Solving the Heat and Wave Equations with the (Fast) Discrete Fourier Transform

Alkiviadis G. Akritas†, Jerry Uhl‡ and Panagiotis S. Vigklas†

†University of Thessaly
Department of Computer and Communication Engineering
37 Glavani & 28th October
GR-38221, Volos
Greece
and
‡University of Illinois at Urbana-Champaign
Department of Mathematics
273 Altgeld Hall (mc 382)
1409 W. Green
Urbana, IL 61801
USA
akritas@uth.gr, juhl@cm.math.uiuc.edu, pviglas@uth.gr

Motivated by the excellent work of Bill Davis and Jerry Uhl "Differential Equations & *Mathematica*" [2], we present in detail a little known application of the *f*ast *D*iscrete *F*ourier *T*ransform (DFT), also known as FFT. Namely, we first examine the use of FFT in approximating polynomials with sines and cosines (also known as *F*ast *F*ourier *F*it or FFF) and then derive the heat and wave equations. This presentation is ideally suited for educational purposes.

■ Introduction

We begin with a review of the basic definition needed.

Let R be a ring, $n \in \mathbb{Z}_{\geq 1}$, and $\omega \in R$ be a primitive nth root of unity; that is, $\omega^n = 1$ and $\omega^{n/t}$ - 1 is not a zero divisor (or, $\omega^{n/t} - 1 \neq 0$) for any prime divisor t of n. We represent the polynomial $f = \sum_{i=0}^{n-1} f_i\, x^i \in R[x]$, of degree $< n$ by the coefficient list, in reverse order, $\{ f_0, \ldots, f_{n-1}\} \in R^n$.

Definition 1 (DFT): The R-linear map DFT_ω: $R^n \to R^n$, which evaluates a polynomial at the powers of ω, i.e. DFT_ω: $\{ f_0, \ldots, f_{n-1}\} \to \frac{1}{\sqrt{n}} \{f(1), f(\omega), \ldots, f(\omega^{n-1})\}$, is called the *Discrete Fourier Transform* (DFT).

In other words, the Discrete Fourier Transform is a special multipoint evaluation at the powers $1, \omega, \ldots, \omega^{n-1}$ of a primitive nth root of unity ω. The fast implementation of the DFT is known as the fast DFT, or simply as FFT; it can be performed in time $O(n \log n)$. Details can be found in the literature [5]. Keeping it simple, we mention in passing that the **inverse** Discrete Fourier Transform is defined as the problem of interpolation at the powers of ω and is easily solved.

In *Mathematica* the map DFT$_\omega$ and its inverse are implemented — for the complex numbers — by the functions `Fourier[]`, and `InverseFourier[]`. The fast Fourier transform is implemented in `Fourier[]`. So, for example, the definition is verified by

```
f[x_] := x^3 - 7 x + 7;
{Fourier[CoefficientList[f[x], x]]} ==
 {n = 4; ω = ⅇ^((2 π ⅈ)/n); 1/Sqrt[n] {f[1], f[ω], f[ω^2], f[ω^3]}}

True
```

■ FFT is the basis of fast Fourier fit (FFF)

We next turn our attention to the problem of fast Fourier fit or FFF, i.e. the problem of approximating functions with sines and/or cosines.

Definition 2: Periodic functions $f:\mathbb{R}\to\mathbb{C}$, in one real variable and with values in the complex plane, can be approximated (or fitted) by complex trigonometric polynomials of the form

$$f(t) = \sum_{k=-n}^{n} c_k e^{k\,\omega\, i\, t} = \frac{\alpha_0}{2} + \sum_{k=1}^{n} (\alpha_k \cos(k\omega t) + \beta_k \sin(k\omega t))$$

where c_k are the Fourier fit coefficients satisfying

$$c_0 = \frac{\alpha_0}{2},\ c_k = \frac{(\alpha_k - i\,\beta_k)}{2},\ c_{-k} = \frac{(\alpha_k + i\,\beta_k)}{2}$$

and

$$\alpha_0 = 2c_0,\ \alpha_k = c_k + c_{-k},\ \beta_k = i(c_k - c_{-k})$$

for $k = 1, \ldots, n$, and $\omega = \frac{2\pi}{L}$ with $L > 0$ [4].

The problem of fast Fourier fit has attracted the attention of some of the best scientific minds of all times. Gauss came up with a fast Fourier fit algorithm in 1866. The modern version of the fast Fourier fit is due to John Tukey and his cohorts at IBM and Princeton [3].

