Title: Front End - Mathematica - 2

Date: Aug 26, 2016 10:30 AM

URL: http://pirsa.org/16080085

Abstract:

Solving Partial Differential Equations Approximately

PSI Lecture, 2016-08-26

Erik Schnetter

Discretization

Basis functions

```
n = 4

Plot[Table[ChebyshevT[i, x], {i, 0, n}], {x, -1, +1}]

cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]
```

$$2 \int_{-1}^{+1} \frac{f[x]}{\sqrt{}} \, dx$$

150%

Approximately

PSI Lecture, 2016-08-26

Erik Schnetter

1. Approximate functions on a manifold
2. Find discrete versions of operatiors (e.g. derivative operator)
3. Solve a PDE (stationary heat equation)

Discretization

Basis functions

```
n = 4
```

```
Plot[Table[ChebyshevT[i, x], {i, 0, n}], {x, -1, +1}]
```

```
cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]
```

$$\texttt{chebInt}[f\_] := \frac{2 \int_{-1}^{+1} \frac{f[x]}{\sqrt{1-x^2}}\, dx}{\pi}$$

150%

Approximately

PSI Lecture, 2016-08-26

Erik Schnetter

1. Approximate functions on a manifold
2. Find discrete versions of operatiors (e.g. derivative operator)
3. Solve a PDE (stationary heat equation)

Discretization

Manifold: Interval [-1; +1]

Basis functions

```
n = 4
```

```
Plot[Table[ChebyshevT[i, x], {i, 0, n}], {x, -1, +1}]
```

```
cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]
```

$$2 \int_{+1} \frac{f[x]}{} dx$$

150% ▸

Approximately

PSI Lecture, 2016-08-26

Erik Schnetter

1. Approximate functions on a manifold
2. Find discrete versions of operatiors (e.g. derivative operator)
3. Solve a PDE (stationary heat equation)

Discretization

Manifold: Interval [-1; +1]
Choose basis
Truncate basis
Represent functions as coefficients for that basis

Basis functions

```
n = 4

Plot[Table[ChebyshevT[i, x], {i, 0, n}], {x, -1, +1}]
```

Discretization

Manifold: Interval [-1; +1]
Choose basis
Truncate basis
Represent functions as coefficients for that basis

Basis functions

Keep 4 basis functions:

In[1]:=
```
n = 4
```

Out[1]=
```
4
```

```
Plot[Table[ChebyshevT[i, x], {i, 0, n}], {x, -1, +1}]
```

```
cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]
```

$$\text{chebInt}[f\_] := \frac{2\int_{-1}^{+1} \frac{f[x]}{\sqrt{1-x^2}}\,dx}{\pi}$$

```
Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

150%

Discretization

Manifold: Interval [-1; +1]
Choose basis
Truncate basis
Represent functions as coefficients for that basis

Basis functions

Keep 4 basis functions:

In[1]:=
```
n = 4
```
Out[1]=
```
4
```

Choose Chebyshev functions as basis

```
Plot[Table[ChebyshevT[i, x], {i, 0, n}], {x, -1, +1}]
```

```
cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]
```

$$\text{chebInt}[f\_] := \frac{2\int_{-1}^{+1} \frac{f[x]}{\sqrt{1-x^2}}\,dx}{\pi}$$

```
Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

150%

Basis functions

Keep 4 basis functions:

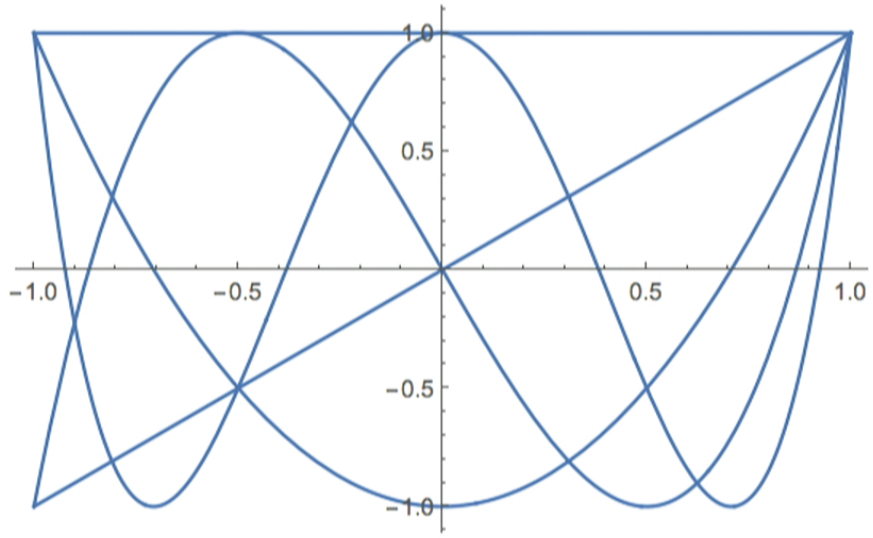
In[1]:=
```
n = 4
```

Out[1]= 4

Choose Chebyshev functions as basis

In[2]:=
```
Plot[Table[ChebyshevT[i, x], {i, 0, n}], {x, -1, +1}]
```

Out[2]=

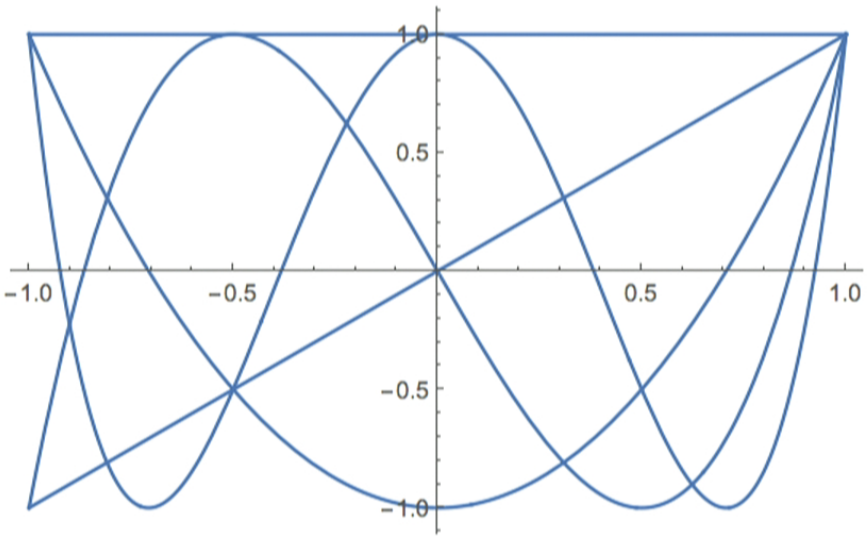



```
cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]
```

150%

In[2]:=
```
Plot[Table[ChebyshevT[i, x], {i, 0, n}], {x, -1, +1}]
```

Out[2]=



Chebyshev functions are orthogonal

$$\texttt{cheb}[i\_, x\_] := \texttt{If}\left[i == 0, \frac{1}{\sqrt{2}}, 1\right] \texttt{ChebyshevT}[i, x]$$

$$\texttt{chebInt}[f\_] := \frac{2\int_{-1}^{+1} \frac{f[x]}{\sqrt{1-x^2}}\, dx}{\pi}$$

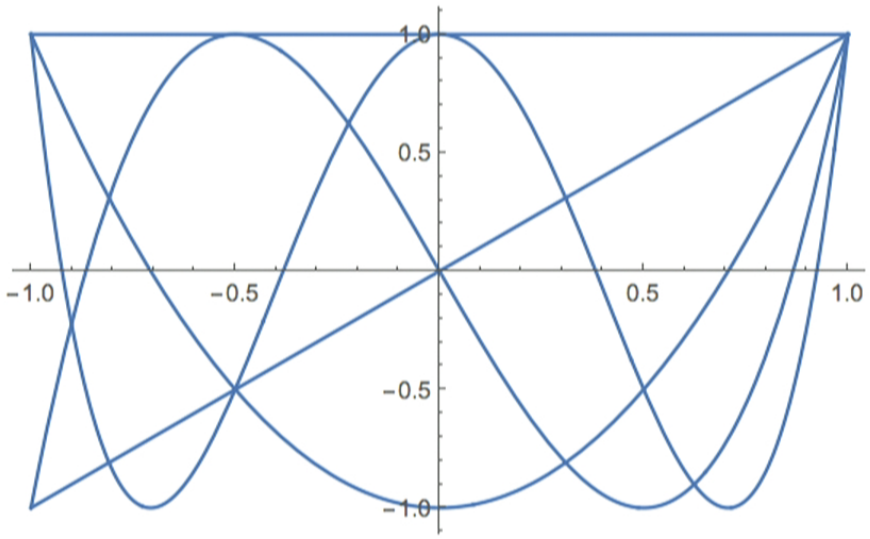
```
Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

Reconstruction and Projection onto basis

150% ▸

In[2]:=
```
Plot[Table[ChebyshevT[i, x], {i, 0, n}], {x, -1, +1}]
```

Out[2]=



Chebyshev functions are orthogonal with respect to this integral

$$\text{chebInt}[f\_] := \left| \frac{2\int_{-1}^{+1} \frac{f[x]}{\sqrt{1-x^2}}\, dx}{\pi} \right|$$

```
cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]
```

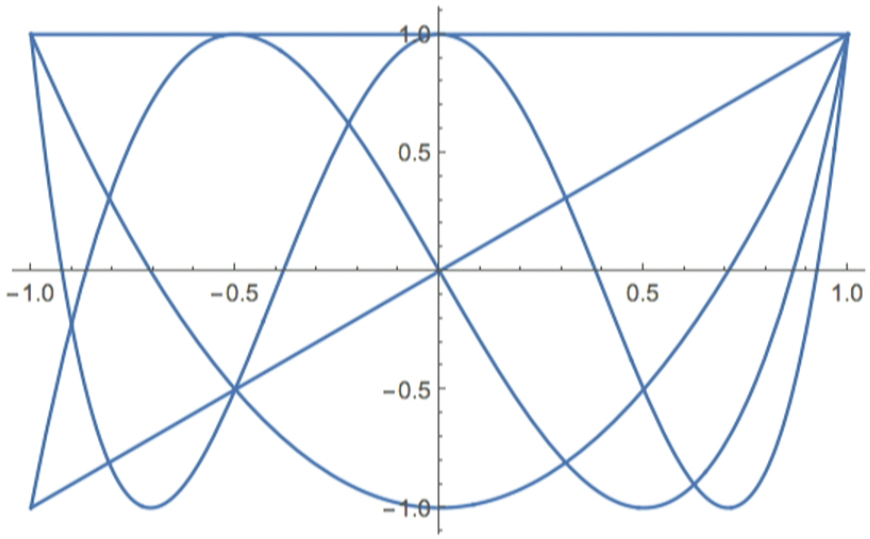
```
Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

Reconstruction and Projection onto basis

150%

In[2]:=
```
Plot[Table[ChebyshevT[i, x], {i, 0, n}], {x, -1, +1}]
```

Out[2]=



Chebyshev functions are orthogonal with respect to this integral

In[3]:=
$$\text{chebInt}[f\_] := \frac{2\int_{-1}^{+1} \frac{f[x]}{\sqrt{1-x^2}}\,dx}{\pi}$$

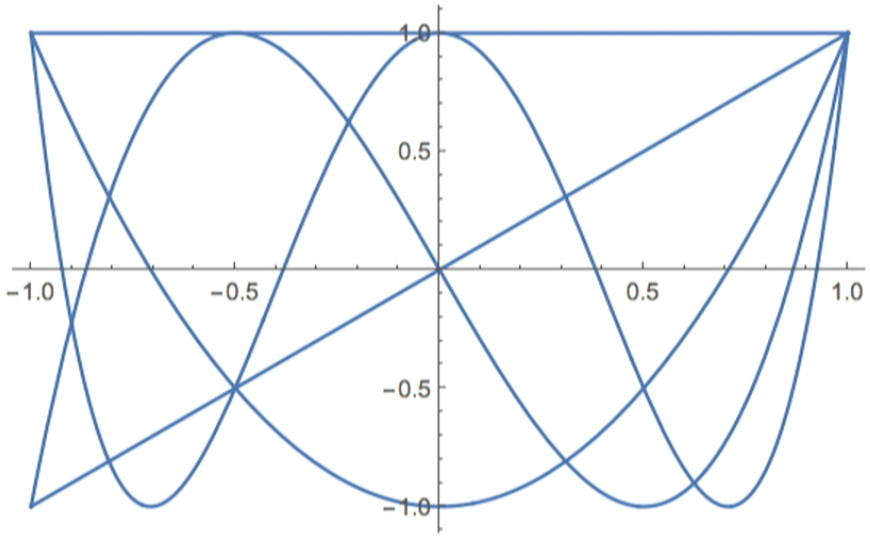
```
chebInt[ChebyshevT[0, #] ChebyshevT[0, #]]|
```

```
cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]
```

```
Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

150% ▸

Out[2]=



Chebyshev functions are orthogonal with respect to this integral

In[3]:=
```
chebInt[f_] := 2 Integrate[f[x]/Sqrt[1-x^2], {x, -1, +1}, Assumptions -> Element[x, Reals]]/Pi
```

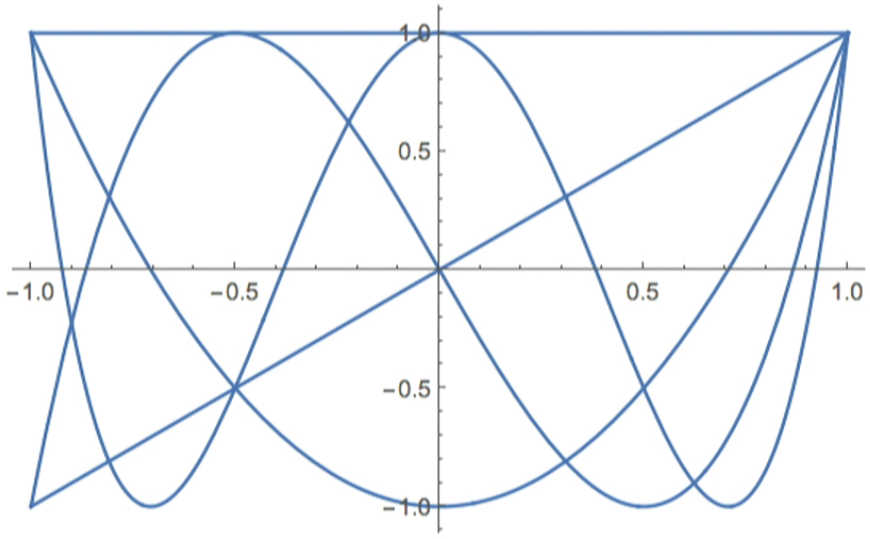
In[5]:=
```
chebInt[ChebyshevT[0, #] ChebyshevT[0, #] &]
```

Out[5]= 2

