ECE 204 *Numerical methods*: Course summary

Douglas Wilhelm Harder

April 2023

Issues with floating-point arithmetic

(i) it is not associative: $x + (y + z)$ need not equal $(x + y) + z$

(ii) $x + y = x$ if y is sufficiently small

(iii) $x - y$ loses precision if x is sufficiently close to y

To mitigate these issues with floating-point numbers:

- add numbers of the same sign from smallest to largest
- avoid adding large numbers onto smaller numbers if the significance of the smaller numbers matters
- avoid subtracting numbers of approximately equal value

1 The seven tools:

Here we list the seven tools that will be used throughout this course, and each time a tool is used, we will refer to it with the number in square parentheses; for example, Newton's method is derived using Taylor series [**5**] and uses iteration [**2**].

1. Weighted averages
   - A weighted average of n items $x_1, \ldots, x_n$ (scalars or vectors) is any linear combination of these $c_1x_1 + \cdots + c_nx_n$ where $c_1 + \cdots c_n = 1$.
   - A convex combination of n items is any weighted average where all coefficients $c_k \geq 0$ for all $k = 1, \ldots, n$. For real numbers, any convex combination of n real numbers must evaluate to a value between $\min\{x_1, \ldots, x_n\}$ and $\max\{x_1, \ldots, x_n\}$, inclusive.
   - An average of n items is a weighted average where all coefficients are equal: $c_k = \frac{1}{n}$ for $k = 1, \ldots, n$.
   - If n items $x_1, \ldots, x_n$ approximate the same value x, then a weighted average of those n items is not subject to issues with floating-point arithmetic, and that weighted average continues to approximate x.

2. Iteration

   - The fixed-point theorem says that if we are solving $x = f(x)$ and we start with an x_0 and define $x_{k+1} \leftarrow f(x_k)$, if this sequence converges, it converges to a solution of $x = f(x)$.
   - Approximating $\sqrt{(2)}$ requires an average [**1**].
   - Estimating a and b given samples from a uniform distribution sampled from $[a, b]$ used a weighted average [**1**] with some coefficients being negative.

3. Linear algebra

   - Using partial pivoting, we avoid adding a large multiple of one equation (row) onto equation, thus losing information about the second.
   - We introduce the **Jacobi method** which rewrites $A\mathbf{x} = \mathbf{b}$ in the form $\mathbf{x} = f(\mathbf{x})$ and then we iterate [**2**].

4. Interpolation

   - We find interpolating polynomials using linear algebra [**3**]. Given n distinct x-values, there is a unique polynomial of degree $n - 1$ that passes through n points $(x_1, y_1), \ldots, (x_n, y_n)$.
   - We introduce shifting and scaling that will be used to mitigate issues with floating-point arithmetic. Shifting is necessary if the x-values are large, and scaling applies best if the x-values are equally spaced.

5. Taylor series

   - We replace the representation in first year

     $$f(x) = f(x_0) + f^{(1)}(x_0)(x - x_0) + \frac{1}{2}f^{(2)}(x_0)(x - x_0)^2 + \cdots$$

     with the representation

     $$f(x + h) = f(x) + f^{(1)}(x)h + \frac{1}{2}f^{(2)}(x)h^2 + \cdots$$

6. Bracketing

   - We introduce bracketing by showing it is a numerical equivalent of one algorithm already covered, the **binary search**, and under appropriate conditions, this can be modified to the **interpolation search** (using interpolation [**4**]) makes convergence even faster.

7. Intermediate-value theorem

   - In first year, the intermediate value theorem says that if f is continuous and y lies between $f(a)$ and (b) inclusive, then there is a point x between a and b, inclusive, such that $y = f(x)$.

- We observe that if $x_1 \leq \ldots \leq x_n$ are n points between a and b, inclusive, and f is a continuous function then any convex combination if $f(x_1), \ldots, f(x_n)$ must be a value that lies between $\min\{f(x_1), \ldots, f(x_n)\}$ and $\max\{f(x_1), \ldots, f(x_n)\}$, and thus, there must be an $x_1 \leq x \leq x_n$ such that $f(x)$ equals this convex combination, and this x also satisfies $a \leq x \leq b$.