We will be using the function FastFourierFit`[]` taken from Bill Davis and Jerry Uhl "Differential Equations & *Mathematica*" [2] to compute the approximating complex trigonometric polynomials mentioned in Definition 2 above.

```
jump[n_] := jump[n] = 1/(2 n);
Fvalues[F_, L_, n_] :=
 N[Table[F[L t], {t, 0, 1 - jump[n], jump[n]}]];

numtab[n_] := numtab[n] = Table[k, {k, 1, n}];

FourierFitters[L_, n_, t_] := Table[E^(2 π I k t/L),
    {k, -n + 1, n - 1}];
coeffs[n_, list_] :=
 Join[Reverse[Part[Fourier[list], numtab[n]]],
    Part[InverseFourier[list], Drop[numtab[n], 1]]] /
      N[Sqrt[Length[list]]]

FastFourierFit[F_, L_, n_, t_] :=
  Chop[FourierFitters[L, n, t].
    coeffs[n, Fvalues[F, L, n]]];
```

The code works as follows: the functions `jump[]`and `Fvalues[]` produce a list of $2n$ - 1 equally spaced data points off the plot of the function $f(t)$ between $t = 0$ and $t = L$. Then, the function `numtab[]` creates a list of integers from 1 to n, which is used by `coeffs[]` to concatenate two lists. The first of these lists is the Fourier transform (taken in *reversed* order) of the first n points, while the second list is the *inverse* Fourier transform (with the first element removed) of the same n points. To wit, the list generated by `coeffs[]` has a total of $2n$ - 1 points.

Finally, the function FastFourierFit`[]` takes the dot product of the list $\{e^{(-n+1)\,2\,\pi\,i\,t/L}$, …, 1, …, $e^{(n-1)\,2\,\pi\,i\,t/L}\}$ generated by `FourierFitters[]` and the list concatenated by `coeffs[]`. (All numbers in the list with magnitude less than 10^{-10} are rounded to 0.)

`FastFourierFit[]` takes four arguments; the first one is the periodic function or in general the list of data points which we want to fit; the second argument is the period L of the function; the third argument is the number n for the equally spaced $2n$ - 1 data points and the last argument is the variable we want to use. Note that `FastFourierFit[]` uses the built-in functions `Fourier[]` and `InverseFourier[]`, with computational cost $n \log n$.

Example 2: To see how the function FastFourierFit`[]` is used, consider the periodic function $f(x) = \cos(2\pi x) \sin(1 - \cos(3\pi x))$ with period $L = 2$. A plot is given in Figure 1.

```
f[x_] := Cos[2 π x] Sin[1 - Cos[3 π x]];
L = 2;
cycles = 2;
Plot[f[x], {x, 0, cycles L},
  AxesLabel → {"x", "f(x)"},
  PlotStyle → {{Thickness[0.007], RGBColor[0, 0, 1]}},
  PlotLabel → "cycles" cycles,
  Epilog → {{RGBColor[1, 0, 0],
     Thickness[0.007], Line[{{0, 0}, {L, 0}}]},
    {Text["One Period", {L/2, 0.1}]}}];
```

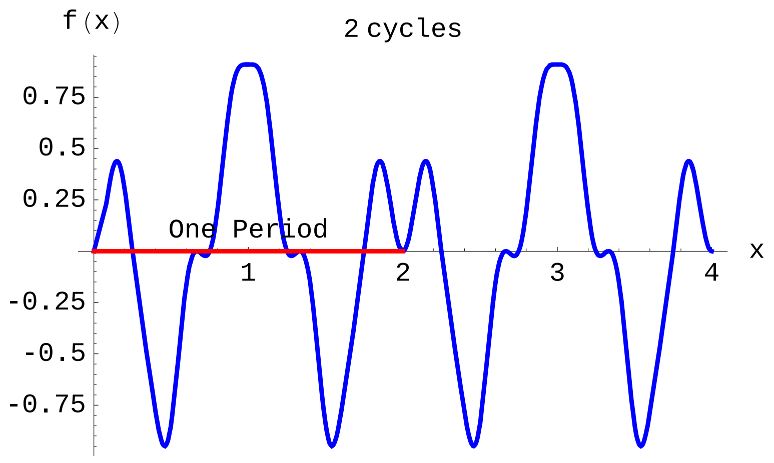

Figure 1. Approximating *f*(*x*) with *n* = 4 we obtain

```
L = 2; n = 4;
fApproximation[t_] = FastFourierFit[f, L, n, t]
```

$$-0.0967056 - 0.113662\, e^{-i \pi t} - 0.113662\, e^{i \pi t} + 0.32403\, e^{-2 i \pi t} + 0.32403\, e^{2 i \pi t} - 0.113662\, e^{-3 i \pi t} - 0.113662\, e^{3 i \pi t}$$

or its real (non-complex) version

```
fApproximationReal[t_] =
 Chop[ComplexExpand[fApproximation[t]]]
```

```
-0.0967056 - 0.227324 Cos[π t] +
 0.64806 Cos[2 π t] - 0.227324 Cos[3 π t]
```