```
cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]

Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

150%

Out[2]=



Chebyshev functions are orthogonal with respect to this integral

In[3]:=
$$\text{chebInt}[f\_] := \frac{2\int_{-1}^{+1} \frac{f[x]}{\sqrt{1-x^2}}\, dx}{\pi}$$

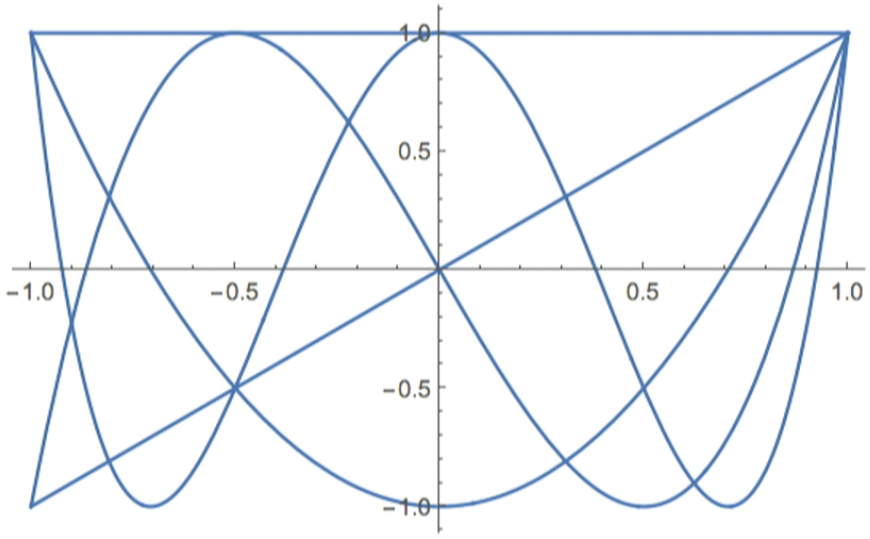
```
chebInt[ChebyshevT[1, #] ChebyshevT[1, #] &]
```

Out[5]= 2

```
cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]

Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

150%

Out[2]=



Chebyshev functions are orthogonal with respect to this integral

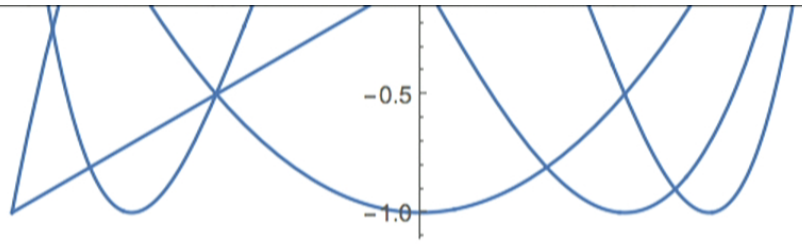
In[3]:=
```
chebInt[f_] := 2 Integrate[f[x]/Sqrt[1-x^2], {x, -1, +1}, dx] / π
```

In[7]:=
```
chebInt[ChebyshevT[0, #] ChebyshevT[1, #] &]
```

Out[7]= 0

```
cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]

Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

150%



Chebyshev functions are orthogonal with respect to this integral

In[3]:=
$$\text{chebInt}[f\_] := \frac{2\int_{-1}^{+1} \frac{f[x]}{\sqrt{1-x^2}}\, dx}{\pi}$$

In[7]:=
```
chebInt[ChebyshevT[0, #] ChebyshevT[1, #] &]
```

Out[7]= 0

Define modified orthonormal Chebyshev functions

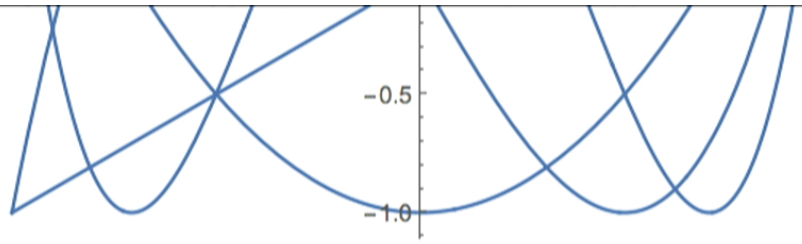
$$\text{cheb}[i\_, x\_] := \text{If}\left[i == 0, \frac{1}{\sqrt{2}}, 1\right] \text{ChebyshevT}[i, x]$$

```
Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

Reconstruction and Projection onto basis

$$\text{c2f}[cs\_] := \text{Function}\left[x, \sum_{i=0}^{n} cs[[i+1]]\, \text{cheb}[i, x]\right]$$

150%



Chebyshev functions are orthogonal with respect to this integral

In[3]:=
$$\text{chebInt}[f\_] := \frac{2\int_{-1}^{+1} \frac{f[x]}{\sqrt{1-x^2}}\, dx}{\pi}$$

In[7]:=
```
chebInt[ChebyshevT[0, #] ChebyshevT[1, #] &]
```

Out[7]= 0

Define modified orthonormal Chebyshev functions

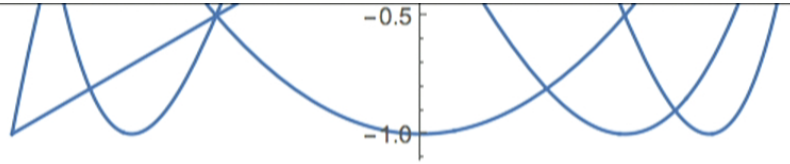
In[8]:=
$$\text{cheb}[i\_, x\_] := \text{If}\left[i == 0, \frac{1}{\sqrt{2}}, 1\right] \text{ChebyshevT}[i, x]$$

```
Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

Reconstruction and Projection onto basis

$$\text{c2f}[cs\_] := \text{Function}\left[x, \sum_{i=0}^{n} cs[[i+1]]\, \text{cheb}[i, x]\right]$$

150%



Chebyshev functions are orthogonal with respect to this integral

In[3]:=
$$\text{chebInt}[f\_] := \frac{2\int_{-1}^{+1} \frac{f[x]}{\sqrt{1-x^2}}\,dx}{\pi}$$

```
In[7]:= chebInt[ChebyshevT[0, #] ChebyshevT[1, #] &]
```

Out[7]= 0

Define modified orthonormal Chebyshev functions

In[8]:=
$$\text{cheb}[i\_, x\_] := \text{If}\left[i == 0, \frac{1}{\sqrt{2}}, 1\right] \text{ChebyshevT}[i, x]$$

```
In[9]:= Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

Out[9]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Reconstruction and Projection onto basis

150% ▸

In[3]:= $\texttt{chebInt}[f\_] := \frac{\quad}{\pi}$

In[7]:= `chebInt[ChebyshevT[0, #] ChebyshevT[1, #] &]`

Out[7]= 0

Define modified orthonormal Chebyshev functions

In[8]:= $\texttt{cheb}[i\_, x\_] := \texttt{If}\left[i == 0, \frac{1}{\sqrt{2}}, 1\right] \texttt{ChebyshevT}[i, x]$

Check orthonormality

In[9]:= `Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm`

Out[9]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Reconstruction and Projection onto basis

$$\texttt{c2f}[cs\_] := \texttt{Function}\left[x, \sum_{i=0}^{n} cs[\![i + 1]\!] \texttt{cheb}[i, x]\right]$$

`f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]`

150%

Define modified orthonormal Chebyshev functions

In[8]:=
```
cheb[i_, x_] := If[i == 0, 1/Sqrt[2], 1] ChebyshevT[i, x]
```

Check orthonormality

In[9]:=
```
Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

Out[9]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Reconstruction and Projection onto basis

```
c2f[cs_] := Function[x, Sum[cs[[i + 1]] cheb[i, x], {i, 0, n}]]

f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]

cs = Table[c_i, {i, 0, n}]

cs // c2f

cs // c2f // f2c
```

150%

$\sqrt{2}$

Check orthonormality

In[9]:=
```
Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

Out[9]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Reconstruction and Projection onto basis

Approximated functions are represented as a vector of basis coefficients

c2f converts such a vector to a function

```
c2f[cs_] := Function[x, Sum[cs[[i + 1]] cheb[i, x], {i, 0, n}]]

f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]

cs = Table[c_i, {i, 0, n}]

cs // c2f

cs // c2f // f2c
```

150%

$\sqrt{2}$

Check orthonormality

In[9]:=
```
Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

Out[9]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Reconstruction and Projection onto basis

Approximated functions are represented as a vector of basis coefficients

c2f converts such a vector to a function

In[10]:=
```
c2f[cs_] := Function[x, Sum[cs[[i + 1]] cheb[i, x], {i, 0, n}]]
```

In[12]:=
```
c2f[{1, 0, 1, 0, 0}][x]
```

Out[12]= $-1 + \frac{1}{\sqrt{2}} + 2\,x^2$

```
f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]
```

150%

Check orthonormality

In[9]:=
```
Table[chebInt[cheb[i, #] cheb[j, #] &], {i, 0, n}, {j, 0, n}] // MatrixForm
```

Out[9]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Reconstruction and Projection onto basis

Approximated functions are represented as a vector of basis coefficients

c2f converts such a vector to a function

In[10]:=
```
c2f[cs_] := Function[x, Sum[cs[[i + 1]] cheb[i, x], {i, 0, n}]]
```

In[15]:=
```
c2f[{Sqrt[2]/2, 0, 1/2, 0, 0}][x] // Simplify
```

Out[15]=
$$x^2$$

```
f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]

cs = Table[c_i, {i, 0, n}]
```

150%

Reconstruction and Projection onto basis

Approximated functions are represented as a vector of basis coefficients

c2f converts such a vector to a function

In[10]:=
```
c2f[cs_] := Function[x, Sum[cs[[i + 1]] cheb[i, x], {i, 0, n}]]
```

Example:

In[15]:=
```
c2f[{Sqrt[2]/2, 0, 1/2, 0, 0}][x] // Simplify
```

Out[15]=
$$x^2$$

Project any function f onto our set of basis vectors

```
f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]
```

```
cs = Table[c_i, {i, 0, n}]
```

```
cs // c2f
```

```
cs // c2f // f2c
```

Derivatives

150%

Reconstruction and Projection onto basis

Approximated functions are represented as a vector of basis coefficients

c2f converts such a vector to a function

In[10]:=
```
c2f[cs_] := Function[x, Sum[cs[[i + 1]] cheb[i, x], {i, 0, n}]]
```

Example:

In[15]:=
```
c2f[{Sqrt[2]/2, 0, 1/2, 0, 0}][x] // Simplify
```

Out[15]=
$$x^2$$

Project any function f onto our set of basis vectors

In[16]:=
```
f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]
```

Example

In[17]:=
```
f2c[#^2 &]
```

Out[17]=
$$\left\{\frac{1}{\sqrt{2}}, 0, \frac{1}{2}, 0, 0\right\}$$

```
cs = Table[c_i, {i, 0, n}]

cs // c2f
```

150%

In[10]:=
```
c2f[cs_] := Function[x, Sum[cs[[i + 1]] cheb[i, x], {i, 0, n}]]
```

Example:

In[15]:=
```
c2f[{Sqrt[2]/2, 0, 1/2, 0, 0}][x] // Simplify
```

Out[15]=
$$x^2$$

Project any function f onto our set of basis vectors

In[16]:=
```
f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]
```

Example

In[17]:=
```
f2c[#^2 &]
```

Out[17]=
$$\left\{\frac{1}{\sqrt{2}}, 0, \frac{1}{2}, 0, 0\right\}$$

In[18]:=
```
f2c[Sin]
```

Out[18]=
```
{0, 2 BesselJ[1, 1], 0, -2 BesselJ[3, 1], 0}
```

```
cs = Table[c_i, {i, 0, n}]
```

```
cs // c2f
```

150%

In[10]:=
```
c2f[cs_] := Function[x, Sum[cs[[i + 1]] cheb[i, x], {i, 0, n}]]
```

Example:

In[15]:=
```
c2f[{Sqrt[2]/2, 0, 1/2, 0, 0}][x] // Simplify
```

Out[15]=
$$x^2$$

Project any function f onto our set of basis vectors

In[16]:=
```
f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]
```

Example

In[17]:=
```
f2c[#^2 &]
```

Out[17]=
$$\left\{\frac{1}{\sqrt{2}}, 0, \frac{1}{2}, 0, 0\right\}$$

In[18]:=
```
f2c[Sin]
```

Out[18]=
```
{0, 2 BesselJ[1, 1], 0, -2 BesselJ[3, 1], 0}
```

In[19]:=
```
f2c[Sin] // N
```

Out[19]=
```
{0., 0.880101, 0., -0.0391267, 0.}
```

```
cs = Table[c_i, {i, 0, n}]
```

150%

In[15]:=
```
c2f[{Sqrt[2]/2, 0, 1/2, 0, 0}][x] // Simplify
```

Out[15]=
$$x^2$$

Project any function f onto our set of basis vectors

In[16]:=
```
f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]
```

Example

In[17]:=
```
f2c[#^2 &]
```

Out[17]=
$$\left\{\frac{1}{\sqrt{2}}, 0, \frac{1}{2}, 0, 0\right\}$$

In[18]:=
```
f2c[Sin]
```

Out[18]=
```
{0, 2 BesselJ[1, 1], 0, -2 BesselJ[3, 1], 0}
```

In[19]:=
```
f2c[Sin] // N
```

Out[19]=
```
{0., 0.880101, 0., -0.0391267, 0.}
```

In[20]:=
```
c2f[f2c[Sin]]
```

Out[20]=
$$\text{Function}\left[x\$, \sum_{i=0}^{n} \{0, 2\,\text{BesselJ}[1, 1], 0, -2\,\text{BesselJ}[3, 1], 0\}[\![i + 1]\!]\,\text{cheb}[i, x\$]\right]$$

```
cs = Table[c_i, {i, 0, n}]
```

