Lecture 17 - Uncertainty Propagation: Polynomial Chaos II

Ilias Bilionis
ibilion@purdue.edu
School of Mechanical Engineering
Purdue University
predictivesciencelab.org

PREDICTIVE
SCIENCE LABORATORY

PURDUE
UNIVERSITY

Lecture objectives

- Reminder: Orthogonal polynomials.
- Introduce quadrature rules in 1D.
- Expand quadrature rules in multiple dimensions using sparse grids.

1D Orthogonal Polynomials

1D Polynomials

A **polynomial** is:

$$g(x; a) = a_0 + a_1 x + a_2 x^2 + \cdots + a_\rho x^\rho$$

Scalar Square Integrable Functions

Fix the space to:

$$\mathcal{F}_1 = \left\{ f : \mathbb{R} \to \mathbb{R} \text{ such that } \mathbb{E}_p \left[f^2(X) \right] < \infty \right\}$$

A set of **orthonormal polynomials** $\phi_1, \phi_2, \ldots$

satisfies the following properties:

$$\phi_i(x) = a_{0i} + a_{1i}x + \cdots + a_{\rho_i i}x^{\rho_i}$$

$$\forall i \neq j \Rightarrow \langle \phi_i, \phi_j \rangle = \mathbb{E}_p[\phi_i(X)\phi_j(X)] = \int_{-\infty}^{\infty} \phi_i(x)\phi_j(x)p(x)dx = 0$$

$$\| \phi_i \|^2 = \langle \phi_i, \phi_i \rangle = \int_{-\infty}^{\infty} \phi_i^2(x)p(x)dx = 1$$

Constructing Orthogonal Polynomials

- Start from standard monomials: 1, x, …
- Think of them as vectors in the Hilbert space of square p-integrable scalar functions.
- Perform the Gram-Schmidt orthogonalization process.
- But not very stable…

Constructing Orthogonal Polynomials

Algorithm 726: ORTHPOL—A Package of Routines for Generating Orthogonal Polynomials and Gauss-Type Quadrature Rules

WALTER GAUTSCHI
Purdue University

A collection of subroutines and examples of their uses, as well as the underlying numerical methods, are described for generating orthogonal polynomials relative to arbitrary weight functions. The object of these routines is to produce the coefficients in the three-term recurrence relation satisfied by the orthogonal polynomials. Once these are known, additional data can be generated, such as zeros of orthogonal polynomials and Gauss-type quadrature rules, for which routines are also provided.

Categories and Subject Descriptors: G.1.2 [**Numerical Analysis**]: Approximation; G.1.4 [**Numerical Analysis**]: Quadrature and Numerical Differentiation; G.4 [**Mathematical Software**]

General Terms: Algorithms

Additional Key Words and Phrases: Gauss-type quadrature rules, orthogonal polynomials

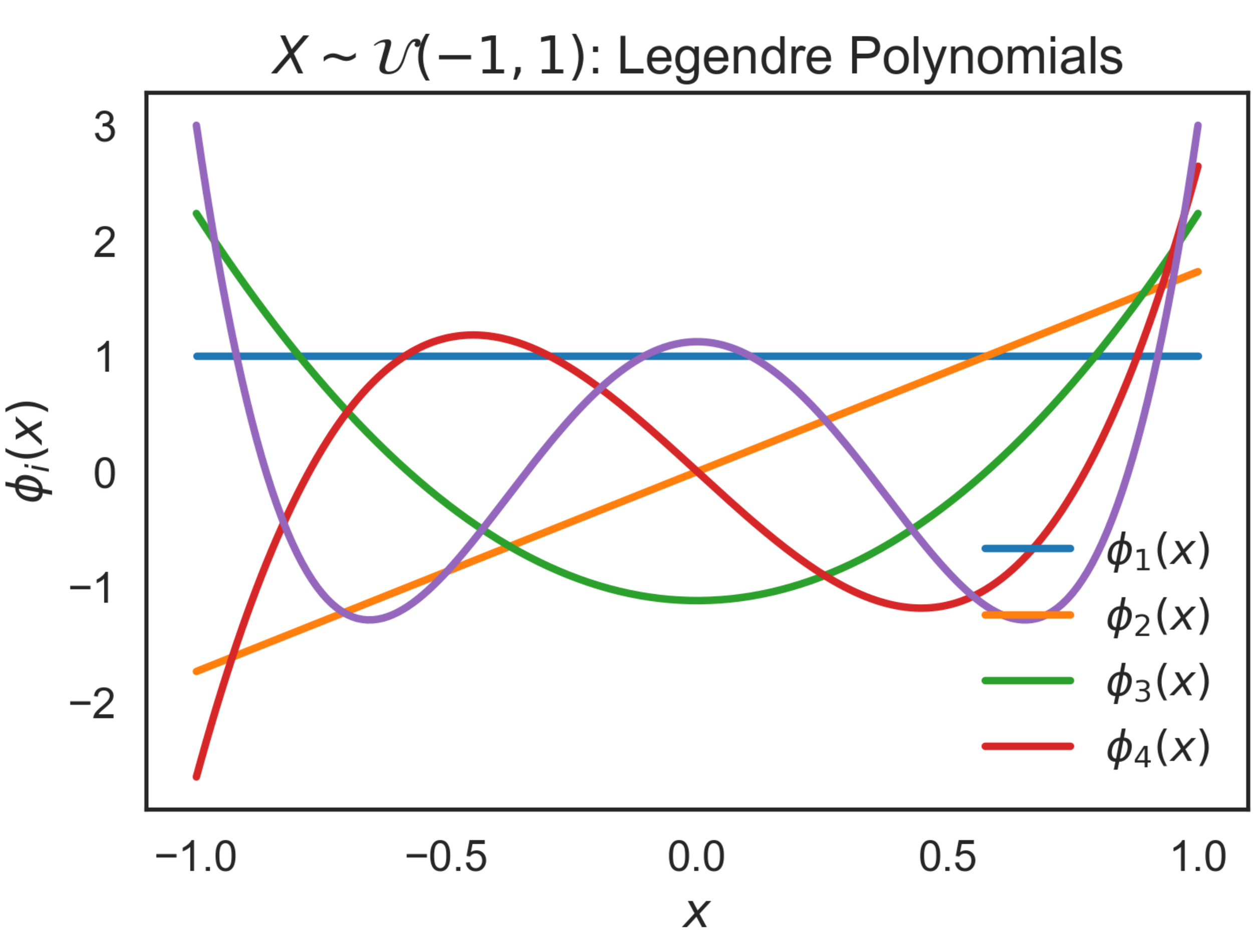
Python interface: py-orthpol

The Uniform and the Legendre Polynomials

$$X \sim \mathcal{U}(-1, 1)$$

The first few are:

$$\begin{aligned} \phi_1(x) &= 1, \\ \phi_2(x) &= x, \\ \phi_3(x) &= \tfrac{1}{2}\left(3x^2 - 1\right). \end{aligned}$$


$X \sim \mathcal{U}(-1, 1)$: Legendre Polynomials
3
2
1
0
−1
−2
$\phi_i(x)$
−1.0
−0.5
0.0
0.5
1.0
x
$\phi_1(x)$
$\phi_2(x)$
$\phi_3(x)$
$\phi_4(x)$

Orthogonal Polynomials in Higher Dimensions

Polynomials in Higher Dimensions

A **polynomial** is:

$$g(x;a) = \sum_{|i_1+\cdots+i_d|\leq\rho} a_{i_1\ldots i_d} x_1^{i_1} \ldots x_d^{i_d}$$

Square Integrable Functions

Fix the space to:

$$\mathcal{F} \equiv \mathcal{F}_d = \left\{ f : \mathbb{R}^d \to \mathbb{R} \text{ such that } \mathbb{E}_p \left[f^2(X) \right] < \infty \right\}$$

A set of **orthonormal polynomials** $\phi_1, \phi_2, \ldots$

satisfies the following properties:

$$\phi_i(x) = \sum_{|i_1 + \cdots + i_d| \leq \rho} a_{i_1 \ldots i_d} x_1^{i_1} \ldots x_d^{i_d}$$

$$\langle \phi_i, \phi_j \rangle = \mathbb{E}_p[\phi_i(X)\phi_j(X)] = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \phi_i(x)\phi_j(x)p(x)dx_1 \ldots dx_d = 0$$

$$\| \phi_i \|^2 = \langle \phi_i, \phi_i \rangle = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \phi_i^2(x)p(x)dx_1 \ldots dx_d = 1$$

Special Case: Random Vector with Independent Components

$$X = (X_1, \ldots, X_d) \sim p$$

$$p(x) = p(x_1, \ldots, x_d) = \prod_{i=1}^{d} p_i(x_i)$$

Build orthogonal polynomials for each component and then take the tensor product:

$$\begin{aligned}
\phi_1(x) &= \phi_{11}(x_1) \cdot \cdots \cdot \phi_{d1}(x_d) = 1, \\
\phi_2(x) &= \phi_{11}(x_1) \cdot \cdots \cdot \phi_{d2}(x_d) = \phi_{d2}(x_d), \\
&\cdots \\
\phi_{i_1,\ldots,i_d}(x) &= \phi_{1i_1}(x_1) \cdot \cdots \cdot \phi_{di_d}(x_d).
\end{aligned}$$

Relation to Uncertainty Propagation

Let X be a random vector with pdf p(x).

Assume that your model is in:

$$\mathcal{F} \equiv \mathcal{F}_d = \left\{ f : \mathbb{R}^d \to \mathbb{R} \text{ such that } \mathbb{E}_p \left[f^2(X) \right] < \infty \right\}$$

and that you (somehow) have found the polynomial expansion:

$$f = \sum_{i=1}^{\infty} c_i \phi_i$$

Then:

$$\mathbb{E}_p[f(X)] = \langle f, \phi_1 \rangle = c_1$$

$$\mathbb{V}_p[f(X)] = \sum_{i=2}^{\infty} c_i^2$$

Quadrature Rules

1D Quadrature Rule

We want to evaluate the integral:

$$I = \int_a^b f(x)dx$$

A quadrature rule is a *linear functional*:

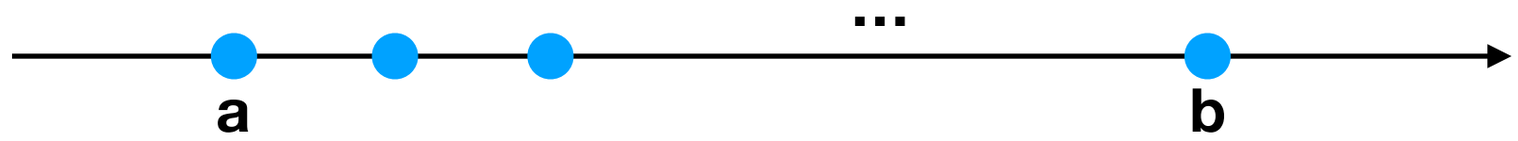
$$Q(f) = \sum_{k=1}^{n} w_k f(x_k)$$

weights **nodes**

Newton-Cotes Rule

$$x_k = a + hk$$

$$h = \frac{b-a}{n+1}$$

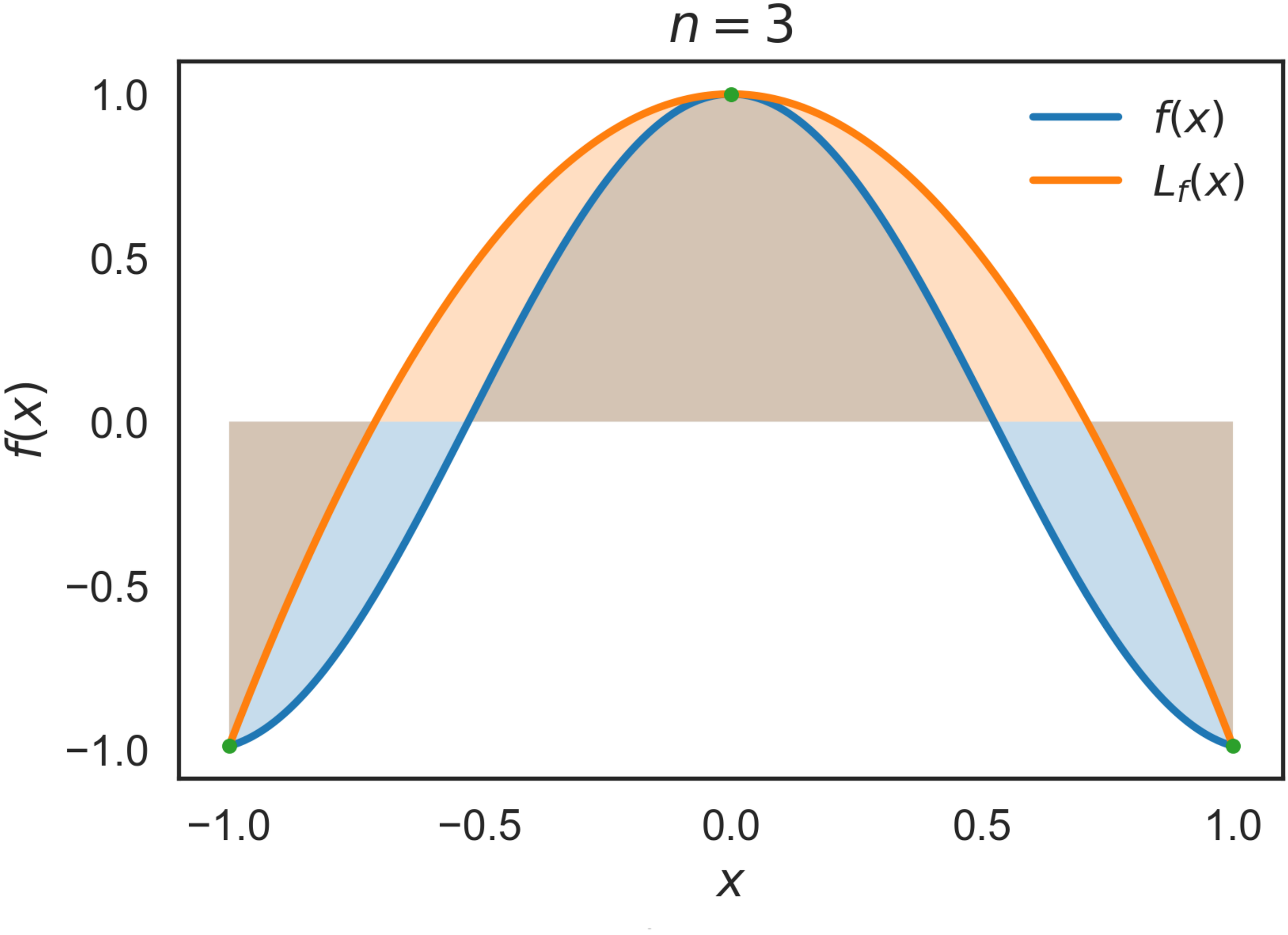


Approximate function with Lagrange interpolating polynomials:

$$f(x) \approx \sum_{k=1}^{n} f(x_k)\ell_k(x)$$

Approximate integral by:

$$Q_{nc}(f) = \sum_{k=1}^{n} \int_a^b \ell_k(x)dx \cdot f(x_k)$$


$n = 3$
$f(x)$
$L_f(x)$
1.0
0.5
0.0
−0.5
−1.0
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
x

