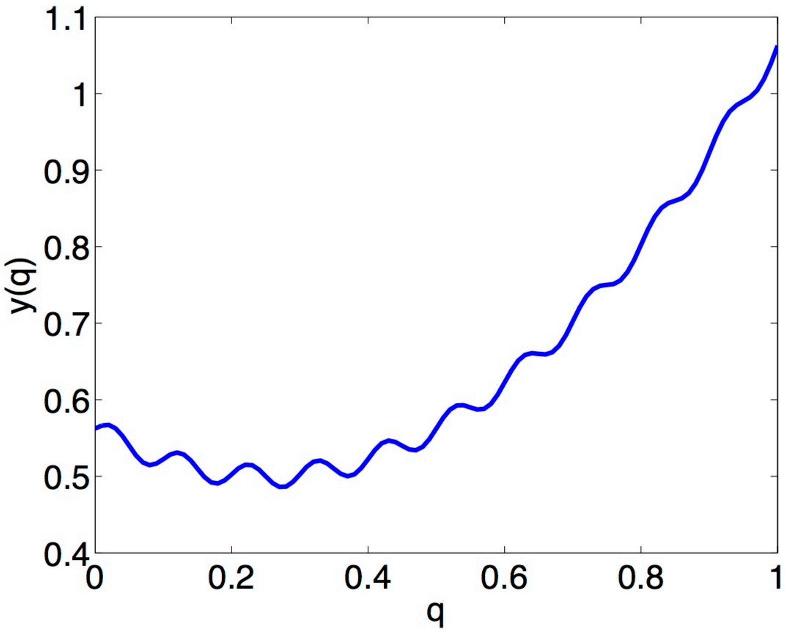
More Accuracy: Central differences

$$f'(x) = \frac{f(x+h) - f(x-h)}{h} + \mathcal{O}(h^2)$$

Numerical Differentiation

Issue: Suppose we have the following “noisy” function or data

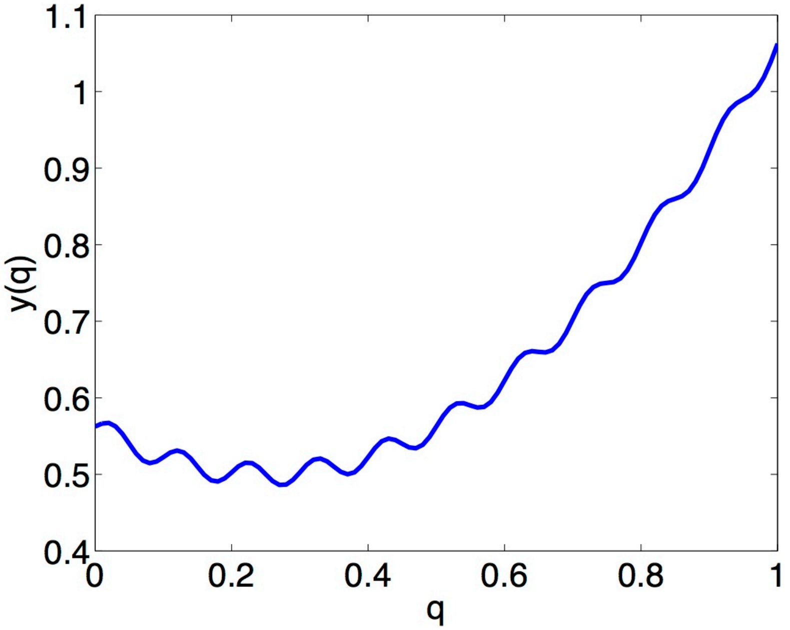
- What is the issue with doing finite-differences to approximate derivative?



