20. Approximating functions globally: Periodic functions and Fourier analysis

Summary of the previous class

- Numerical methods for ODEs
- Taylor methods: calculate the Taylor series of the exact solution
- Adaptivity: Choosing the step size to control the error

Goals for today

- Approximating functions globally

- Fourier series for periodic functions
- Rate of decay of Fourier coefficients

- Trigonometric interpolation

Manipulating functions

- Suppose that we are given a complicated function f over an interval $[a, b]$

Manipulating functions

- Suppose that we are given a complicated function f over an interval $[a, b]$
- We want to do calculations with f such as
  - Addition: $h = f + g$
  - Multiplication: $h(x) = \alpha\, f(x)\, g(x)$,
  - Derivative: f'
  - Definite integral: $\int_a^b f$
  - Find roots of f in $[a, b]$

Manipulating functions

- Suppose that we are given a complicated function f over an interval $[a, b]$
- We want to do calculations with f such as
  - Addition: $h = f + g$
  - Multiplication: $h(x) = \alpha\, f(x)\, g(x)$,
  - Derivative: f'
  - Definite integral: $\int_a^b f$
  - Find roots of f in $[a, b]$
- Idea: Manipulate an **approximation** of f instead

Approximating functions

- We have seen some possibilities for approximating f:

Approximating functions

- We have seen some possibilities for approximating f:
- Taylor series:
  - A polynomial that reproduces f locally near a point
  - Works well only near that point

Approximating functions

- We have seen some possibilities for approximating f:
- Taylor series:
  - A polynomial that reproduces f locally near a point
  - Works well only near that point
- Lagrange interpolation:
  - Expensive to calculate and manipulate
  - Need to be careful about location of interpolation points

Collaboration I

Representing functions globally

1. What other way could we represent a function on an interval?
2. If f is a 2π-periodic function, i.e. $f(t+2\pi) = f(t) \quad \forall t$, how can we represent it?
3. What do we need to calculate in order to do so?

Global approximation

- Idea:

  *Approximate a function f with a linear combination of **basis functions***

Global approximation

- Idea:

  Approximate a function f with a linear combination of ***basis functions***

- We will start with **periodic** functions

Global approximation

- Idea:

  Approximate a function f with a linear combination of ***basis functions***

- We will start with **periodic** functions
- We will do “linear algebra with functions” – i.e. **functional analysis**

Periodic functions

- **Periodic** functions f are a special class that we have not yet discussed

Periodic functions

- **Periodic** functions f are a special class that we have not yet discussed
- Functions f such that there is a (smallest) T for which

$$f(t+T) = f(t) \quad \forall t$$

Periodic functions

- **Periodic** functions f are a special class that we have not yet discussed
- Functions f such that there is a (smallest) T for which

$$f(t+T) = f(t) \quad \forall t$$

- T is the **period** of function

Fourier series

- Consider a **periodic** function f with period 2π
- We will assume f is not “nasty”, say continuous

Fourier series

- Consider a **periodic** function f with period 2π
- We will assume f is not “nasty”, say continuous
- Then (far from obvious) f can be written as

$$f(x) = \sum_{n=-\infty}^{\infty} \hat{f}_n \, \mathrm{e}^{\mathrm{i} n x}$$

- Linear combination of **basis functions** $\phi_n(x) := \mathrm{e}^{\mathrm{i} n x}$

Fourier series

- Consider a **periodic** function f with period 2π
- We will assume f is not "nasty", say continuous
- Then (far from obvious) f can be written as

$$f(x) = \sum_{n=-\infty}^{\infty} \hat{f}_n \, \mathrm{e}^{\mathrm{i}\, nx}$$

- Linear combination of **basis functions** $\phi_n(x) := \mathrm{e}^{\mathrm{i}\, nx}$
- Alternatively

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)]$$

Infinite-dimensional vector space

- This is an *infinite* linear combination of basis functions
- How can we make sense of that?

Infinite-dimensional vector space

- This is an *infinite* linear combination of basis functions
- How can we make sense of that?

- Can *any* periodic function can be expressed like this?

Infinite-dimensional vector space

- This is an *infinite* linear combination of basis functions
- How can we make sense of that?