2 Numerical algorithms and analysis

We will apply these tools to approximate solutions to four categories of problems:

A. evaluating an expression,

B. approximating a solution to an algebraic equation or system of algebraic equations,

C. approximating a solution to an analytic equation or a system of analytic equations, and

D. unconstrained optimization.

These four categories are described here:

A. **Evaluating an expression** We begin by evaluating an expression to a numeric value. This includes evaluating a polynomial at a point, evaluating derivatives and integrals, and evaluating in the presence of noise.

   a. We begin by introducing Horner's rule for evaluating polynomials. We can mitigate the effects of numeric arithmetic if the variable is small and coefficients of the higher terms decreases.

   b. We continue by estimating a value between given points in either space
   $$\ldots, (x_{k-1}, f_{k-1}), (x_k, f_k), (x_{k+1}, f_{k+1}), \ldots$$
   or time
   $$\ldots, (t_{k-2}, y_{k-2}), (t_{k-1}, y_{k-1}), (t_k, y_k).$$

      (a) Assuming the points in space x_k are equally spaced and we have values of the functions at points on both side of the interval on which we'd like to approximate the function, shift and scale the surrounding two points to $-\frac{1}{2}, \frac{1}{2}$, or shift and scale the surrounding four points to $-\frac{3}{2}, -\frac{1}{2}, \frac{1}{2}, \frac{3}{2}$, and use interpolating linear or cubic polynomial **[4]**, respectively.

      (b) Assuming the points in time t_k are equally spaced and we have only the most recent point to the right of the interval on which we'd like to approximate the function, shift and scale the most recent three points to $-\frac{3}{2}, -\frac{1}{2}, \frac{1}{2}$, or shift and scale the most recent four points to $-\frac{5}{2}, -\frac{3}{2}, -\frac{1}{2}, \frac{1}{2}$, and use interpolating linear or cubic polynomial **[4]**, respectively.

(c) To approximate the derivative, we could use rise-over run on two points, but the error is $O(h)$ where h is the width of the interval (the *run*).

c. **Divided-difference formulas:** We continue by estimating derivatives at given points.

(a) Assuming the points in space x_k are equally spaced and we have values to the left and right of the point we'd like to find $O(h^2)$ approximations of the first and second derivatives (so we have values at $x_k - h, x_k, x_k + h$), find the interpolating quadratic polynomial [**4**], differentiate it once or twice, and then evaluate that polynomial at x_k to get the approximation. Error analysis is done with Taylor series [**5**] and the intermediate-value theorem [**7**] because the errors can be written as convex combinations.

(b) Assuming the points in time t_k are equally spaced and we'd like to find an $O(h^2)$ approximations of the first and second derivatives, find an interpolating quadratic polynomial [**4**] through the last three points, differentiate it and evaluate it at t_k for the derivative, or find an interpolating cubic polynomial [**4**] through the last four points, differentiate it twice and evaluate it at t_k for the second derivative. Error analysis is done with Taylor series [**5**].

d. We finish by estimating integrals between given points.

(a) For any integral, $\int_a^b f(x)\mathrm{d}x = \overline{f}_{[a,b]}(b-a)$ where $\overline{f}_{[a,b]}$ is the average value of f on $[a, b]$, so we will approximate the average value of the function by evaluating the function at various points and taking a weighted average [**1**] of those values; however, the actual formulas will be found by taking interpolating polynomials [**4**] between a number of points and integrating that interpolating polynomial.

(b) Assuming the points in space x_k are equally spaced and we have values to the left and right of the interval we'd like to approximate the integral from $x_k - h$ to x_k, we find an interpolating linear polynomial [**4**] through the points $x_k - h$ and x_k or an interpolating cubic polynomial [**4**] through the points $x_k - 2h, x_k - h, x_k, x_k + h$ and integrate that from $x_k - h$ to x_k.

