Matrices, Geometry&*Mathematica*

Authors: Bruce Carpenter, Bill Davis and Jerry Uhl ©2001

Producer: Bruce Carpenter

Publisher: Math Everywhere, Inc.

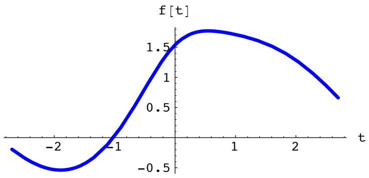
MGM.11 Function Spaces and Root-Mean-Square Approximation *TUTORIALS*

T.1) Scaling for other intervals

T.1.a.i) Polynomial approximation on intervals [-L,L] with L ≠ 1

Here is a function f[t] plotted on [-L,L] with L = 2.7

```
L = 2.7;
Clear[f, t];
f[t_] = Sin[t] + (2 / (1.3 + t^2));
fplot = Plot[f[t], {t, -L, L},
  PlotStyle -> {{Thickness[0.01], Blue}}, AxesLabel -> {"t", "f[t]"}];
```



Come up with a pretty good root-mean square approximation of f[t] on [-L.L] by polynomials.

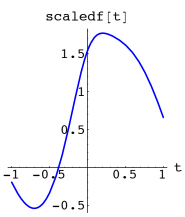
Answer:

If the given interval had been [-1,1], this would have been just a matter of copying, pasting and editing from the Basics.

To take advantage of this fact, you just scale f[t] on [-L,L] this way:

```
Clear[scaledf];
scaledf[t_] = f[t L]
scaledfplot =
 Plot[scaledf[t], {t, -1, 1}, PlotStyle -> {{Thickness[0.01], Blue}},
  AxesLabel -> {"t", "scaledf[t]"}];
```

$$\frac{2}{1.3+7.29\,t^2}+\mathrm{Sin}[2.7\,t]$$

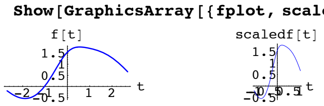


The formula

scaledf[t] = f[t L]

tells you that as t advances from -1 to 1, scaledf[t] does the same thing f[t] does as t advances from -L to L.

```
Show[GraphicsArray[{fplot, scaledfplot}]];
```



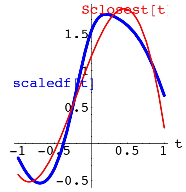
Now approximate scaledf[t] with polynomials on [-1,1]:

```
khigh = 4;
Clear[s, k, t];
s_k_[t_] := LegendreP[k - 1, t];
a = -1;
b = 1;

Clear[fouriercoeff, Sclosest, k];
fouriercoeff[k_] := NIntegrate[scaledf[t] s_k[t], {t, a, b}] / NIntegrate[s_k[t] s_k[t], {t, a, b}];

Sclosest[t_] = Sum_{k=1}^{khigh} fouriercoeff[k] s_k[t]
```

$$0.760913+1.1804\,t-0.862352\left(-\frac{1}{2}+\frac{3t^2}{2}\right)-0.861059\left(-\frac{3t}{2}+\frac{5t^3}{2}\right)$$

Check the quality of the approximation:

```
scaledfitplot = Plot[{scaledf[t], Sclosest[t]}, {t, -1, 1},
  PlotStyle -> {{Thickness[0.02], Blue}, {Thickness[0.01], Red}},
  AxesLabel -> {"t", ""},
  Epilog -> {{Blue, Text["scaledf[t]", {a + 0.5, f[a + 0.5]}]},
    {Red, Text["Sclosest[t]", {b - 0.5, Sclosest[b - 0.5]}]}}];
```



Not great.

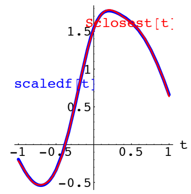
Use more Legendre Polynomials to go after a better approximation:

```
khigh = 12;
Clear[s, k, t];
s_k_[t_] := LegendreP[k - 1, t];
a = -1;
b = 1;

Clear[fouriercoeff, Sclosest, k];
fouriercoeff[k_] := NIntegrate[scaledf[t] s_k[t], {t, a, b}] / NIntegrate[s_k[t] s_k[t], {t, a, b}];

Sclosest[t_] = Sum_{k=1}^{khigh} fouriercoeff[k] s_k[t]
```

$$0.760913+1.1804\,t-0.862352\left(-\frac{1}{2}+\frac{3t^2}{2}\right)-0.861059\left(-\frac{3t}{2}+\frac{5t^3}{2}\right)+$$
$$0.515619\left(\frac{3}{8}-\frac{15t^2}{4}+\frac{35t^4}{8}\right)+0.114087\left(\frac{15t}{8}-\frac{35t^3}{4}+\frac{63t^5}{8}\right)-$$
$$0.273808\left(-\frac{5}{16}+\frac{105t^2}{16}-\frac{315t^4}{16}+\frac{231t^6}{16}\right)-$$
$$0.00623132\left(-\frac{35t}{16}+\frac{315t^3}{16}-\frac{693t^5}{16}+\frac{429t^7}{16}\right)+$$
$$0.138033\left(\frac{35}{128}-\frac{315t^2}{32}+\frac{3465t^4}{64}-\frac{3003t^6}{32}+\frac{6435t^8}{128}\right)+$$
$$0.000185783\left(\frac{315t}{128}-\frac{1155t^3}{32}+\frac{9009t^5}{64}-\frac{6435t^7}{32}+\frac{12155t^9}{128}\right)-$$
$$0.0675708\left(-\frac{63}{256}+\frac{3465t^2}{256}-\frac{15015t^4}{128}+\frac{45045t^6}{128}-\frac{109395t^8}{256}+\frac{46189t^{10}}{256}\right)-3.49184\times10^{-6}$$
$$\left(-\frac{693t}{256}+\frac{15015t^3}{256}-\frac{45045t^5}{128}+\frac{109395t^7}{128}-\frac{230945t^9}{256}+\frac{88179t^{11}}{256}\right)$$

Check the quality of the approximation:

```
scaledfitplot = Plot[{scaledf[t], Sclosest[t]}, {t, -1, 1},
  PlotStyle -> {{Thickness[0.02], Blue}, {Thickness[0.01], Red}},
  AxesLabel -> {"t", ""},
  Epilog -> {{Blue, Text["scaledf[t]", {a + 0.5, f[a + 0.5]}]},
    {Red, Text["Sclosest[t]", {b - 0.5, Sclosest[b - 0.5]}]}}];
```



Lookin' pretty good.

Now unscale Sclosest[t] via

unscaledSclosest[t] = Sclosest[$\frac{t}{L}$]:

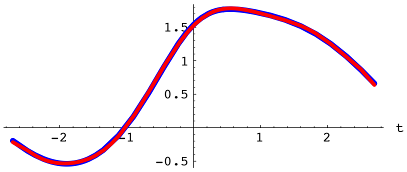
As as t advances from -L to L, unscaledSclosest[t] does the same thing Sclosest[t] does as t advances from -1,1.

```
Clear[unscaledSclosest];
unscaledSclosest[t_] = Sclosest[t/L]
```

$$0.760913+0.437186\,t-0.862352\left(-\frac{1}{2}+0.205761\,t^2\right)-$$
$$0.861059\,(-0.555556\,t+0.127013\,t^3)+$$
$$0.515619\left(\frac{3}{8}-0.514403\,t^2+0.0823233\,t^4\right)+$$
$$0.114087\,(0.694444\,t-0.444546\,t^3+0.0548822\,t^5)-$$
$$0.273808\left(-\frac{5}{16}+0.900206\,t^2-0.370455\,t^4+0.0372657\,t^6\right)-$$
$$0.00623132\,(-0.810185\,t+1.00023\,t^3-0.301852\,t^5+0.0256325\,t^7)+$$
$$0.138033\left(\frac{35}{128}-1.35031\,t^2+1.01875\,t^4-0.242227\,t^6+0.0178003\,t^8\right)+$$
$$0.000185783$$
$$(0.911458\,t-1.83375\,t^3+0.98102\,t^5-0.192244\,t^7+0.0124529\,t^9)-$$
$$0.0675708\left(-\frac{63}{256}+1.85667\,t^2-2.20729\,t^4+0.908352\,t^6-0.151303\,t^8+\right.$$
$$\left.0.00876316\,t^{10}\right)-3.49184\times10^{-6}\,(-1.0026\,t+2.97985\,t^3-$$
$$2.45255\,t^5+0.817036\,t^7-0.118303\,t^9+0.00619618\,t^{11})$$

See the fit:

```
fitplot = Plot[{f[t], unscaledSclosest[t]}, {t, -L, L},
  PlotStyle -> {{Thickness[0.015], Blue}, {Thickness[0.01], Red}},
  AxesLabel -> {"t", ""}];
```



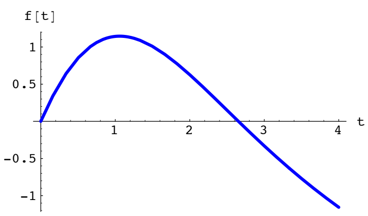
Just fine!.