150%

In[15]:= `c2f[{Sqrt[2]/2, 0, 1/2, 0, 0}][x] // Simplify`

Out[15]= x^2

Project any function f onto our set of basis vectors

In[16]:=
```
f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]
```

Example

In[17]:= `f2c[#^2 &]`

Out[17]= $\left\{\frac{1}{\sqrt{2}}, 0, \frac{1}{2}, 0, 0\right\}$

In[18]:= `f2c[Sin]`

Out[18]= `{0, 2 BesselJ[1, 1], 0, -2 BesselJ[3, 1], 0}`

In[19]:= `f2c[Sin] // N`

Out[19]= `{0., 0.880101, 0., -0.0391267, 0.}`

In[21]:= `c2f[f2c[Sin]][x]`

Out[21]= $2\,x\,\text{BesselJ}[1, 1] - 2\left(-3\,x + 4\,x^3\right)\text{BesselJ}[3, 1]$

```
cs = Table[c_i, {i, 0, n}]

cs // c2f
```

150%

In[15]:= `c2f[{Sqrt[2]/2, 0, 1/2, 0, 0}][x] // Simplify`

Out[15]= x^2

Project any function f onto our set of basis vectors

In[16]:=
```
f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]
```

Example

In[17]:= `f2c[#^2 &]`

Out[17]= $\left\{\frac{1}{\sqrt{2}}, 0, \frac{1}{2}, 0, 0\right\}$

In[18]:= `f2c[Sin]`

Out[18]= `{0, 2 BesselJ[1, 1], 0, -2 BesselJ[3, 1], 0}`

In[19]:= `f2c[Sin] // N`

Out[19]= `{0., 0.880101, 0., -0.0391267, 0.}`

In[21]:= `c2f[f2c[Sin]][x]`

Out[21]= $2\,x\,\text{BesselJ}[1, 1] - 2\left(-3\,x + 4\,x^3\right)\text{BesselJ}[3, 1]$

```
cs = Table[c_i, {i, 0, n}]

cs // c2f
```

150%

In[16]:=
```
f2c[f_] := Table[chebInt[f[#] cheb[i, #] &], {i, 0, n}]
```

Example

In[17]:=
```
f2c[#^2 &]
```

Out[17]=
$$\left\{\frac{1}{\sqrt{2}}, 0, \frac{1}{2}, 0, 0\right\}$$

In[18]:=
```
f2c[Sin]
```

Out[18]=
```
{0, 2 BesselJ[1, 1], 0, -2 BesselJ[3, 1], 0}
```

In[19]:=
```
f2c[Sin] // N
```

Out[19]=
```
{0., 0.880101, 0., -0.0391267, 0.}
```

In[21]:=
```
c2f[f2c[Sin]][x]
```

Out[21]=
$$2\,x\,\text{BesselJ}[1, 1] - 2\left(-3\,x + 4\,x^3\right)\text{BesselJ}[3, 1]$$

Define generic coefficient

In[22]:=
```
cs = Table[c_i, {i, 0, n}]
```

Out[22]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

```
cs // c2f
```

```
cs // c2f // f2c
```

150%

Out[17]= $\left\{\frac{1}{\sqrt{2}}, 0, \frac{1}{2}, 0, 0\right\}$

In[18]:=
```
f2c[Sin]
```
Out[18]=
```
{0, 2 BesselJ[1, 1], 0, -2 BesselJ[3, 1], 0}
```

In[19]:=
```
f2c[Sin] // N
```
Out[19]=
```
{0., 0.880101, 0., -0.0391267, 0.}
```

In[21]:=
```
c2f[f2c[Sin]][x]
```
Out[21]= $2\,x\,\text{BesselJ}[1, 1] - 2\left(-3\,x + 4\,x^3\right)\text{BesselJ}[3, 1]$

Define generic coefficient

In[22]:=
```
cs = Table[cᵢ, {i, 0, n}]
```
Out[22]= $\{c_0, c_1, c_2, c_3, c_4\}$

In[24]:=
```
(cs // c2f)[x]
```
Out[24]= $\frac{c_0}{\sqrt{2}} + x\,c_1 + \left(-1 + 2\,x^2\right)c_2 + \left(-3\,x + 4\,x^3\right)c_3 + \left(1 - 8\,x^2 + 8\,x^4\right)c_4$

```
cs // c2f // f2c
```

Derivatives

Out[17]= $\left\{\frac{1}{\sqrt{2}}, 0, \frac{1}{2}, 0, 0\right\}$

In[18]:=
```
f2c[Sin]
```
Out[18]=
```
{0, 2 BesselJ[1, 1], 0, -2 BesselJ[3, 1], 0}
```

In[19]:=
```
f2c[Sin] // N
```
Out[19]=
```
{0., 0.880101, 0., -0.0391267, 0.}
```

In[21]:=
```
c2f[f2c[Sin]][x]
```
Out[21]= $2\,x\,\text{BesselJ}[1, 1] - 2\left(-3\,x + 4\,x^3\right)\text{BesselJ}[3, 1]$

Define generic coefficient

In[22]:=
```
cs = Table[c_i, {i, 0, n}]
```
Out[22]= $\{c_0, c_1, c_2, c_3, c_4\}$

In[24]:=
```
(cs // c2f)[x]
```
Out[24]= $\frac{c_0}{\sqrt{2}} + x\,c_1 + \left(-1 + 2\,x^2\right)c_2 + \left(-3\,x + 4\,x^3\right)c_3 + \left(1 - 8\,x^2 + 8\,x^4\right)c_4$

In[25]:=
```
cs // c2f // f2c
```
Out[25]= $\{c_0, c_1, c_2, c_3, c_4\}$

150%

In[18]:=
```
f2c[Sin]
```

Out[18]=
```
{0, 2 BesselJ[1, 1], 0, -2 BesselJ[3, 1], 0}
```

In[19]:=
```
f2c[Sin] // N
```

Out[19]=
```
{0., 0.880101, 0., -0.0391267, 0.}
```

In[21]:=
```
c2f[f2c[Sin]][x]
```

Out[21]=
$$2\,x\,\text{BesselJ}[1,1]-2\left(-3\,x+4\,x^3\right)\text{BesselJ}[3,1]$$

Define generic coefficient

In[22]:=
```
cs = Table[cᵢ, {i, 0, n}]
```

Out[22]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

In[24]:=
```
(cs // c2f)[x]
```

Out[24]=
$$\frac{c_0}{\sqrt{2}}+x\,c_1+\left(-1+2\,x^2\right)c_2+\left(-3\,x+4\,x^3\right)c_3+\left(1-8\,x^2+8\,x^4\right)c_4$$

In[25]:=
```
cs // c2f // f2c
```

Out[25]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

This shows that f2c is the inverse of c2f (for this choice of n)

Derivatives

150%

In[18]:=
```
f2c[Sin]
```
Out[18]=
```
{0, 2 BesselJ[1, 1], 0, -2 BesselJ[3, 1], 0}
```

In[19]:=
```
f2c[Sin] // N
```
Out[19]=
```
{0., 0.880101, 0., -0.0391267, 0.}
```

In[21]:=
```
c2f[f2c[Sin]][x]
```
Out[21]=
$$2\,x\,\text{BesselJ}[1,1]-2\left(-3\,x+4\,x^3\right)\text{BesselJ}[3,1]$$

Define generic coefficient

In[22]:=
```
cs = Table[c_i, {i, 0, n}]
```
Out[22]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

In[24]:=
```
(cs // c2f)[x]
```
Out[24]=
$$\frac{c_0}{\sqrt{2}}+x\,c_1+\left(-1+2\,x^2\right)c_2+\left(-3\,x+4\,x^3\right)c_3+\left(1-8\,x^2+8\,x^4\right)c_4$$

In[25]:=
```
cs // c2f // f2c
```
Out[25]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

This shows that f2c is the inverse of c2f (for this choice of n)

Derivatives

150%

In[22]:=
```
cs = Table[c_i, {i, 0, n}]
```

Out[22]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

In[24]:=
```
(cs // c2f)[x]
```

Out[24]=
$$\frac{c_0}{\sqrt{2}} + x\,c_1 + \left(-1 + 2\,x^2\right) c_2 + \left(-3\,x + 4\,x^3\right) c_3 + \left(1 - 8\,x^2 + 8\,x^4\right) c_4$$

In[25]:=
```
cs // c2f // f2c
```

Out[25]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

This shows that f2c is the inverse of c2f (for this choice of n)

Derivatives

We need to be able to represent e.g. derivative operators in our discrete (truncated) bases to solve PDEs

```
dd = Module[{dcs}, dcs = f2c[c2f[cs]′];
    Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];

dd // MatrixForm

dd.dd // MatrixForm

dsin = dd.(Sin // f2c) // c2f

Plot[{Sin'[x], dsin[x]}, {x, -1, +1}]
```

150% ▸

$\sqrt{2}$

In[25]:=
```
cs // c2f // f2c
```

Out[25]= $\{c_0, c_1, c_2, c_3, c_4\}$

This shows that f2c is the inverse of c2f (for this choice of n)

Derivatives

We need to be able to represent e.g. derivative operators in our discrete (truncated) bases to solve PDEs

Next steps:
1. Take a generic function represented in our basis
2. Calculate its derivative
3. Determine ("read off") the matrix representation of the derivative operator in our basis

```
dd = Module[{dcs}, dcs = f2c[c2f[cs]′];
   Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];

dd // MatrixForm

dd.dd // MatrixForm

dsin = dd.(Sin // f2c) // c2f

Plot[{Sin'[x], dsin[x]}, {x, -1, +1}]
```

150%

In[25]:=
```
cs // c2f // f2c
```

Out[25]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

This shows that f2c is the inverse of c2f (for this choice of n)

Derivatives

We need to be able to represent e.g. derivative operators in our discrete (truncated) bases to solve PDEs

Next steps:
1. Take a generic function represented in our basis
2. Calculate its derivative
3. Determine ("read off") the matrix representation of the derivative operator in our basis

Generic set of coefficients:

In[27]:=
```
cs
```

Out[27]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

In[26]:=
```
dd = Module[{dcs}, dcs = f2c[c2f[cs]′];
    Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];

dd // MatrixForm

dd.dd // MatrixForm

dsin = dd.(Sin // f2c) // c2f
```

150%

Derivatives

We need to be able to represent e.g. derivative operators in our discrete (truncated) bases to solve PDEs

Next steps:
1. Take a generic function represented in our basis
2. Calculate its derivative
3. Determine ("read off") the matrix representation of the derivative operator in our basis

Generic set of coefficients:

In[27]:=
```
cs
```

Out[27]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

Generic function:

In[28]:=
```
c2f[cs][x]
```

Out[28]=
$$\frac{c_0}{\sqrt{2}} + x\,c_1 + \left(-1 + 2\,x^2\right) c_2 + \left(-3\,x + 4\,x^3\right) c_3 + \left(1 - 8\,x^2 + 8\,x^4\right) c_4$$

Take

In[26]:=
```
dd = Module[{dcs}, dcs = f2c[c2f[cs]′];
   Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];

dd // MatrixForm
```

150%

Next steps:

1. Take a generic function represented in our basis
2. Calculate its derivative
3. Determine ("read off") the matrix representation of the derivative operator in our basis

Generic set of coefficients:

In[27]:=
```
cs
```

Out[27]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

Generic function:

In[28]:=
```
c2f[cs][x]
```

Out[28]=
$$\frac{c_0}{\sqrt{2}} + x\,c_1 + \left(-1 + 2\,x^2\right) c_2 + \left(-3\,x + 4\,x^3\right) c_3 + \left(1 - 8\,x^2 + 8\,x^4\right) c_4$$

Take derivative of this function:

In[29]:=
```
c2f[cs]'[x]
```

Out[29]=
$$c_1 + 4\,x\,c_2 + 3\left(-1 + 4\,x^2\right) c_3 + 4\left(-4\,x + 8\,x^3\right) c_4$$

In[26]:=
```
dd = Module[{dcs}, dcs = f2c[c2f[cs]'];
    Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];

dd // MatrixForm
```

150%

1. Take a generic function represented in our basis
2. Calculate its derivative
3. Determine ("read off") the matrix representation of the derivative operator in our basis

Generic set of coefficients:

In[27]:=
```
cs
```

Out[27]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

Generic function:

In[28]:=
```
c2f[cs][x]
```

Out[28]=
$$\frac{c_0}{\sqrt{2}} + x\,c_1 + \left(-1 + 2\,x^2\right) c_2 + \left(-3\,x + 4\,x^3\right) c_3 + \left(1 - 8\,x^2 + 8\,x^4\right) c_4$$

Take derivative of this function:

In[29]:=
```
c2f[cs]'[x]
```

Out[29]=
$$c_1 + 4\,x\,c_2 + 3\left(-1 + 4\,x^2\right) c_3 + 4\left(-4\,x + 8\,x^3\right) c_4$$

Project back onto our basis:

```
f2c[c2f[cs]']
```

In[26]:=
```
dd = Module[{dcs}, dcs = f2c[c2f[cs]′];
    Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];
```

150%

Generic set of coefficients:

In[27]:=
```
cs
```

Out[27]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

Generic function:

In[28]:=
```
c2f[cs][x]
```

Out[28]=
$$\frac{c_0}{\sqrt{2}} + x\,c_1 + \left(-1 + 2\,x^2\right) c_2 + \left(-3\,x + 4\,x^3\right) c_3 + \left(1 - 8\,x^2 + 8\,x^4\right) c_4$$

Take derivative of this function:

In[29]:=
```
c2f[cs]'[x]
```

Out[29]=
$$c_1 + 4\,x\,c_2 + 3\left(-1 + 4\,x^2\right) c_3 + 4\left(-4\,x + 8\,x^3\right) c_4$$

Project back onto our basis:

In[30]:=
```
f2c[c2f[cs]']
```

Out[30]=
$$\left\{\sqrt{2}\ (c_1 + 3\,c_3),\ 4\ (c_2 + 2\,c_4),\ 6\,c_3,\ 8\,c_4,\ 0\right\}$$

In[26]:=
```
dd = Module[{dcs}, dcs = f2c[c2f[cs]′];
   Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];

dd // MatrixForm

dd.dd // MatrixForm
```

150%

Out[28]= $\frac{c_0}{\sqrt{2}} + x\,c_1 + \left(-1 + 2x^2\right) c_2 + \left(-3x + 4x^3\right) c_3 + \left(1 - 8x^2 + 8x^4\right) c_4$

Take derivative of this function:

In[29]:=
```
c2f[cs]'[x]
```

Out[29]= $c_1 + 4x\,c_2 + 3\left(-1 + 4x^2\right) c_3 + 4\left(-4x + 8x^3\right) c_4$

Project back onto our basis:

In[30]:=
```
f2c[c2f[cs]']
```

Out[30]= $\left\{\sqrt{2}\ (c_1 + 3c_3),\ 4\ (c_2 + 2c_4),\ 6c_3,\ 8c_4,\ 0\right\}$

In[26]:=
```
dd = Module[{dcs}, dcs = f2c[c2f[cs]'];
    Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];
```

In[31]:=
```
dd // MatrixForm
```

Out[31]//MatrixForm=

$$\begin{pmatrix} 0 & \sqrt{2} & 0 & 3\sqrt{2} & 0 \\ 0 & 0 & 4 & 0 & 8 \\ 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
dd.dd // MatrixForm
```

```
dsin = dd.(Sin // f2c) // c2f
```

150%

Out[28]= $\frac{c_0}{\sqrt{2}} + x\,c_1 + \left(-1 + 2x^2\right) c_2 + \left(-3x + 4x^3\right) c_3 + \left(1 - 8x^2 + 8x^4\right) c_4$

Take derivative of this function:

In[29]:=
```
c2f[cs]'[x]
```

Out[29]= $c_1 + 4x\,c_2 + 3\left(-1 + 4x^2\right) c_3 + 4\left(-4x + 8x^3\right) c_4$

Project back onto our basis:

In[30]:=
```
f2c[c2f[cs]']
```

Out[30]= $\left\{\sqrt{2}\ (c_1 + 3c_3),\ 4\ (c_2 + 2c_4),\ 6\,c_3,\ 8\,c_4,\ 0\right\}$

Matrix representation of derivative operator for our trubasis (modified Chebyshev polynomials)

In[26]:=
```
dd = Module[{dcs}, dcs = f2c[c2f[cs]′];
    Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];
```

In[31]:=
```
dd // MatrixForm
```

Out[31]//MatrixForm=

$$\begin{pmatrix} 0 & \sqrt{2} & 0 & 3\sqrt{2} & 0 \\ 0 & 0 & 4 & 0 & 8 \\ 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
dd.dd // MatrixForm
```

150% ▸

Out[28]= $\frac{c_0}{\sqrt{2}} + x\,c_1 + \left(-1 + 2x^2\right) c_2 + \left(-3x + 4x^3\right) c_3 + \left(1 - 8x^2 + 8x^4\right) c_4$

Take derivative of this function:

In[29]:=
```
c2f[cs]'[x]
```

Out[29]= $c_1 + 4\,x\,c_2 + 3\left(-1 + 4x^2\right) c_3 + 4\left(-4x + 8x^3\right) c_4$

Project back onto our basis:

In[30]:=
```
f2c[c2f[cs]']
```

Out[30]= $\left\{\sqrt{2}\ (c_1 + 3\,c_3),\ 4\ (c_2 + 2\,c_4),\ 6\,c_3,\ 8\,c_4,\ 0\right\}$

Matrix representation of derivative operator for our truncated basis (modified Chebyshev polynomials)

In[26]:=
```
dd = Module[{dcs}, dcs = f2c[c2f[cs]′];
    Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];
```

In[31]:=
```
dd // MatrixForm
```

Out[31]//MatrixForm=

$$\begin{pmatrix} 0 & \sqrt{2} & 0 & 3\sqrt{2} & 0 \\ 0 & 0 & 4 & 0 & 8 \\ 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
dd.dd // MatrixForm
```

150% ▸

Project back onto our basis:

In[30]:=
```
f2c[c2f[cs]']
```

Out[30]=
$$\left\{\sqrt{2}\ (c_1 + 3\,c_3),\ 4\ (c_2 + 2\,c_4),\ 6\,c_3,\ 8\,c_4,\ 0\right\}$$

Matrix representation of derivative operator for our truncated basis (modified Chebyshev polynomials)

In[26]:=
```
dd = Module[{dcs}, dcs = f2c[c2f[cs]'];
    Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];
```

In[31]:=
```
dd // MatrixForm
```

Out[31]//MatrixForm=
$$\begin{pmatrix} 0 & \sqrt{2} & 0 & 3\sqrt{2} & 0 \\ 0 & 0 & 4 & 0 & 8 \\ 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Second derivative

```
dd.dd // MatrixForm
```

```
dsin = dd.(Sin // f2c) // c2f
```

```
Plot[{Sin'[x], dsin[x]}, {x, -1, +1}]
```

Boundaries

150%

Matrix representation of derivative operator for our truncated basis (modified Chebyshev polynomials)

In[26]:=
```
dd = Module[{dcs}, dcs = f2c[c2f[cs]′];
    Table[Coefficient[dcs[[i + 1]], cs[[j + 1]]], {i, 0, n}, {j, 0, n}]];
```

In[31]:=
```
dd // MatrixForm
```

Out[31]//MatrixForm=

$$\begin{pmatrix} 0 & \sqrt{2} & 0 & 3\sqrt{2} & 0 \\ 0 & 0 & 4 & 0 & 8 \\ 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Second derivative

In[32]:=
```
dd.dd // MatrixForm
```

Out[32]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 4\sqrt{2} & 0 & 32\sqrt{2} \\ 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 48 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
dsin = dd.(Sin // f2c) // c2f

Plot[{Sin'[x], dsin[x]}, {x, -1, +1}]
```

150%

Out[31]//MatrixForm=

$$\begin{pmatrix} 0 & \sqrt{2} & 0 & 3\sqrt{2} & 0 \\ 0 & 0 & 4 & 0 & 8 \\ 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Second derivative

In[32]:=
```
dd.dd // MatrixForm
```

Out[32]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 4\sqrt{2} & 0 & 32\sqrt{2} \\ 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 48 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Example: Derivative of sine function

```
dsin = dd.(Sin // f2c) // c2f

Plot[{Sin'[x], dsin[x]}, {x, -1, +1}]
```

Boundaries

```
bb = Module[{bcs}, bcs = c2f[cs][{-1, +1}];
    Table[Coefficient[bcs[[i]], cs[[j + 1]]], {i, 1, 2}, {j, 0, n}]];

bb // MatrixForm
```

150%

$$\begin{pmatrix} 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Second derivative

In[32]:=
```
dd.dd // MatrixForm
```

Out[32]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 4\sqrt{2} & 0 & 32\sqrt{2} \\ 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 48 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Example: Derivative of sine function

In[33]:=
```
dsin = dd.(Sin // f2c) // c2f
```

Out[33]=
```
Function[x$,
 Sum[{2 Sqrt[2] BesselJ[1, 1] - 6 Sqrt[2] BesselJ[3, 1], 0, -12 BesselJ[3, 1], 0, 0}[[i + 1]] cheb[i, x$], {i, 0, n}]]
```

```
Plot[{Sin'[x], dsin[x]}, {x, -1, +1}]
```

Boundaries

```
bb = Module[{bcs}, bcs = c2f[cs][{-1, +1}];
```

150%

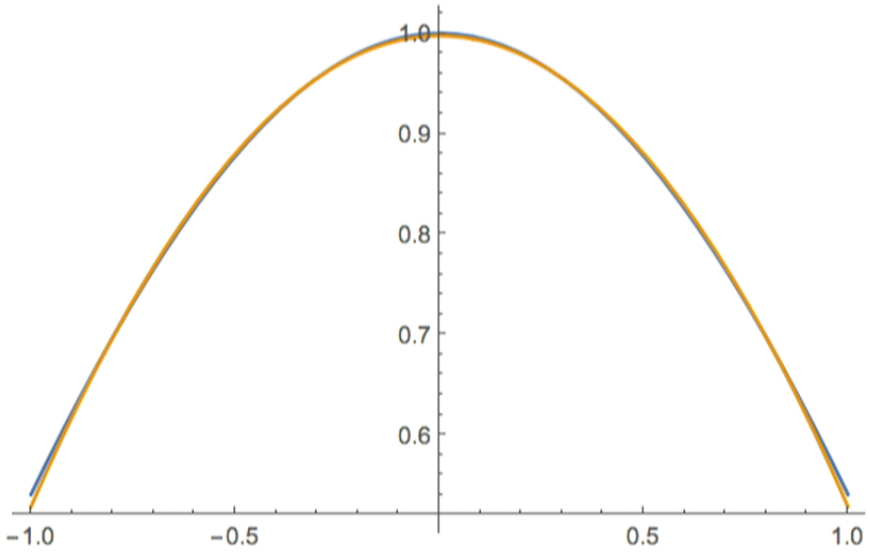
Example: Derivative of sine function

In[33]:=
```
dsin = dd.(Sin // f2c) // c2f
```

Out[33]=
```
Function[x$,
 Sum_{i=0}^{n} {2 Sqrt[2] BesselJ[1, 1] - 6 Sqrt[2] BesselJ[3, 1], 0, -12 BesselJ[3, 1], 0, 0}[[i + 1]] cheb[i, x$]]
```

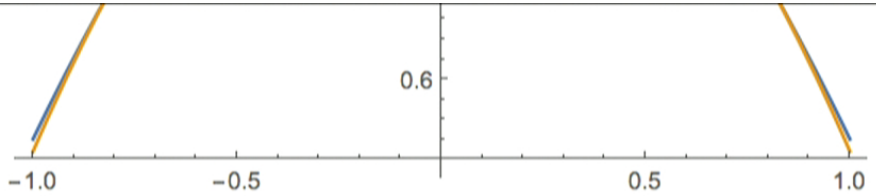
In[34]:=
```
Plot[{Sin'[x], dsin[x]}, {x, -1, +1}]
```

Out[34]=



Boundaries

150%



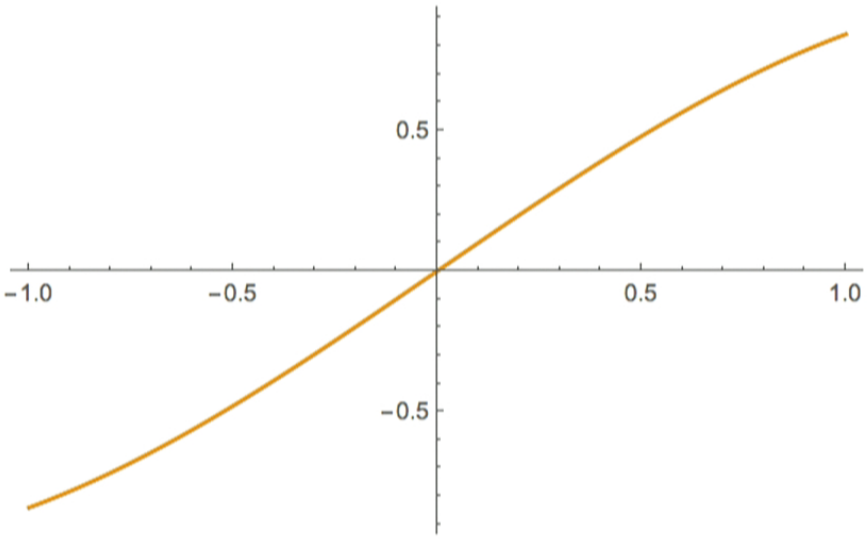
Example: sine function itself

In[35]:=
```
sin = (Sin // f2c) // c2f
```

Out[35]=
$$\text{Function}\left[\text{x\$}, \sum_{i=0}^{n} \{0, 2\,\text{BesselJ}[1, 1], 0, -2\,\text{BesselJ}[3, 1], 0\}[\![i + 1]\!]\,\text{cheb}[i, \text{x\$}]\right]$$