Please note that the coefficients of *fApproximation*(*t*) and *fApproximationReal*(*t*) satisfy the relations mentioned in Definition 2. Moreover, *f*(*x*) has pure cosine fit. This was expected because the function *f*(*x*) = cos(2*πx*) sin(1 - cos(3*πx*)) is *even*; that is, for the function *evenf*(*x*), defined on the extended interval $0 \le x \le 2L$, we have *evenf*(*x*) = *f*(*x*), $0 \le x \le L$, and *evenf*(*x*) = *f*(2*L* - *x*), $L < x \le 2L$. See also its plot in Figure 1. Later on we will meet odd functions as well; those have pure sine fits.

The functions *f*(*x*) and *fApproximationReal*(*t*) are plotted together in Figure 2. As we see, what FastFourierFit[ ] does is to pick 2*n* - 1 equally spaced data points off the plot of *f*(*x*) between *x* = 0 and *x* = *L*; it then tries to fit these points with a combination of complex exponentials.

```
fplot = Plot[f[x], {x, 0, L},
   PlotStyle → {Thickness[0.008], RGBColor[0, 0, 1]},
   AspectRatio → 1/GoldenRatio,
   DisplayFunction :→ Identity];
fapproxPlot = Plot[fApproximationReal[t],
   {t, 0, L}, PlotStyle → {{Thickness[0.008],
      RGBColor[1, 0, 0], Dashing[{0.03, 0.03}]}},
   AspectRatio → 1/GoldenRatio,
   DisplayFunction :→ Identity];
fdata = Table[N[{x, f[x]}], {x, 0, L - L/(2 n - 1), L/(2 n - 1)}];
fdataplot =
  ListPlot[fdata, PlotStyle → PointSize[0.02],
   DisplayFunction → Identity];
Show[fplot, fapproxPlot, fdataplot,
  DisplayFunction → $DisplayFunction];
```

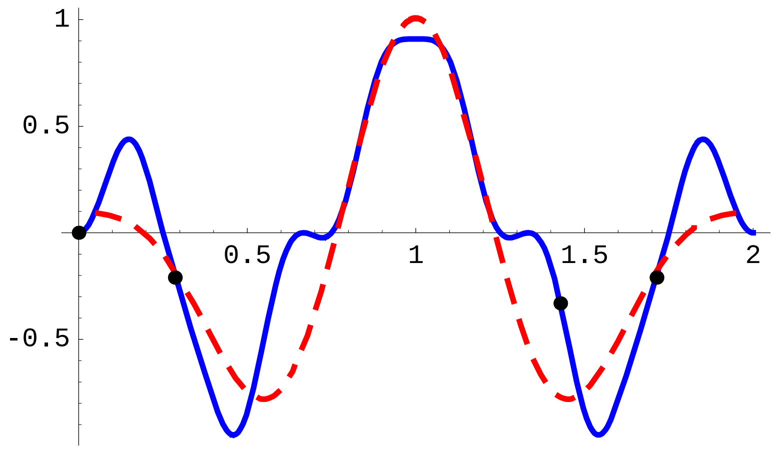

Figure 2. The dashed red plot is that of the approximating function.

As we mentioned before, the coefficients c_k of the approximating polynomial in Definition 2 are computed using the fast Fourier transform—incorporated in the function FastFourierFit[ ]. Another way of computing those coefficients is using the integrals

$$c_k = \frac{1}{L}\int_0^L f(t)\, e^{-\frac{i k (2\pi) t}{L}}\, dt.$$

This results in the integral Fourier fit.

This formula for the coefficients is obtained if we assume that for a fixed n, the function $f(t)$ is being approximated by the function

$$complexApproximation(t) = \sum_{k=-n}^{n} c_k e^{\frac{k(2\pi) i t}{L}},$$

where $L > 0$, and we set

$$f(t) = complexApproximation(t).$$

Then, we will definitely have

$$\int_0^L complexApproximation(t)\, e^{-\frac{j(2\pi)i\,t}{L}}\, dt = \int_0^L f(t)\, e^{-\frac{j(2\pi)i\,t}{L}}\, dt.$$

But

$$\int_0^L complexApproximation(t)\, e^{-\frac{j(2\pi)i\,t}{L}}\, dt = Lc_j$$

and, hence, the formula for the coefficients.

The two approximations resulting from the fast Fourier fit and the integral Fourier fit are pretty close, and almost identical for large values of *n*.

The disadvantage of the integral Fourier fit is that the integrals that need to be computed sometimes are very hard and impractical even for numerical integration. Nonetheless, the method is useful for hand computations, whereas doing fast Fourier fit by hand is completely out of the question.