$n = 5$

$f(x)$

$L_f(x)$

1.0

0.5

0.0

−0.5

−1.0

$f(x)$

−1.0

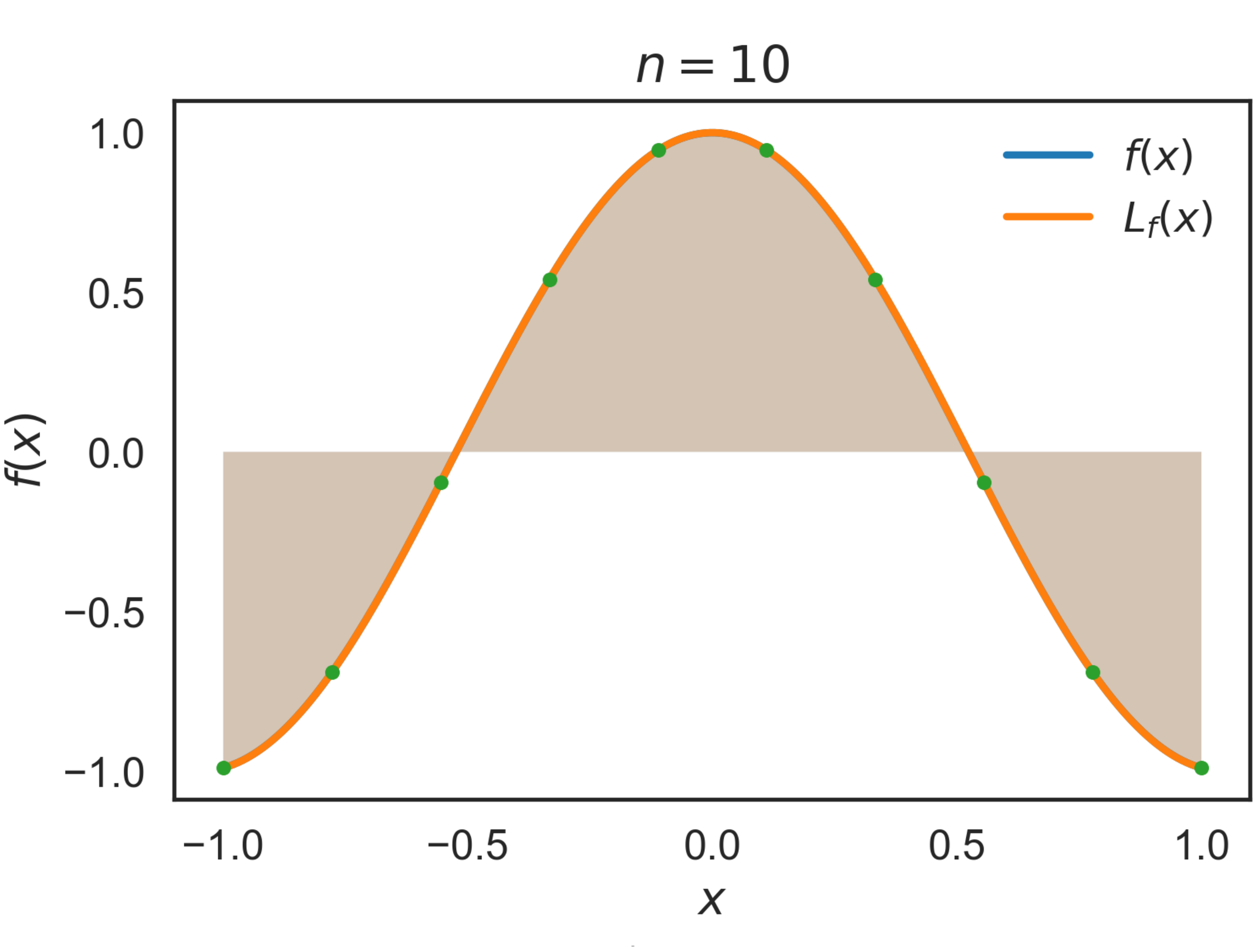
−0.5

0.0

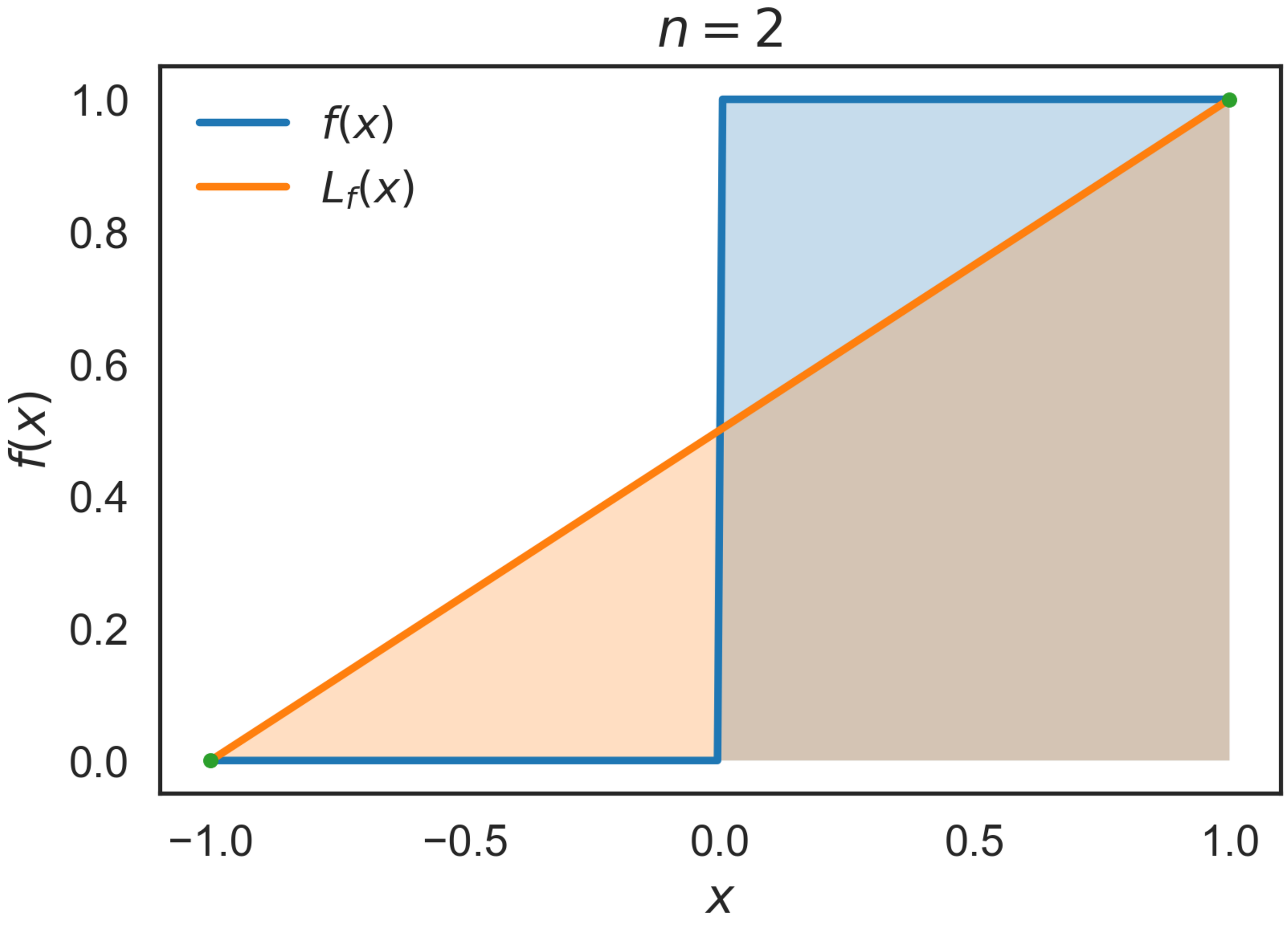
0.5

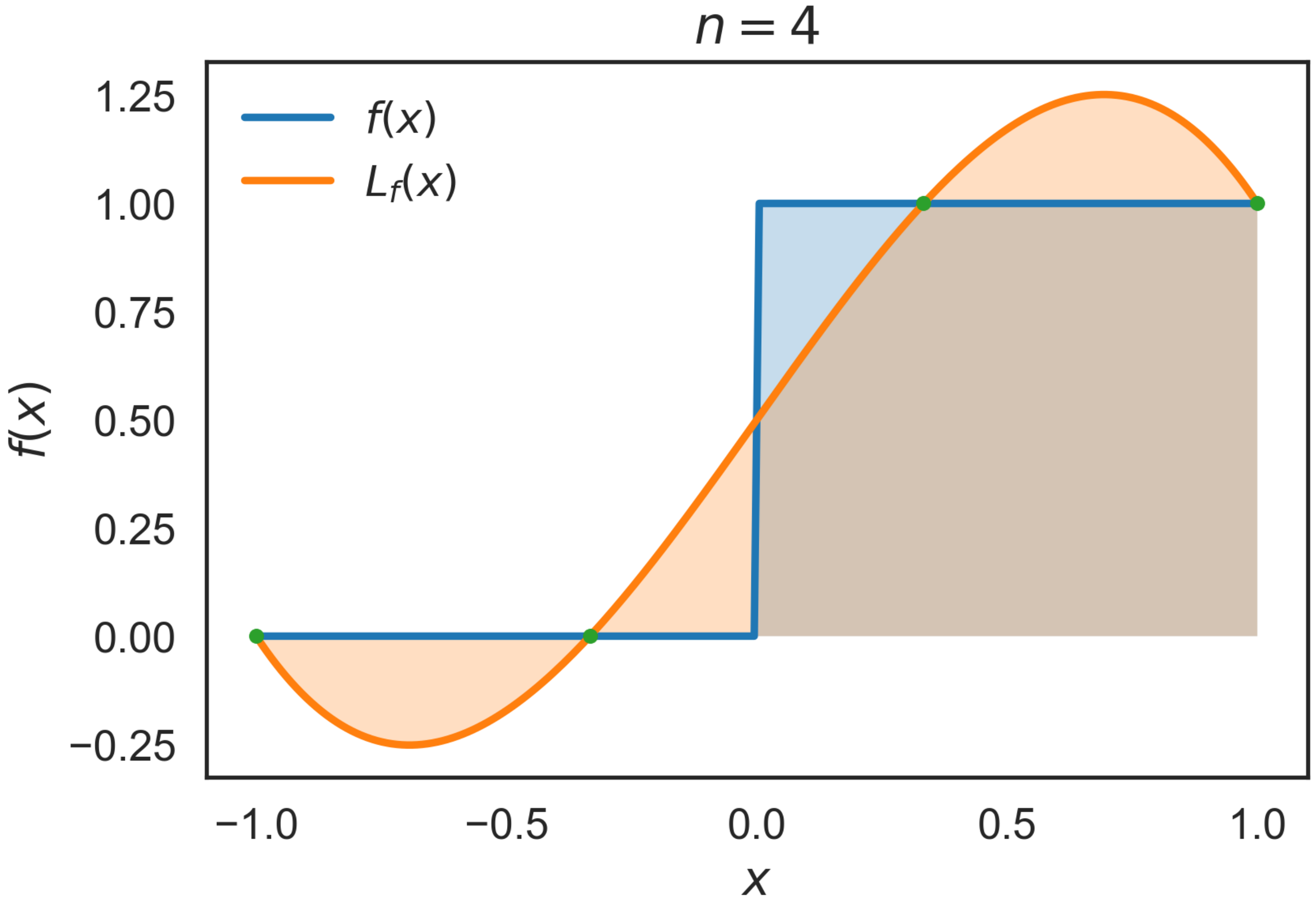
1.0

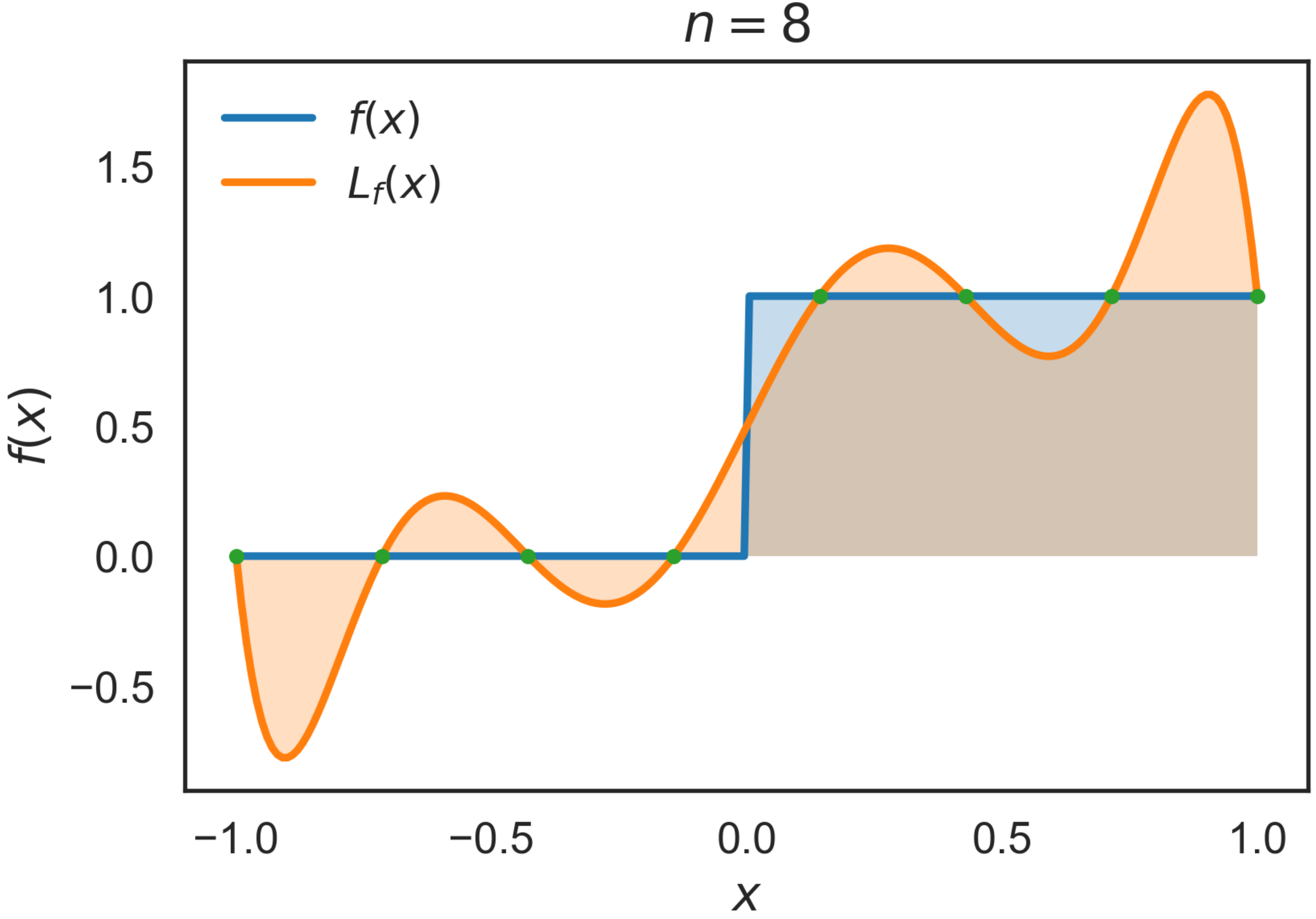
x


$n = 10$
1.0
0.5
0.0
−0.5
−1.0
$f(x)$
$L_f(x)$
−1.0
−0.5
0.0
0.5
1.0
x

But it doesn’t always work…


$n = 2$
$f(x)$
$L_f(x)$
1.0
0.8
0.6
0.4
0.2
0.0
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
x


$n = 4$
$f(x)$
$L_f(x)$
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
x


$n = 8$
$f(x)$
$L_f(x)$
1.5
1.0
0.5
0.0
−0.5
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
x

$n = 10$

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0

$f(x)$

−1.0
−0.5
0.0
0.5
1.0

x

$f(x)$
$L_f(x)$

$n = 16$

$f(x)$

$L_f(x)$

40

20

0

−20

−40

$f(x)$

−1.0

−0.5

0.0

0.5

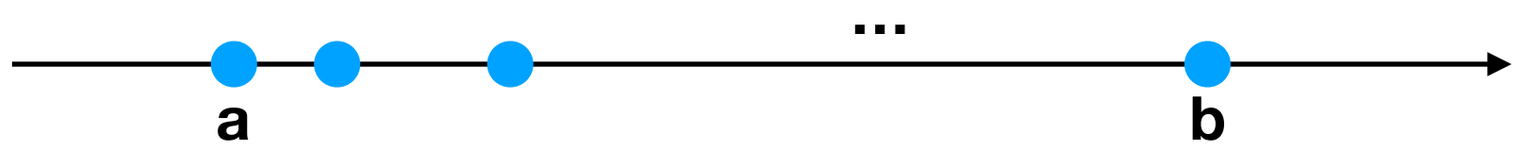
1.0

x

Gaussian Quadrature (n nodes)

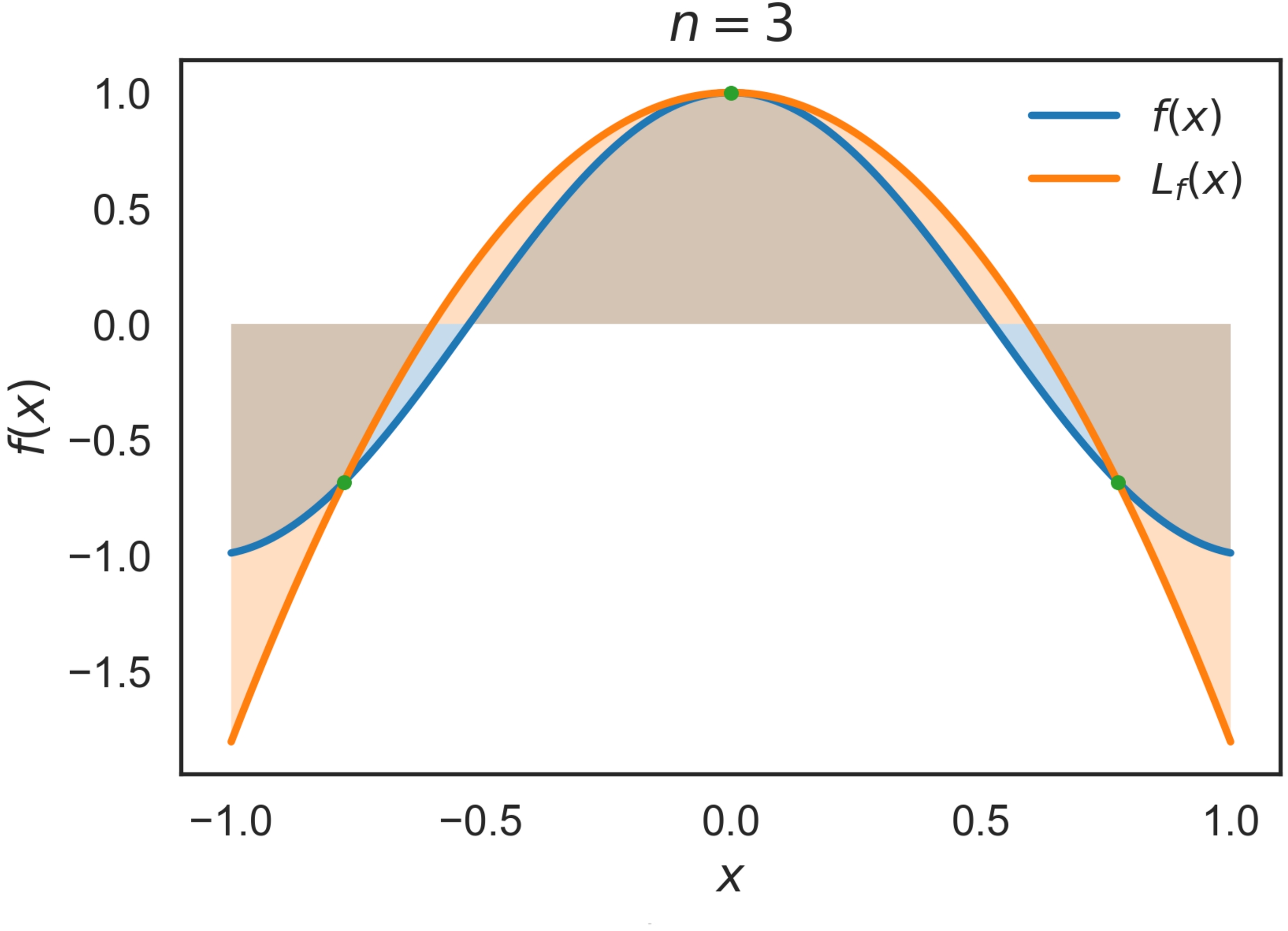
Just like Newton-Cotes but:

x_k **are the zeros of the n-th orthogonal polynomial**



Theorem 1: It interpolates exactly polynomials of order 2n + 1.

Theorem 2: No other quadrature rule with n nodes can do that.