- Can *any* periodic function can be expressed like this?
- We need to specify which **space of functions** we allow

Hilbert space

- We define the space of **square–integrable functions**

$$L^2 := \{f : \int_0^{2\pi} |f|^2 < \infty\}$$

Hilbert space

- We define the space of **square–integrable functions**

$$L^2 := \{f : \int_0^{2\pi} |f|^2 < \infty\}$$

- L^2 is a **Hilbert space** – a complete inner-product space

Hilbert space

- We define the space of **square–integrable functions**

$$L^2 := \{f : \int_0^{2\pi} |f|^2 < \infty\}$$

- L^2 is a **Hilbert space** – a complete inner-product space
- Infinite-dimensional generalization of Euclidean space

Hilbert space

- We define the space of **square–integrable functions**

$$L^2 := \{f : \int_0^{2\pi} |f|^2 < \infty\}$$

- L^2 is a **Hilbert space** – a complete inner-product space
- Infinite-dimensional generalization of Euclidean space
- It can be shown that the ϕ_n form a **complete basis** of L^2:

Hilbert space

- We define the space of **square–integrable functions**

$$L^2 := \{f : \int_0^{2\pi} |f|^2 < \infty\}$$

- L^2 is a **Hilbert space** – a complete inner-product space
- Infinite-dimensional generalization of Euclidean space
- It can be shown that the ϕ_n form a **complete basis** of L^2:
- Eigenfunctions of Laplacian operator $\frac{\partial^2}{\partial x^2}$
- Sturm–Liouville theory
- Spectral theorem for compact operatorss

Fourier coefficients

- $\widehat{f}_n$ are the **Fourier coefficients** of f
- How can they be calculated?

Fourier coefficients

- $\widehat{f}_n$ are the **Fourier coefficients** of f
- How can they be calculated?

- We saw that coefficients of a linear combination can be easy to find

Fourier coefficients

- $\widehat{f}_n$ are the **Fourier coefficients** of f
- How can they be calculated?

- We saw that coefficients of a linear combination can be easy to find
- Provided the basis vectors are **orthogonal**

Fourier coefficients

- $\widehat{f}_n$ are the **Fourier coefficients** of f
- How can they be calculated?

- We saw that coefficients of a linear combination can be easy to find
- Provided the basis vectors are **orthogonal**
- We need to be able to talk about orthogonality of *functions*

Inner product

- Inner product **generalizes** the dot product in Euclidean space

Inner product

- Inner product **generalizes** the dot product in Euclidean space
- We define

$$(f, g) := \int_0^{2\pi} \overline{f(x)}\, g(x)\, dx$$

Inner product

- Inner product **generalizes** the dot product in Euclidean space
- We define

$$(f, g) := \int_0^{2\pi} \overline{f(x)}\, g(x)\, dx$$

- For $z = x + iy \in \mathbb{C}$, the **complex conjugate** is $\overline{z} := x - iy$

Inner product

- Inner product **generalizes** the dot product in Euclidean space
- We define

$$(f,g) := \int_0^{2\pi} \overline{f(x)}\, g(x)\, dx$$

- For $z = x + iy \in \mathbb{C}$, the **complex conjugate** is $\overline{z} := x - iy$
- Properties: (f,g) is linear in g, and $(f,f) \geq 0$
- Two functions are **orthogonal** if $(f,g) = 0$

Orthogonality

- Let's look at inner products of our basis functions:

$$(\phi_n, \phi_m) = \int_0^{2\pi} \mathrm{e}^{-\mathrm{i}\,nx}\, \mathrm{e}^{\mathrm{i}\,mx} dx$$

$$= \int_0^{2\pi} \mathrm{e}^{\mathrm{i}\,(m-n)x} dx$$

$$= \frac{1}{i(m-n)} \left[\mathrm{e}^{\mathrm{i}\,(m-n)x}\right]_{x=0}^{2\pi}$$

- So $(\phi_n, \phi_m) = 0$ for $m \neq n$ and $= 2\pi$ for $m = n$
- Hence the basis functions (ϕ_n) are orthogonal

