An Engineering-oriented Calculus Core

Daniel Kaplan

Florida Poly 02-15-2022

Math at Florida Poly

- Required by all majors
  - MAC 2311/12 - Analytic Geometry and Calculus 1/2 (Stewart)
- Required by many majors
  - MAC 2313 - Analytic Geometry and Calculus 3 (Stewart)
  - Differential Equations (NA)
- Required by some majors
  - Statistics
    - OpenIntro
    - Walpole
  - Linear algebra
    - Lay
    - Larson

A Compact Core

- Taken by all students
  - Something new for students with previous calculus
  - Shared experience
- Accessible to students without previous calculus
- Comprehensive—topics that all downstream courses can build on
  - Avoid pre-mature branching. ("Can I manage MAC 2313?" "Do I want to commit a semester to it?")
- Short enough to complete in first year.

Accessible yet engaging

How to make calculus more comprehensive yet accessible.

- Purposefully select theoretical topics and calculation techniques.
- Calculus is about describing change and relationships.
  - Infinitesimal and infinite play supporting roles, not the main characters.
- Notation and type:
  - Make it possible to determine from the symbol what kind of thing is being referred to.
  - Have a close computer-language equivalent for names and operations.

Accessible but engaging (cont.)

- **Avoid gratuitous algebra.**
  - Very little drill on differentiation and anti-differentiation of functions made up for the purpose of drill.
  - Use small set of "basic modeling functions."
- Emphasize functions of two variables.
  - No special notation for functions of one variable. (Such special notation leads to bad habits, even if it might simplify things in the short run.)

Calculus theory: Functions, not equations

- Functions take *inputs* and produce an *output*.
- The output is computed by an algorithm.
- Formulas are one form of algorithm, but there are others.
- All the calculus operations can be applied easily to any form of function.
- The word "variable" is reserved for data and statistics,

Not $y = x^2$. Instead, $f(x) \equiv x^2$ or `f <- makeFun(x^2 ~ x)`.

x^2 is an *expression* not a function.

Calculus theory: Approximation

- Approximation, not exactitude
  - low-order polynomial approximation; constant, linear, interaction, quadratic
  - Taylor polynomials (not "series") are one form, but there are others.
  - "Euler method"
- Differentiation and anti-differentiation are about relationships, not process.
  - Teach differentiation and anti-differentiation hand in hand for pattern-book functions
  - Don't pull a rabbit (fundamental theorem of calculus) out of a hat (area).

Calculus theory: Confirmation not derivation.

Example: is finite-difference derivative numerically stable as dt gets smaller?

Calculus techniques

The computational techniques used in applications of calculus

- Small, enumerated set of calculus operations: "something you do to a function."
  i. function evaluation,
  ii. differentiation,
  iii. anti-differentiation,
  iv. solving,
  v. argmax,
  vi. iteration.
- All have computational implementation.
- Most have graphical implementation.

Avoiding gratuitous algebra.

- Use almost exclusively a small set of "basic modeling functions."
- Everyone: Symbolic derivatives and anti-derivatives of "basic modeling functions" on first sight.
- Allow computing instead of algebra.
- Allow graphics instead of algebra.

Principles of notation

i. By looking at a bit of notation, you should be able to say what kind of thing it refers to.
ii. Strive to have one way of saying something and to be consistent.
iii. Notation for all functions should support functions of multiple inputs.
iv. Make **definition** explicit.
v. Strive to be compatible with computer notation and parallel it if possible. Example: hardly any computer language will let you name something `f'` or `df/dx`.
vi. Avoid ambiguous terms, even if they are traditional.

Examples of notation

Functions: A name followed by a parenthesis. So $\sin(x)$ not $\sin x$. Can use sin() to name the function. - exception: exponentials are often written with superscripts rather than parentheses: e^{kt}, x^n.

Function inputs: From back of alphabet: u, v, w, x, y, z. (These are *inputs*, not "variables." Let's leave "variables" for statistics.)

Function evaluation: $f(x = 4, t = 3)$

Function definition: - $f(x) \equiv xe^{kx}$ - `f <- makeFun(x * exp(k*x) ~ x)` - NEVER $y = 2x^2$

Examples of notation (cont.)

Parameters: Following tradition, e.g. k, P, A, ω,

Coefficients: Mostly from front of alphabet. Numerical subscripts.

Specific values of inputs: e.g. - x_0 - $x^\star$ for argmax. (But maybe I don't need to allow superscripts.)

Vectors and matrices are decorated with harpoons: $\overrightharpoon{\mathbf{u}}$ and $\overleftrightharpoons{\mathbf{A}}$.

Derivatives:

- $\partial_t f(t)$.
- `dt_f <- D(f(t) ~ t`

Small set of core modeling functions

"Pattern-book functions" don't have parameters but do have English names. All have a single input.

- one()
- identity()
- recip()
- Exponential: e^x
- Logarithm: ln()
- Power-law: x^p
- Sinusoid: sin()
- Gaussian: dnorm()
- Sigmoid: pnorm()

Students should be able to sketch any pattern-book functions on demand, identify gaps in domain (e.g. ln() and recip()), special input/output pairs, and horizontal and vertical asymptotes.

Basic modeling functions

Parameterized versions of pattern-book functions, e.g.,

- $\sin\left(\frac{2\pi}{P}t\right)$
- $\text{dnorm}(x, \text{mean}, \text{sd})$
- e^{kt}

Must also be able to write on sight symbolic derivative and anti-derivative of every pattern-book and basic-modeling function.

One exception for anti-differentiation: Don't need to know $\int \text{dnorm}(x)dx$.

Functions of with multiple inputs

Primarily constructed from single-input basic modeling functions by

- linear combination
- products

Graphics modalities

Be able to construct and interpret

- Graph of function of one input
- Contour plot for functions with two inputs
- Vector field
- Path
- Inequality constraint
- Point plot for data

Quantities vs numbers

The inputs to basic modeling functions are ***quantities*** and have a dimension and units. The outputs are also quantities.

Functions created by differentiation or anti-differentiation (typically) have different output dimension.

Linear algebra goals

1. Understand least squares in a way that supports statistical inference in a later course.
2. Provide concrete, accessible framework to practice modeling/interpretation skills.
3. Eigenvalues/vectors (in dynamics)

Matrices are collections of (column) vectors. A matrix times a vector is a linear combination of the column vectors.

Linear algebra topics

1. Vectors: length, angle, dot product
2. Linear combinations and subspaces
   - builds on earlier construction of multi-input functions and low-order polynomial approximation
   - Matrix is a collection of (column) vectors; defines a subspace.
3. Projection of a vector $\vec{\mathbf{b}}$ onto a subspace to produce $\widehat{\mathbf{b}}$
   - *confirm* that claimed solution has the right properties
4. "Target problem" — find a linear combination of vectors that reaches $\widehat{\mathbf{b}}$.
5. Introduction to "least squares" and R^2
6. Basis set for function decomposition: Fourier

Linear algebra topics not needed

- determinants: not on topic
- inverses/singular matrices: not needed since hardly any of our matrices will be square
- gaussian elimination or similar non-trivial mechanics

Dynamics/ODE topics

- First-order linear and nonlinear DEs
- Fixed points and stability
- Sets of first-order linear and nonlinear DEs
    - Nonlinear: to reinforce modeling concepts and low-order polynomial approximations.
    - Linear: Stability and oscillation
- Eigenvalues and eigenvectors of linear flows, complex exponentials.