T.1.a.ii) Sine wave approximation on intervals [0,L] with L ≠ π

Here is a function f[t] plotted on [0,L] with L = 4:

```
L = 4
Clear[f, t];
f[t_] = t (6.2/(1.3 + e^t) - 0.4);
fplot = Plot[f[t], {t, 0, L},
   PlotStyle → {{Thickness[0.01], Blue}}, AxesLabel → {"t", "f[t]"}];
```

4



Come up with a pretty good root-mean square approximation of f[t] on [0,L] by sine waves.

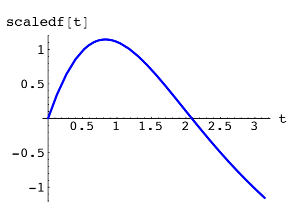
Answer:

If the given interval had been [0,π], this would have been just a matter of copying, pasting and editing from the Basics.

To take advantage of this fact, you just scale f[t] on [0,L] this way:

```
Clear[scaledf];
scaledf[t_] = f[t L/π];
```

```
scaledfplot =
  Plot[scaledf[t], {t, 0, π}, PlotStyle → {{Thickness[0.01], Blue}},
   AxesLabel → {"t", "scaledf[t]"}];
```

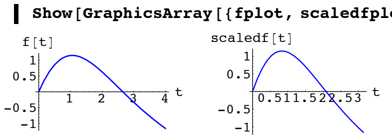


The formula

scaledf[t] = f[t $\frac{L}{\pi}$]

tells you that as t advances from 0 to π, scaledf[t] does the same thing as f[t] does as t advances from 0 to L.

```
Show[GraphicsArray[{fplot, scaledfplot}]];
```



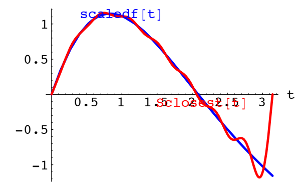
Now approximate scaledf[t] with Sine waves on [0,π]:

```
khigh = 15;
Clear[s, k, t];
s_k[t_] := Sin[k t]
a = 0;
b = π;

Clear[fouriercoeff, Sclosest, k];
fouriercoeff[k_] := NIntegrate[scaledf[t] s_k[t], {t, a, b}] / NIntegrate[s_k[t] s_k[t], {t, a, b}];

Sclosest[t_] = Sum[fouriercoeff[k] s_k[t], {k, 1, khigh}]
```

0.566507 Sin[t] + 0.826316 Sin[2 t] − 0.133321 Sin[3 t] +
0.231253 Sin[4 t] − 0.127908 Sin[5 t] + 0.135412 Sin[6 t] −
0.098461 Sin[7 t] + 0.0972656 Sin[8 t] − 0.0786887 Sin[9 t] +
0.0762768 Sin[10 t] − 0.0652339 Sin[11 t] + 0.0628788 Sin[12 t] −
0.0556078 Sin[13 t] + 0.0535447 Sin[14 t] − 0.0484149 Sin[15 t]

Check the quality of the approximation:

```
scaledfitplot = Plot[{scaledf[t], Sclosest[t]}, {t, a, b},
   PlotStyle -> {{Thickness[0.01], Blue}, {Thickness[0.01], Red}},
   AxesLabel -> {"t", ""},
   Epilog -> {{Blue, Text["scaledf[t]", {a + 1, f[a + 1]}]},
     {Red, Text["Sclosest[t]", {b - 1, Sclosest[b - 1]}]}}];
```



Not bad - except at the right endpoint.

Now unscale Sclosest[t] via

unscaledSclosest[t] = Sclosest[$\frac{t\pi}{L}$]:

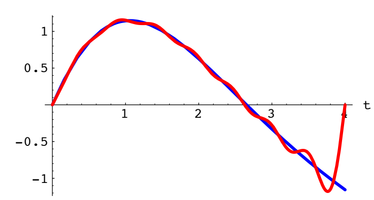
As as t advances from 0 to L, unscaledSclosest[t] does the same thing Sclosest[t] does as t advances from 0 to π.

```
Clear[unscaledSclosest];
unscaledSclosest[t_] = Sclosest[t π/L]
```

$0.566507 \sin\left[\frac{\pi t}{4}\right] + 0.826316 \sin\left[\frac{\pi t}{2}\right] - 0.133321 \sin\left[\frac{3\pi t}{4}\right] + 0.231253 \sin[\pi t] - 0.127908 \sin\left[\frac{5\pi t}{4}\right] + 0.135412 \sin\left[\frac{3\pi t}{2}\right] - 0.098461 \sin\left[\frac{7\pi t}{4}\right] + 0.0972656 \sin[2\pi t] - 0.0786887 \sin\left[\frac{9\pi t}{4}\right] + 0.0762768 \sin\left[\frac{5\pi t}{2}\right] - 0.0652339 \sin\left[\frac{11\pi t}{4}\right] + 0.0628788 \sin[3\pi t] - 0.0556078 \sin\left[\frac{13\pi t}{4}\right] + 0.0535447 \sin\left[\frac{7\pi t}{2}\right] - 0.0484149 \sin\left[\frac{15\pi t}{4}\right]$

See the fit:

```
fitplot = Plot[{f[t], unscaledSclosest[t]}, {t, 0, L},
  PlotStyle -> {{Thickness[0.01], Blue}, {Thickness[0.01], Red}},
  AxesLabel -> {"t", ""}];
```



Not bad except at the right endpoint.

T.2) Weighted root-mean -square -distance between two functions

Weighted dot product of two functions

Families orthogonal with respect to a weight function

Chebyshev polynomials

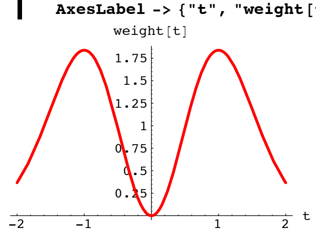
Best weighted root-mean-square -distance approximation

T.2.a.i) Examples of weight functions

A weight function on an interval [a,b] is any function weight[t] with weight[t] ≥ 0 for all t's with a ≤ t ≤ b.

Here's an example of a weight function on [-2,2]

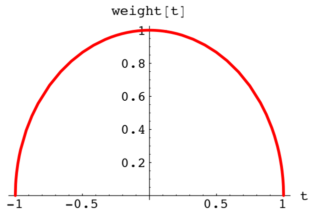
```
a = -2;
b = 2;
Clear[weight, t];
weight[t_] = 5 t^2 E^(-t^2);

Plot[weight[t], {t, a, b}, PlotStyle -> {{Thickness[0.01], Red}},
  AxesLabel -> {"t", "weight[t]"}, AspectRatio -> 1 / GoldenRatio];
```



This function puts no weight on t = 0 and puts heavier weight on what's going on around t = 1 and t = -1.

Here's an example of a weight function on [-1,1]

```
a = -1;
b = 1;
Clear[weight, t];
weight[t_] = √(1 - t^2);

Plot[weight[t], {t, a, b}, PlotStyle -> {{Thickness[0.01], Red}},
  AxesLabel -> {"t", "weight[t]"}, AspectRatio -> 1 / GoldenRatio];
```

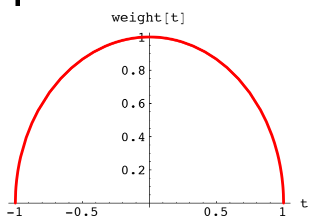


This function puts no weight on t = -1 and no weight on t = 1 but puts heavier weight on what's going on around t =0.

□T.2.a.ii) Weighted root-mean-square distance between two functions on an interval.

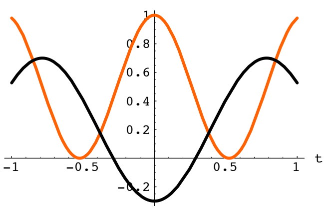
.Here's a weight function on [a,b] with a = -1 and b = 1:

```
a = -1;
b = 1;
Clear[weight, t];
weight[t_] = √(1 - t^2);
Plot[weight[t], {t, a, b}, PlotStyle → {{Thickness[0.01], Red}},
  AxesLabel → {"t", "weight[t]"}, AspectRatio → 1/GoldenRatio];
```



Here are two functions f[t] and g[t] on [a,b] with a = -1 and b = 1:

```
Clear[f, g, t];
f[t_] = Cos[3 t]^2;
g[t_] = Sin[2 t]^2 - 0.3;
Plot[{f[t], g[t]}, {t, a, b},
  PlotStyle -> {{Thickness[0.01], CadmiumOrange},
    {Thickness[0.01], Black}}, AxesLabel -> {"t", ""}];
```



Calculate the weighted root-mean-square distance from f[t] to g[t] with respect to the given weight function on [a,b].