Numerical Differentiation

Issue: Suppose we have the following “noisy” function or data

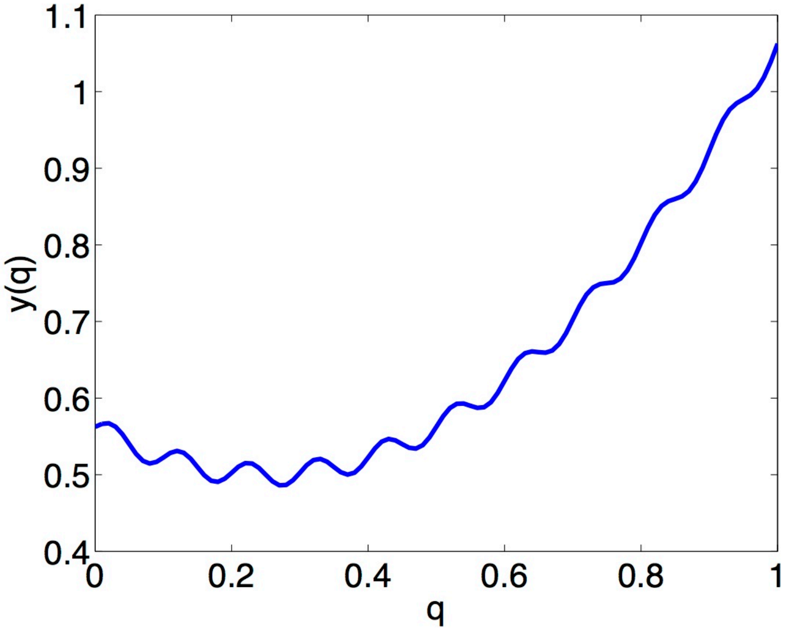
- What is the issue with doing finite-differences to approximate derivative?
- Derivatives can grow unboundedly due to noise.



Numerical Differentiation

Issue: Suppose we have the following “noisy” function or data

- What is the issue with doing finite-differences to approximate derivative?
- Derivatives can grow unboundedly due to noise.



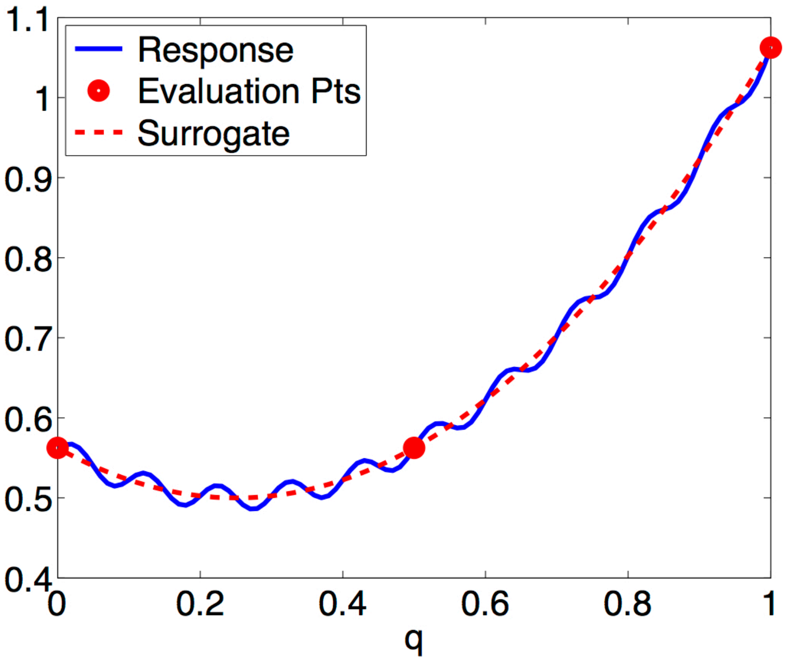
Example: Quadratic polynomial

$$y_s(q) = (q - 0.25)^2 + 0.5$$

Note: Solve linear system

Solution:

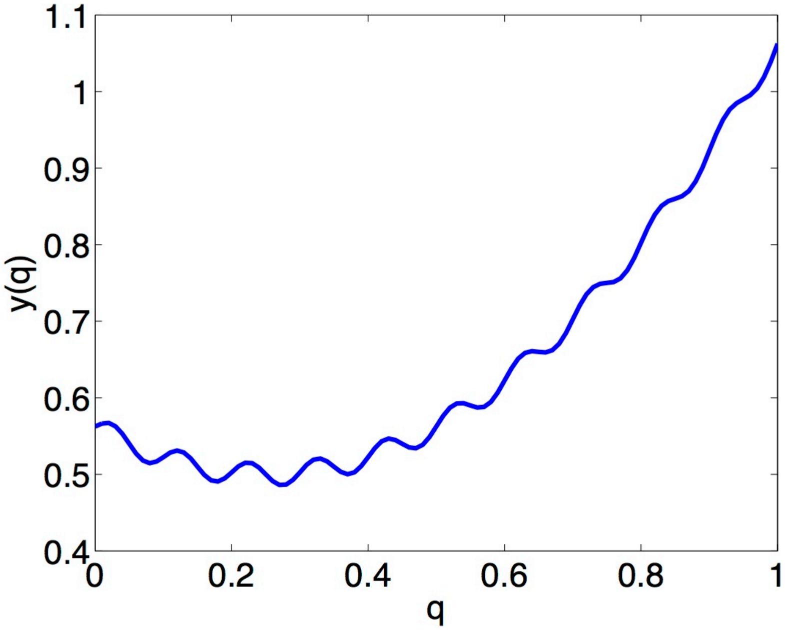
- Fit "smooth” function that is easy to differentiate.
- Interpolation



Numerical Differentiation

Issue: Suppose we have the following “noisy” function or data

- What is the issue with doing finite-differences to approximate derivative?
- Derivatives can grow unboundedly due to noise.