(c) Alternatively, assuming we want to integrate from $x_k - h$ to $x_k + h$, we derive Simpson's rule by finding the interpolating quadratic polynomial [**4**] through the points $x_k - h, x_k, x_k + h$ and integrate that from $x_k - h$ to $x_k + h$.

(d) Assuming the points in time t_k are equally spaced and we'd like to approximate the integral from $t_k - h$ to t_k, find an interpolating quadratic polynomial [**4**] through the last three points, or find an interpolating cubic polynomial [**4**] through the last four points, and integrate that polynomial from $t_k - h$ to t_k.

e. In the presence of noise, it is inappropriate to use interpolating polynomials, so instead we use least-squares best-fitting polynomials.
   (a) Recall that to find the best approximation of a target N-dimensional vector $\mathbf{y}$ given n linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ where $n < N$, let $V = (\mathbf{v}_1 \cdots \mathbf{v}_n)$, solve $V^{\mathrm{T}} V\mathbf{a} = V^{\mathrm{T}}\mathbf{y}$ with linear algebra **[3]** to get $\mathbf{y} \approx a_1\mathbf{v}_1 + \cdots + a_n\mathbf{v}_n$.
   (b) First shift and scale the m most recent points to $-m+1, \ldots, -1, 0$, find the least-squares best-fitting linear polynomial $at + b$ or quadratic polynomial $at^2 + bt + c$ and then evaluate, differentiate or integrate that polynomial to get the necessary result.

B. **Approximating a solution to an algebraic equation or system of algebraic equations** We continue approximating a solution to an algebraic equation or a system of algebraic equations. For linear equations, we use linear algebra **[4]**, and for all other non-linear equations, we convert the problem into a root-finding problem, either $f(x) = 0$ or $\mathbf{f}(\mathbf{x}) = \vec{\mathbf{0}}$.
   a. It is trivial to solve a linear equation $ax = b$: $x = -\frac{b}{a}$.
   b. To approximate a root of a non-linear function, we may proceed as follows:
      (a) **Newton's method:** If x approximates a root, use the Taylor series **[5]** to find a better approximation of the root. We then iterate **[2]**. We deduce the error with the Taylor series.
      (b) **Bisection method:** If f is continuous and a and b are such that $f(a)$ and $f(b)$ have opposite signs, the intermediate-value theorem **[7]** says that there is a root on $[a, b]$, so this interval brackets a root **[6]**. Select the midpoint m and update whichever end-point has the same sign as $f(m)$ to continue bracketing the root. We then iterate **[2]**.
      (c) **Bracketed secant method:** Like the bisection method, we brackets the root **[6]**, but then we use an interpolating linear polynomial **[4]** between $(a, f(a))$ and $(b, f(b))$, and find the root m of that linear polynomial, and update whichever end-point has the same sign as $f(m)$ to continue bracketing the root. We then iterate **[2]**. The formula for the root m is not subject to subtractive cancellation.
      (d) **Secant method:** Starting with two approximations to a root, we find the next approximation by finding the interpolating linear polynomial **[4]** between the two. The root of this becomes the next approximation, and we discard that point x_k such that $|f(x_k)|$ is largest. We then iterate **[2]**. Unlike the bracketed secant method, the formula for the root is subject to subtractive cancellation.
      (e) **Muller's method:** Starting with three approximations to a root x_0, x_1 and x_2 (the last being the best), we shift and find the interpolating quadratic polynomial **[4]** between $(x_0 - x_2, f(x_0))$,

$(x_1 - x_2, f(x_1))$ and $(0, f(x_2))$ and we find the root of this interpolating polynomial using $\frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$ to mitigate the effects of floating-point arithmetic. We then iterate [**2**].