□Answer:

The weighted root-mean-square distance from f[t] to g[t] with respect to this weight function on [a,b] is

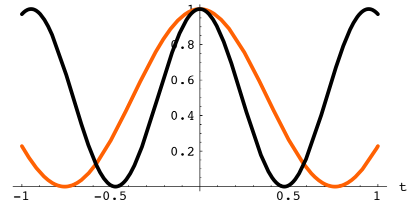
$$\sqrt{\int_a^b (f[t] - g[t])^2\, \text{weight}[t]\, dt}:$$

```
√(NIntegrate[(f[t] - g[t])^2 weight[t], {t, a, b}])
```

0.842109

See what happens when you go with new functions f[t] and g[t]:

```
Clear[f, g, t];
m = Random[Real, {1, 3}];
n = Random[Real, {3, 6}];
f[t_] = Cos[m t]^2;
g[t_] = Cos[n t]^2;

Plot[{f[t], g[t]}, {t, a, b},
  PlotStyle -> {{Thickness[0.01], CadmiumOrange},
    {Thickness[0.01], Black}}, AxesLabel -> {"t", ""}];
```



The weighted root-mean-square distance from f[t] to g[t] with respect to this weight function on [a,b] is

$$\sqrt{\int_a^b (f[t] - g[t])^2\, \text{weight}[t]\, dt}:$$

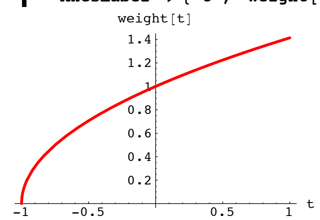
```
√(NIntegrate[(f[t] - g[t])^2 weight[t], {t, a, b}])
```

0.506702

□T.2.a.iii) Weighted dot product of two functions on an interval

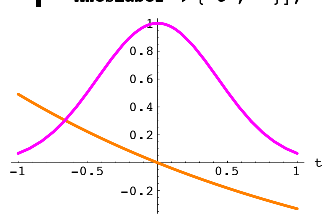
.Here's a weight function on [a,b] with a = -1 and b = 1:

```
a = -1;
b = 1;
Clear[weight, t];
weight[t_] = Sqrt[1 + t];

Plot[weight[t], {t, a, b}, PlotStyle -> {{Thickness[0.01], Red}},
  AxesLabel -> {"t", "weight[t]"}, AspectRatio -> 1 / GoldenRatio];
```



Here are two functions f[t] and g[t] on [a,b] with a = -1 and b = 1:

```
Clear[f, g, t];
f[t_] = E^(-0.4 t) - 1.0;
g[t_] = E^(-2.7 t^2);

Plot[{f[t], g[t]}, {t, a, b}, PlotStyle ->
   {{Thickness[0.01], Orange}, {Thickness[0.01], Magenta}},
  AxesLabel -> {"t", ""}];
```



Calculate weighted dot product f•g of f[t] and g[t] on [a,b] with respect to the given weight function.

□Answer:

Just calculate:

$$f \bullet g = \int_a^b f[t]\, g[t]\, \text{weight}[t]\, dt:$$

```
NIntegrate[f[t] g[t] weight[t] , {t, a, b}]
```

-0.0244278

They are almost perpendicular with respect to the weighted dot product.

□T.2.b.i) Weighted root-mean-square approximation via Chebyshev polynomials system

The set of functions

$\{s_1[t], s_2[t], s_3[t], \ldots, s_k[t], \ldots\}$

for

$s_k[t] = \text{ChebyshevT}[k - 1, t]$

is an orthogonal set with respect to the weight function

$\text{weight}[t] = 1/\sqrt{1 - t^2}$

on the interval $a \le t \le b$ with $a = -1$ and $b = 1$.

Here are the formulas for the first seven:

```
Clear[k];
ColumnForm[Table[ChebyshevT[k - 1, t], {k, 1, 7}]]
```

1

t

$-1 + 2 t^2$

$-3 t + 4 t^3$

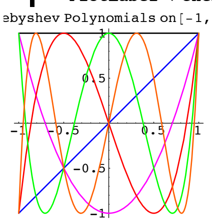
$1 - 8t^2 + 8t^4$

$5t - 20t^3 + 16t^5$

$-1 + 18t^2 - 48t^4 + 32t^6$

Here are plots of the first six:

```
Clear[s, g, k, t];
s_k_[t_] := ChebyshevT[k - 1, t];

a = -1;
b = 1;

Plot[{s1[t], s2[t], s3[t], s4[t], s5[t], s6[t]}, {t, a, b}, PlotStyle →
    {{Black}, {Blue}, {Magenta}, {Red}, {Green}, {CadmiumOrange}},
  PlotLabel → Chebyshev Polynomials on[a, b]];
```



To illustrate the fact that this set of functions

$\{s_1[t], s_2[t], s_3[t], \ldots, s_k[t], \ldots\}$

is an orthogonal set with respect to the weight function

$\text{weight}[t] = \frac{1}{\sqrt{1-t^2}}$

on the interval $a \le t \le b$ with $a = -1$ and $b = 1$,
look at this calculation of

$\int_a^b s_k[t]\, s_p[t]\, \text{weight}[t]\, dt$

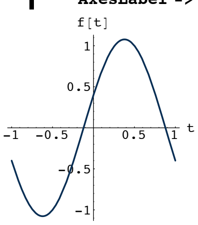
for random positive integers p and k with $p \neq k$

```
Clear[weight];
weight[t_] = 1/Sqrt[1 - t^2];
k = Random[Integer, {1, 8}];
p = Random[Integer, {k + 1, 2 k}];
∫_a^b s_k[t] s_p[t] weight[t] ⅆt
```

0

Here is a function f[t] plotted on [a,b] = [-1,1]:

```
Clear[f, t];
a = 1;
b = -1;
f[t_] = Sin[π t] + 0.4 Cos[π t];

fplot = Plot[f[t], {t, a, b}, PlotStyle -> {{Thickness[0.01], Indigo}}
   AxesLabel -> {"t", "f[t]"}];
```



Go with the function space S[a,b] spanned by:

```
khigh = 5;
Table[ s_k[t], {k, 1, khigh}]
```

$\{1, t, -1 + 2t^2, -3t + 4t^3, 1 - 8t^2 + 8t^4\}$

Come up with the function Sclosest[t] in S[a,b] so that
the weighted root-mean-square distance between f[t] and Sclosest is as small as possible

□ Answer:

It's not bad at all.

For each k you take the weighted component of f[t] in the direction of $s_k[t]$:

$\frac{f \bullet s_k}{s_k \bullet s_k}\, s_k[t]$

(This uses the weighted dot product$f \bullet g = \int\int_a^b f[t]\, g[t]\, \text{weight}[t]\, dt$.)

and then you add 'em up to get

$\text{Sclosest}[t] = \sum_{k=1}^{\text{khigh}} \frac{f \bullet s_k}{s_k \bullet s_k}\, s_k[t]$

See the formula:

```
Clear[weightedfouriercoeff, Sclosest, k];
weightedfouriercoeff[k_] :=
  NIntegrate[f[t] s_k[t] weight[t], {t, a, b}] / NIntegrate[s_k[t] s_k[t] weight[t], {t, a, b}];

Sclosest[t_] = Sum[weightedfouriercoeff[k] s_k[t], {k, 1, khigh}]
```

Power::infy : Infinite expression $\frac{1}{\sqrt{0.}}$ encountered.

Power::infy : Infinite expression $\frac{1}{\sqrt{0.}}$ encountered.

Power::infy : Infinite expression $\frac{1}{\sqrt{0.}}$ encountered.

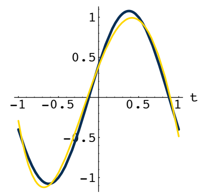
General::stop : Further output of Power::infy will be suppressed during this calculation.

$-0.121697 + 0.569231\,t - 0.388347\,(-1 + 2t^2) - 0.666917\,(-3t + 4t^3) + 0.12114\,(1 - 8t^2 + 8t^4)$

Disregard the error messages.

See the quality of the approximation of f[t] by Sclosest[t]:

```
fitplot = Plot[{f[t], Sclosest[t]},
   {t, a, b}, PlotStyle -> {{Thickness[0.015], Indigo},
     {Thickness[0.01], Gold}}, AxesLabel -> {"t", ""},
   Epilog -> {{Indigo, Text["f[t]", {a + 0.5, f[a + 0.5]}]},
     {Gold, Text["Sclosest[t]", {b - 0.5, Sclosest[b - 0.5]}]}}];
```



□ T.2.b.ii) Increasing the quality of the approximation

Keep everything the same as in part i), but go with the function space S[a,b] spanned by more Chebyshev polynomials:

```
khigh = 8;
Table[ s_k[t], {k, 1, khigh}]
```

$\{1, t, -1 + 2t^2, -3t + 4t^3, 1 - 8t^2 + 8t^4, 5t - 20t^3 + 16t^5, -1 + 18t^2 - 48t^4 + 32t^6, -7t + 56t^3 - 112t^5 + 64t^7\}$

Come up with the function Sclosest[t] in S[a,b] so that
the weighted root-mean-square distance between f[t] and Sclosest is as small as possible

□ Answer:

```
Clear[weightedfouriercoeff, Sclosest, k];
weightedfouriercoeff[k_] :=
  NIntegrate[f[t] s_k[t] weight[t], {t, a, b}] / NIntegrate[s_k[t] s_k[t] weight[t], {t, a, b}];

Sclosest[t_] = Sum[weightedfouriercoeff[k] s_k[t], {k, 1, khigh}]
```

Power::infy : Infinite expression $\frac{1}{\sqrt{0.}}$ encountered.