The advantage of the integral Fourier fit is that, in theoretical situations, it provides a specific formula to work with. However, after the theory is developed and calculations begin, people switch to the fast Fourier fit.

Recapping, note that `FastFourierFit[]` is a “double” approximation. It first uses sines and cosines to approximate a continuous periodic function and then uses discrete Fourier transform to approximate integrals involving these trigonometric polynomials — in effect replacing numerical integration by sampling.

■ FFF in deriving the heat and wave equations

The fast Fourier is a very useful tool and we now demonstrate its power by deriving the heat and wave equations. Like most of the applications, this one again can be found in the excellent work of Bill Davis and Jerry Uhl [2]. However, some preliminaries are in order.

□ Preliminaries

As we have seen in the previous examples, using the fast Fourier fit we usually get a mixture of both sines and cosines. But sometimes we get pure sines as in

```
f[x_] = x (1 - x) (2 - x);
Chop[ComplexExpand[FastFourierFit[f, L = 2, n = 4, t]]]
```

```
0.386374 Sin[π t] +
 0.046875 Sin[2 π t] + 0.0113738 Sin[3 π t]
```

or pure cosines as in

```
f[x_] = x (1 - x)^2 (2 - x);
Chop[ComplexExpand[FastFourierFit[f, L = 2, n = 4, t]]]
```

```
0.123047 + 0.0662913 Cos[π t] -
 0.09375 Cos[2 π t] - 0.0662913 Cos[3 π t]
```

We will focus our attention on pure sine approximations, obtained for *odd* functions. (Even functions were mentioned earlier.)

To get a pure sine fit of $f(x)$ in the interval $0 \le x \le L$, two things need to be satisfied: (**a**) $f(0) = f(L) = 0$, and (**b**) for the function $oddf(x)$, defined on the extended interval $0 \le x \le 2L$, we have $oddf(x) = f(x)$, $0 \le x \le L$, and $oddf(x) = -f(2L - x)$, $L < x \le 2L$.

That is, the part of the plot to the right of the center line, has to be the negative mirror image of its plot to the left of the center line.

The requirement $f(0) = f(L) = 0$, stated above, comes from the fast fourier fit of $oddf(x)$ on $0 \le x \le 2L$. This means the function $oddf(x)$ is approximated with the functions $\sin(\frac{k\pi t}{L})$, which are all zeroed out at $t = 0$ and $t = L$. Therefore, if we want a good sines approximation of $f(x)$ on $0 \le x \le L$ we have to have $f(0) = f(L) = 0$.

If a given function $f(x)$ is approximated with a mixture of sines and cosines, there is a way to obtain a pure sine fit. Namely, if $f(x)$ is defined in the interval $0 \le x \le L$, and $f(0) = f(L) = 0$, we define a new function, $oddf(x)$ on the interval $0 \le x \le 2L$ as follows:

$$\begin{aligned} oddf(x) &= f(x), & 0 \le x \le L, \\ oddf(x) &= -f(2L - x), & L < x \le 2L. \end{aligned}$$

This new function, $odd(x)$, will clearly have a pure sine fit in the interval $0 \le x \le 2L$.

Example 4: Consider the function $f(x) = 6x\,(4 - x)\,e^{-x}$, which is approximated with a mixture of sines and cosines on $0 \le x \le L = 4$.

```
f[x_] = 6 x (4 - x) E^-x;
Chop[ComplexExpand[FastFourierFit[f, L = 4, n = 3, t]]]
```

$$2.94583 - 1.04677\,\text{Cos}\left[\frac{\pi t}{2}\right] - 1.32181\,\text{Cos}[\pi t] + 3.03426\,\text{Sin}\left[\frac{\pi t}{2}\right] + 0.643404\,\text{Sin}[\pi t]$$

Since it satisfies $f(0) = f(L) = 0$, we can define the new function $oddf(x)$, which has a pure sine fit in the interval $0 \le x \le 2L$.

```
oddf[x_] := f[x] /; 0 ≤ x ≤ L;
oddf[x_] := -f[2 L - x] /; L < x ≤ 2 L;
Chop[ComplexExpand[FastFourierFit[oddf, 2 L, 3, t]]]
```

$$4.10249\,\text{Sin}\left[\frac{\pi t}{4}\right] + 2.39086\,\text{Sin}\left[\frac{\pi t}{2}\right]$$

If the condition $f(0) = f(L) = 0$ is *not* satisfied then the function $f(x)$ needs a bit more work before it can have a pure sine approximation. Namely, to “fix” the problem we first run a line through the endpoints $f(0)$, $f(L)$, say $line(x) = f(0) + \frac{f(L) - f(0)}{L}\,x$, and then define the function $adjustedf(x) = f(x) - line(x)$, for which $adjustedf(0) = adjustedf(L) = 0$.