$n = 3$
1.0
0.5
0.0
−0.5
−1.0
−1.5
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
x
$f(x)$
$L_f(x)$

$n = 6$

$f(x)$

$L_f(x)$

1.0

0.5

0.0

−0.5

−1.0

$f(x)$

−1.0

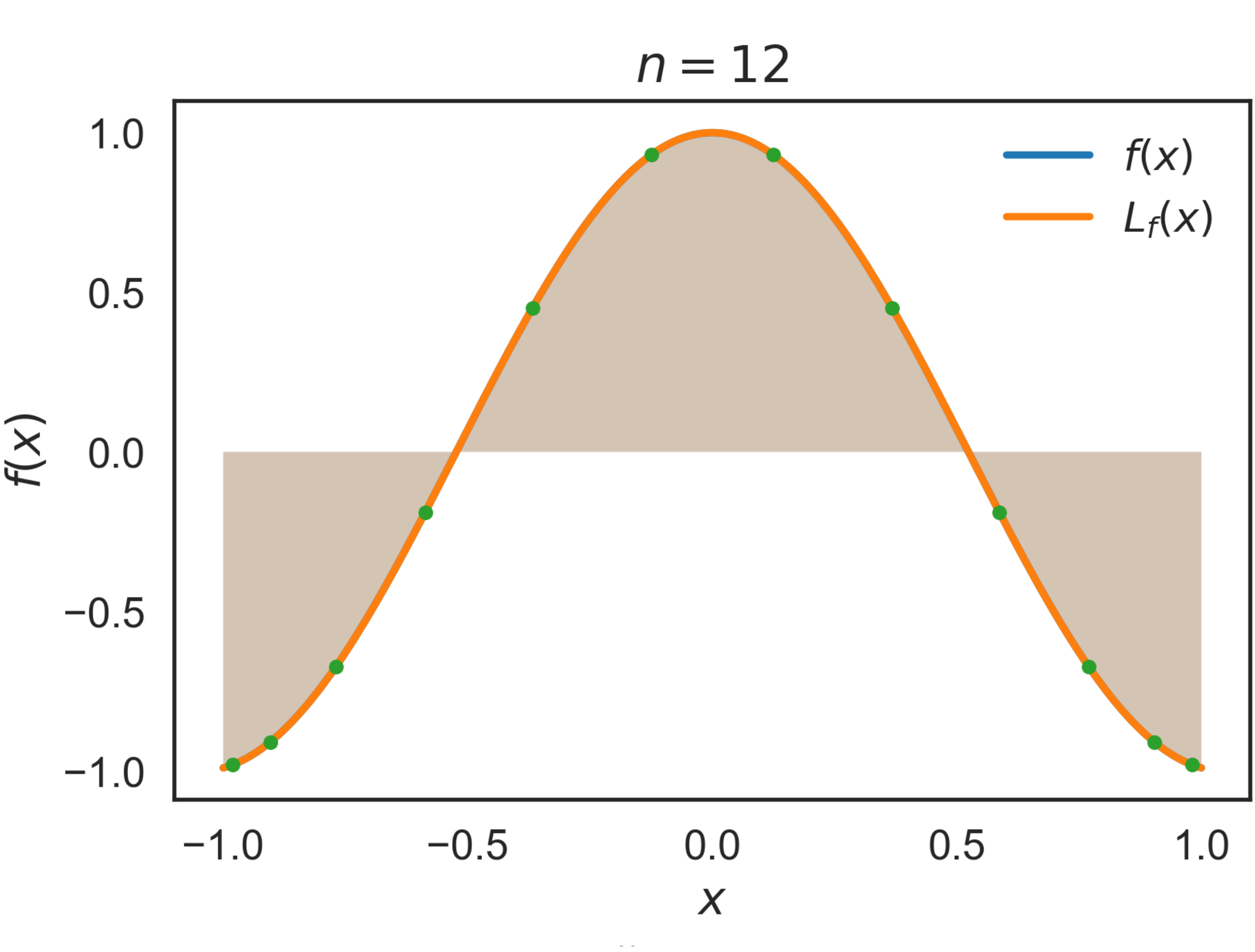
−0.5

0.0

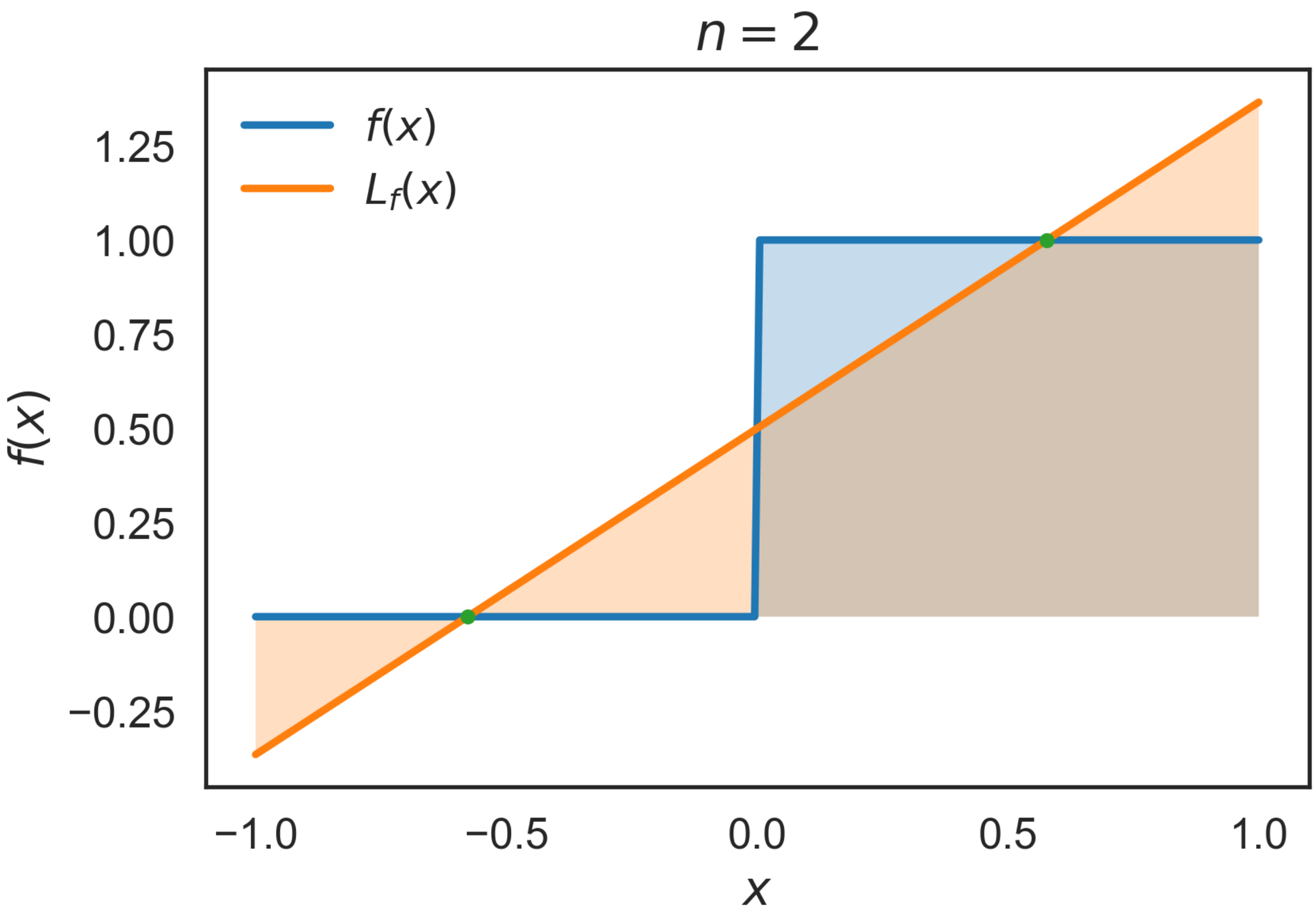
0.5