Power::infy : Infinite expression $\frac{1}{\sqrt{0.}}$ encountered.

Power::infy : Infinite expression $\frac{1}{\sqrt{0.}}$ encountered.

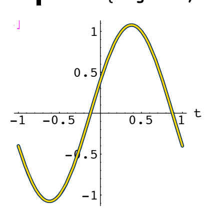
General::stop : Further output of Power::infy will be suppressed during this calculation.

$-0.121697 + 0.569231\,t - 0.388347\,(-1 + 2t^2) - 0.666917\,(-3t + 4t^3) + 0.12114\,(1 - 8t^2 + 8t^4) + 0.104282\,(5t - 20t^3 + 16t^5) - 0.0116368\,(-1 + 18t^2 - 48t^4 + 32t^6) - 0.00684063\,(-7t + 56t^3 - 112t^5 + 64t^7)$

Disregard the error messages.

See the quality of the approximation of f[t] by Sclosest[t]:

```
fitplot = Plot[{f[t], Sclosest[t]}, {t, a, b},
   PlotStyle -> {{Thickness[0.02], Indigo}, {Thickness[0.01], Gold}},
   AxesLabel -> {"t", ""},
   Epilog -> {{Indigo, Text["f[t]", {a + 0.5, f[a + 0.5]}]},
     {Magenta, Text["Sclosest[t]", {b - 0.5, Sclosest[b - 0.5]}]}}];
```



Sharing ink all the way..

Great!

The weighted root-mean-square distance between f[t] and Sclosest[t] is:

```
Sqrt[NIntegrate[(f[t] - Sclosest[t])^2 weight[t], {t, a, b}]]
```

Power::infy : Infinite expression $\frac{1}{\sqrt{0.}}$ encountered.

Power::infy : Infinite expression $\frac{1}{\sqrt{0.}}$ encountered.

Power::infy : Infinite expression $\frac{1}{\sqrt{0.}}$ encountered.

General::stop : Further output of Power::infy will be suppressed during this calculation.

$0.000765375\,i$

Disregard the error messages.

Love those Chebyshev polynomials.

T.3) Fourier Sine approximation and the heat equation

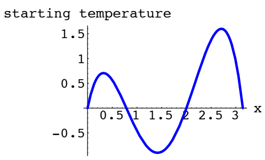
Fourier Sine approximation and the wave equation

Fourier invented Fourier approximations for
the purpose of working on this very problem.

T.3.a.i) Fourier sine approximation and the heat equation

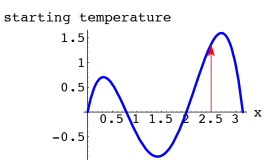
Start with a heated wire L units long with the temperature allowed to vary from position to position on the wire. Think of the wire as the interval $0 \le x \le L$. At the start of the experiment, you instantly cool the ends at $x = 0$ and $x = L$ and maintain these ends at temperature 0, and you take pains to guarantee that the rest of the wire is perfectly insulated.
At the start of this particular experiment, the temperature of the wire at position x (for $0 \le x \le L = \pi$) is given by the following function startertemp[x]:

```
L = π;
Clear[startertemp, x];
startertemp[x_] = x (x - π/4) (x - 2) (π - x);

starterplot = Plot[startertemp[x],
   {x, 0, L}, PlotStyle → {{Thickness[0.015], Blue}},
   AxesLabel → {"x", "starting temperature"}];
```



To fully understand this plot, look at this:

```
xvalue = 2.5;
Clear[pointer];
pointer[x_] :=
  Arrow[{0, startertemp[x]}, Tail → {x, 0}, VectorColor → Red];

Show[starterplot, pointer[xvalue]];
```



Think of the interval $[0,L] = [0\ \pi]$ as the wire.
The tip of the pointer tells you the starting temperature (at time $t = 0$) at the tail of the pointer.
Your problem here is to use Fourier Sine approximation on $[0,\pi]$ of startertemp[x] come up with a function temp[x,t] that estimates the temperature of the wire at position x at time t after the experiment begins.
Do it.

Answer:

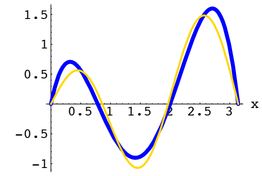
First do a Sine approximation of f[x] = startertemp[x] on [0,Pi]:

```
khigh = 3;
Clear[s, g, f, k, x];
f[x_] = startertemp[x];
s_k[x_] := Sin[k x];
a = 0;
b = π;

Clear[fouriercoeff, Sclosest, k];
fouriercoeff[k_] :=
  fouriercoeff[k] = NIntegrate[f[t] s_k[t], {t, a, b}] / NIntegrate[s_k[t] s_k[t], {t, a, b}];

Sclosest[x_] = Sum_{k=1}^{khigh} fouriercoeff[k] s_k[x]
```

-0.000229884 Sin[x] - 0.534292 Sin[2 x] + 1.00601 Sin[3 x]

Check out the quality of the fit:

```
fitplot = Plot[{startertemp[x], Sclosest[x]}, {x, a, b},
   PlotStyle → {{Thickness[0.02], Blue}, {Thickness[0.01], Gold}},
   AxesLabel → {"x", ""}];
```



That's not the greatest fit.

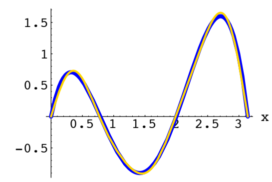
Make khigh bigger:

```
khigh = 8;
Clear[s, g, f, k, x];
f[x_] = startertemp[x];
s_k[x_] := Sin[k x];
a = 0;
b = π;

Clear[fouriercoeff, Sclosest, k];
fouriercoeff[k_] :=
  fouriercoeff[k] = NIntegrate[f[t] s_k[t], {t, a, b}] / NIntegrate[s_k[t] s_k[t], {t, a, b}];

Sclosest[x_] = Sum_{k=1}^{khigh} fouriercoeff[k] s_k[x]
```

-0.000229884 Sin[x] - 0.534292 Sin[2 x] +
1.00601 Sin[3 x] - 0.0667865 Sin[4 x] + 0.234682 Sin[5 x] -
0.0197886 Sin[6 x] + 0.0872708 Sin[7 x] - 0.00834831 Sin[8 x]

Check out the quality of the fit:

```
fitplot = Plot[{startertemp[x], Sclosest[x]}, {x, a, b},
   PlotStyle → {{Thickness[0.02], Blue}, {Thickness[0.01], Gold}},
   AxesLabel → {"x", ""}];
```



Beautiful fit - go with this one.

To get temp[x,t], you look at:

```
Sclosest[x]
```

-0.000229884 Sin[x] - 0.534292 Sin[2 x] +
1.00601 Sin[3 x] - 0.0667865 Sin[4 x] + 0.234682 Sin[5 x] -
0.0197886 Sin[6 x] + 0.0872708 Sin[7 x] - 0.00834831 Sin[8 x]

Pick off the coefficients of the Sin[k x] terms:

```
Clear[A, k];
A[k_] := Coefficient[Sclosest[x], Sin[k x]];

Table[A[k], {k, 1, khigh}]
```

{-0.000229884, -0.534292, 1.00601, -0.0667865,
0.234682, -0.0197886, 0.0872708, -0.00834831}

Now you're done because you can write down temp[x,t].