Fourier coefficients II

- We have $f = \sum_{-\infty}^{\infty} \hat{f}_n \, \phi_n$

Fourier coefficients II

- We have $f = \sum_{-\infty}^{\infty} \hat{f}_n \, \phi_n$
- Take the inner product with ϕ_m to obtain

$$(\phi_m, f) = \sum_n \hat{f}_n \, (\phi_m, \phi_n)$$

$$= 2\pi \hat{f}(m)$$

Fourier coefficients II

- We have $f = \sum_{-\infty}^{\infty} \hat{f}_n \phi_n$
- Take the inner product with ϕ_m to obtain

$$(\phi_m, f) = \sum_n \hat{f}_n (\phi_m, \phi_n)$$

$$= 2\pi \hat{f}(m)$$

- Thus

$$\hat{f}_m = \frac{1}{2\pi} \int_0^{2\pi} \mathrm{e}^{-\mathrm{i}mx} f(x) dx$$

Representation on a computer

- The natural idea to represent a periodic function on a computer is then to truncate to a *finite* sum

$$f \simeq f_N := \sum_{n=-N}^{N} \hat{f}(n)\phi_n$$

Representation on a computer

- The natural idea to represent a periodic function on a computer is then to truncate to a *finite* sum

$$f \simeq f_N := \sum_{n=-N}^{N} \hat{f}(n)\phi_n$$

- The natural question is *how many* terms should we take?

Representation on a computer

- The natural idea to represent a periodic function on a computer is then to truncate to a *finite* sum

$$f \simeq f_N := \sum_{n=-N}^{N} \hat{f}(n)\phi_n$$

- The natural question is *how many* terms should we take?
- We want to take enough terms so that we represent the function well, so the neglected terms are "small"
- What does this depend on?

Representation on a computer

- The natural idea to represent a periodic function on a computer is then to truncate to a *finite* sum

$$f \simeq f_N := \sum_{n=-N}^{N} \hat{f}(n)\phi_n$$

- The natural question is *how many* terms should we take?
- We want to take enough terms so that we represent the function well, so the neglected terms are "small"
- What does this depend on?
- It depends on *how fast* the $\hat{f}(n)$ *decay* as $n \to \infty$

Fourier coefficients of derivatives

- Suppose f is periodic and differentiable

Fourier coefficients of derivatives

- Suppose f is periodic and differentiable
- f' is also periodic, so can expand it in a Fourier series:

$$f' = \sum \widehat{f'}_n \phi_n$$

where $\widehat{f'}_n = \frac{1}{2\pi} \int_0^{2\pi} \mathrm{e}^{-\mathrm{i}nx} f'(x) dx$

Fourier coefficients of derivatives

- Suppose f is periodic and differentiable
- f' is also periodic, so can expand it in a Fourier series:

$$f' = \sum \widehat{f'}_n \phi_n$$

where $\widehat{f'}_n = \frac{1}{2\pi} \int_0^{2\pi} \mathrm{e}^{-\mathrm{i}\, nx} f'(x) dx$

- Integrate by parts to get

$$\widehat{f'}_n = \mathrm{i}\, n \, \hat{f}_n$$

Rate of decay of Fourier coefficients

- Can show: If f is integrable, e.g. continuous, on an interval, then

$$\widehat{f}_n \to 0 \quad \text{as } n \to \infty$$

- **Riemann–Lebesgue lemma**

Rate of decay of Fourier coefficients

- Can show: If f is integrable, e.g. continuous, on an interval, then

$$\hat{f}_n \to 0 \quad \text{as } n \to \infty$$

- **Riemann–Lebesgue lemma**

- If $f \in C^k$ then $\widehat{f^{(k)}}_n = (\mathrm{i}\, n)^k \hat{f}_n \to 0$

Rate of decay of Fourier coefficients

- Can show: If f is integrable, e.g. continuous, on an interval, then

$$\hat{f}_n \to 0 \quad \text{as } n \to \infty$$

- **Riemann–Lebesgue lemma**

- If $f \in C^k$ then $\widehat{f^{(k)}}_n = (\mathrm{i}\, n)^k \hat{f}_n \to 0$
- Hence $\hat{f}_n = o(n^{-k})$

