Scientific Computation

Spring, 2019

Lecture 11

Imperial College
London

Notes

- HW2: Posted online today ~7pm, due by end of next Friday (1/3)
- HW1 solutions will be posted on Friday, feedback, early next week
- Today's office hour is in 6M20
- Anonymous online feedback: https://goo.gl/forms/K5YEXbxZFTbUdfda2

Overview of time-marching methods

- **Discussed: implicit/explicit, fixed time-step and variable time-step methods**

- **Fixed time-step methods:**
  - **Accuracy characterized by truncation error**
  - **Also considered stability (model equation,** $dy/dt = ay$, $\mathrm{Real}(a) \le 0$)

- **Variable time-step methods**
  - **Time step automatically reduced to reach error**
  - ***Stiffness* rather than stability is main concern**

Overview of time-marching methods

- **Discussed: implicit/explicit, fixed time-step and variable time-step methods**

- **Fixed time-step methods:**
  - **Accuracy characterized by truncation error**
  - **Also considered stability (model equation,** $dy/dt = ay$, $Real(a)<=0$)

❑ **Explicit Euler: truncation error ~** Δt, ~~***unconditionally unstable***~~ **for model eqn.**

❑ **RK4: truncation error ~** Δt^4, ***conditionally stable*** **for model eqn.**

❑ **Implicit Euler: truncation error ~** Δt, ***unconditionally stable*** **for model eqn.**

Explicit Euler: truncation error ~ Δt, ***conditionally stable*** **for model eqn.**
Unconditionally unstable if $Real(a)=0$

Overview of time-marching methods

- **Discussed: implicit/explicit, fixed time-step and variable time-step methods**

- **Variable time-step methods**
  - **Time step automatically reduced to reach error**
  - ***Stiffness* rather than stability is main concern**

❑ **Explicit variable time-step (e.g. RK45): Good for linear dynamics (single time scale)**

❑ **Implicit variable time-step (e.g. odeint, BDF): Good for *stiff* systems with multiple time scales – problems with strong nonlinearity**

Overview of time-marching methods

- **Discussed: implicit/explicit, fixed time-step and variable time-step methods**

- **Variable time-step methods**
  - **Time step automatically reduced to reach error**
  - ***Stiffness* rather than stability is main concern**

❑ **Explicit variable time-step (e.g. RK45): Good for linear dynamics (single time scale)**

❑ **Implicit variable time-step (e.g. odeint, BDF): Good for *stiff* systems with multiple time scales – problems with strong nonlinearity**

- **How do these models select their time steps?**
  - **By computing the solution** y_{i+1} **from** y_i **with 2 (or more) different time steps, they can estimate the *error* for** y_{i+1}, E_{i+1}
  - **Then, this error is compared to tolerance for the *absolute* and *relative* errors:** ε_{abs} **and** ε_{rel}
  - **The solution is accepted if** $E_{i+1} < |y_{i+1}| \, \varepsilon_{rel} + \varepsilon_{abs}$

Overview of time-marching methods

- **Discussed: implicit/explicit, fixed time-step and variable time-step methods**

- **Fixed time-step methods:**
  - **Accuracy characterized by truncation error**
  - **Also considered stability (model equation,** dy/dt = ay, Real(a)<=0**)**

- **Variable time-step methods**
  - **Time step automatically reduced to reach error**
  - ***Stiffness* rather than stability is main concern**

- **How do you choose one method vs. another?**
  - **Generally: Fastest method that provides a desired level of accuracy**
    - **Desired accuracy is context-dependent**
    - **The best we can hope for with double-precision arithmetic is ~1e-13**
  - **My (idiosyncratic) approach:**
    - **Use** odeint **(or something similar) for systems of ODEs, and single PDEs**
    - **Use something like RK4 for systems of PDEs**

Data science

- No particularly standard definition

- Combination of analysis, processing, and machine learning
  - Analysis and processing – “shape” or simplify data to facilitate understanding of underlying dynamics and trends

  - Learning – train models to classify data and/or predict trends

  - Often requires use of optimization methods (or ideas)

- Method 1: Principal Component Analysis (PCA)

- But first, let’s take a detour through an optimization problem

Matrix operators

- **How can we interpret:** $Ax = b \qquad x \in \mathbb{R}^N,\ b \in \mathbb{R}^M$
  - **If** A **and** b **are known: system of equations**
  - **If** A **and** x **are known: transformation of** x **to** b
- **Let's think about the general case where *neither*** x **nor** b **are known**

Matrix operators

- **How can we interpret:** $Ax = b \qquad x \in \mathbb{R}^N,\ b \in \mathbb{R}^M$
  - **If** A **and** b **are known: system of equations**
  - **If** A **and** x **are known: transformation of** x **to** b
- **Let's think about the general case where *neither*** x **nor** b **are known**

$\boxed{\text{Given a matrix A, find x so that |b| is maximized}}$

Application: system of linear ODEs: $\frac{dx}{dt} = Mx$

$$x(t) = A(t)x_0,\ A = exp(Mt)$$

(here, *exp* is the *matrix exponential*)

More generally, a broad range of dynamical process can be stated (sometimes approximately) in an iterative form: $x_{i+1} = Ax_i$

And the optimization problem aims to find the maximum possible "growth"

Maximum growth

- We need to be more careful with our problem statement:

Given a matrix A**, find** x **so that** $|b|$ **is maximized with** $|x|=1$

1. We first set an upper bound for the growth

2. Then, we see if we can reach that upper bound

Maximum growth

- **We need to be more careful with our problem statement:**

Given a matrix A, find x so that |b| is maximized with |x|=1

1. **We first set an upper bound for the growth**

2. **Then, we see if we can reach that upper bound**

Task 1:

We have: $Ax = b$ **which implies,** $x^T A^T A x = b^T b$

Note that $A^T A$ **is symmetric and thus can be orthogonally diagonalized:**

$$A^T A = V S V^T$$

Here, V is the orthogonal eigenvector matrix and S is a diagonal matrix with the eigenvalues of $A^T A$ on the diagonal.

Maximum growth

Task 1:

We have: $Ax = b$ **which implies,** $x^T A^T A x = b^T b$ **(1)**

Note that $A^T A$ **is symmetric and thus can be orthogonally diagonalized:**

$$A^T A = V S V^T$$

Here, V **is the orthogonal eigenvector matrix and** S **is a diagonal matrix with the eigenvalues of** $A^T A$ **on the diagonal.**

- **Note that the eigenvalues of** $A^T A$ **are real and non-negative, and we construct** S **so that the eigenvalues appear in descending order,**

$$S_{i,i} = \lambda_i, \;\; \lambda_1 \geq \lambda_2 ... \geq \lambda_N$$

- **So (1) can be re-written as:** $x^T V S V^T x = b^T b$
- **Now, if we take** x = V z**, we have:** $z^T S z = b^T b$

Maximum growth

Task 1:

- **So (1) can be re-written as:** $x^T V S V^T x = b^T b$
- **Now, if we take** x = V z**, we have:** $z^T S z = b^T b$ **which can be rearrange**

as: $b^T b = \lambda_1 z_1^T z_1 + \lambda_2 z_2^T z_2 + \ldots + \lambda_N z_N^T z_N$ **and** z_i **is the** i^{th} **element of** z

Then: $b^T b \leq \lambda_1 \left(z_1^T z_1 + z_2^T z_2 + \ldots + z_N^T z_N \right)$

$$b^T b \leq \lambda_1 z^T z$$

$$x^T x = z^T z$$

$$b^T b \leq \lambda_1 x^T x$$

Task 2: Find x such that $x^T x = 1$ and $b^T b = \lambda_1$