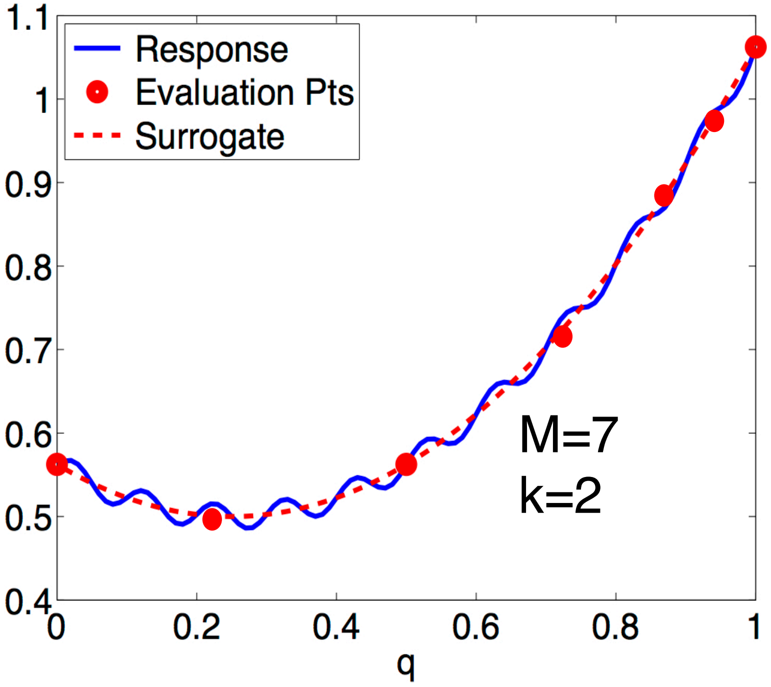


Example: Quadratic polynomial

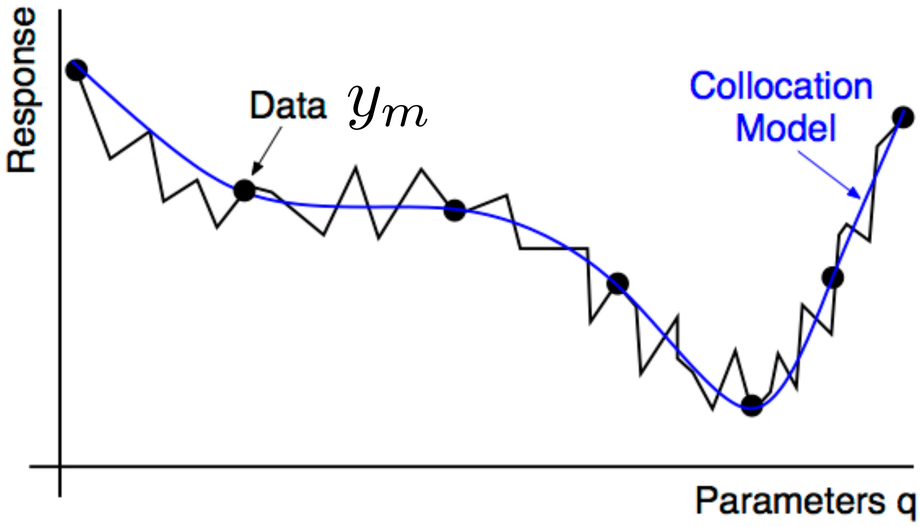
$$y_s(q) = (q - 0.25)^2 + 0.5$$

Solution:

- Fit "smooth” function that is easy to differentiate.
- Regression: See Statistics presentation



Lagrange Polynomials



Strategy: Consider high fidelity model

$$y = f(q)$$

with M model evaluations

$$y_m = f(q^m) \ , \ m = 1, \dots, M$$

Lagrange Polynomials:

$$Y^M(q) = \sum_{m=1}^{M} y_m L_m(q)$$

where $L_m(q)$ is a Lagrange polynomial, which in 1-D, is represented by

$$L_m(q) = \prod_{\substack{j=0 \\ j \neq m}}^{M} \frac{q - q^j}{q^m - q^j} = \frac{(q-q^1)\cdots(q-q^{m-1})(q-q^{m+1})\cdots(q-q^M)}{(q^m-q^1)\cdots(q^m-q^{m-1})(q^m-q^{m+1})\cdots(q^m-q^M)}$$

Note:

$$L_m(q^j) = \delta_{jm} = \begin{cases} 0 & , \quad j \neq m \\ 1 & , \quad j = m \end{cases}$$

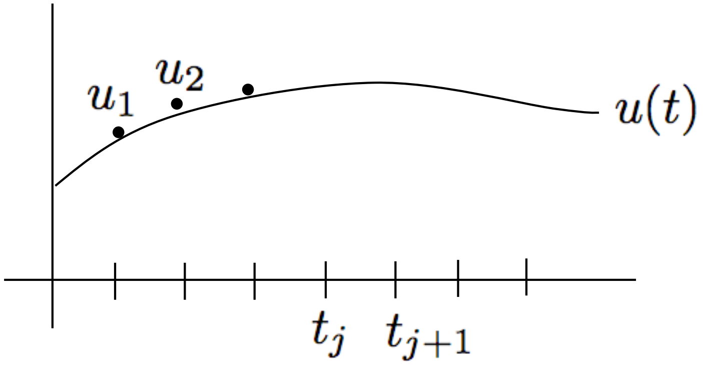
Result: $Y^M(q^m) = y_m$

Numerical Methods for IVP: Euler's Method

Initial Value Problem:

$$\frac{du}{dt} = f(t,u)\ ,\ t \geq 0$$

$$u(0) = u_0$$



Notation: $t_j = jk$ for $j = 0, 1, \cdots$

$u_j \approx u(t_j)$

Taylor Series:

$$u(t_{j+1}) = u(t_j) + k\frac{du}{dt}(t_j) + \frac{k^2}{2}\frac{d^2u}{dt^2}(t_j) + \cdots + \frac{k^n}{n!}\frac{d^nu}{dt^n}(t_j) + \frac{k^{n+1}}{(n+1)!}\frac{d^{n+1}u}{dt^{n+1}}(\xi)$$

Euler's Method:

$$u(t_{j+1}) = u(t_j) + k\dot{u}(t_j) + \mathcal{O}(k^2)$$

$$\Rightarrow u_{j+1} = u_j + kf(t_j, u_j)$$

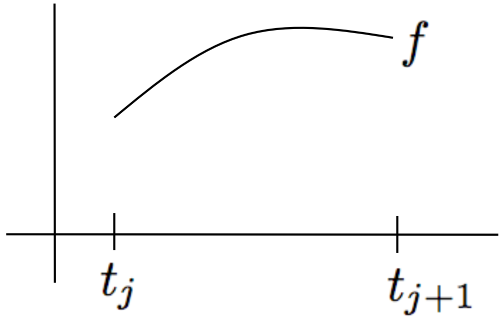
Accuracy: Local truncation error $\mathcal{O}(k^2)$