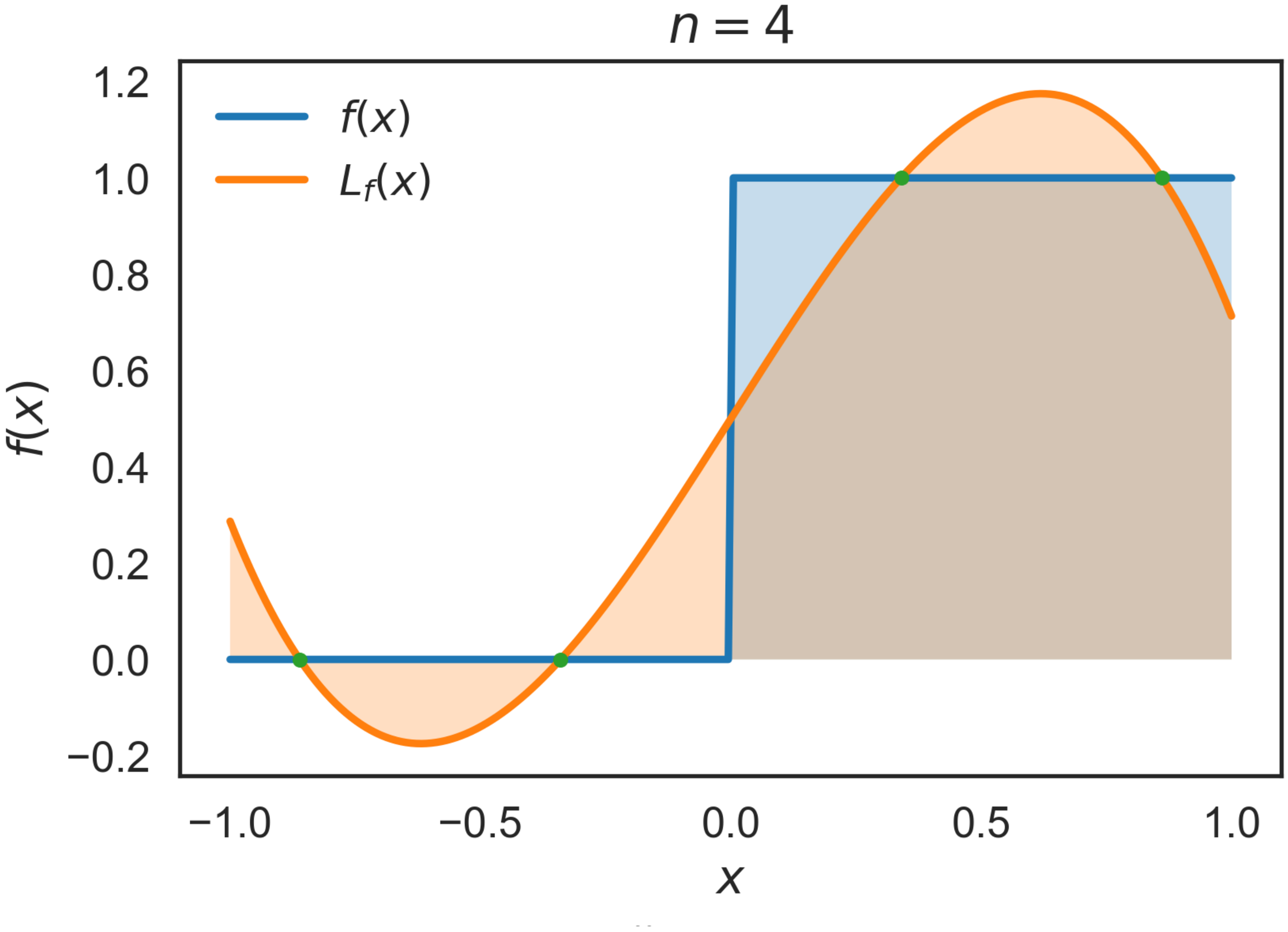
1.0

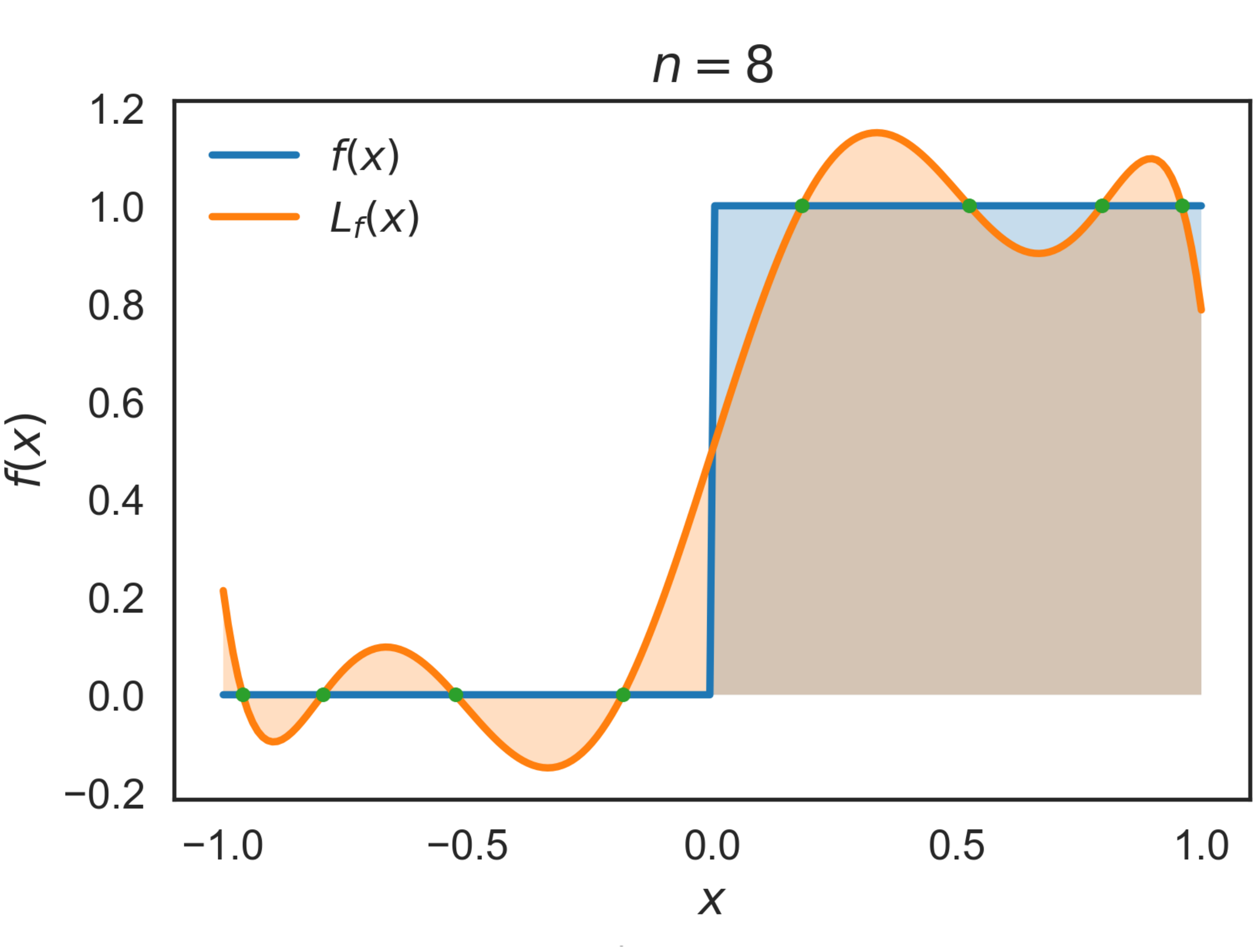
x

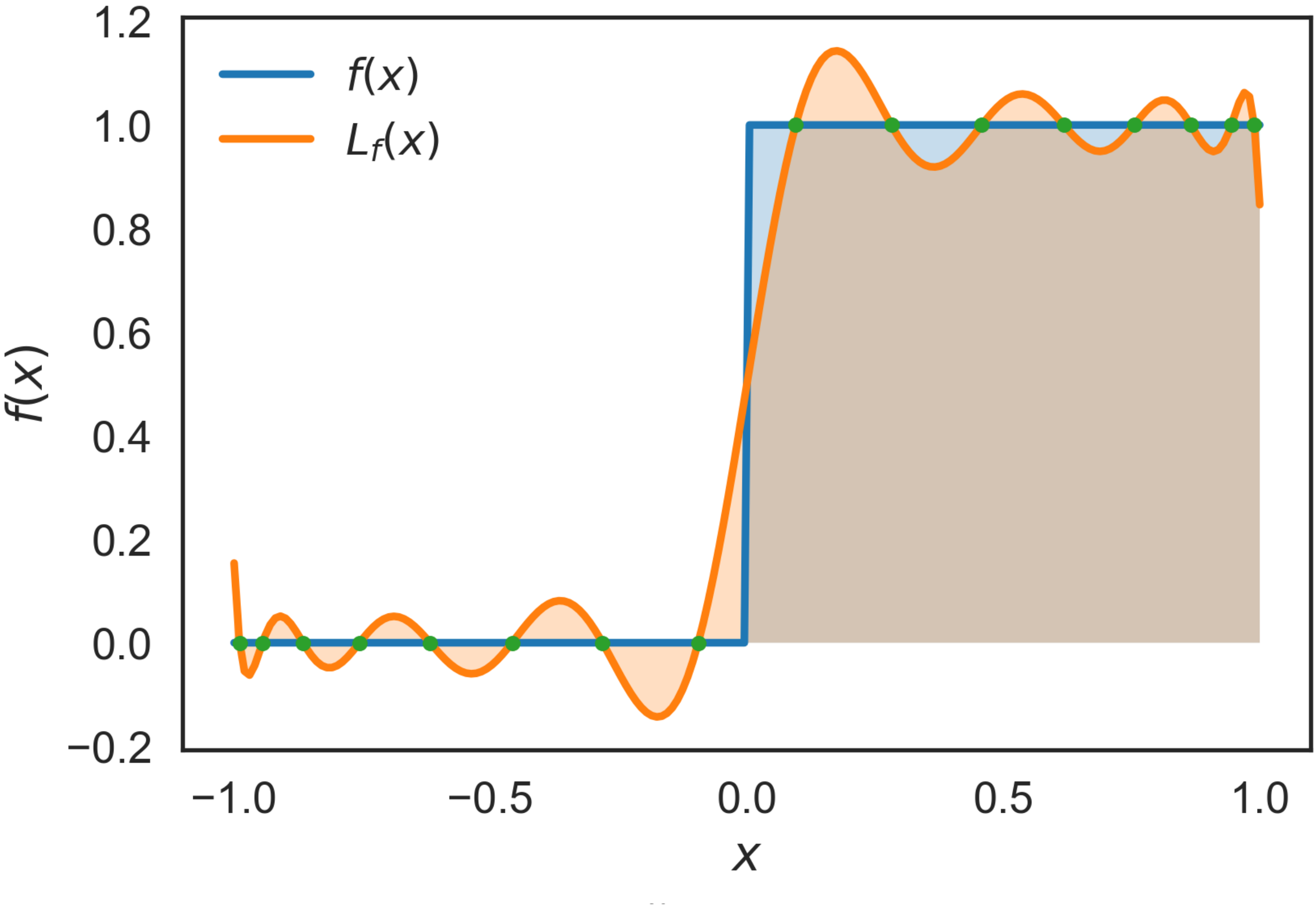

$n = 12$
1.0
0.5
0.0
−0.5
−1.0
$f(x)$
$L_f(x)$
−1.0
−0.5
0.0
0.5
1.0
x

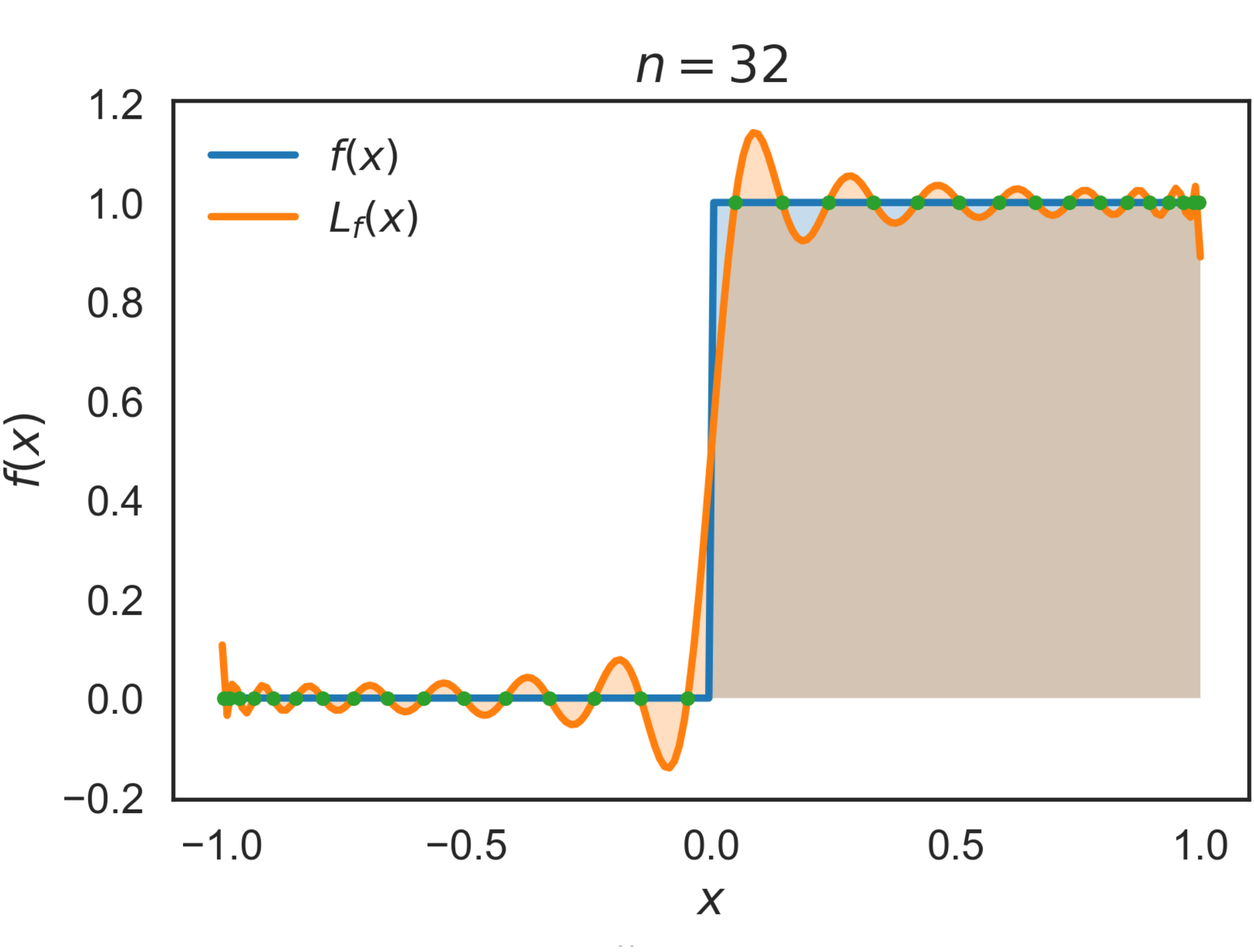
It does behave better with things like step functions…