In[36]:=
```
Plot[{Sin[x], sin[x]}, {x, -1, +1}]
```

Out[36]=

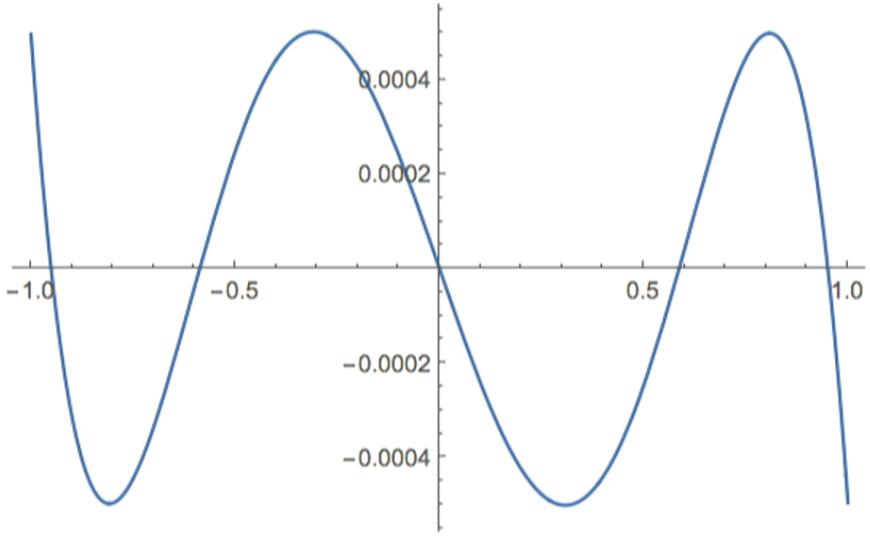


150%

Show error

In[37]:=
```
Plot[Sin[x] - Sin[x], {x, -1, +1}]
```

Out[37]=

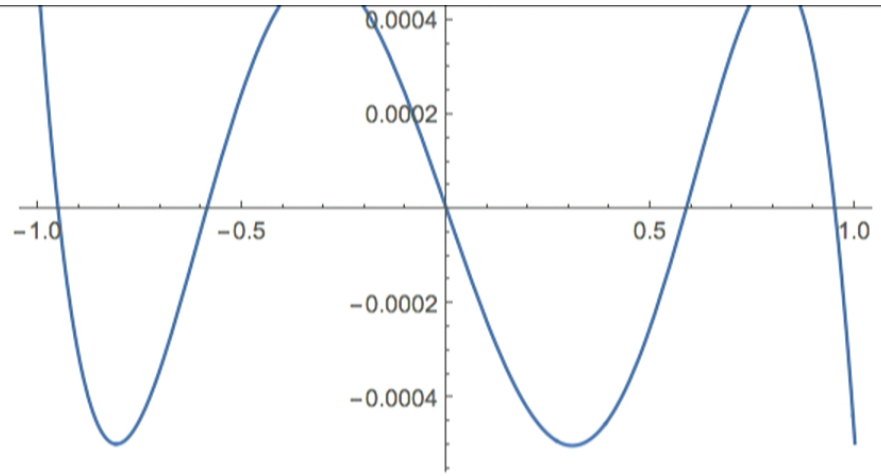


Boundaries

```
bb = Module[{bcs}, bcs = c2f[cs][{-1, +1}];
    Table[Coefficient[bcs[[i]], cs[[i + 1]]], {i, 1, 2}, {i, 0, n}]];
```

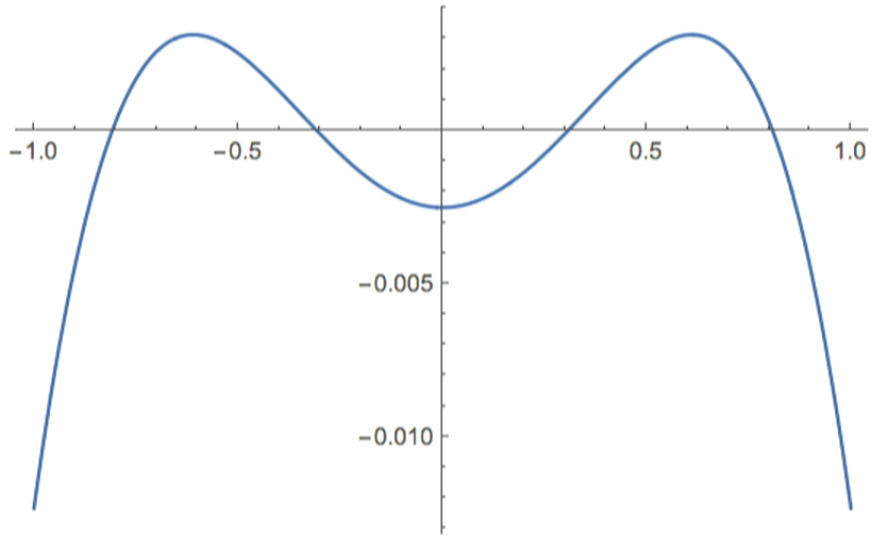
150%

Out[37]=



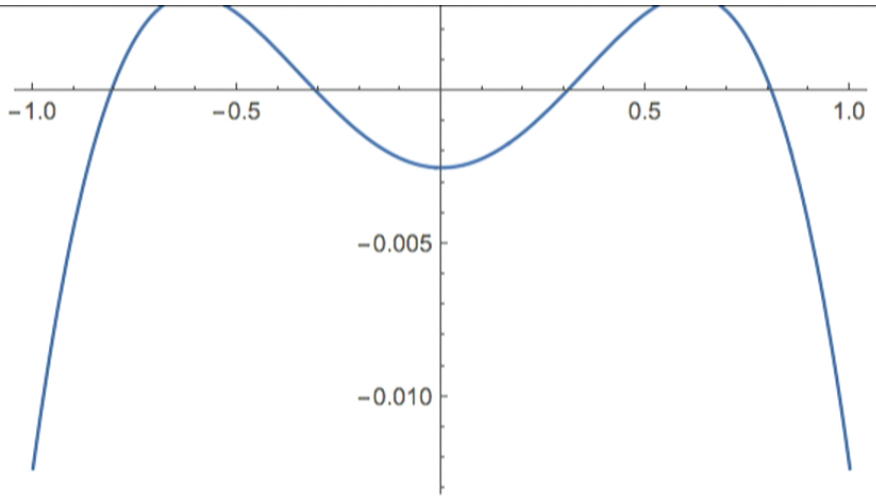
In[38]:=
```
Plot[dsin[x] - Sin'[x], {x, -1, +1}]
```

Out[38]=



150%

Out[38]=



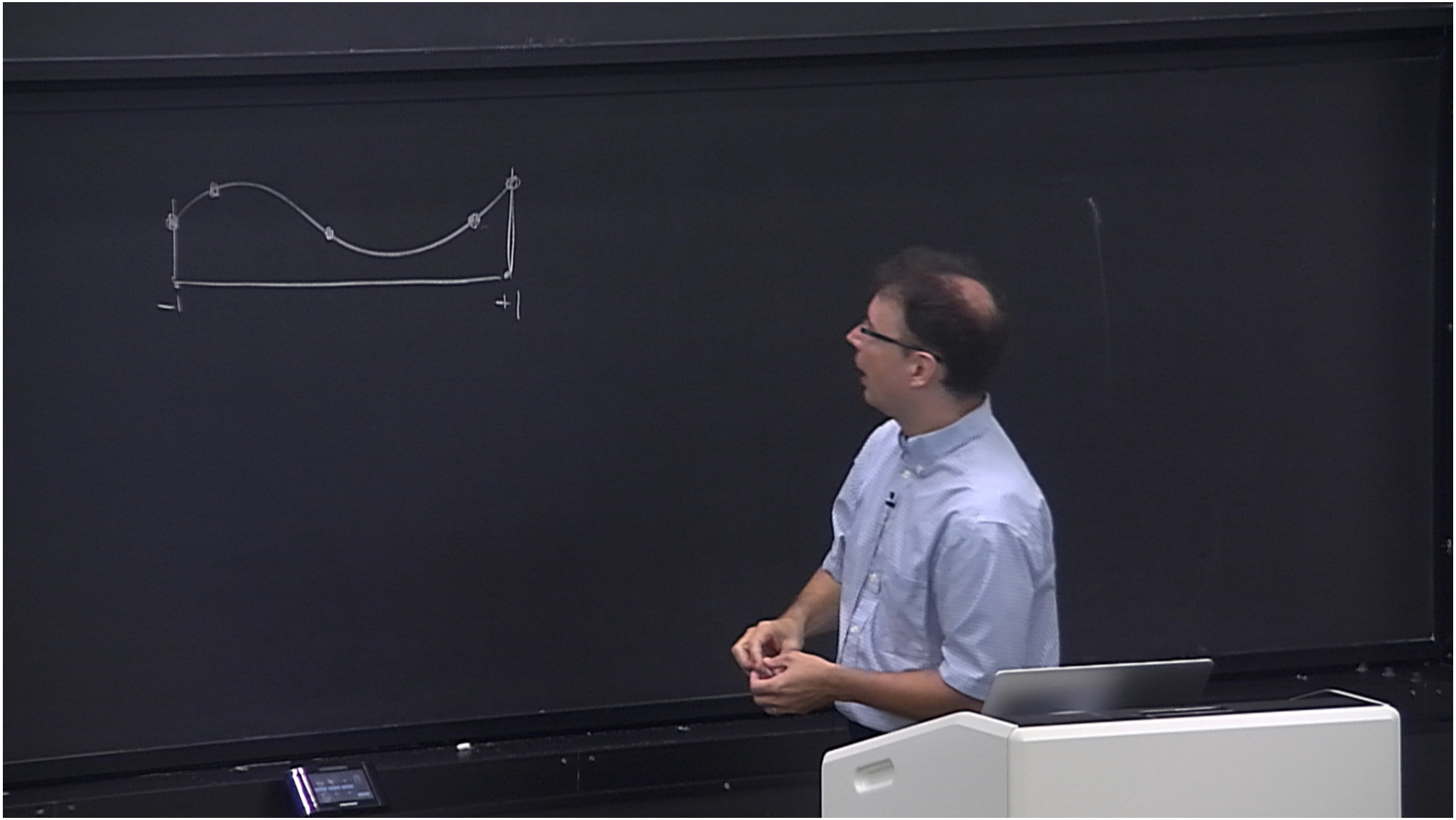
Boundaries

Define “boundary operator” that evaluates a function at the boundary

```
bb = Module[{bcs}, bcs = c2f[cs][{-1, +1}];
    Table[Coefficient[bcs[[i]], cs[[j + 1]]], {i, 1, 2}, {j, 0, n}]];

bb // MatrixForm

bb1 = Join[Array[0 &, {n - 2 + 1, n + 1}], bb];

bb1 // MatrixForm

dd + bb1 // MatrixForm
```

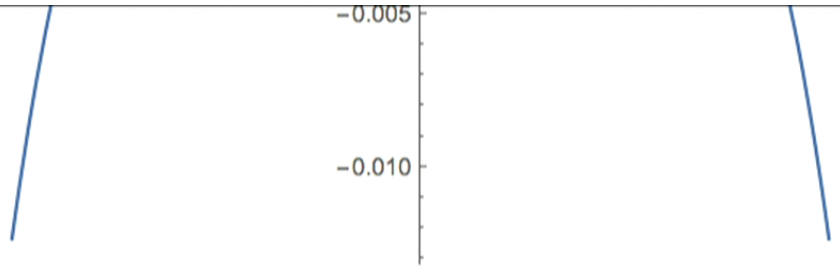
150%


−1
+1


−1
+1


−1
+1


−1
+1



Boundaries

Define “boundary operator” that evaluates a function at the boundary

Generic set of coefficients

In[39]:=
```
cs
```

Out[39]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

Reconstruct generic functions

In[40]:=
```
c2f[cs]
```

Out[40]=
$$\text{Function}\left[x\$, \sum_{i=0}^{n} \{c_0, c_1, c_2, c_3, c_4\}[\![i+1]\!]\ \text{cheb}[i, x\$]\right]$$

```
bb = Module[{bcs}, bcs = c2f[cs][{-1, +1}];
    Table[Coefficient[bcs[[i]], cs[[j + 1]]], {i, 1, 2}, {j, 0, n}]];

bb // MatrixForm
```

150%

Boundaries

Define “boundary operator” that evaluates a function at the boundary

Generic set of coefficients

In[39]:=
```
cs
```

Out[39]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

Reconstruct generic functions

In[40]:=
```
c2f[cs]
```

Out[40]=
$$\text{Function}\left[x\$, \sum_{i=0}^{n} \{c_0, c_1, c_2, c_3, c_4\}[[i+1]]\ \text{cheb}[i, x\$]\right]$$

Evaluate function at boundary

In[42]:=
```
c2f[cs][+1]
```

Out[42]=
$$\frac{c_0}{\sqrt{2}} + c_1 + c_2 + c_3 + c_4$$

```
bb = Module[{bcs}, bcs = c2f[cs][{-1, +1}];
   Table[Coefficient[bcs[[i]], cs[[j + 1]]], {i, 1, 2}, {j, 0, n}]];
```

150%

In[39]:=
```
cs
```

Out[39]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

Reconstruct generic functions

In[40]:=
```
c2f[cs]
```

Out[40]=
$$\text{Function}\left[x\$, \sum_{i=0}^{n} \{c_0, c_1, c_2, c_3, c_4\}[\![i+1]\!]\ \text{cheb}[i, x\$]\right]$$

Evaluate function at boundary

In[44]:=
```
c2f[cs][{-1, +1}] // MatrixForm
```

Out[44]//MatrixForm=
$$\begin{pmatrix} \frac{c_0}{\sqrt{2}} - c_1 + c_2 - c_3 + c_4 \\ \frac{c_0}{\sqrt{2}} + c_1 + c_2 + c_3 + c_4 \end{pmatrix}$$

```
bb = Module[{bcs}, bcs = c2f[cs][{-1, +1}];
    Table[Coefficient[bcs[[i]], cs[[j + 1]]], {i, 1, 2}, {j, 0, n}]];

bb // MatrixForm

bb1 = Join[Array[0 &, {n - 2 + 1, n + 1}], bb];

bb1 // MatrixForm

dd + bb1 // MatrixForm
```

150%

In[39]:=
```
cs
```

Out[39]=
$$\{c_0, c_1, c_2, c_3, c_4\}$$

Reconstruct generic functions

In[40]:=
```
c2f[cs]
```

Out[40]=
$$\text{Function}\left[x\$, \sum_{i=0}^{n} \{c_0, c_1, c_2, c_3, c_4\}[\![i+1]\!]\ \text{cheb}[i, x\$]\right]$$

Evaluate function at boundary

In[44]:=
```
c2f[cs][{-1, +1}] // MatrixForm
```

Out[44]//MatrixForm=
$$\begin{pmatrix} \frac{c_0}{\sqrt{2}} - c_1 + c_2 - c_3 + c_4 \\ \frac{c_0}{\sqrt{2}} + c_1 + c_2 + c_3 + c_4 \end{pmatrix}$$

In[45]:=
```
bb = Module[{bcs}, bcs = c2f[cs][{-1, +1}];
    Table[Coefficient[bcs[[i]], cs[[j + 1]]], {i, 1, 2}, {j, 0, n}]];

bb // MatrixForm

bb1 = Join[Array[0 &, {n - 2 + 1, n + 1}], bb];

bb1 // MatrixForm

dd + bb1 // MatrixForm
```