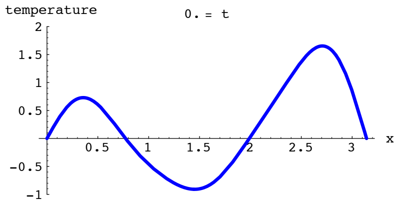
It's just:

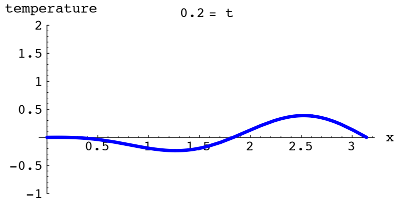
```
Clear[temp, x, t];
temp[x_, t_] = Sum_{k=1}^{khigh} A[k] E^(-k^2 t) Sin[k x]
```

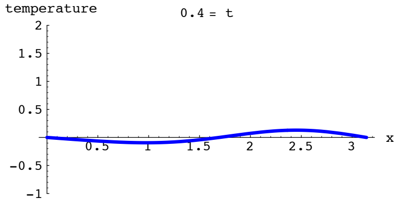
$$-0.000229884\, e^{-t} \operatorname{Sin}[x] - 0.534292\, e^{-4t} \operatorname{Sin}[2x] + 1.00601\, e^{-9t} \operatorname{Sin}[3x] - 0.0667865\, e^{-16t} \operatorname{Sin}[4x] + 0.234682\, e^{-25t} \operatorname{Sin}[5x] - 0.0197886\, e^{-36t} \operatorname{Sin}[6x] + 0.0872708\, e^{-49t} \operatorname{Sin}[7x] - 0.00834831\, e^{-64t} \operatorname{Sin}[8x]$$

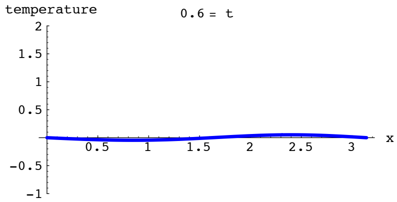
See the wire cool down:

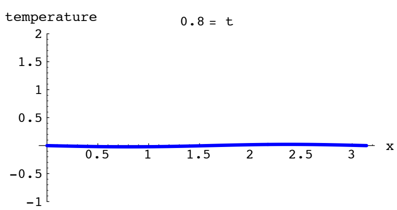
```
Clear[tempplotter];
tempplotter[t_] :=
  Plot[temp[x, t], {x, 0, π}, PlotStyle → {{Thickness[0.01], Blue}},
   PlotRange → {-1, 2}, AxesLabel → {"x", " temperature"},
   PlotLabel → N[t] "= t", AspectRatio → 1/2];

timejump = 0.2;

Table[tempplotter[t], {t, 0, 1.4, timejump}]
```

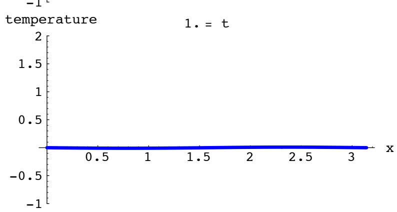


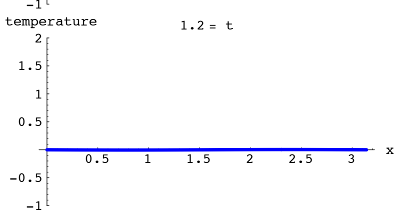


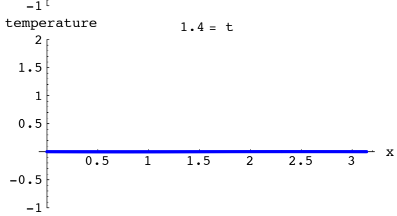













```
{- Graphics -, - Graphics -, - Graphics -, - Graphics -,
 - Graphics -, - Graphics -, - Graphics -, - Graphics -}
```

Grab all the plot and animate.

It takes about 1.2 time units for the wire to cool all the way down to 0.

T.3.a.ii) The heat equation explains why you want the exponentials

You get temp[x,t] by taking a rigged sine fit of starterposition[x]:

```
Sclosest[x]
```

$-0.000229884\,\mathrm{Sin}[x] - 0.534292\,\mathrm{Sin}[2x] + 1.00601\,\mathrm{Sin}[3x] - 0.0667865\,\mathrm{Sin}[4x] + 0.234682\,\mathrm{Sin}[5x] - 0.0197886\,\mathrm{Sin}[6x] + 0.0872708\,\mathrm{Sin}[7x] - 0.00834831\,\mathrm{Sin}[8x]$

And you get temp[x,t] by inserting Cosines into each term of the rigged sine fit:

```
temp[x, t]
```

$-0.000229884\,e^{-t}\,\mathrm{Sin}[x] - 0.534292\,e^{-4t}\,\mathrm{Sin}[2x] + 1.00601\,e^{-9t}\,\mathrm{Sin}[3x] - 0.0667865\,e^{-16t}\,\mathrm{Sin}[4x] + 0.234682\,e^{-25t}\,\mathrm{Sin}[5x] - 0.0197886\,e^{-36t}\,\mathrm{Sin}[6x] + 0.0872708\,e^{-49t}\,\mathrm{Sin}[7x] - 0.00834831\,e^{-64t}\,\mathrm{Sin}[8x]$

Why did that trick of inserting the exponentials work?

Answer:

Engineering studies have shown that after the appropriate unit adjustments are made, the function temp[x,t] satisfies the partial differential equation known as the heat equation

$\partial_{\{x,2\}}\,\mathrm{temp}[x, t] = \partial_t\,\mathrm{temp}[x, t]$

This is the same as D[temp[x,t], {x,2}] = D[temp[x,t], t];
in other words
the second derivative of temp[x,t[with respect to x equals the first derivative of temp[x,t] with respect to t.

with

-> temp[x, 0] = Sclosest[x]

and

-> temp[0, t] = 0 and temp[L, t] = 0 for all t's.

Inserting the exponentials made tempt[x,t] so all this:

Check

$\partial_{\{x,2\}}\,\mathrm{temp}[x, t] = \partial_t\,\mathrm{temp}[x, t]$:

```
∂{x,2} temp[x, t]
```

$0.000229884\,e^{-t}\,\mathrm{Sin}[x] + 2.13717\,e^{-4t}\,\mathrm{Sin}[2x] - 9.05407\,e^{-9t}\,\mathrm{Sin}[3x] + 1.06858\,e^{-16t}\,\mathrm{Sin}[4x] - 5.86704\,e^{-25t}\,\mathrm{Sin}[5x] + 0.712389\,e^{-36t}\,\mathrm{Sin}[6x] - 4.27627\,e^{-49t}\,\mathrm{Sin}[7x] + 0.534292\,e^{-64t}\,\mathrm{Sin}[8x]$

```
∂t temp[x, t]
```

$0.000229884\,e^{-t}\,\mathrm{Sin}[x] + 2.13717\,e^{-4t}\,\mathrm{Sin}[2x] - 9.05407\,e^{-9t}\,\mathrm{Sin}[3x] + 1.06858\,e^{-16t}\,\mathrm{Sin}[4x] - 5.86704\,e^{-25t}\,\mathrm{Sin}[5x] + 0.712389\,e^{-36t}\,\mathrm{Sin}[6x] - 4.27627\,e^{-49t}\,\mathrm{Sin}[7x] + 0.534292\,e^{-64t}\,\mathrm{Sin}[8x]$

Good.

Check the rest:

```
temp[x, 0]
```

$-0.000229884\,\mathrm{Sin}[x] - 0.534292\,\mathrm{Sin}[2x] + 1.00601\,\mathrm{Sin}[3x] - 0.0667865\,\mathrm{Sin}[4x] + 0.234682\,\mathrm{Sin}[5x] - 0.0197886\,\mathrm{Sin}[6x] + 0.0872708\,\mathrm{Sin}[7x] - 0.00834831\,\mathrm{Sin}[8x]$

```
Sclosest[x]
```

$-0.000229884\,\mathrm{Sin}[x] - 0.534292\,\mathrm{Sin}[2x] + 1.00601\,\mathrm{Sin}[3x] - 0.0667865\,\mathrm{Sin}[4x] + 0.234682\,\mathrm{Sin}[5x] - 0.0197886\,\mathrm{Sin}[6x] + 0.0872708\,\mathrm{Sin}[7x] - 0.00834831\,\mathrm{Sin}[8x]$

```
temp[0, t]
```

0

```
temp[π, t]
```

0

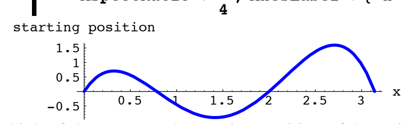
Done!

T.3.b.i) Fourier Sine approximation and the wave equation

The ends of a guitar string are anchored at 0 and L on the x-axis and the string is pulled to an initial position and then allowed to vibrate on its own starting with initial velocity 0.

At the start of this particular experiment, the position of the guitar string at position x (for $0 \le x \le L = \mathrm{Pi}$) is given by the following function starterposition[x]:

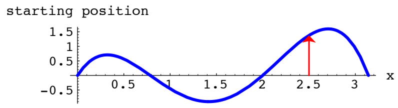
```
L = π;
Clear[starterposition, x];
starterposition[x_] = x (x - π/4) (x - 2) (π - x);

starterplot = Plot[starterposition[x], {x, 0, L},
   PlotStyle → {{Thickness[0.01], Blue}}, PlotRange → All,
   AspectRatio → 1/4, AxesLabel → {"x", "starting position"}];
```



Think of the curve as the starting position of the guitar string.
To fully understand this plot, look at this:

```
xvalue = 2.5;
Clear[pointer];
pointer[x_] :=
```

```
Arrow[{0, starterposition[x]}, Tail → {x, 0}, VectorColor → Red];

Show[starterplot, pointer[xvalue]];
```



The tip of the pointer tells you the starting position (at time t = 0) at the tail of the pointer.
Your problem here is to use Fourier Sine approximation on [0,π] of starterposition[x] come up come up with a function position[x,t] that estimates the position of the guitar string at position x on the x-axis at time t after the experiment begins.
Do it.