Global truncation error $\mathcal{O}(k)$

Euler and Implicit Euler Methods

Note:

$$u(t_{j+1}) = u(t_j) + \int_{t_j}^{t_{j+1}} f(t, u(t))dt$$



Euler's Method: Left Endpoint

$$u_{j+1} = u_j + kf(t_j, u_j)$$

Implicit Euler: Right Endpoint

$$u_{j+1} = u_j + kf(t_{j+1}, u_{j+1})$$

Stability: Apply method to

$$\dot{u}(t) = \lambda u \ , \ \ \lambda < 0$$

$$u(0) = u_0$$

Forward Euler

$$\begin{aligned} u_{j+1} &= u_j + \lambda k u_j \\ &= (1 + \lambda k)^{j+1} u_0 \end{aligned}$$

$$\Rightarrow |1 + \lambda k| < 1$$

$$\Rightarrow k < \frac{-2}{\lambda}$$

Implicit Euler

$$\begin{aligned} u_{j+1} &= u_j + \lambda u_{j+1} \\ &= \left(\frac{1}{1 - \lambda k} \right)^{j+1} u_0 \end{aligned}$$

$$\Rightarrow 1 < |1 - \lambda k|$$

$$\Rightarrow k > 0$$

Runge-Kutta-Feylberg Methods

4th Order Runge-Kutta:

$$k_1 = kf(t_j, u_j)$$

$$k_2 = kf\left(t_j + \frac{k}{2}, u_j + \frac{k_1}{2}\right)$$

$$k_3 = kf\left(t_j + \frac{k}{2}, u_j + \frac{k_2}{2}\right)$$

$$k_4 = kf(t_{j+1}, u_j + k_3)$$

$$u_{j+1} = u_j + \frac{1}{6}(k_1 + k_2 + k_3 + k_4)$$

Accuracy: Local Truncation error is 4th-order if u(t) has five continuous derivatives.

Runge-Kutta-Feylberg: Use R-K method with 5th order truncation error to estimate local error in 4th order R-K method to choose appropriate stepsize.

MATLAB ODE Routines

Algorithms: From the MATLAB ODE documentation

- **ode45** is based on an explicit Runge-Kutta (4,5) formula, the Dormand-Prince pair. It is a one-step solver - in computing y(tn), it needs only the solution at the immediately preceding time point, y(tn-1). In general, ode45 is the best function to apply as a "first try" for most problems.

- **ode23** is an implementation of an explicit Runge-Kutta (2,3) pair of Bogacki and Shampine. It may be more efficient than ode45 at crude tolerances and in the presence of moderate stiffness. Like ode45, ode23 is a one-step solver.

- **ode113** is a variable order Adams-Bashforth-Moulton PECE solver. It may be more efficient than ode45 at stringent tolerances and when the ODE file function is particularly expensive to evaluate. ode113 is a multistep solver - it normally needs the solutions at several preceding time points to compute the current solution.

- The above algorithms are intended to solve nonstiff systems. If they appear to be unduly slow, try using one of the stiff solvers below.

- **ode15s** is a variable order solver based on the numerical differentiation formulas (NDFs). Optionally, it uses the backward differentiation formulas (BDFs, also known as Gear's method) that are usually less efficient. Like ode113, ode15s is a multistep solver. Try ode15s when ode45 fails, or is very inefficient, and you suspect that the problem is stiff, or when solving a differential-algebraic problem.

- **ode23s** is based on a modified Rosenbrock formula of order 2. Because it is a one-step solver, it may be more efficient than ode15s at crude tolerances. It can solve some kinds of stiff problems for which ode15s is not effective.

- **ode23t** is an implementation of the trapezoidal rule using a "free" interpolant. Use this solver if the problem is only moderately stiff and you need a solution without numerical damping. ode23t can solve DAEs.