Rate of decay II

- If f is **smooth** (C^∞) then decays faster than any polynomial

- In fact, if f is analytic in a suitable region then

$$\hat{f}_n = \mathcal{O}(e^{-\alpha n})$$

Rate of decay II

- If f is **smooth** (C^∞) then decays faster than any polynomial

- In fact, if f is analytic in a suitable region then

$$\hat{f}_n = \mathcal{O}(e^{-\alpha n})$$

- I.e. the Fourier coefficients decay **exponentially** fast

Rate of decay II

- If f is **smooth** (C^∞) then decays faster than any polynomial

- In fact, if f is analytic in a suitable region then

$$\hat{f}_n = \mathcal{O}(e^{-\alpha n})$$

- I.e. the Fourier coefficients decay **exponentially** fast
- This is called **spectral convergence**

Representing functions on the computer

- This suggests that we should calculate *enough* terms

Representing functions on the computer

- This suggests that we should calculate *enough* terms
- After a while they should start decaying fast

Representing functions on the computer

- This suggests that we should calculate *enough* terms
- After a while they should start decaying fast
- Stop when close to machine epsilon

Representing functions on the computer

- This suggests that we should calculate *enough* terms
- After a while they should start decaying fast
- Stop when close to machine epsilon
- We get an approximation of our general periodic function to ~ machine epsilon!

Representing functions on the computer

- This suggests that we should calculate *enough* terms
- After a while they should start decaying fast
- Stop when close to machine epsilon
- We get an approximation of our general periodic function to $\sim$ machine epsilon!

- **Implementations**:
  - The **Chebfun** package in Matlab
  - The **ApproxFun.jl** package in Julia

Trigonometric interpolation

Trigonometric interpolation

- Calculating Fourier integrals numerically is not too nice
- Is there an alternative?

Trigonometric interpolation

- Calculating Fourier integrals numerically is not too nice
- Is there an alternative?
- How else could we look for a Fourier series that approximates f?

Trigonometric interpolation

- Calculating Fourier integrals numerically is not too nice
- Is there an alternative?
- How else could we look for a Fourier series that approximates f?
- Return to an idea from earlier in the course: **interpolation**

Trigonometric interpolation

- Calculating Fourier integrals numerically is not too nice
- Is there an alternative?
- How else could we look for a Fourier series that approximates f?
- Return to an idea from earlier in the course: **interpolation**
- Let's try to **interpolate** f using trigonometric functions

Trigonometric interpolation II

- Recall the setting of interpolation:

  Given x_j *and* $f_j = f(x_j)$ *for* $j = 0, \ldots, N$*, find* g *(in some class) such that* $g(x_j) = f_j$

Trigonometric interpolation II

- Recall the setting of interpolation:

  Given x_j *and* $f_j = f(x_j)$ *for* $j = 0, \ldots, N$, *find* g *(in some class) such that* $g(x_j) = f_j$

- In the context of periodic functions we take $x_N = x_0$

Trigonometric interpolation II

- Recall the setting of interpolation:

  Given x_j *and* $f_j = f(x_j)$ *for* $j = 0, \ldots, N$, *find* g *(in some class) such that* $g(x_j) = f_j$

- In the context of periodic functions we take $x_N = x_0$
- We will look for trigonometric functions

$$g = \sum_{k=0}^{N-1} g_k \, \phi_k$$

- For technical reasons restrict to odd N

Interpolation II

- Condition for interpolation:

$$g(x_j) = f_j$$

- So interpolate N points (x_j, f_j) with a sum of basis functions with N unknown coefficients g_n:

$$\sum_{k=0}^{N-1} g_k \, \phi_k(x_j) = f_j$$

Interpolation II

- Condition for interpolation:

$$g(x_j) = f_j$$

- So interpolate N points (x_j, f_j) with a sum of basis functions with N unknown coefficients g_n:

$$\sum_{k=0}^{N-1} g_k \, \phi_k(x_j) = f_j$$

- Which points should we interpolate in?