$n = 2$
$f(x)$
$L_f(x)$
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
x

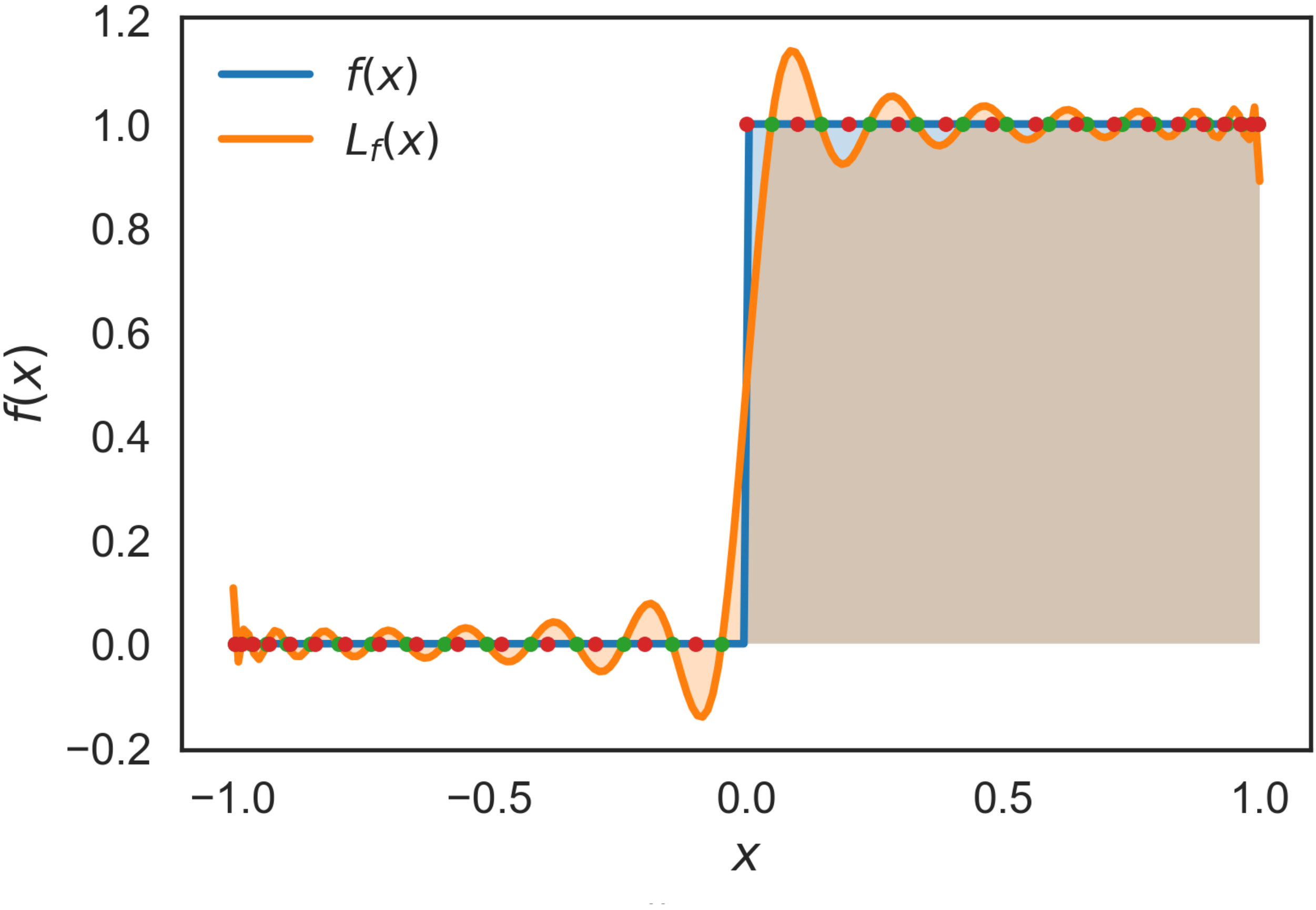

$n = 4$
$f(x)$
$L_f(x)$
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
x


$n = 8$
$f(x)$
$L_f(x)$
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
x


$n = 16$
$f(x)$
$L_f(x)$
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
x


$n = 32$
$f(x)$
$L_f(x)$
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
x

The major problem with Gaussian quadrature is that it is not nested


$n = 32$
$f(x)$
$L_f(x)$
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
x

Clenshaw-Curtis Quadrature

This is a *nested rule*.

$$\int_{-1}^{1} f(x)dx = \int_{0}^{\pi} f(\cos\theta)\sin(\theta)d\theta$$

$$f(\cos\theta) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(k\theta)$$

$$a_k = \frac{2}{\pi}\int_{0}^{\pi} f(\cos\theta)\cos(k\theta)d\theta$$

$$\int_{0}^{\pi} f(\cos\theta)\sin\theta d\theta = a_0 + \sum_{k=1}^{\infty}\frac{2a_k}{1-(2k)^2}$$

Clenshaw-Curtis Quadrature

Nyquist-Shannon sampling theorem:

$$a_k = \frac{2}{n}\left(\left(-1\right)^k \frac{f(-1)}{2} + \frac{f(1)}{2} + \sum_{j=1}^{n-1} f\left(\cos\frac{j\pi}{n}\right)\cos\frac{kj\pi}{n}\right)$$

So the nodes are:

$$x_j = \cos\frac{j\pi}{n}$$

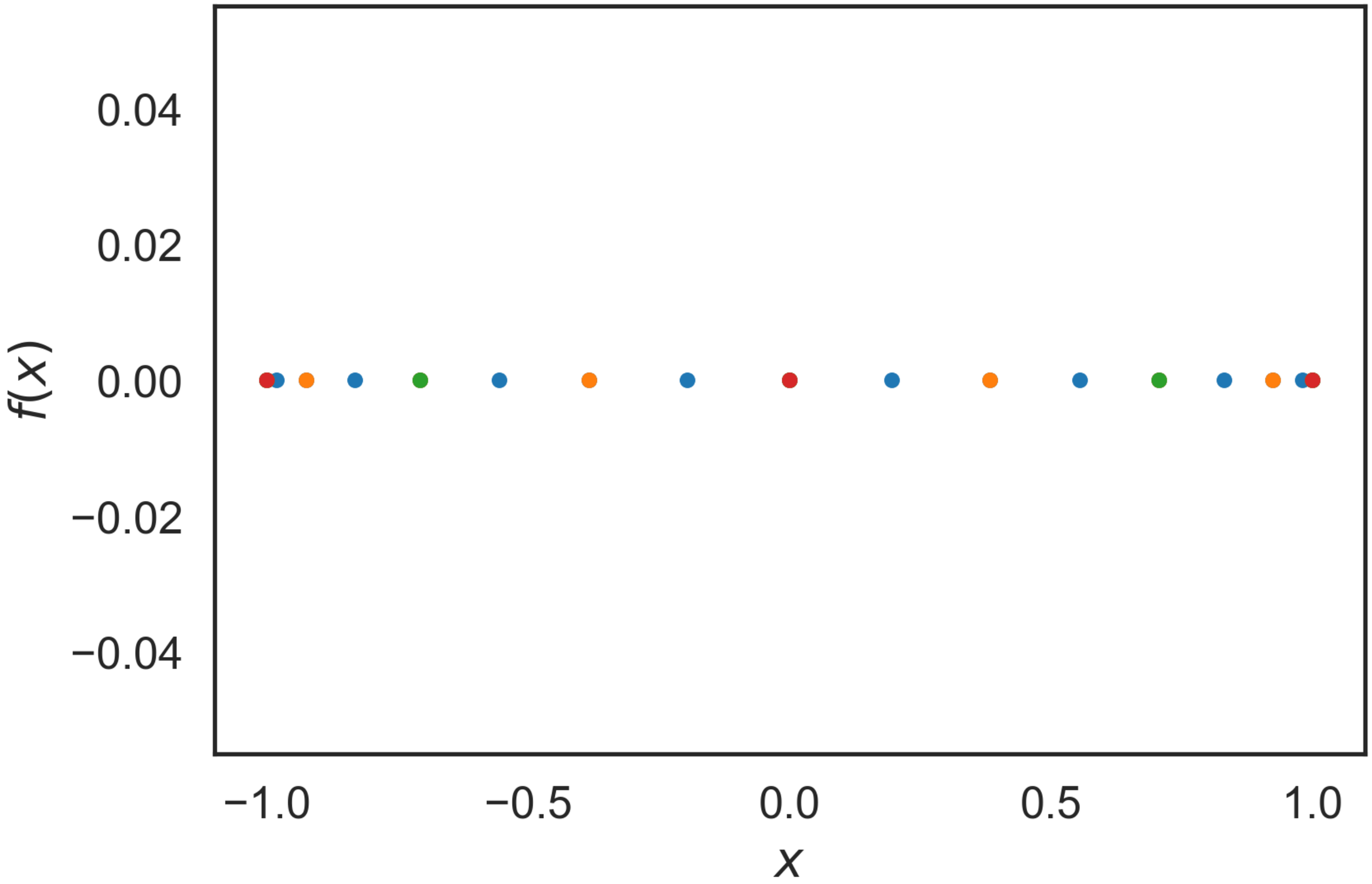
By matching terms we can find the weights.

Clenshaw-Curtis Quadrature

Theorem 1: CC is nested (double n).

Theorem 2: CC integrates polynomials of degree n + 1.

We will only use CC from now on.


0.04
0.02
0.00
−0.02
−0.04
f(x)
−1.0
−0.5
0.0
0.5
1.0
x

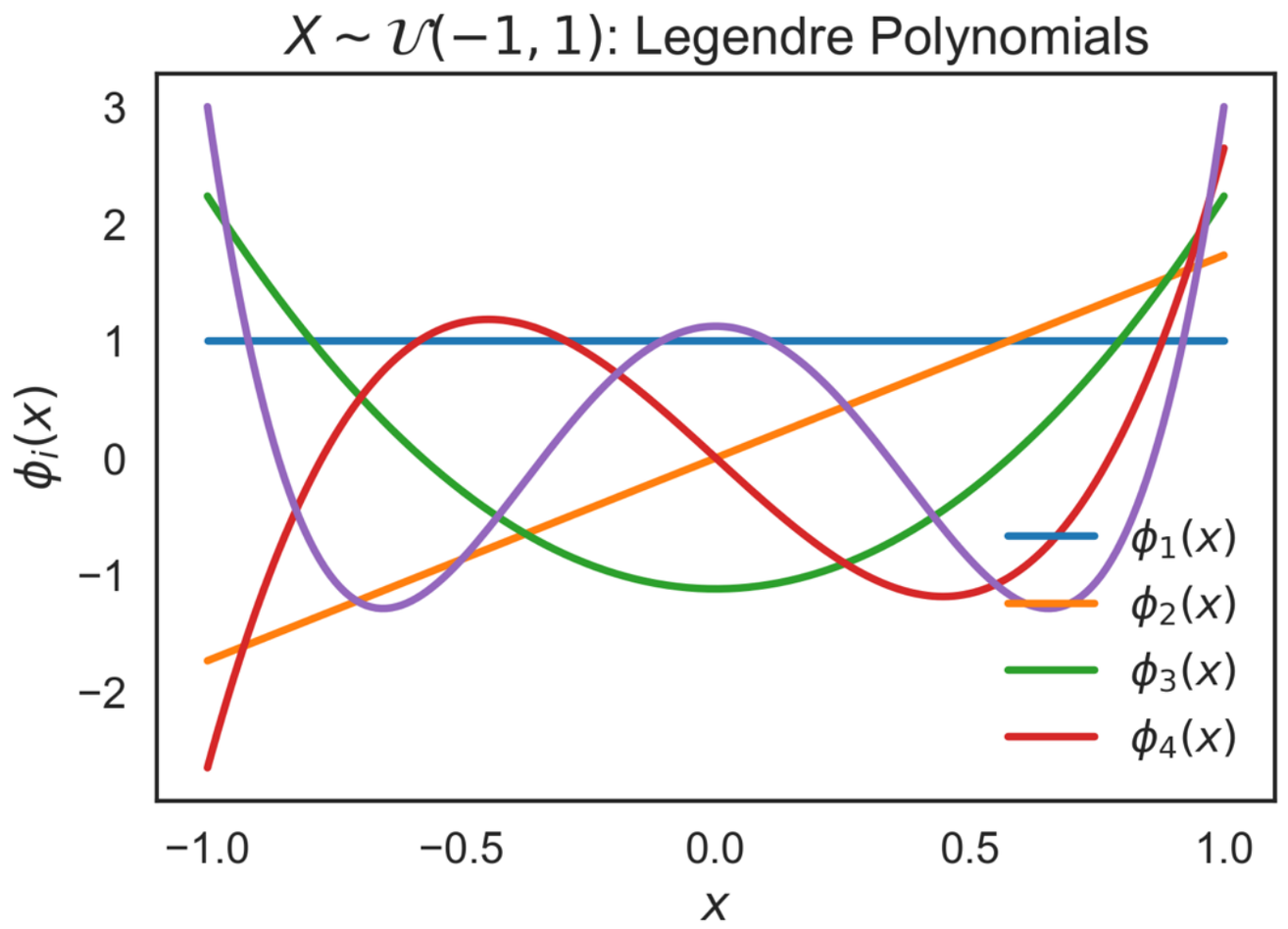
Example 1: Verify the Legendre Polynomials are Orthogonal