Answer:

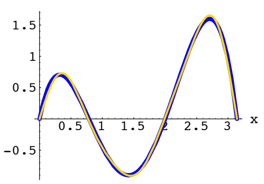
First do a Sine approximation of f[x] = starterposition[x] on [0,Pi]:

```
khigh = 8;
Clear[s, g, f, k, x];
f[x_] = starterposition[x];
s_k_[x_] := Sin[k x];
a = 0;
b = Pi;

Clear[fouriercoeff, Sclosest, k];
fouriercoeff[k_] :=
  fouriercoeff[k] = NIntegrate[f[t] s_k[t], {t, a, b}] / NIntegrate[s_k[t] s_k[t], {t, a, b}];

Sclosest[x_] = Sum[fouriercoeff[k] s_k[x], {k, 1, khigh}]
```

```
-0.000229884 Sin[x] - 0.534292 Sin[2 x] +
 1.00601 Sin[3 x] - 0.0667865 Sin[4 x] + 0.234682 Sin[5 x] -
 0.0197886 Sin[6 x] + 0.0872708 Sin[7 x] - 0.00834831 Sin[8 x]
```

Check out the quality of the fit:

```
fitplot = Plot[{starterposition[x], Sclosest[x]}, {x, a, b},
   PlotStyle → {{Thickness[0.02], Blue}, {Thickness[0.01], Gold}},
   AxesLabel → {"x", ""}];
```



Beautiful fit - go with this one.

To get position[x,t], you look at:

```
Sclosest[x]
```

```
-0.000229884 Sin[x] - 0.534292 Sin[2 x] +
 1.00601 Sin[3 x] - 0.0667865 Sin[4 x] + 0.234682 Sin[5 x] -
 0.0197886 Sin[6 x] + 0.0872708 Sin[7 x] - 0.00834831 Sin[8 x]
```

Pick off the coefficients of the Sin[k x] terms:

```
Clear[A, k];
A[k_] := Coefficient[Sclosest[x], Sin[k x]];

Table[A[k], {k, 1, khigh}]
```

```
{-0.000229884, -0.534292, 1.00601, -0.0667865,
 0.234682, -0.0197886, 0.0872708, -0.00834831}
```

Now you're done because you can write down position[x,t].

It's just:

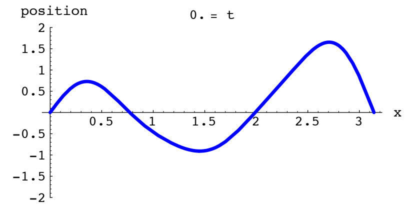
```
Clear[position, x, t];
position[x_, t_] = Sum[A[k] Cos[k t] Sin[k x], {k, 1, khigh}]
```

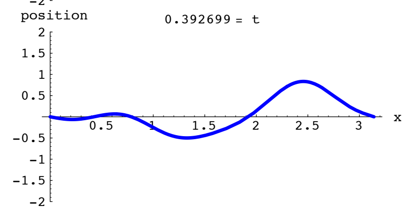
```
-0.000229884 Cos[t] Sin[x] - 0.534292 Cos[2 t] Sin[2 x] +
 1.00601 Cos[3 t] Sin[3 x] - 0.0667865 Cos[4 t] Sin[4 x] +
 0.234682 Cos[5 t] Sin[5 x] - 0.0197886 Cos[6 t] Sin[6 x] +
 0.0872708 Cos[7 t] Sin[7 x] - 0.00834831 Cos[8 t] Sin[8 x]
```

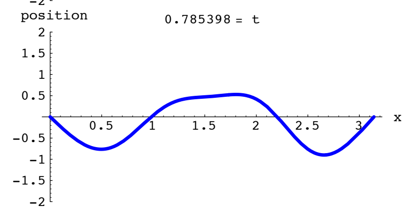
See the string vibrate::

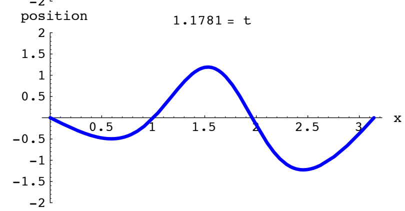
```
Clear[tempplotter];
tempplotter[t_] := Plot[position[x, t],
   {x, 0, π}, PlotStyle → {{Thickness[0.01], Blue}},
   PlotRange → {-2, 2}, AxesLabel → {"x", " position"},
   PlotLabel → N[t] "= t", AspectRatio → 1/2];
```

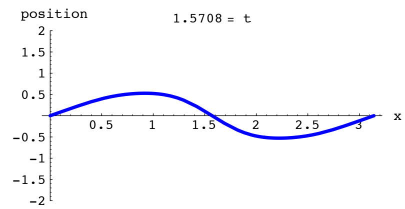
```
timejump = π/8;

Table[tempplotter[t], {t, 0, 2 π, timejump}]
```

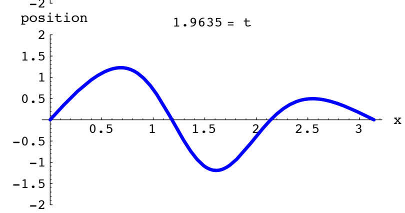


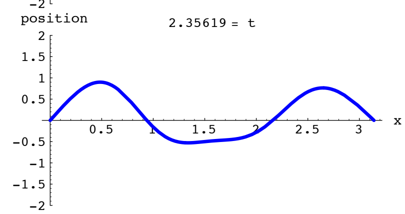


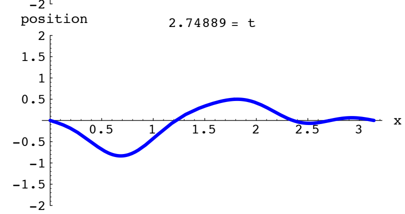


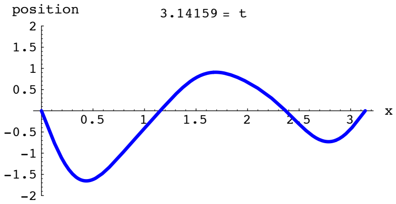


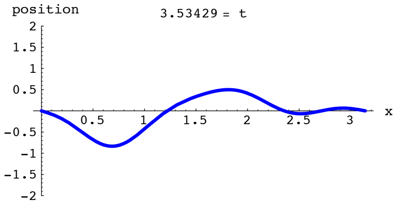


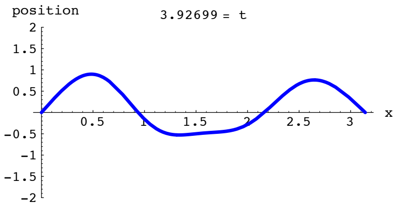


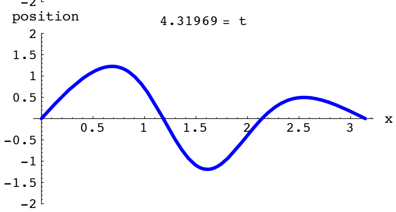


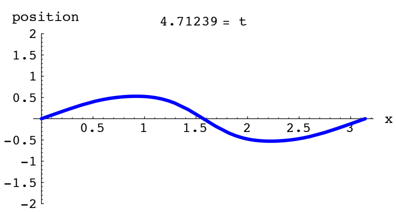


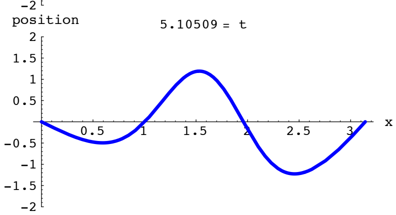


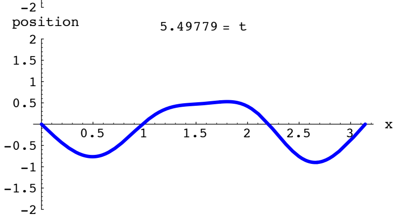


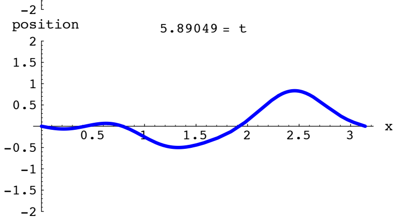


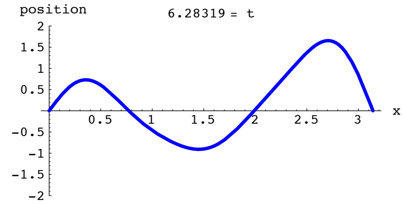













```
{- Graphics -, - Graphics -, - Graphics -, - Graphics -,
 - Graphics -, - Graphics -, - Graphics -, - Graphics -,
 - Graphics -, - Graphics -, - Graphics -, - Graphics -,
 - Graphics -, - Graphics -, - Graphics -, - Graphics -, - Graphics -}
```

Grab all the plot and animate.