150%

In[40]:=
```
c2f[cs]
```

Out[40]=
$$\text{Function}\left[x\$, \sum_{i=0}^{n} \{c_0, c_1, c_2, c_3, c_4\}[[i+1]]\ \text{cheb}[i, x\$]\right]$$

Evaluate function at boundary

In[44]:=
```
c2f[cs][{-1, +1}] // MatrixForm
```

Out[44]//MatrixForm=
$$\begin{pmatrix} \frac{c_0}{\sqrt{2}} - c_1 + c_2 - c_3 + c_4 \\ \frac{c_0}{\sqrt{2}} + c_1 + c_2 + c_3 + c_4 \end{pmatrix}$$

Matrix representation of boundary operator (i.e. projection onto boundary)

In[45]:=
```
bb = Module[{bcs}, bcs = c2f[cs][{-1, +1}];
    Table[Coefficient[bcs[[i]], cs[[j + 1]]], {i, 1, 2}, {j, 0, n}]];
```

In[46]:=
```
bb // MatrixForm
```

Out[46]//MatrixForm=
$$\begin{pmatrix} \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

```
bb1 = Join[Array[0 &, {n - 2 + 1, n + 1}], bb];

bb1 // MatrixForm

dd + bb1 // MatrixForm
```

150%

$$\left(\frac{}{\sqrt{2}} + c_1 + c_2 + c_3 + c_4\right)$$

Matrix representation of boundary operator (i.e. projection onto boundary)

In[45]:=
```
bb = Module[{bcs}, bcs = c2f[cs][{-1, +1}];
    Table[Coefficient[bcs[[i]], cs[[j + 1]]], {i, 1, 2}, {j, 0, n}]];
```

In[46]:=
```
bb // MatrixForm
```

Out[46]//MatrixForm=

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

Prolongate boundary operator to (n+1) dimensions

In[47]:=
```
bb1 = Join[Array[0 &, {n - 2 + 1, n + 1}], bb];
```

In[48]:=
```
bb1 // MatrixForm
```

Out[48]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

```
dd + bb1 // MatrixForm
```

150%

```
Table[Coefficient[bcs[[i]], cs[[j + 1]]], {i, 1, 2}, {j, 0, n}]];
```

In[46]:=
```
bb // MatrixForm
```

Out[46]//MatrixForm=

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

Prolongate boundary operator to (n+1) dimensions

In[47]:=
```
bb1 = Join[Array[0 &, {n - 2 + 1, n + 1}], bb];
```

In[48]:=
```
bb1 // MatrixForm
```

Out[48]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

```
dd + bb1 // MatrixForm
```

Equations and Boundary Conditions

```
rhs[x_] = Sin[π x]
```

150% ▸

Out[46]//MatrixForm=

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

Prolongate boundary operator to (n+1) dimensions

In[47]:=
```
bb1 = Join[Array[0 &, {n - 2 + 1, n + 1}], bb];
```

In[48]:=
```
bb1 // MatrixForm
```

Out[48]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

In[50]:=
```
dd.dd + bb1 // MatrixForm
```

Out[50]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 4\sqrt{2} & 0 & 32\sqrt{2} \\ 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 48 \\ \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

150%

$\left(\sqrt{2}\right)$

Prolongate boundary operator to (n+1) dimensions

In[47]:=
```
bb1 = Join[Array[0 &, {n - 2 + 1, n + 1}], bb];
```

In[48]:=
```
bb1 // MatrixForm
```

Out[48]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

Combined second derivative and boundary operator

In[50]:=
```
dd.dd + bb1 // MatrixForm
```

Out[50]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 4\sqrt{2} & 0 & 32\sqrt{2} \\ 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 48 \\ \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

Equations and Boundary Conditions

150% ▸

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

As example, the combined second derivative and boundary operator

In[50]:=
```
dd.dd + bb1 // MatrixForm
```

Out[50]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 4\sqrt{2} & 0 & 32\sqrt{2} \\ 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 48 \\ \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

Equations and Boundary Conditions

We want to solve this PDE: f''[x] == sin(pi x)

```
rhs[x_] = Sin[π x]
```

```
(*rhs[x_]=If[x≥0&&x≤1/2,1,0]*)
```

```
Plot[rhs[x], {x, -1, +1}]
```

```
deq = ∂x,x f[x] == rhs[x]
```

150%

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

As example, the combined second derivative and boundary operator

In[50]:=
```
dd.dd + bb1 // MatrixForm
```

Out[50]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 4\sqrt{2} & 0 & 32\sqrt{2} \\ 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 48 \\ \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

Equations and Boundary Conditions

We want to solve this PDE:

f''[x] == sin(pi x)

f[0] == f0

f[1] == f1

```
rhs[x_] = Sin[π x]
```

```
(*rhs[x_]=If[x≥0&&x≤1/2,1,0]*)
```

150%

Out[50]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 4\sqrt{2} & 0 & 32\sqrt{2} \\ 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 48 \\ \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 1 \\ \frac{1}{\sqrt{2}} & 1 & 1 & 1 & 1 \end{pmatrix}$$

Equations and Boundary Conditions

We want to solve this PDE:

f''[x] == sin(pi x)

f[0] == f0

f[1] == f1

Define RHS:

```
rhs[x_] = Sin[π x]
```

```
(*rhs[x_]=If[x≥0&&x≤1/2,1,0]*)
```

```
Plot[rhs[x], {x, -1, +1}]
```

```
deq = ∂x,x f[x] == rhs[x]
```

```
bcs = {f[-1] == 1/2, f[+1] == 0}
```

150%

We want to solve this PDE:

f''[x] == sin(pi x)

f[0] == f0

f[1] == f1

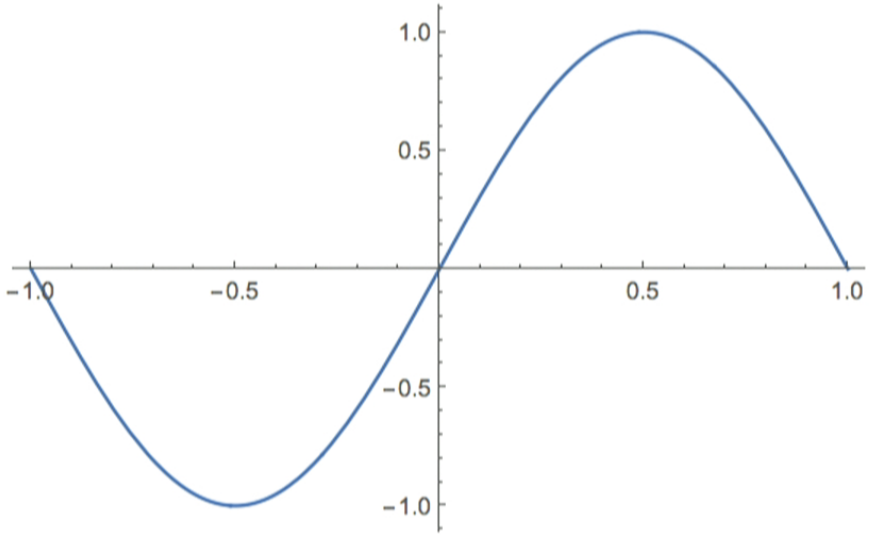
Define RHS:

In[51]:=
```
rhs[x_] = Sin[π x]
```

Out[51]=
```
Sin[π x]
```

In[52]:=
```
Plot[rhs[x], {x, -1, +1}]
```

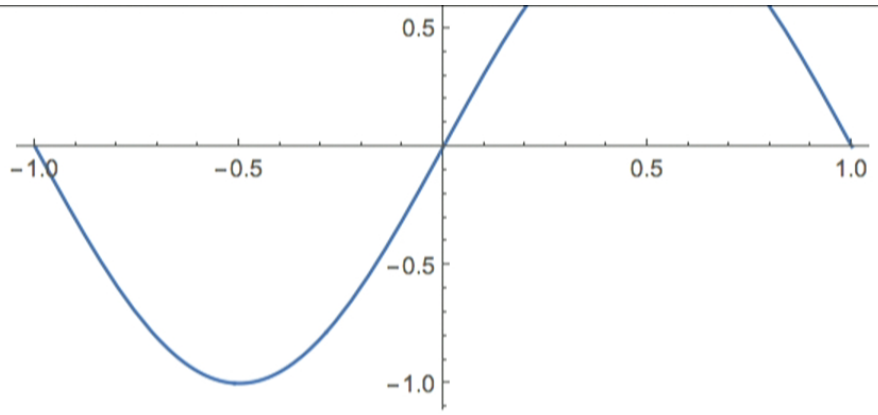
Out[52]=




```
deq = ∂x,x f[x] == rhs[x]
```

150%

Out[52]=



Define differential equation:

In[53]:= `deq = ∂x,x f[x] == rhs[x]`

Out[53]= `f''[x] == Sin[π x]`

Define boundary conditions:

In[54]:= `bcs = {f[-1] == 1/10, f[+1] == 0}`

Out[54]= `{f[-1] == 1/10, f[1] == 0}`

Analytic Solution

Define differential equation:

In[53]:=
```
deq = ∂x,x f[x] == rhs[x]
```

Out[53]=
$$f''[x] == \text{Sin}[\pi\, x]$$

Define boundary conditions:

In[54]:=
```
bcs = {f[-1] == 1/10, f[+1] == 0}
```

Out[54]=
$$\left\{f[-1] == \frac{1}{10},\ f[1] == 0\right\}$$

Analytic Solution

Ask Mathematica to solve this PDE:

In[55]:=
```
asol = DSolve[Join[{deq}, bcs], f, x]
```

Out[55]=
$$\left\{\left\{f \to \text{Function}\left[\{x\},\ \frac{\pi^2 - \pi^2\, x - 20\, \text{Sin}[\pi\, x]}{20\, \pi^2}\right]\right\}\right\}$$

```
Simplify[Join[{deq}, bcs] /. asol[[1]]]
```

```
Plot[f[x] /. asol[[1]], {x, -1, 1}]
```

Numeric solution from NDSolve

150%

In[53]:= deq = $\partial_{x,x}$f[x] == rhs[x]

Out[53]= f''[x] == Sin[π x]

Define boundary conditions:

In[54]:= bcs = $\left\{f[-1] == \frac{1}{10}, f[+1] == 0\right\}$

Out[54]= $\left\{f[-1] == \frac{1}{10}, f[1] == 0\right\}$

Analytic Solution

Ask Mathematica to solve this PDE:

In[55]:= asol = DSolve[Join[{deq}, bcs], f, x]

Out[55]= $\left\{\left\{f \to \text{Function}\left[\{x\}, \frac{\pi^2 - \pi^2 x - 20 \text{Sin}[\pi x]}{20 \pi^2}\right]\right\}\right\}$

In[56]:= Simplify[Join[{deq}, bcs] /. asol[[1]]]

Out[56]= {True, True, True}

Plot[f[x] /. asol[[1]], {x, -1, 1}]

Numeric solution from NDSolve

150%

In[54]:=
```
bcs = {f[-1] == 1/10, f[+1] == 0}
```

Out[54]=
$$\left\{f[-1] == \frac{1}{10},\ f[1] == 0\right\}$$

Analytic Solution

Ask Mathematica to solve this PDE:

In[55]:=
```
asol = DSolve[Join[{deq}, bcs], f, x]
```

Out[55]=
$$\left\{\left\{f \to \text{Function}\left[\{x\},\ \frac{\pi^2 - \pi^2\, x - 20\, \text{Sin}[\pi\, x]}{20\, \pi^2}\right]\right\}\right\}$$

Check whether our solution solves the equations:

In[56]:=
```
Simplify[Join[{deq}, bcs] /. asol[[1]]]
```

Out[56]=
```
{True, True, True}
```

```
Plot[f[x] /. asol[[1]], {x, -1, 1}]
```

Numeric solution from NDSolve

```
ndsol = NDSolve[Join[{deq}, bcs], f, {x, -1, +1}]
```

150%

Out[54]= $\left\{f[-1] == \frac{1}{10}, f[1] == 0\right\}$

Analytic Solution

Ask Mathematica to solve this PDE:

In[55]:=
```
asol = DSolve[Join[{deq}, bcs], f, x]
```

Out[55]= $\left\{\left\{f \to \text{Function}\left[\{x\}, \frac{\pi^2 - \pi^2 x - 20 \text{Sin}[\pi x]}{20 \pi^2}\right]\right\}\right\}$

Check whether our solution solves the equations:

In[56]:=
```
Simplify[Join[{deq}, bcs] /. asol[[1]]]
```

Out[56]= {True, True, True}

Show analytic solution:

```
Plot[f[x] /. asol[[1]], {x, -1, 1}]
```

Numeric solution from NDSolve

```
ndsol = NDSolve[Join[{deq}, bcs], f, {x, -1, +1}]
```

```
Plot[{f[x] /. asol[[1]], f[x] /. ndsol[[1]]}, {x, -1, 1}]
```

150%

In[55]:=
```
asol = DSolve[Join[{deq}, bcs], f, x]
```

Out[55]=
$$\left\{\left\{f \to \text{Function}\left[\{x\}, \frac{\pi^2 - \pi^2\, x - 20\, \text{Sin}[\pi\, x]}{20\, \pi^2}\right]\right\}\right\}$$

Check whether our solution solves the equations:

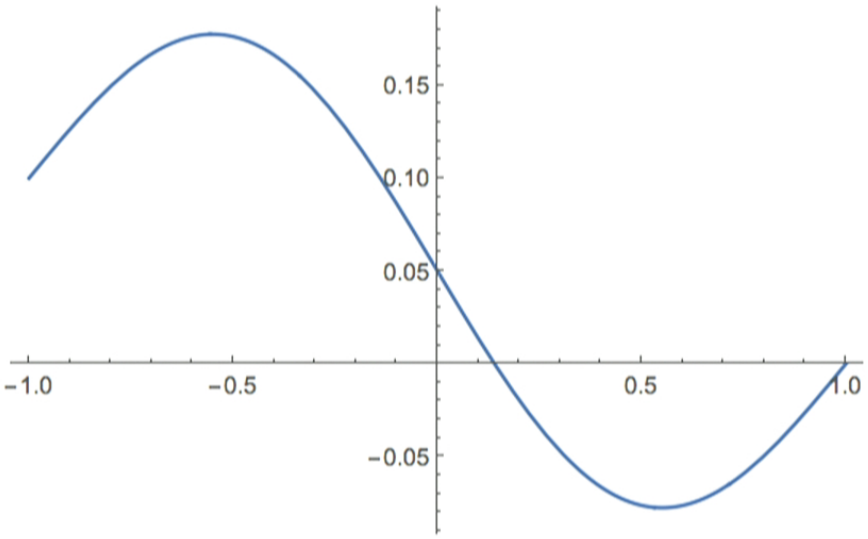
In[56]:=
```
Simplify[Join[{deq}, bcs] /. asol[[1]]]
```