We use the CC rule to find:

$$\int_{-1}^{1} \phi_i(x)\phi_j(x)\frac{1}{2}dx \approx \sum_{k=1}^{n} \frac{w_k}{2}\phi_i(x_k)\phi_j(x_k)$$

Result:

```
<0, 0>   = 1.000
<0, 1>   = 0.000
<0, 2>   = -0.000
<0, 3>   = 0.000
<0, 4>   = 0.000
<1, 1>   = 1.000
<1, 2>   = 0.000
<1, 3>   = 0.000
<1, 4>   = 0.000
<2, 2>   = 1.000
<2, 3>   = 0.000
<2, 4>   = 0.000
<3, 3>   = 1.000
<3, 4>   = 0.000
<4, 4>   = 1.000
```



Evaluating Arbitrary Expectations with the CC Quadrature Rule

CC only works with: $$\int_{-1}^{1} f(x)dx$$

We need: $$\mathbb{E}[f(X)] := \int_{-\infty}^{\infty} f(x)p(x)dx$$

Transform the integration using the CDF of p(x): $z = 2F(x) - 1$

$$x = F^{-1}\left(\frac{z+1}{2}\right) \quad dz = 2F'(x)dx = 2p(x)dx$$

$$\mathbb{E}[f(X)] = \frac{1}{2}\int_{-1}^{1} f\left(F^{-1}\left(\frac{z+1}{2}\right)\right)dz$$

Evaluating Arbitrary Expectations with the CC Quadrature Rule

CC only works with: $$\int_{-1}^{1} f(x)dx$$

We need: $$\mathbb{E}[f(X)] := \int_{-\infty}^{\infty} f(x)p(x)dx$$

Transform the integration using the CDF of p(x): $z = 2F(x) - 1$

If v_k and z_k are weights and nodes of the CC rule, then use:

$$w_k = \frac{1}{2}v_k \qquad x_k = F^{-1}\left(\frac{z_k + 1}{2}\right)$$

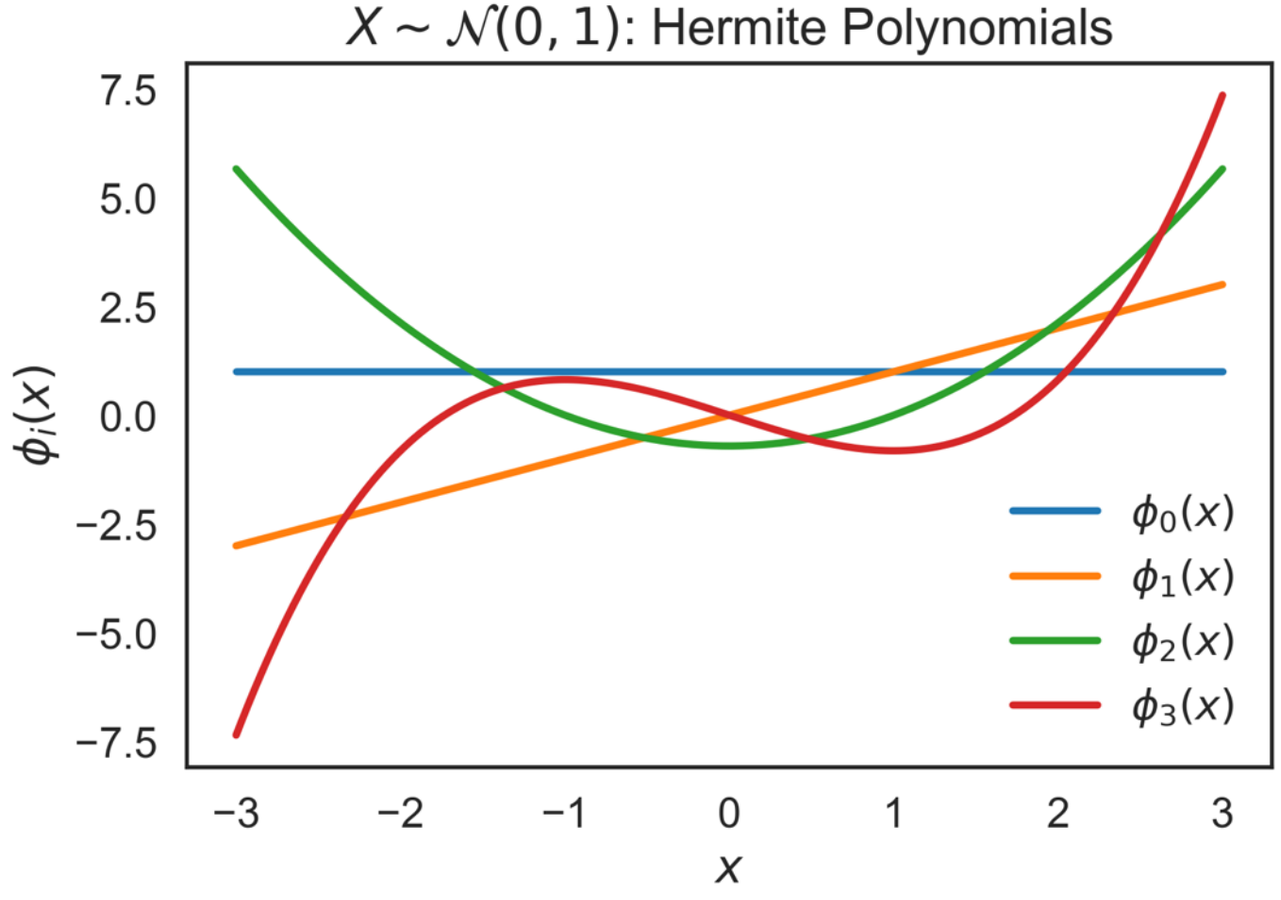
Example 2: Verify the Hermite Polynomials are Orthogonal

We use the CC (open) rule to find:

$$\int_{-\infty}^{\infty} \phi_i(x)\phi_j(x)\mathcal{N}(x|0,1)dx \approx \sum_{k=1}^{n} \frac{v_k}{2} \phi_i\left(\Phi^{-1}\left(\frac{z_k+1}{2}\right)\right) \phi_j\left(\Phi^{-1}\left(\frac{z_k+1}{2}\right)\right)$$

Result:

```
<0, 0>   = 1.000
<0, 1>   = 0.000
<0, 2>   = 0.000
<0, 3>   = -0.000
<1, 1>   = 1.000
<1, 2>   = -0.000
<1, 3>   = 0.000
<2, 2>   = 1.000
<2, 3>   = -0.000
<3, 3>   = 1.000
```



Quadrature Rules in High-Dimensions

Tensor Products of Quadrature Rules

Consider a 1D quadrature rule:

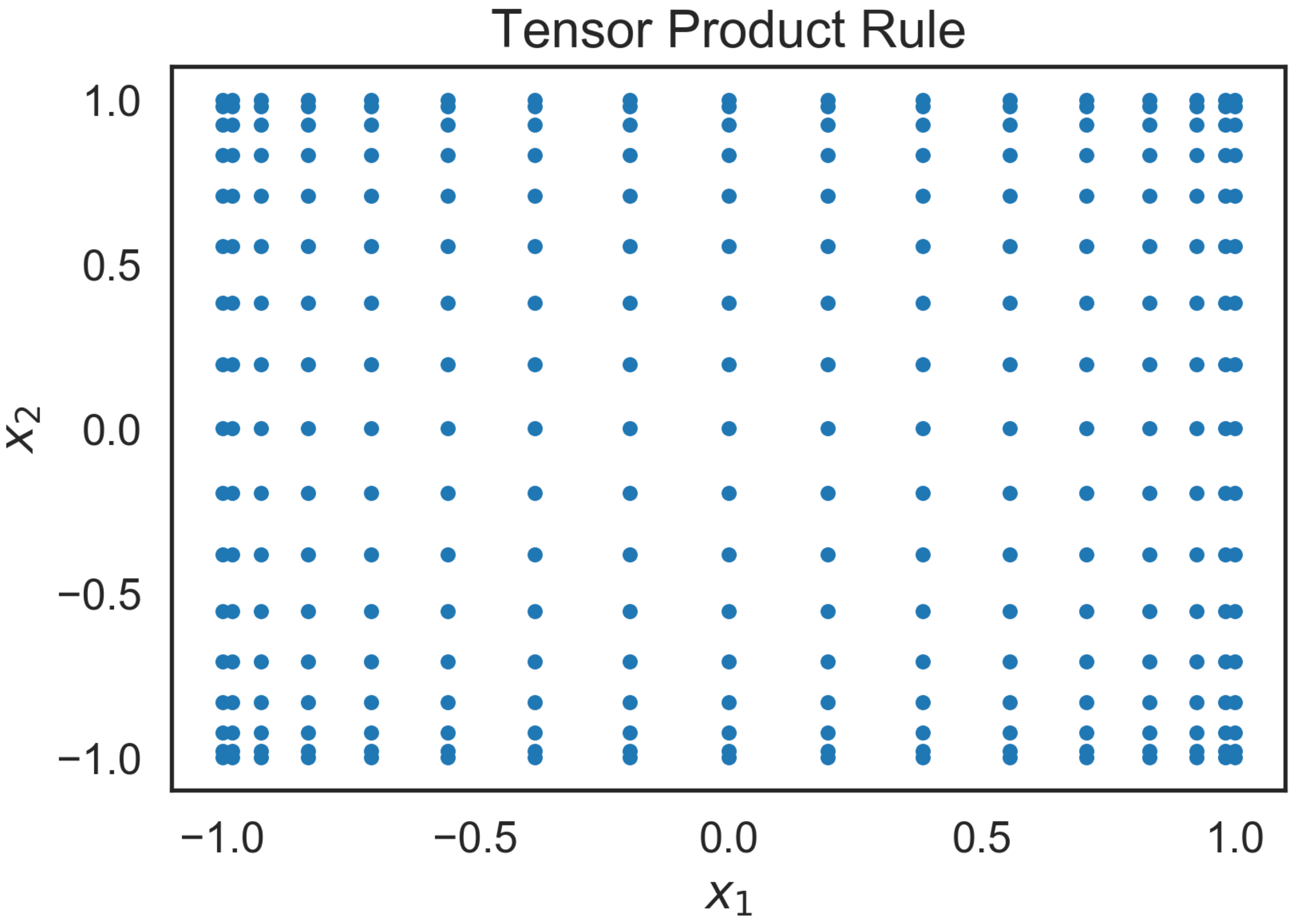
$$Q^{(1)}(f) = \sum_{k=1}^{n} w_k f(x_k)$$

The tensor product:

$$Q^{(2)} = Q^{(1)} \otimes Q^{(1)}$$

Is defined by:

$$Q^{(2)}(f) = \left(Q^{(1)} \otimes Q^{(1)}\right)(f) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j f(x_i, x_j)$$