(f) **Inverse quadratic interpolating:** Starting with three approximations to a root x_0, x_1 and x_2 (the last being the best), we find the interpolating quadratic polynomial [**4**] between $(f(x_0), x_0)$, $(f(x_1), x_1)$ and $(f(x_2), x_2)$ and we find the constant coefficient of this interpolating polynomial. If we are close to a root, all the abscissa values are already small, so no shifting is required. We then iterate [**2**].

c. We have already seen how to mitigate the effects of floating-point arithmetic for solving a system of linear equations [**3**] using partial pivoting. We also saw the Jacobi method. We then introduce two additional techniques:

(a) **Gauss-Seidel method:** Given $\mathbf{x}_k$, instead of calculating $\mathbf{x}_{k+1}$, instead we assign $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$ and then update the entries one at a time.

(b) **Successive over-relaxation:** Having found $\mathbf{x}_{k+1}$ using $\mathbf{x}_k$, we use as the next approximation a weighted average [**1**] of these two approximations $\mathbf{x}_{k+1} \leftarrow \omega \mathbf{x}_{k+1} + (1 - \omega)\mathbf{x}_k$.

d. We approximate a solution to finding a root of a system of nonlinear expressions by converting it into a system of linear equations and finding the simultaneous root using linear algebra [**3**].

(a) For Newton's method, given an approximation $\mathbf{x}_0$, we find the tangent plane/hyper-plane using the Jacobian $J(\mathbf{f})(\mathbf{x}_0)$ and solve $J(\mathbf{f})(\mathbf{x}_0)\Delta\mathbf{x}_0 = -\mathbf{f}(\mathbf{x}_0)$ and then $\mathbf{x}_1 \leftarrow \mathbf{x}_0 + \Delta\mathbf{x}_0$ We then iterate [**2**]. Like Newton's method for a real-valued function of a real variable, this formula is found using Taylor series [**5**].

C. **Approximating a solution to an analytic equation or a system of analytic equations:** For ordinary-differential equations, We look at both initial-value problems and boundary-value problems. For partial differential equations, we look at the heat and wave equations, and Laplace's equation for a steady-state solution.

a. Given a 1^{st}-order initial-value problem $y^{(1)}(t) = f(t, y(t))$, and $y(t_0) = y_0$, we want to find $y(t_0 + h) = y_0 + \int_{t_0}^{t_0+h} y^{(1)}(t)\mathrm{d}t$, so as before, we rewrite this as $y(t_0 + h) = y_0 + h\overline{y^{(1)}}_{[t_0, t_0+h]}$, and then iterate [**2**]. Like integration, all formulas will use weighted averages [**1**] of samples of $y^{(1)}(t)$ to estimate this average value. The error analysis of the composite rules is done with the intermediate-value theorem [**7**].

(a) **Euler's method:** Approximate this value by $f(t_0, y_0)$. This is a consequence of Taylor series [**5**] and the error analysis also comes from this.

(b) **Heun's method:** Approximate this value by the average [**1**] of $s_0 \leftarrow (t_0, y_0)$ and $s_1 \leftarrow f(t_0 + h, y_0 + hs_0)$.

(c) 4^{th}**-order Runge-Kutta method:** Approximate this value by a weighted average [**1**] of $s_0 \leftarrow (t_0, y_0)$ and $s_1 \leftarrow f(t_0 + 0.5h, y_0 + 0.5/_2 hs_0)$, $s_2 \leftarrow f(t_0 + 0.5h, y_0 + 0.5/_2 hs_1)$ and $s_3 \leftarrow f(t_0 + 0.5h, y_0 + 0.5/_2 hs_2)$.

(d) Adaptive techniques use Taylor series [**4**] to allow one to approximate the error of a worse approximation using a better approximation. We discuss the Euler-Heun method and the Dormand-Prince method.

(e) To find an approximation to a system of initial-value problems, we simply use vector arithmetic.

(f) We use calculus to convert a higher-order initial value problem or a system of higher-order initial-value problems into a system of 1^{st}-order initial-value problems.

b. Given a 2^{nd}-order ordinary differential equation, we may have two boundary values, which may either specified values (Dirichlet) or slopes (Neumann).