Out[56]=
```
{True, True, True}
```

Show analytic solution:

In[58]:=
```
Plot[f[x] /. asol[[1]], {x, -1, 1}]
```

Out[58]=

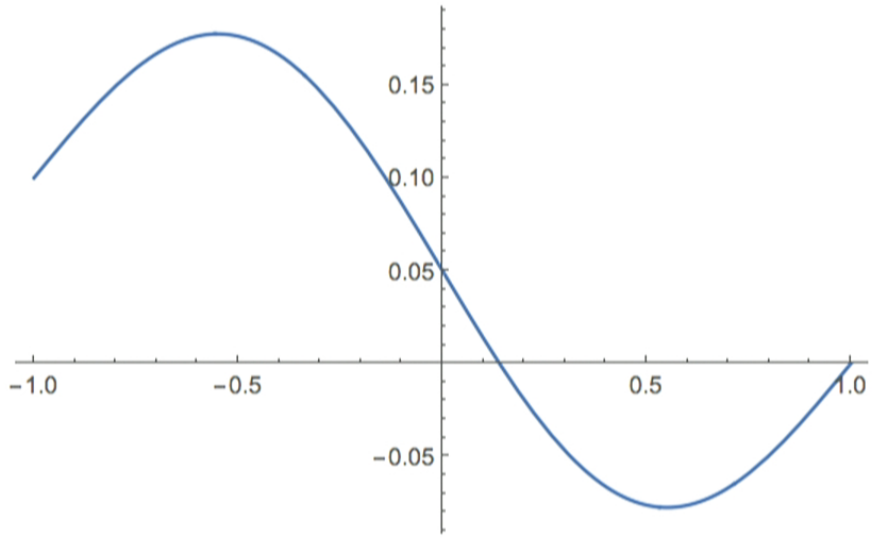


150%

Show analytic solution:

In[58]:=
```
Plot[f[x] /. asol[[1]], {x, -1, 1}]
```

Out[58]=



Numeric solution from NDSolve

Repeat this with Mathematica's numerical method

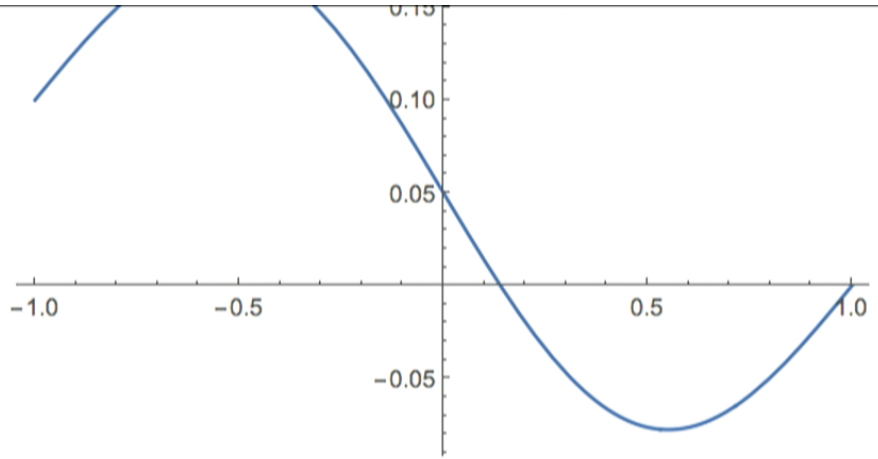
```
ndsol = NDSolve[Join[{deq}, bcs], f, {x, -1, +1}]
```

```
Plot[{f[x] /. asol[[1]], f[x] /. ndsol[[1]]}, {x, -1, 1}]
```

Numeric Solution

150% ▸

Out[58]=



Numeric solution from NDSolve

Repeat this with Mathematica's numerical method

In[59]:=
```
ndsol = NDSolve[Join[{deq}, bcs], f, {x, -1, +1}]
```

Out[59]= {{f → InterpolatingFunction[Domain: {{−1., 1.}} Output: scalar]}}

```
Plot[{f[x] /. asol[[1]], f[x] /. ndsol[[1]]}, {x, -1, 1}]
```

Numeric Solution

150%

Numeric solution from NDSolve

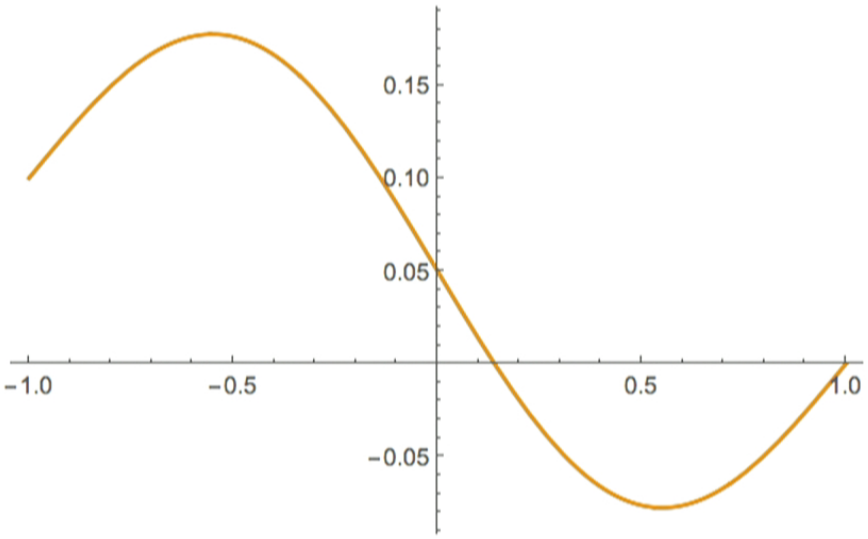
Repeat this with Mathematica's numerical method

In[59]:=
```
ndsol = NDSolve[Join[{deq}, bcs], f, {x, -1, +1}]
```

Out[59]=
```
{{f → InterpolatingFunction[  Domain: {{-1., 1.}}
                                                                                Output: scalar ]}}
```

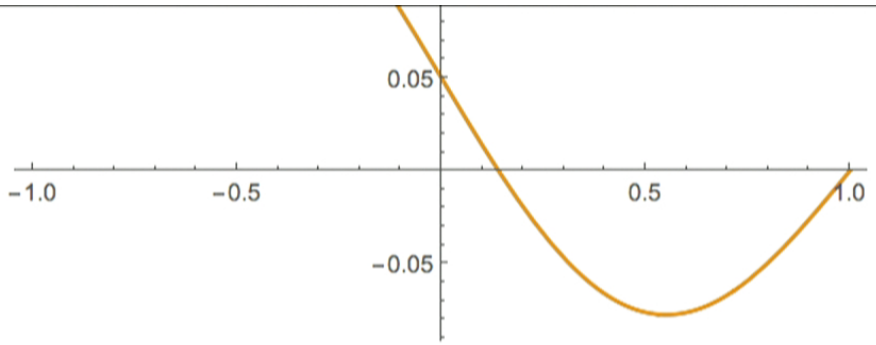
In[60]:=
```
Plot[{f[x] /. asol[[1]], f[x] /. ndsol[[1]]}, {x, -1, 1}]
```

Out[60]=



150%

Out[60]=



Numeric Solution

Now we repeat this with our own representation of approximated functions

1. Take differential equation and boundary condition, and find their matrix representation (for our basis)
2. Solve this a linear (matrix) equation

RHS

```
drhs = rhs // f2c // N

Plot[{rhs[x], (drhs // c2f)[x]}, {x, -1, +1}]

dbcs = {f[-1], f[+1]} /. (bcs /. (lhs_ == rhs_) → (lhs → rhs)) // N

drhs1 = Join[Take[drhs, n - 2 + 1], {0, 0}]
```

150%

Numeric Solution

Now we repeat this with our own representation of approximated functions

1. Take differential equation and boundary condition, and find their matrix representation (for our basis)
2. Solve this a linear (matrix) equation

RHS

Project RHS onto our basis functions

```
drhs = rhs // f2c
```

f2c

```
Plot[{rhs[x], (drhs // c2f)[x]}, {x, -1, +1}]

dbcs = {f[-1], f[+1]} /. (bcs /. (lhs_ == rhs_) → (lhs → rhs)) // N

drhs1 = Join[Take[drhs, n - 2 + 1], {0, 0}]

dbcs1 = Join[Table[0, {i, n - 2 + 1}], dbcs]

nsol = LinearSolve[dd.dd + bb1, drhs1 + dbcs1]

Plot[{f[x] /. asol[[1]], c2f[nsol][x]}, {x, -1, +1}]
```

150%

Numeric Solution

Now we repeat this with our own representation of approximated functions

1. Take differential equation and boundary condition, and find their matrix representation (for our basis)
2. Solve this a linear (matrix) equation

RHS

Project RHS onto our basis functions

```
In[61]:= drhs = rhs // f2c
Out[61]= {0, 2 BesselJ[1, π], 0, -2 BesselJ[3, π], 0}
```

```
Plot[{rhs[x], (drhs // c2f)[x]}, {x, -1, +1}]

dbcs = {f[-1], f[+1]} /. (bcs /. (lhs_ == rhs_) → (lhs → rhs)) // N

drhs1 = Join[Take[drhs, n - 2 + 1], {0, 0}]

dbcs1 = Join[Table[0, {i, n - 2 + 1}], dbcs]

nsol = LinearSolve[dd.dd + bb1, drhs1 + dbcs1]

Plot[{f[x] /. asol[[1]], c2f[nsol][x]}, {x, -1, +1}]
```

150%

Numeric Solution

Now we repeat this with our own representation of approximated functions

1. Take differential equation and boundary condition, and find their matrix representation (for our basis)
2. Solve this a linear (matrix) equation

RHS

Project RHS onto our basis functions

In[62]:=
```
drhs = rhs // f2c // N
```

Out[62]=
```
{0., 0.569231, 0., -0.666917, 0.}
```

```
Plot[{rhs[x], (drhs // c2f)[x]}, {x, -1, +1}]

dbcs = {f[-1], f[+1]} /. (bcs /. (lhs_ == rhs_) → (lhs → rhs)) // N

drhs1 = Join[Take[drhs, n - 2 + 1], {0, 0}]

dbcs1 = Join[Table[0, {i, n - 2 + 1}], dbcs]

nsol = LinearSolve[dd.dd + bb1, drhs1 + dbcs1]

Plot[{f[x] /. asol[[1]], c2f[nsol][x]}, {x, -1, +1}]
```

150%

Now we repeat this with our own representation of approximated functions

1. Take differential equation and boundary condition, and find their matrix representation (for our basis)
2. Solve this a linear (matrix) equation