Example 5: Consider the function $3|0.25x - \langle 0.25x\rangle| + 1$, defined on the interval $0 \le x \le L = 3$, where $\langle 0.25x\rangle$ denotes the closest integer to $0.25x$. For this function we have $f(0) \ne f(L)$, with both being $\ne 0$. Its Fourier approximation includes both sines and cosines.

```
f[x_] = 3 Abs[0.25 x - Round[0.25 x]] + 1; {f[0], f[L = 3]}
{1, 1.75}
Chop[ComplexExpand[FastFourierFit[f, L = 3, n = 4, t]]]
```

$$1.82031 - 0.446978\,\text{Cos}\left[\frac{2\pi t}{3}\right] - 0.1875\,\text{Cos}\left[\frac{4\pi t}{3}\right] - 0.115522\,\text{Cos}[2\pi t] - 0.419519\,\text{Sin}\left[\frac{2\pi t}{3}\right] - 0.046875\,\text{Sin}\left[\frac{4\pi t}{3}\right] - 0.0445194\,\text{Sin}[2\pi t]$$

Since $f(0) \ne f(L)$ we clearly cannot, yet, apply our technique to get a pure sine approximation of this function. To “fix” the problem we first run $line(x)$ through the endpoints of the plot, and then we define the function $adjustedf(x) = f(x) - line(x)$, with $adjustedf(0) = adjustedf(L) = 0$. See Figure 5.

```
line[x_] = (f[L] - f[0]) x / L + f[0];
adjustedf[x_] = f[x] - line[x];
Plot[adjustedf[x], {x, 0, L},
  PlotStyle → {{Thickness[0.01], RGBColor[0, 0, 1]}},
  AxesLabel → {"x", "adjustedf"}];
```

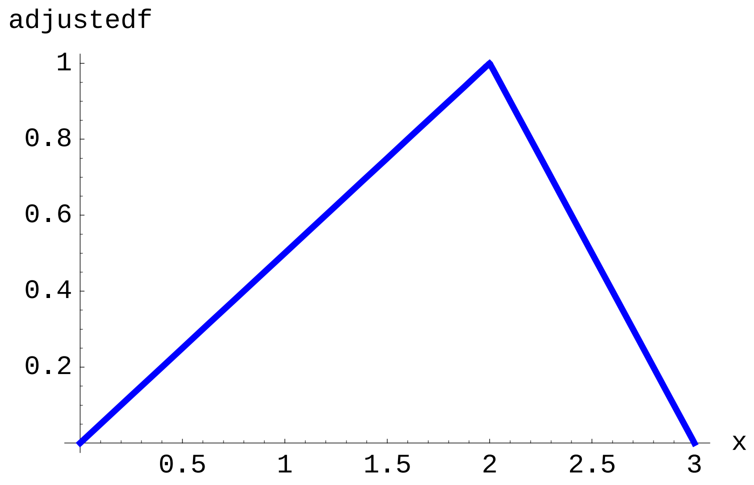

Figure 5. The adjusted function.

We can now apply our technique to *adjustedf*(*t*) and obtain a pure sine approximation of it.

```
oddAdjustedf[x_] := adjustedf[x] /; 0 ≤ x ≤ L;
oddAdjustedf[x_] := -adjustedf[2 L - x] /; L < x ≤ 2 L;

Chop[ComplexExpand[
  FastFourierFit[oddAdjustedf, 2 L, n = 4, t]]]
```

$$0.772748\,\mathrm{Sin}\left[\frac{\pi t}{3}\right] - 0.1875\,\mathrm{Sin}\left[\frac{2\pi t}{3}\right] + 0.0227476\,\mathrm{Sin}[\pi t]$$

□ The heat equation $\frac{\partial^2 temp(x,t)}{\partial x^2} = \frac{\partial\, temp(x,t)}{\partial t}$

We can now tackle the heat equation.

Problem: Start with a heated wire *L* units long with the temperature allowed to vary from position to position on the wire. The function *startertemp*(*x*) gives the temperature at the point *x*, for $0 \le x \le L$, of the wire at the start of the experiment. Because of the previous discussion we consider, without loss of generality, functions *startertemp*(*x*) for which *startertemp*(0) = *startertemp*(*L*) = 0. That way we can easily obtain a pure sine approximation.

Think of the wire as the interval $0 \le x \le L$. At the start of the experiment, we instantly cool the ends at $x = 0$ and $x = L$ and maintain these ends at temperature 0; we also take pains to guarantee that the rest of the wire is perfectly insulated.