(a) **Shooting method:** With Dirichlet conditions, we can convert the boundary-value problem into an initial-value problem with initial values $u(a) = u_a$ and $u^{(1)}(a) = s$, and then let $u_s(x)$ be the approximation of the solution with the initial slope s. We want $u_s(b) = u_b$ so we define $u_s(b) - u_b$ to be an expression in s and we want to find a root of this expression. We then start with two initial slopes s_0 and s_1 and we proceed by using the **secant method**.

(b) **Finite-difference method:** Approximate a linear ordinary-differential equation by a finite-difference equation by substituting the centered **divided-difference formulas** for the first and second derivatives. By dividing the interval $[a, b]$ into n subintervals with $h \leftarrow \frac{b-a}{n}$ and $x_k \leftarrow a + hk$, we create a system of $n - 1$ linear equations in $n - 1$ unknowns and use linear algebra [**3**] to find an approximation.

c. For linear partial-differential equations such as the heat equation, the wave equation and Laplace's equation, we approximate partial-differential equation by a finite-difference equation by substituting the centered or forward **divided-difference formulas** for the various partial derivatives.

(a) **Heat equation:** For the heat equation, we solve for $u_{k,\ell+1}$ and then use the initial conditions and the boundary conditions to approximate the $u_{k,\ell+1}$ for a given ℓ for all $k = 1, \ldots, n-1$, then finding the boundary values based on whether they are Dirichlet or Neumann.

(b) **Wave equation:** As with the heat equation, we solve for $u_{k,\ell+1}$, but this now depends on $u_{k,\ell-1}$, so when $\ell = 0$, we must use a second initial condition, the initial rate-of-change of the amplitude of the wave. Everything else is the same.

(c) **Laplace's equation:** In one dimension, the solution to Laplace's equation is a straight line between the boundary values, as appropriate. In higher dimensions, the finite-difference approximation says the value at a point is the average of the values around it. This creates a system of linear equations **[3]** which we then solve.

D. **Unconstrained optimization:** Given a real-valued function of either a single variable $f(x)$ or multiple variables $f(\mathbf{x})$, we look at a number of techniques for finding local minima.

a. For real-valued functions of a single variable $f(x)$, we generalize root-finding techniques:

(a) **Step-by-step iteration:** We move in the direction of a minimum with a step size h until the next value is higher and then continue to halve the step size. This is a naive bracketing technique **[6]**.

(b) **Newton's method:** We can find a root-finding technique such as Newton's method but apply it to the derivative $f^{(1)}(x)$ As before, this uses Taylor series **[5]** and iteration **[2]**.

(c) **Golden-ratio search:** A generalization of the bisection method, we bracket the minimum **[6]** and then evaluate the function at two intermediate points to reduce the width of the interval. We then iterate **[2]**.

(d) **Successive parabolic interpolation:** A generalization of the secant method, with three approximations of the minimum, we find an interpolating quadratic polynomial **[4]** and find the minimum of this polynomial. We then iterate **[2]**.

b. For real-valued functions of multiple variables $f(\mathbf{x})$, we may proceed as follows:

(a) **Hooke-Jeeves method:** Approximate the direction of the gradient and then step in that direction until a minimum in that direction is found. Then iterate **[2]**, possibly halving the step size.

(b) Newton's method: Find a simultaneous root of the gradient $\vec{\nabla} f(\mathbf{x}) = \vec{\mathbf{0}}$.

(c) Gradient descent: Find the gradient at a point $\vec{\nabla} f(\mathbf{x}_0)$ and then convert the problem into one of one variable: minimize $f(\mathbf{x}_0 - s\vec{\nabla} f(\mathbf{x}_0)))$. Having found the minimum s_0, let $\mathbf{x}_1 \leftarrow \mathbf{x}_0 + s_0\vec{\nabla} f(\mathbf{x}_0))$. Then iterate **[2]**.