- **ode23tb** is an implementation of TR-BDF2, an implicit Runge-Kutta formula with a first stage that is a trapezoidal rule step and a second stage that is a backward differentiation formula of order two. By construction, the same iteration matrix is used in evaluating both stages. Like ode23s, this solver may be more efficient than ode15s at crude tolerances.

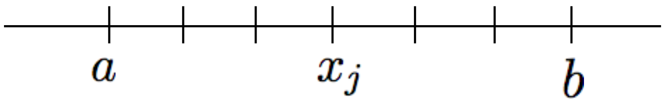
MATLAB ODE Routines: From the Documentation

Solver	Problem Type	Order of Accuracy	When to Use
ode45	Nonstiff	Medium	Most of the time. This should be the first solver you try.
ode23	Nonstiff	Low	For problems with crude error tolerances or for solving moderately stiff problems.
ode113	Nonstiff	Low to High	For problems with stringent error tolerances or for solving computationally intensive problems.
ode15s	Stiff	Low to Medium	If ode45 is slow because the problem is stiff
ode23s	Stiff	Low	If using crude error tolerances to solve stiff systems and the mass matrix is constant.
ode23t	Moderately Stiff	Low	For moderately stiff problems if you need a solution without numerical damping.
ode23tb	Stiff	Low	If using crude error tolerances to solve stiff systems.

Numerical Methods for BVP: Finite Differences

Problem:

$$y'' = p(x)y' + q(x)y + f(x)\ ,\ \ a \le x \le b$$
$$y(a) = \alpha\ ,\ \ y(b) = \beta$$



Grid: $x_j = a + jh\ ,\ \ h = (b-a)/(N+1)$

Note: N interior grid points

Centered Difference Formulas: (From Taylor expansions)

$$y''(x_j) = \frac{1}{h^2}\left[y(x_{j+1}) - 2y(x_j) + y(x_{j-1})\right] - \frac{h^2}{24}y^{(4)}(\xi_j)$$
$$y'(x_j) = \frac{1}{2h}\left[y(x_{j+1}) - y(x_{j-1})\right] - \frac{h^2}{6}y'''(\eta_j)$$

System:

$$\begin{aligned}\frac{y(x_{j+1}) - 2y(x_j) + y(x_{j-1})}{h^2} &= p(x_j)\left[\frac{y(x_{j+1}) - y(x_{j-1})}{2h}\right] + q(x_j)y(x_j)\\ &\quad + f(x_j) - \frac{h^2}{12}\left[2p(x_j)y'''(\eta_j) - y^{(4)}(\xi_j)\right]\end{aligned}$$

Finite Difference Method for BVP

Finite Difference System: Define $y_0 = \alpha\ ,\ y_{N+1} = \beta$ and consider

$$\left(\frac{2y_j - y_{j+1} - y_{j-1}}{h^2}\right) + p(x_j)\left(\frac{y_{j+1} - y_{j-1}}{2h}\right) + q(x_j)y_j = -f(x_j)$$

$$\Rightarrow -\left(1 + \frac{h}{2}p(x_j)\right) y_{j-1} + (2 + h^2 q(x_j))y_j - \left(1 - \frac{h}{2}p(x_j)\right) y_{j+1} = -h^2 f(x_j)$$

for $j = 1, 2, \cdots, N$

Matrix System:

$$\begin{bmatrix} 2 + h^2 q(x_1) & -1 + \frac{h}{2}p(x_1) & 0 & \\ -1 - \frac{h}{2}p(x_2) & 2 + h^2 q(x_2) & -1 + \frac{h}{2}p(x_2) & \\ & & & \\ & \ddots & \ddots & \ddots \\ & & & -1 + \frac{h}{2}p(x_{N-1}) \\ & 0 & -1 - \frac{h}{2}p(x_N) & 2 + h^2 q(x_N) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{bmatrix} = \begin{bmatrix} -h^2 f(x_1) + \left(1 + \frac{h}{2}p(x_1)\right)\alpha \\ -h^2 f(x_2) \\ \vdots \\ -h^2 f(x_{N-1}) \\ -h^2 f(x_N) + \left(1 - \frac{h}{2}p(x_N)\right)\beta \end{bmatrix}$$