The problem is to find a formula for *temp*(*x*, *t*), so that we can tell the temperature at a given point *x*, at time *t* after the start of the experiment.

We first give a theoretical solution, which we then compare with one obtained based on FFF.

Theoretical solution (based on Fourier Transforms): Given the problem statement, the previous discussion on odd functions and using the integral computation of the coefficients, we have:

$$\alpha_k = 0$$

$$\beta_k = \int_0^L f(x)\sin\left(\frac{k\pi x}{L}\right) dx$$

Specifically for the heat equation, β_k becomes:

$$\beta_k = \int_0^L temp(y,\ t)\sin\left(\frac{k\pi y}{L}\right) dy$$

Taking the first derivative of the expression above, we obtain

$$\frac{d\beta_k}{dt} = \beta_k^{(1)} = \int_0^L \frac{\partial\, temp(y,t)}{\partial t}\sin\left(\frac{k\pi y}{L}\right) dy$$

Applying the Sine Fourier Transform on both of the heat equation we have:

$$\int_0^L \frac{\partial\, temp(y,t)}{\partial t}\sin\left(\frac{k\pi y}{L}\right) dy = \int_0^L \frac{\partial^2\, temp(y,t)}{\partial x^2}\sin\left(\frac{k\pi y}{L}\right) dy\,.$$

We observe that the left hand side above is identical to the first derivative of β_k, i.e. to $\beta_k^{(1)}$, while the right hand side is equal to $\beta_k^{(2)}$. So we have:

$$\frac{d\beta_k}{dt} = -\frac{k^2\pi^2}{L^2}\beta_k,$$

with verification being left to the reader. Solving this last differential equation, we obtain

$$\beta_k = \lambda\, e^{-\frac{k^2\pi^2}{L^2}t},$$

where $\lambda = \beta_k(0)$.

Similarly, from the initial condition $temp(x, 0) = startertemp(x)$, we obtain

$$\beta_k(0) = \int_0^L temp(y,\ 0)\sin\left(\frac{k\pi y}{L}\right) dy = \int_0^L startertemp(y)\sin\left(\frac{k\pi y}{L}\right) dy$$

Hence, β_k becomes

$$\beta_k = e^{-\frac{k^2\pi^2}{L^2}t}\int_0^L startertemp(y)\sin\left(\frac{k\pi y}{L}\right) dy$$

It easily follows that the Inverse Fourier transform gives the solution of the heat equation:

$$temp(x,t) = \frac{2}{L}\sum_{k=1}^{\infty}\beta_k\sin\left(\frac{k\pi x}{L}\right) =$$
$$\frac{2}{L}\sum_{k=1}^{\infty} e^{-\frac{k^2\pi^2}{L^2}t}\sin\left(\frac{k\pi x}{L}\right)\int_0^L startertemp(y)\sin\left(\frac{k\pi y}{L}\right) dy.$$

Indeed, we have:

```
startertemp[x_] = 0.2 Sin[2 x]^2 (x - 3)^2;
L = 3; temp[x_, t_] :=
```

$$\frac{2}{L}\sum_{k=1}^{8} e^{-\frac{k^2\pi^2}{L^2}t}\,\mathrm{Sin}\left[\frac{k\pi x}{L}\right]\int_0^L startertemp[y]\,\mathrm{Sin}\left[\frac{k\pi y}{L}\right] dy;$$

```
temp[2.5, 1]
```

```
0.0590846
```

Solution based on Fast Fourier Fit: Using FFF we can easily obtain the formula for $temp(x, t)$
proceeding as follows:

a. Adjust the function *startertemp*(x) (that is, make *startertemp*(x) an *odd* function) and using the fast Fourier fit obtain a sines only approximation of it, for a given accuracy n. Given the original assumptions $startertemp(0) = startertemp(L) = 0$, this step is easily done.

b. Pick off the coefficients $A(k)$ of the $\sin(\frac{k\pi x}{L})$ terms of the sines only approximation of the adjusted $startertemp(x)$, and

c. Write down the formula for $temp(x, t)$ as:

$$temp(x, t) = \sum_{k=1}^{n} A(k)\, e^{-\left(\frac{k\pi}{L}\right)^2 t} \sin(\frac{k\pi x}{L})$$

Note: The terms $\sin(\frac{k\pi t}{L})$ in the sines only approximation of the adjusted $startertemp(x)$ are written as $\sin(\frac{k\pi x}{L})$, i.e. the usual t's are replaced by x's.

The central question that needs to be answered is why were the terms $e^{-\left(\frac{k\pi}{L}\right)^2 t}$ introduced. This is explained below.