RHS

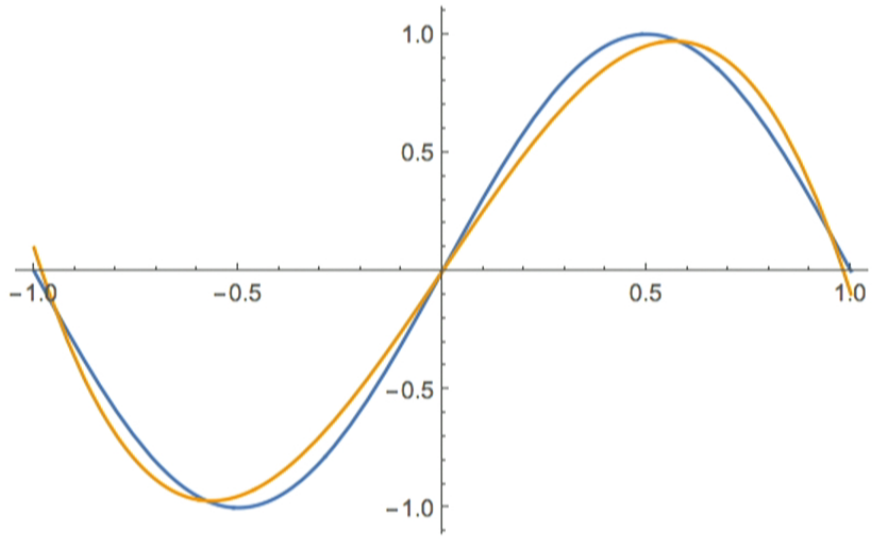
Project RHS onto our basis functions

In[62]:=
```
drhs = rhs // f2c // N
```

Out[62]=
```
{0., 0.569231, 0., -0.666917, 0.}
```

In[63]:=
```
Plot[{rhs[x], (drhs // c2f)[x]}, {x, -1, +1}]
```

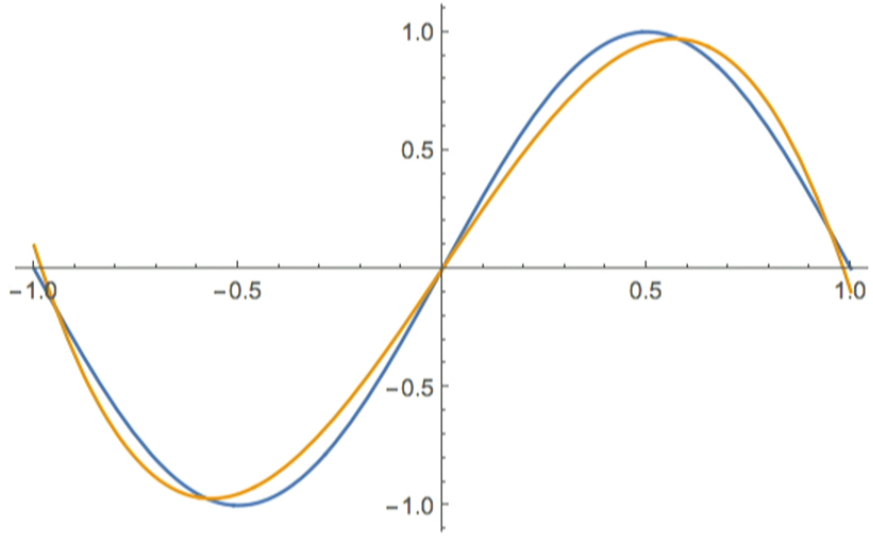
Out[63]=



150% ▸

In[63]:=
```
Plot[{rhs[x], (drhs // c2f)[x]}, {x, -1, +1}]
```

Out[63]=



Project boundary conditions onto our basis

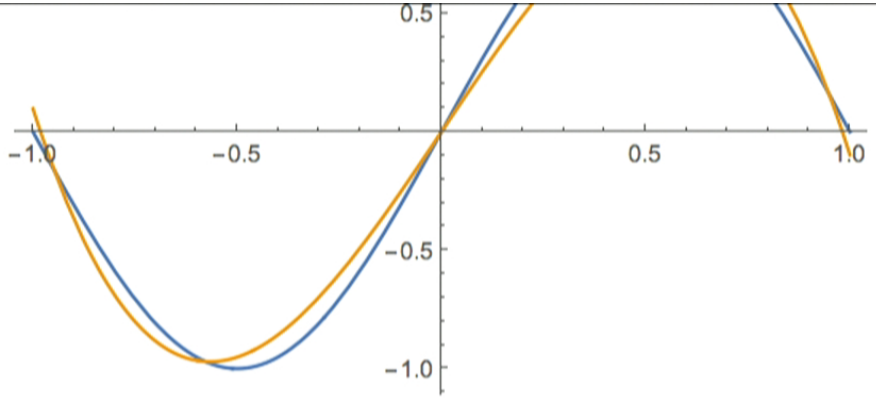
In[64]:=
```
dbcs = {f[-1], f[+1]} /. (bcs /. (lhs_ == rhs_) → (lhs → rhs)) // N
```

Out[64]=
```
{0.1, 0.}
```

```
drhs1 = Join[Take[drhs, n - 2 + 1], {0, 0}]

dbcs1 = Join[Table[0, {i, n - 2 + 1}], dbcs]

nsol = LinearSolve[dd.dd + bb1, drhs1 + dbcs1]

Plot[{f[x] /. asol[[1]], c2f[nsol][x]}, {x, -1, +1}]
```

150%

Out[63]=



Project boundary conditions onto our basis

In[64]:=
```
dbcs = {f[-1], f[+1]} /. (bcs /. (lhs_ == rhs_) → (lhs → rhs)) // N
```

Out[64]=
```
{0.1, 0.}
```

In[65]:=
```
drhs1 = Join[Take[drhs, n - 2 + 1], {0, 0}]
```

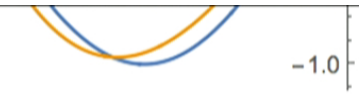
Out[65]=
```
{0., 0.569231, 0., 0, 0}
```

In[66]:=
```
dbcs1 = Join[Table[0, {i, n - 2 + 1}], dbcs]
```

Out[66]=
```
{0, 0, 0, 0.1, 0.}
```

```
nsol = LinearSolve[dd.dd + bb1, drhs1 + dbcs1]

Plot[{f[x] /. asol[[1]], c2f[nsol][x]}, {x, -1, +1}]
```

150%



Project boundary conditions onto our basis

In[64]:=
```
dbcs = {f[-1], f[+1]} /. (bcs /. (lhs_ == rhs_) → (lhs → rhs)) // N
```

Out[64]=
```
{0.1, 0.}
```

Take first (n-1) conditions from differential equation

In[65]:=
```
drhs1 = Join[Take[drhs, n - 2 + 1], {0, 0}]
```

Out[65]=
```
{0., 0.569231, 0., 0, 0}
```

Take 2 conditions from boundary conditions

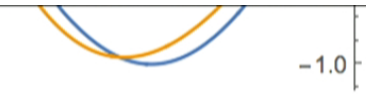
In[66]:=
```
dbcs1 = Join[Table[0, {i, n - 2 + 1}], dbcs]
```

Out[66]=
```
{0, 0, 0, 0.1, 0.}
```

```
nsol = LinearSolve[dd.dd + bb1, drhs1 + dbcs1]
```

```
Plot[{f[x] /. asol[[1]], c2f[nsol][x]}, {x, -1, +1}]
```

150% ▸



Project boundary conditions onto our basis

```
In[64]:= dbcs = {f[-1], f[+1]} /. (bcs /. (lhs_ == rhs_) → (lhs → rhs)) // N
Out[64]= {0.1, 0.}
```

Take first (n-1) conditions from differential equation

```
In[65]:= drhs1 = Join[Take[drhs, n - 2 + 1], {0, 0}]
Out[65]= {0., 0.569231, 0., 0, 0}
```

Take 2 conditions from boundary conditions

```
In[66]:= dbcs1 = Join[Table[0, {i, n - 2 + 1}], dbcs]
Out[66]= {0, 0, 0, 0.1, 0.}
```

Stitch conditions together; solve them

```
In[67]:= nsol = LinearSolve[dd.dd + bb1, drhs1 + dbcs1]
Out[67]= {0.0707107, -0.0737179, 0., 0.0237179, 0.}
```

```
Plot[{f[x] /. asol[[1]], c2f[nsol][x]}, {x, -1, +1}]
```

150% ▸

Take 2 conditions from boundary conditions

In[66]:=
```
dbcs1 = Join[Table[0, {i, n - 2 + 1}], dbcs]
```

Out[66]=
```
{0, 0, 0, 0.1, 0.}
```

Stitch conditions together; solve them

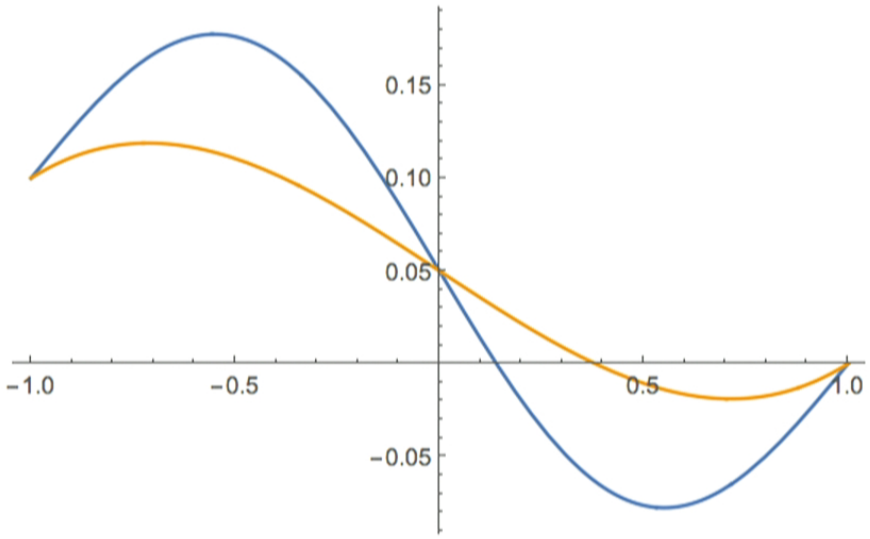
In[67]:=
```
nsol = LinearSolve[dd.dd + bb1, drhs1 + dbcs1]
```

Out[67]=
```
{0.0707107, -0.0737179, 0., 0.0237179, 0.}
```

Show solution

In[68]:=
```
Plot[{f[x] /. asol[[1]], c2f[nsol][x]}, {x, -1, +1}]
```

Out[68]=



150%

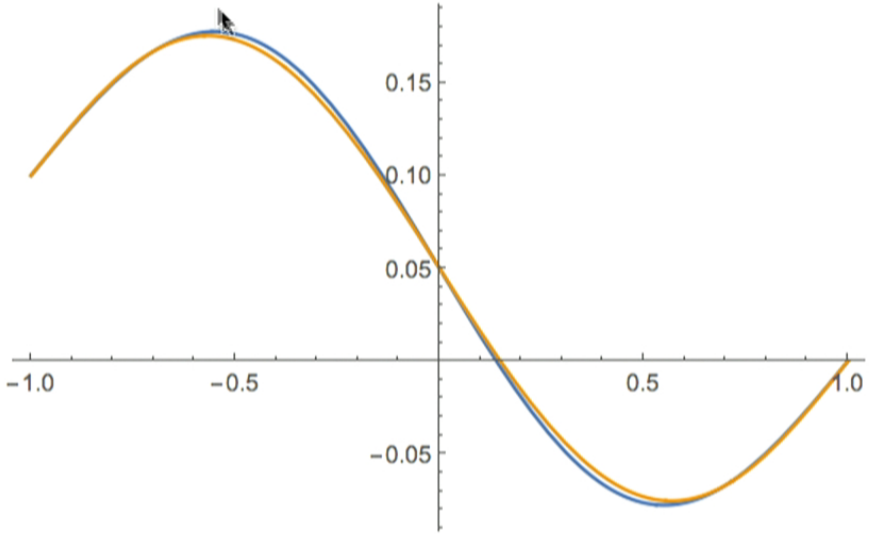
In[52]:=
```
nsol = LinearSolve[dd.dd + bb1, drhs1 + dbcs1]
```

Out[52]=
```
{0.0707107, -0.107064, 0., 0.0654002, 0., -0.00833646, 0.}
```

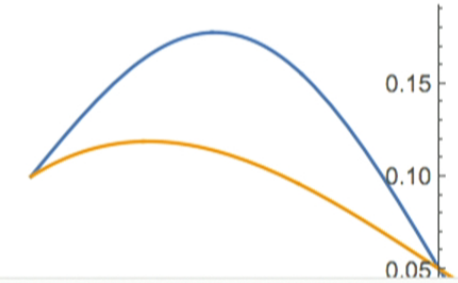
Show solution

In[53]:=
```
Plot[{f[x] /. asol[[1]], c2f[nsol][x]}, {x, -1, +1}]
```

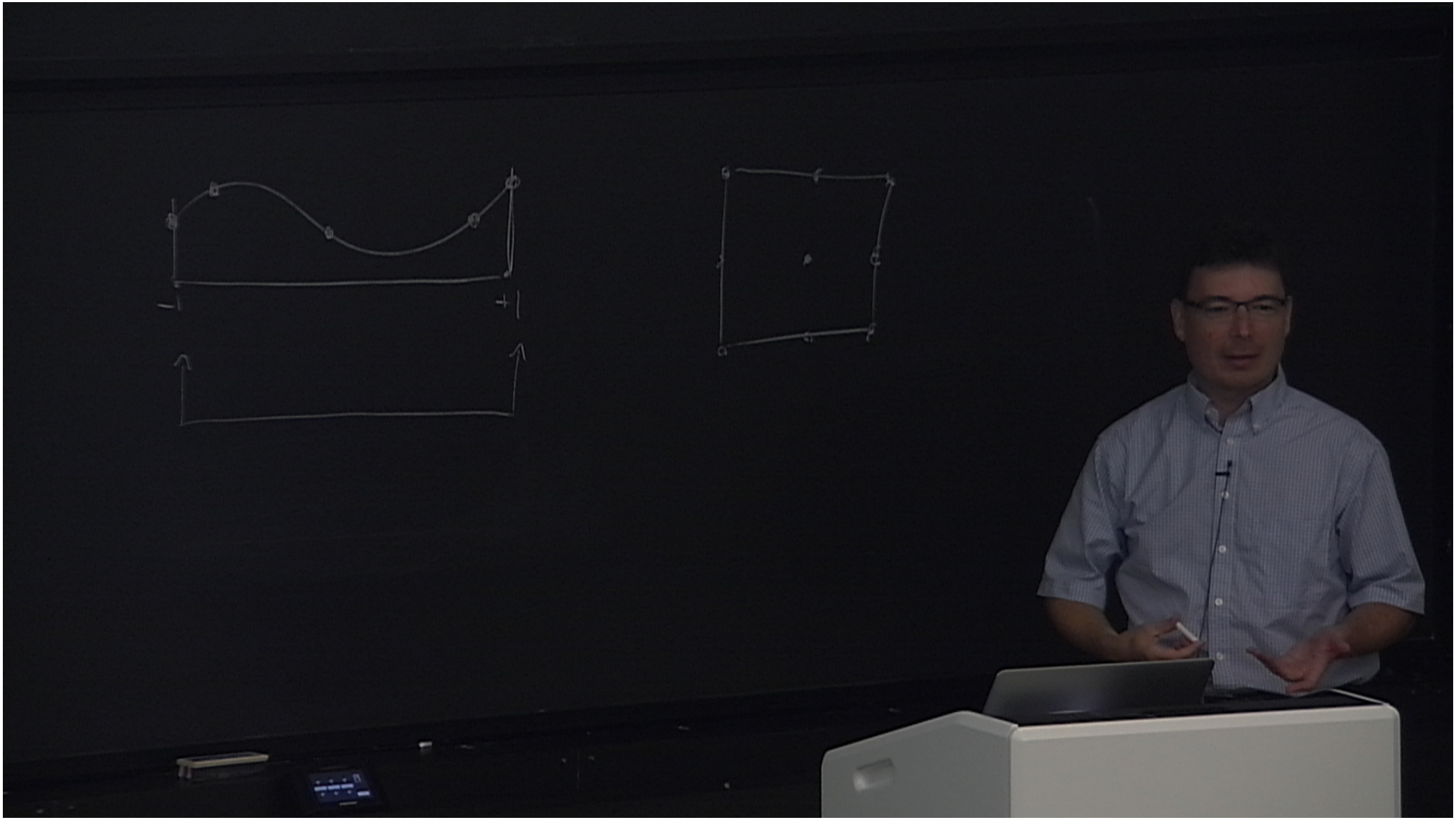
Out[53]=



Out[68]=



150%


−1
+1