It takes 2π time units for the string to return to its original position..

T.3.b.ii) A couple of calculations

Keep everything the same as above and look at these calculations:

```
∂t position[x, t] /. t -> 3
```

```
0.0000324413 Sin[x] - 0.298579 Sin[2 x] -
 1.24378 Sin[3 x] - 0.143343 Sin[4 x] - 0.763053 Sin[5 x] -
 0.0891658 Sin[6 x] - 0.511109 Sin[7 x] - 0.0604804 Sin[8 x]
```

```
∂t position[x, t] /. x -> 2
```

```
0.000209033 Sin[t] - 0.808707 Sin[2 t] +
 0.843283 Sin[3 t] + 0.264303 Sin[4 t] + 0.638359 Sin[5 t] -
 0.0637081 Sin[6 t] - 0.605158 Sin[7 t] - 0.019228 Sin[8 t]
```

What do these calculations measure?

Answer

Take them one at a time:

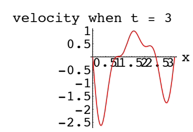
The velocity of the string at x at time t = 3 is:

```
∂t position[x, t] /. t -> 3
```

```
0.0000324413 Sin[x] - 0.298579 Sin[2 x] -
 1.24378 Sin[3 x] - 0.143343 Sin[4 x] - 0.763053 Sin[5 x] -
 0.0891658 Sin[6 x] - 0.511109 Sin[7 x] - 0.0604804 Sin[8 x]
```

See it :

```
Plot[∂t position[x, t] /. t -> 3,
  {x, 0, Pi}, PlotStyle -> {{Red, Thickness[0.01]}},
  AxesLabel -> {"x", "velocity when t = 3"}];
```



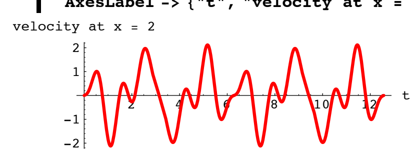
The velocity of the string at x = 2 at time t is:

```
∂t position[x, t] /. x -> 2
```

```
0.000209033 Sin[t] - 0.808707 Sin[2 t] +
 0.843283 Sin[3 t] + 0.264303 Sin[4 t] + 0.638359 Sin[5 t] -
 0.0637081 Sin[6 t] - 0.605158 Sin[7 t] - 0.019228 Sin[8 t]
```

See it :

```
Plot[Evaluate[∂t position[x, t] /. x -> 2],
 {t, 0, 4 Pi}, PlotStyle -> {{Red, Thickness[0.01]}},
 AxesLabel -> {"t", "velocity at x = 2"}];
```



T.3.b.iii) The wave equation explains why you want the Cosines

You get position[x,t] by taking a rigged sine fit of starterposition[x]:

```
Sclosest[x]
```

```
-0.000229884 Sin[x] - 0.534292 Sin[2 x] +
 1.00601 Sin[3 x] - 0.0667865 Sin[4 x] + 0.234682 Sin[5 x] -
 0.0197886 Sin[6 x] + 0.0872708 Sin[7 x] - 0.00834831 Sin[8 x]
```

And you get position[x,t] by inserting Cosines into each term of the rigged sine fit:

```
position[x, t]
```

```
-0.000229884 Cos[t] Sin[x] - 0.534292 Cos[2 t] Sin[2 x] +
 1.00601 Cos[3 t] Sin[3 x] - 0.0667865 Cos[4 t] Sin[4 x] +
 0.234682 Cos[5 t] Sin[5 x] - 0.0197886 Cos[6 t] Sin[6 x] +
 0.0872708 Cos[7 t] Sin[7 x] - 0.00834831 Cos[8 t] Sin[8 x]
```

Why do you insert those Cosines?

Answer

Engineering studies have shown that after the appropriate unit adjustments are made, the function position[x,t] satisfies the wave equation

$$\partial_{\{x,2\}}\text{position}[x, t] = \partial_{\{t,2\}}\text{position}[x, t].$$

This is the same as D[position[x,t], {x,2}] = D[position[x,t], {t,2}].

with

-> position[x, 0] = Sclosest[x]

-> position[0, t] = 0 andposition[π, t] = 0 for all t's

Reason: The ends of the guitar string are attached at the ends.

and with

-> ∂_t position[x, t] = 0 for t = 0 .

Reason: you let the guitar string vibrate giving it initial velocity 0.

Lots of folks call this a partial differential equation.

Inserting the Cosines made position[x,t] do all this:

Check $\partial_{\{x,2\}}$ position[x, t] = $\partial_{\{t,2\}}$ position[x, t]:

```
∂{x,2} position[x, t]
```

0.000229884 Cos[t] Sin[x] + 2.13717 Cos[2 t] Sin[2 x] - 9.05407 Cos[3 t] Sin[3 x] + 1.06858 Cos[4 t] Sin[4 x] - 5.86704 Cos[5 t] Sin[5 x] + 0.712389 Cos[6 t] Sin[6 x] - 4.27627 Cos[7 t] Sin[7 x] + 0.534292 Cos[8 t] Sin[8 x]

```
∂{t,2} position[x, t]
```

0.000229884 Cos[t] Sin[x] + 2.13717 Cos[2 t] Sin[2 x] - 9.05407 Cos[3 t] Sin[3 x] + 1.06858 Cos[4 t] Sin[4 x] - 5.86704 Cos[5 t] Sin[5 x] + 0.712389 Cos[6 t] Sin[6 x] - 4.27627 Cos[7 t] Sin[7 x] + 0.534292 Cos[8 t] Sin[8 x]

This checks.

Check: position[x, 0] = Sclosest[x]

```
position[x, 0]
```

-0.000229884 Sin[x] - 0.534292 Sin[2 x] + 1.00601 Sin[3 x] - 0.0667865 Sin[4 x] + 0.234682 Sin[5 x] - 0.0197886 Sin[6 x] + 0.0872708 Sin[7 x] - 0.00834831 Sin[8 x]

```
Sclosest[x]
```

-0.000229884 Sin[x] - 0.534292 Sin[2 x] + 1.00601 Sin[3 x] - 0.0667865 Sin[4 x] + 0.234682 Sin[5 x] - 0.0197886 Sin[6 x] + 0.0872708 Sin[7 x] - 0.00834831 Sin[8 x]

This checks.

Check: position[0, t] = 0 and position[Pi, t] = 0 for all t's

```
position[0, t]
```

0

```
position[Pi, t]
```

0

This checks.

Check: ∂_t position[x, t] = 0 for t = 0 .

```
∂t position[x, t] /. t -> 0
```

0

T.4) Using Fourier methods to try to bring the Dirac Delta function to life

T.5.a)

For a lot of folks the Dirac Delta function DiracDelta[t] is a total mystery.
If t ≠ 0, then DiracDelta[t] = 0:

```
t = Random[Real, {-4, 4}];
DiracDelta[t]
```

0

If t = 0, then DiracDelta[t] is not defined as a real number:

```
t = 0;
DiracDelta[t]
```

DiracDelta[0]

Yet ,

$$\int_a^b \text{DiracDelta}[t]\, dt = 1$$

provided a < 0 < b.

```
a = Random[Real, {-2, -0.0001}];
b = Random[Real, {0.0001, 2}];
Clear[t];
∫_a^b DiracDelta[t] dt
```

1

But

$$\int_a^b \text{DiracDelta}[t]\, dt = 0$$

for 0 < a < b:

```
a = Random[Real, {0.00001, 3}];
b = Random[Real, {a, 3}];
∫_a^b DiracDelta[t] dt
```

0

And

$$\int_a^b \text{DiracDelta}[t]\, dt = 0$$

for a < b < 0:

```
a = Random[Real, {-2, -0.0001}];
b = Random[Real, {a, -0.0001}];
∫_a^b DiracDelta[t] dt
```

0

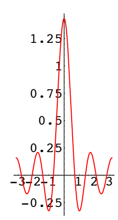
Go with the orthogonal system of Sines and Cosines on [-π,π]; and see what you get when you try to approximate DiracDelta[t] with Sine and Cosine waves::

```
Clear[f, t];
f[t_] = DiracDelta[t];
a = -π;
b = π;
khigh = 9;
Clear[s, k, t];

s_k_[t_] := If[EvenQ[k], Sin[k t/2], Cos[1/2 (k - 1) t]];
Clear[fouriercoeff, SinCosapprox, k];
fouriercoeff[k_] := fouriercoeff[k] = (∫_a^b f[t] s_k[t] dt)/(∫_a^b s_k[t] s_k[t] dt);

SinCosapprox[t_] = Sum[fouriercoeff[k] s_k[t], {k, 1, khigh}]
```

$$\frac{1}{2\pi} + \frac{\text{Cos}[t]}{\pi} + \frac{\text{Cos}[2 t]}{\pi} + \frac{\text{Cos}[3 t]}{\pi} + \frac{\text{Cos}[4 t]}{\pi}$$