Engineering studies have shown that, after the appropriate unit adjustments are made, the function $temp(x, t)$ satisfies the partial differential equation $\frac{\partial^2 temp(x,t)}{\partial x^2} = \frac{\partial temp(x,t)}{\partial t}$, known as the heat equation. That is, the second derivative of $temp(x, t)$ with respect to x equals the first derivative of $temp(x, t)$ with respect to t.

The boundary conditions for this differential equation are:

$temp(x, 0) = startertemp(x)$,
$temp(0, t) = 0$ and $temp(L, t) = 0, \forall t$.

The key boundary conditions are

$$temp(0, t) = 0 \text{ and } temp(L, t) = 0, \forall t,$$

which agree with the fact that $\sin(\frac{k\pi x}{L}) = 0$ for $x = 0$ and $x = L$ for all positive integers k. That means that for each fixed time t, and any value of n, we can approximate $temp(x, t)$ with a sines only fit of the form

$$approxtemp(x, t) = \sum_{k=1}^{n} u(t,\ k) \sin(\frac{k\pi x}{L}),$$

where the Fourier fit coefficients $u(t, k)$ have to be determined. Note that these coefficients depend on t as well as k because you expect a different adjusted sine fit at different times t.

The heat equation says $\frac{\partial^2 temp(x,t)}{\partial x^2} = \frac{\partial temp(x,t)}{\partial t}$. Instead of $temp(x, t)$ we use its approximation, $approxtemp(x, t)$, into the heat equation and see that

$$\sum_{k=1}^{n} u(t,\ k) \left(\frac{k\pi}{L}\right)^2 (-\sin(\frac{k\pi x}{L})) = \sum_{k=1}^{n} \frac{\partial u(t,k)}{\partial t} \sin(\frac{k\pi x}{L})$$

However, the above equation is valid only if the following exponential differential equation is true

$$\frac{\partial u(t,k)}{\partial t} = -\, u(t,\ k) \left(\frac{k\pi}{L}\right)^2.$$

But we already know that the solution to this last differential equation is

$$u(t, k) = A(k)\, e^{-\left(\frac{k\pi}{L}\right)^2 t}.$$

So, in order to compute the Fourier fit coefficients we have to determine the coefficients $A(k)$.

Substituting these $u(t, k)$ into $approxtemp(x, t)$ we obtain

$$approxtemp(x, t) = \sum_{k=1}^{n} A(k)\, e^{-\left(\frac{k\pi}{L}\right)^2 t} \sin(\frac{k\pi x}{L}),$$

which for $t = 0$ becomes

$$approxtemp(x, 0) = \sum_{k=1}^{n} A(k) \sin(\frac{k\pi x}{L}).$$

This is the approximation to the starting temperature, so we pick off the $A(k)$'s from the sine fit of the adjusted *startertemp*(x).

Example 6: At the start of this particular experiment, the temperature of the wire at position x (for $0 \le x \le L = 3$) is given by the following function *startertemp*(x) = $0.2 \sin^2(2\,x)(x - 3)$.

For the given function we have *startertemp*(0) = *startertemp*(L) = 0 and its plot is shown in Figure 6.

```
L = 3;
startertemp[x_] = 0.2 Sin[2 x]^2 (x - 3)^2;
{startertemp[0], startertemp[L]}
```

```
{0, 0}
```

```
Plot[startertemp[x], {x, 0, L},
  PlotStyle → {{Thickness[0.007], RGBColor[0, 0, 1]}},
  AxesLabel → {"x", ""}];
```

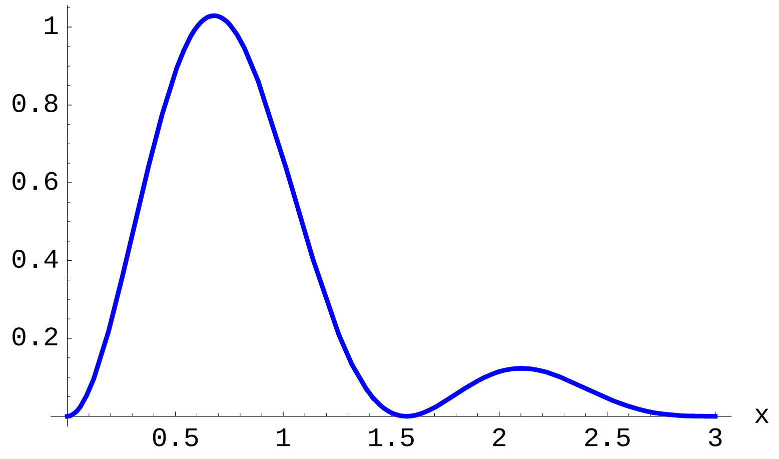

Figure 6. The initial temperature distribution.