Interpolation II

- Condition for interpolation:

$$g(x_j) = f_j$$

- So interpolate N points (x_j, f_j) with a sum of basis functions with N unknown coefficients g_n:

$$\sum_{k=0}^{N-1} g_k \, \phi_k(x_j) = f_j$$

- Which points should we interpolate in?
- For polyomials, equally-spaced points were **bad**

Interpolation III

- For trigonometric interpolation equally spaced is fine:
- Take N **equally-spaced** points $x_j := jh$ with $h := \frac{2\pi}{N}$

Interpolation III

- For trigonometric interpolation equally spaced is fine:
- Take N **equally-spaced** points $x_j := jh$ with $h := \frac{2\pi}{N}$
- The interpolation condition becomes

$$\sum_{k=0}^{N-1} g_k \exp(ikjh) = f_j$$

Collaboration II

1. What kind of equation do we have?
2. How can we solve it?
3. What number of operations do we expect to need to do so?

Interpolation IV

- What kind of equations are these for the unknown coefficients g_n?

Interpolation IV

- What kind of equations are these for the unknown coefficients g_n?
- It's a **linear system** for the g_n:

$$\sum_k M_{jk}\, g_k = f_j$$

Interpolation IV

- What kind of equations are these for the unknown coefficients g_n?
- It's a **linear system** for the g_n:

$$\sum_k M_{jk}\, g_k = f_j$$

- I.e. a matrix equation $\quad \mathsf{M}\,\mathbf{g} = \mathbf{f}$

Interpolation IV

- What kind of equations are these for the unknown coefficients g_n?
- It's a **linear system** for the g_n:

$$\sum_k M_{jk}\, g_k = f_j$$

- I.e. a matrix equation $\quad \mathsf{M}\,\mathbf{g} = \mathbf{f}$
- With solution $\mathbf{g} = \mathsf{M}^{-1}\mathbf{f}$

Interpolation IV

- What kind of equations are these for the unknown coefficients g_n?
- It's a **linear system** for the g_n:

$$\sum_k M_{jk}\, g_k = f_j$$

- I.e. a matrix equation $\mathsf{M}\,\mathbf{g} = \mathbf{f}$
- With solution $\mathbf{g} = \mathsf{M}^{-1}\mathbf{f}$
- When is this easy to solve?

Interpolation V

- This is easy to solve if M is an **orthogonal** matrix

Interpolation V

- This is easy to solve if M is an **orthogonal** matrix
- It turns out that it almost is:

$$\mathsf{M}^*\mathsf{M} = N\mathsf{I}$$

where M^* is the **conjugate transpose** $\mathsf{M}^* := \overline{\mathsf{M}^\top}$

Interpolation V

- This is easy to solve if M is an **orthogonal** matrix
- It turns out that it almost is:

$$\mathsf{M}^*\mathsf{M} = N\mathsf{I}$$

where M^* is the **conjugate transpose** $\mathsf{M}^* := \overline{\mathsf{M}^\top}$

- Hence

$$\mathbf{g} = \frac{1}{N}\mathsf{M}^*\mathbf{f}$$

Discrete Fourier transform (DFT)

- The **Discrete Fourier transform** $\mathscr{F}$ maps *from* function values *to* Fourier coefficients:

$$g_k = \frac{1}{N} \sum_j e^{-ijk\frac{2\pi}{N}} f_j$$

- The **Inverse Discrete Fourier Transform** maps *from* Fourier coefficients *to* function values:

$$f_j = \sum_k e^{ijk\frac{2\pi}{N}} g_k$$

Fast Fourier transform

- It looks like we need to do a matrix–vector multiplication to calculate the DFT – $\mathcal{O}(n^2)$ operations

Fast Fourier transform

- It looks like we need to do a matrix–vector multiplication to calculate the DFT – $\mathcal{O}(n^2)$ operations
- One of the great algorithmic discoveries of the 19th/20th century: **Fast Fourier Transform** (FFT)

Fast Fourier transform

- It looks like we need to do a matrix–vector multiplication to calculate the DFT – $\mathcal{O}(n^2)$ operations

- One of the great algorithmic discoveries of the 19th/20th century: **Fast Fourier Transform** (FFT)
- This uses *structure* in M to calculate in $\mathcal{O}(n \log n)$ time
- Has myriad applications throughout applied mathematics, signal processing, physics, …

Summary

- Periodic functions may be expanded in Fourier series
- Infinite linear combination of orthogonal basis functions
- The Fourier coefficients $\widehat{f}_n$ decay at a rate determined by the smoothness of f

- Trigonometric interpolation corresponds to a discrete Fourier transform