See it:

```
Plot[SinCosapprox[t], {t, a, b}, AspectRatio -> 2,
  PlotRange -> All, PlotStyle -> {{Thickness[0.01], Red}}];
```



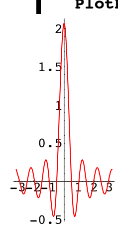
Kick up khigh:

```
khigh = 14;
Clear[SinCosapprox];

SinCosapprox[t_] = Sum[fouriercoeff[k] s_k[t], {k, 1, khigh}]
```

$$\frac{1}{2\pi} + \frac{\text{Cos}[t]}{\pi} + \frac{\text{Cos}[2 t]}{\pi} + \frac{\text{Cos}[3 t]}{\pi} + \frac{\text{Cos}[4 t]}{\pi} + \frac{\text{Cos}[5 t]}{\pi} + \frac{\text{Cos}[6 t]}{\pi}$$

See it:

```
Plot[SinCosapprox[t], {t, a, b}, AspectRatio -> 2,
  PlotRange -> All, PlotStyle -> {{Thickness[0.01], Red}}];
```



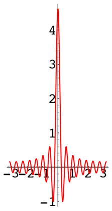
Kick up khigh even more:

```
khigh = 30;
Clear[SinCosapprox];

SinCosapprox[t_] = Sum[fouriercoeff[k] s_k[t], {k, 1, khigh}]
```

$$\frac{1}{2\pi} + \frac{\text{Cos}[t]}{\pi} + \frac{\text{Cos}[2 t]}{\pi} + \frac{\text{Cos}[3 t]}{\pi} + \frac{\text{Cos}[4 t]}{\pi} + \frac{\text{Cos}[5 t]}{\pi} + \frac{\text{Cos}[6 t]}{\pi} + \frac{\text{Cos}[7 t]}{\pi} + \frac{\text{Cos}[8 t]}{\pi} + \frac{\text{Cos}[9 t]}{\pi} + \frac{\text{Cos}[10 t]}{\pi} + \frac{\text{Cos}[11 t]}{\pi} + \frac{\text{Cos}[12 t]}{\pi} + \frac{\text{Cos}[13 t]}{\pi} + \frac{\text{Cos}[14 t]}{\pi}$$

See it:

```
Plot[SinCosapprox[t], {t, a, b}, AspectRatio -> 2,
  PlotRange -> All, PlotStyle -> {{Thickness[0.01], Red}}];
```



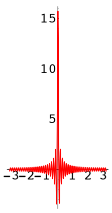
Kick up khigh even more:

```
khigh = 100;
Clear[SinCosapprox];
```

$$\texttt{SinCosapprox[t\_]} = \sum_{k=1}^{khigh} \texttt{fouriercoeff[k]}\, s_k\texttt{[t]};$$

See it:

```
Plot[SinCosapprox[t], {t, a, b}, AspectRatio -> 2,
  PlotRange -> All, PlotStyle -> {{Thickness[0.01], Red}}];
```



That's looking the way you might expect a decent approximation of DiracDelta[t] to look.
And these approximations are settling into a pattern.

$$\text{DiracApprox[khigh, t]} = \frac{1}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \text{Cos[k t]}:$$

When you integrate

$$\text{DiracApprox[khigh, t]} = \frac{1}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \text{Cos[k t]}$$

from - h to h, you get

$$\int_{-h}^{h} \text{DiracApprox[khigh, t]}\, dt = \frac{2h}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \frac{\text{Sin[k h]} - \text{Sin[k (-h)]}}{k} = \frac{1}{\pi}\left(h + \sum_{k=1}^{khigh} \frac{2\,\text{Sin[k h]}}{k}\right)$$

See how

$$\int_{-h}^{h} \text{DiracApprox[khigh, t]}\, dt$$

turns out for h = 0.01 and various choices of khigh.

```
h = 0.01;
khigh = 50;
```

$$\texttt{N}\left[\frac{1}{\pi}\left(h + \sum_{k=1}^{khigh} \frac{2\,\texttt{Sin[k h]}}{k}\right), 30\right]$$

```
0.316973
khigh = 100;
```

$$\frac{1}{\pi}\left(h + \sum_{k=1}^{khigh} \frac{2\,\texttt{Sin[k h]}}{k}\right)$$

```
0.604972
khigh = 500;
```

$$\frac{1}{\pi}\left(h + \sum_{k=1}^{khigh} \frac{2\,\texttt{Sin[k h]}}{k}\right)$$

```
0.986107
khigh = 5000;
```

$$\texttt{N}\left[\frac{1}{\pi}\left(h + \sum_{k=1}^{khigh} \frac{2\,\texttt{Sin[k h]}}{k}\right), 30\right]$$

```
0.987774
```

In fact when you run khigh all the way to infinity,you get:

$$\frac{1}{\pi}\left(h + \sum_{k=1}^{\texttt{Infinity}} \frac{2\,\texttt{Sin[k h]}}{k}\right)$$

```
1.
```

When you integrate $\frac{1}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \text{Cos[k t]}$ from a to b with $0 < a < b \le \pi$, you get

$$\int_{a}^{b} \text{DiracApprox[khigh, t]}\, dt = \frac{(b-a)}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \frac{\text{Sin[k b]} - \text{Sin[k a]}}{k}.$$

See how these turn out for random choices of a and b with $0 < a < b \le \pi$,:

```
a = Random[Real, {0, Pi}];
b = Random[Real, {a, Pi}];
khigh = 50;
```

$$\texttt{N}\left[\frac{(b-a)}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \frac{\texttt{Sin[k b]} - \texttt{Sin[k a]}}{k}, 30\right]$$

```
-0.00184577
khigh = 500;
```

$$\texttt{N}\left[\frac{(b-a)}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \frac{\texttt{Sin[k b]} - \texttt{Sin[k a]}}{k}, 30\right]$$

```
0.000129927
khigh = 1000;
```

$$\texttt{N}\left[\frac{(b-a)}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \frac{\texttt{Sin[k b]} - \texttt{Sin[k a]}}{k}, 30\right]$$

```
-0.000248836
khigh = Infinity;
```

$$\texttt{N}\left[\frac{(b-a)}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \frac{\texttt{Sin[k b]} - \texttt{Sin[k a]}}{k}, 30\right]$$

```
0
```

When you integrate $\frac{1}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \text{Cos[k t]}$ from a to b with $-\pi \le a < b < 0$ you get

$$\int_{a}^{b} \text{DiracApprox[khigh, t]}\, dt = \frac{(b-a)}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \frac{\text{Sin[k b]} - \text{Sin[k a]}}{k}.$$

See how these turn out for random choices of a and b with $-\pi \le a < b < 0$,:

```
a = Random[Real, {-Pi, 0}];
b = Random[Real, {a, 0}];
khigh = 50;
```

$$\texttt{N}\left[\frac{(b-a)}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \frac{\texttt{Sin[k b]} - \texttt{Sin[k a]}}{k}, 30\right]$$

```
0.00121803
khigh = 500;
```

$$\texttt{N}\left[\frac{(b-a)}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \frac{\texttt{Sin[k b]} - \texttt{Sin[k a]}}{k}, 30\right]$$

```
-0.000186486
khigh = 1000;
```

$$\texttt{N}\left[\frac{(b-a)}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \frac{\texttt{Sin[k b]} - \texttt{Sin[k a]}}{k}, 30\right]$$

```
0.000063264
khigh = Infinity;
```

$$\texttt{N}\left[\frac{(b-a)}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \frac{\texttt{Sin[k b]} - \texttt{Sin[k a]}}{k}, 30\right]$$

```
0
```

These experiments indicate that as khigh → Infinity

$$\int_{a}^{b} \text{DiracApprox[khigh, t]}\, dt \to \int_{a}^{b} \text{DiracDelta[t]}\, dt.$$

And if the functions

$$\text{DiracApprox[khigh,t]} = \frac{1}{2\pi} + \frac{1}{\pi}\sum_{k=1}^{khigh} \text{Cos[k t]}$$

converged to a normal function as khigh gets big , then DiracApprox[khigh,t] would converge to DiracDelta[t].
The trouble is that these DiracApprox[khigh,t] do not converge to a normal function as khigh gets big.

To get around this, the French mathematician Laurent Schwartz invented a new kind of convergence and a new kind of function so that it is now perfectly legitimate to say that for purposes of integration, DiracApprox[khigh,t] does converge to DiracDelta[t].