We next adjust the function *startertemp*(x) and arbitrarily setting $n = 8$ we compute its sines *only* approximation with FFF. That is, we have

```
oddStartertemp[x_] := startertemp[x] /; 0 ≤ x ≤ L;
oddStartertemp[x_] :=
  -startertemp[2 L - x] /; L < x ≤ 2 L;

sinesOnlyfit[t_] = Chop[ComplexExpand[
   FastFourierFit[oddStartertemp, 2 L, n = 8, x]]]
```

$$0.379996\,\mathrm{Sin}\left[\frac{\pi x}{3}\right] + 0.395139\,\mathrm{Sin}\left[\frac{2\pi x}{3}\right] + 0.298906\,\mathrm{Sin}[\pi x] + 0.0691479\,\mathrm{Sin}\left[\frac{4\pi x}{3}\right] - 0.0908022\,\mathrm{Sin}\left[\frac{5\pi x}{3}\right] - 0.0548269\,\mathrm{Sin}[2\pi x] - 0.0186735\,\mathrm{Sin}\left[\frac{7\pi x}{3}\right]$$

We then pick off the coefficients of the $\sin(\frac{k\pi t}{L})$ terms:

```
A[k_] := Coefficient[sinesOnlyfit[t], Sin[(k π x / L)]];
Table[A[k], {k, 1, n}]
{0.379996, 0.395139, 0.298906, 0.0691479,
 -0.0908022, -0.0548269, -0.0186735, 0}
```

and the function *temp*(*x*, *t*) is:

$$\texttt{temp[x\_, t\_]} = \sum_{k=1}^{n} A[k]\, E^{-\left(\frac{k\pi}{L}\right)^2 t}\, \mathrm{Sin}\left[\frac{(k\pi)\,x}{L}\right]$$

$$0.379996\, e^{-\frac{\pi^2 t}{9}} \mathrm{Sin}\left[\frac{\pi x}{3}\right] + 0.395139\, e^{-\frac{4\pi^2 t}{9}} \mathrm{Sin}\left[\frac{2\pi x}{3}\right] + 0.298906\, e^{-\pi^2 t} \mathrm{Sin}[\pi x] + 0.0691479\, e^{-\frac{16\pi^2 t}{9}} \mathrm{Sin}\left[\frac{4\pi x}{3}\right] - 0.0908022\, e^{-\frac{25\pi^2 t}{9}} \mathrm{Sin}\left[\frac{5\pi x}{3}\right] - 0.0548269\, e^{-4\pi^2 t} \mathrm{Sin}[2\pi x] - 0.0186735\, e^{-\frac{49\pi^2 t}{9}} \mathrm{Sin}\left[\frac{7\pi x}{3}\right]$$

```
temp[2.5, 1]
0.0592159
```

The result is pretty close to the one obtained earlier using the theoretical solution.

□ The wave equation $\frac{\partial^2 position(x,t)}{\partial x^2} = \frac{\partial^2 position(x,t)}{\partial t^2}$

Moving on to the wave equation, we just state the problem and leave the details to the reader.

Problem: Start with a string fixed at 0 and *L* on the *x*-axis. The string is pulled to an initial position and then allowed to vibrate on its own, starting with initial velocity 0. The function *starterposition*(*x*) gives the position of the wire at the point *x*, for $0 \leq x \leq L$. As in the case of the heat equation, because of the previous discussion we consider, without loss of generality, functions *starterposition*(*x*) for which *starterposition*(0) = *starterposition*(*L*) = 0. That way we can easily adjust them for pure sine approximation.

The problem is to find a formula for *position*(*x*, *t*), so that we can tell the position of the string at a given point *x*, at time *t* after the start of the experiment. Since it is analogous to the heat equation we end our discussion here and refer the reader to [2].

■ Conclusions

Our goal has been to put together several difficult to access applications of FFT for use in the classroom. Hopefully, the programs provided here will be of help for experimentation and further development.

■ References

1. William E. Boyce and Richard C. DiPrima: *Elementary Differential Equations and Boundary Value Problems*, John Wiley & Sons, New York, NY, 1997.

2. Bill Davis and Jerry Uhl: *Differential Equations & Mathematica*, Math Everywhere, Inc., 1999 (Part of the “Calculus & Mathematica” series of books).

3. David Kahaner, Cleve Moler and Stephen Nash: *Numerical Methods and Software*, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

4. Walter Strampp, Victor Ganzha and Evgenij Vorozhtsov: *Höhere Mathematik mit Mathematica*, Vieweg Lehrbuch Computeralgebra, Braunschweig/Wiesbaden, 1997.

5. H. Joseph Weaver: *Applications of Discrete and Continuous Fourier Analysis*, John Wiley & Sons, New York, NY, 1983.