Tensor Product Rule
1.0
0.5
0.0
−0.5
−1.0
x_2
−1.0
−0.5
0.0
0.5
1.0
x_1

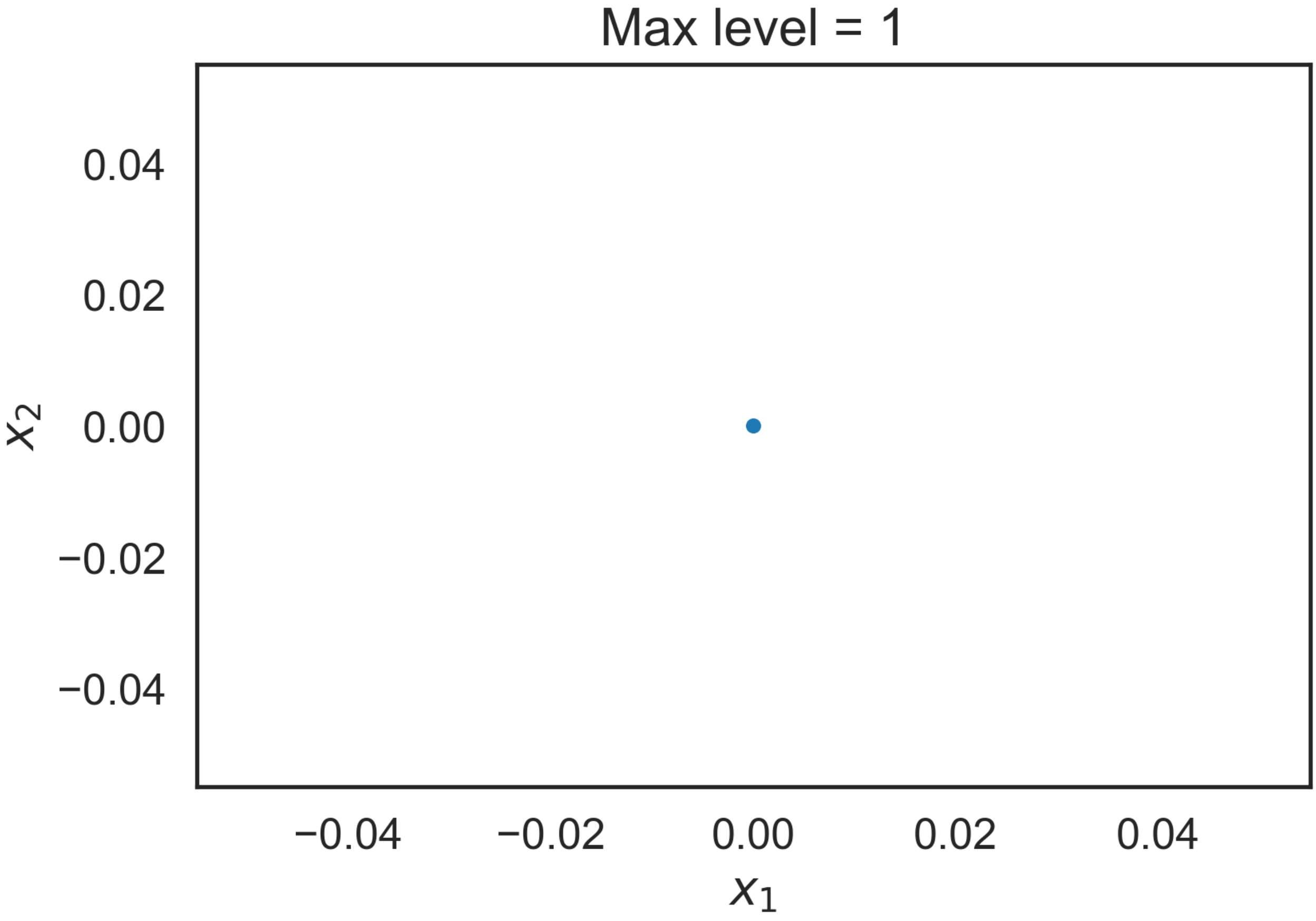
Tensor products grow to fast with dimensionality

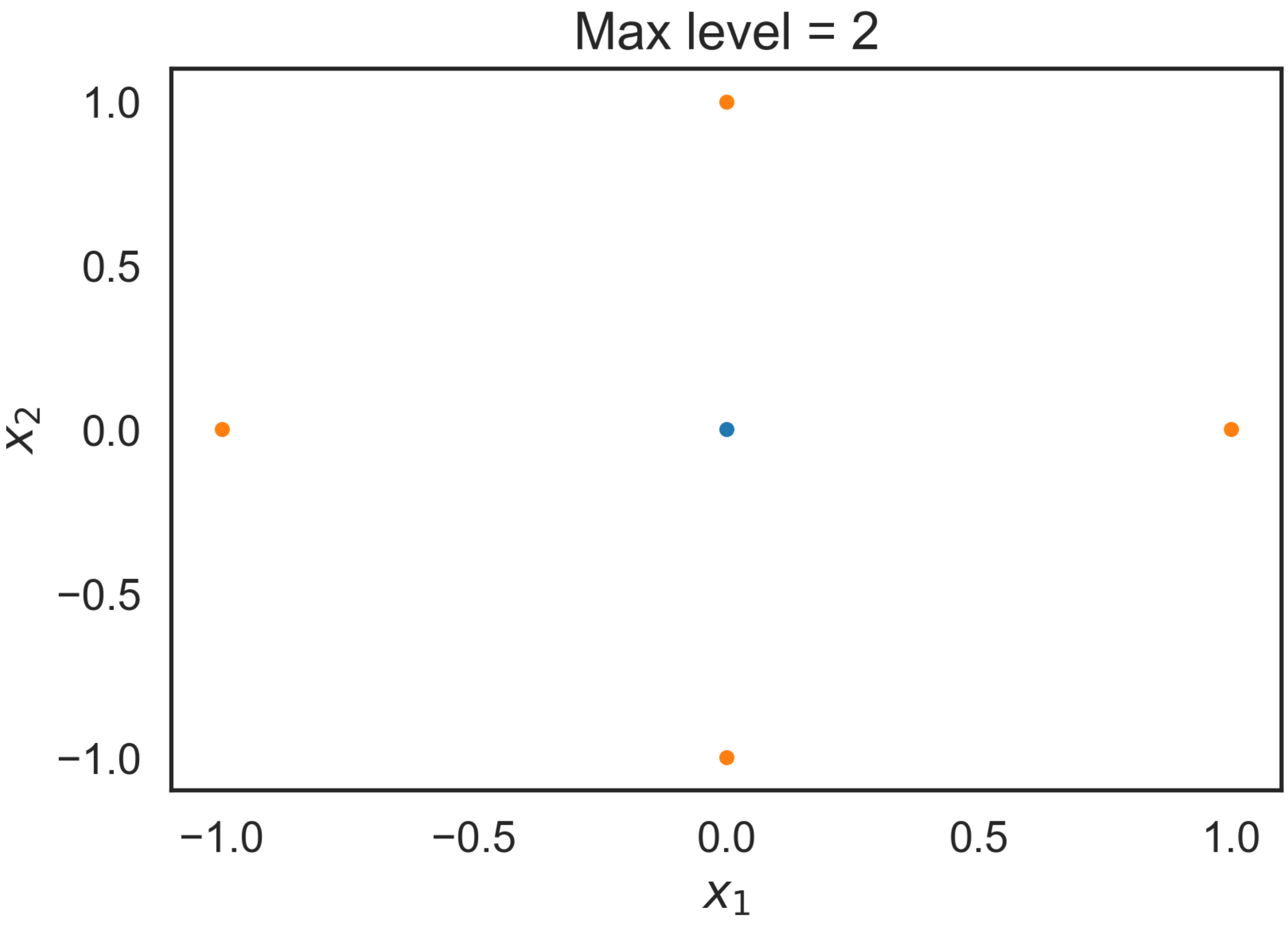
Sparse Grids

Pick any 1D nested, quadrature rule: $Q_\ell^{(1)}$

Define recursively the following rules (Smolyak quadrature):

$$Q_\ell^{(d)}(f) := \left(\sum_{i=1}^{\ell} \left(Q_i^{(1)} - Q_{i-1}^{(1)} \right) \otimes Q_{\ell-i+1}^{(d-1)} \right) (f)$$


Max level = 1
0.04
0.02
0.00
−0.02
−0.04
x_2
−0.04
−0.02
0.00
0.02
0.04
x_1


Max level = 2
1.0
0.5
0.0
−0.5
−1.0
x_2
−1.0
−0.5
0.0
0.5
1.0
x_1

Max level = 3

1.0
0.5
0.0
−0.5
−1.0
x_2

−1.0
−0.5
0.0
0.5
1.0
x_1

Max level = 4

1.0
0.5
0.0
−0.5
−1.0
x_2

−1.0
−0.5
0.0
0.5
1.0
x_1

Max level = 5

1.0
0.5
0.0
−0.5
−1.0
x_2

−1.0
−0.5
0.0
0.5
1.0
x_1

Max level = 6

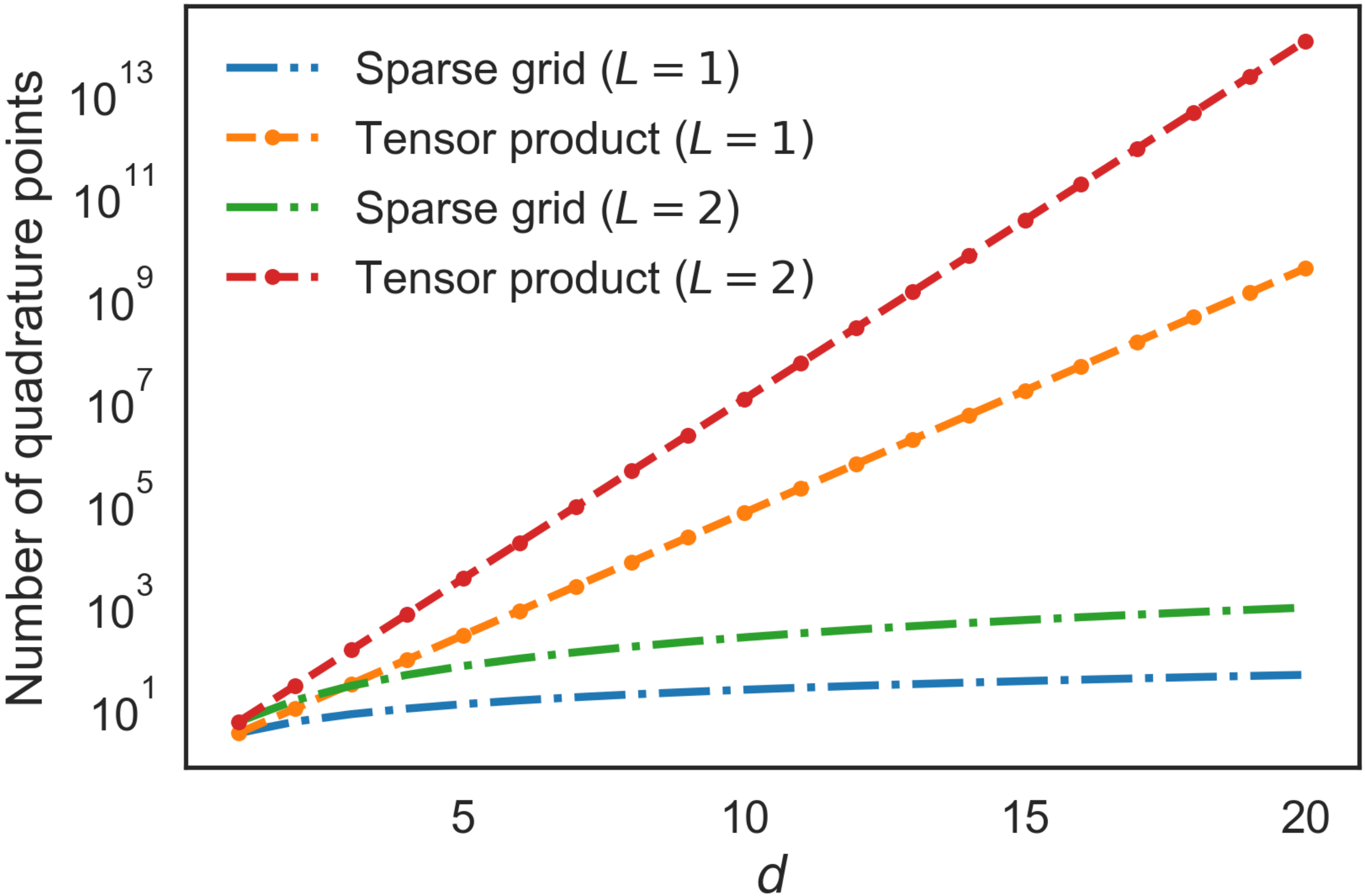
1.0
0.5
0.0
−0.5
−1.0

x_2

−1.0 −0.5 0.0 0.5 1.0

x_1

The number of terms in sparse grids grows much slower than tensor products


Sparse grid ($L = 1$)
Tensor product ($L = 1$)
Sparse grid ($L = 2$)
Tensor product ($L = 2$)
Number of quadrature points
10^{13}
10^{11}
10^{9}
10^{7}
10^{5}
10^{3}
10^{1}
5
10
15
20
d

Hands-on Examples (as much as you can until the end of the lecture)