Nick Trefethen, 27 June 2014

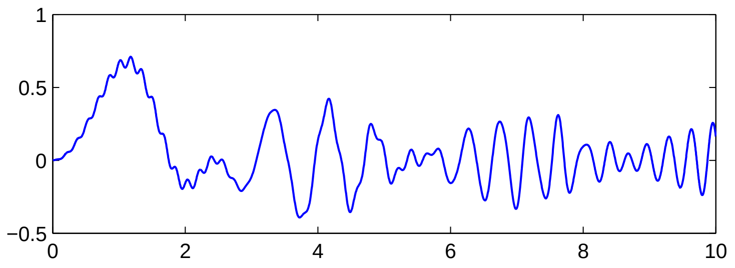
Two-page summary of the main features of Chebfun

Chebfun is a MATLAB-based software system for machine precision numerical computing with functions. It started from the observation that polynomial interpolants in Chebyshev points are powerful and flexible tools for representing functions, computing their roots and extrema and integrals, and other operations. The Chebfun idea is to represent a function $f(x)$ by a polynomial or piecewise polynomial to about 16 digits of precision. When operations are carried out like $g = \exp(f)$ or $h = gf$, the polynomial degrees are trimmed automatically, always aiming to maintain this accuracy. This is achieved by monitoring Chebyshev series coefficients and detecting plateaus caused by rounding errors at approximately the level of machine precision. This is an analogue for functions of the rounding idea in floating point arithmetic for numbers.

For example, to construct a representation of $f(x) = \sin(x^2)J_0(x) + \sin(x)J_1((20-x)^2)$ on the interval $[0, 10]$, one may type

```
f = chebfun('sin(x.^2).*besselj(0,x)+sin(x).*besselj(1,(20-x).^2)',[0 10]);
```

The resulting object, called a chebfun (with a lower-case c), "feels like" f on the chosen interval and can be manipulated with familiar MATLAB commands. For example, `plot(f)` gives this picture:



Here are the roots of f in the subinterval $[4, 5]$, computed via eigenvalues of colleague matrices.

```
roots(f{4,5})
ans =
   4.355408374348725
   4.703277404115000
```

Here are its maximum and definite integral, computed by Clenshaw-Curtis quadrature:

```
max(f)
ans = 0.709809172023478
sum(f)
ans = 0.551807278413019
```

The representation of f is by a polynomial of moderate degree,

```
length(f)
ans = 221
```

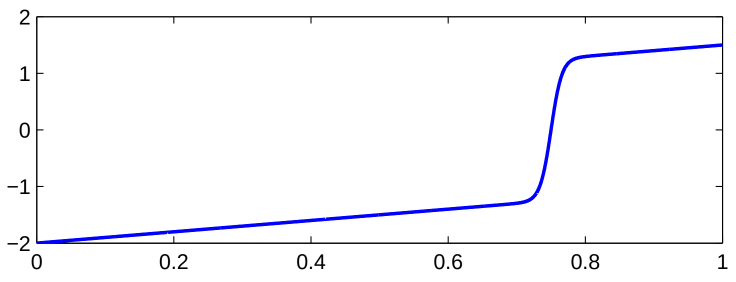
Operations that introduce discontinuities, like `g = exp(abs(f))`, lead to piecewise representations to which all the same operations can be applied in the same way:

```
mean(g)
ans = 1.206846790985504
```

In Maple or Mathematica, computations like these are possible in principle, but they require far more expertise on the part of the user to blend symbolic and numerical operations effectively. Chebfun is completely numerical and avoids the combinatorial explosion of computer time and memory that tends to afflict symbolic systems.

Initially Chebfun was a function calculator, but it has grown. Early on it developed continuous analogues of linear algebra operations such as QR factorization and SVD. More important for users have been its capabilities with ODE boundary-value and eigenvalue problems. In MATLAB, one types `x=A\b` to invoke an algorithm to solve a linear system of equations. Similarly, in Chebfun one types `u=L\f` to solve a boundary-value problem defined by a linear or nonlinear operator L with boundary conditions and a forcing function f. The algorithm used is rectangular adaptive spectral Chebyshev discretization as developed by Toby Driscoll and Nick Hale, with convergence determined in the usual Chebfun fashion by detection of plateaus in Chebyshev series expansions. For example, here is a solution of the nonlinear problem $0.01u'' + uu' - u = 0$ on $[0, 1]$ with Dirichlet boundary conditions:

```
N = chebop(@(u) 0.01*diff(u,2)+u.*diff(u)-u, [0,1]);
N.lbc = -2; N.rbc = 1.5;
u = N\0; plot(u)
```



The nonlinear aspect of the problem is handled by Newton iteration using a continuous analogue of automatic differentiation, and the result is accurate to 11 digits. Here we compute the maximum derivative of the solution in the interior layer:

```
max(diff(u))
ans = 80.484282397754797
```

We are unaware of any other tool for ODE boundary-value problems that is so powerful, accurate, and convenient. Similar commands solve eigenvalue problems and integral equations, and are readily explored with the graphical user interface CHEBGUI. Analogous capabilities in 2D are under development.

Chebfun is implemented in object-oriented MATLAB with about 40 classes. Users mainly interact with the chebfun, chebfun2, chebop, and chebgui classes for 1D functions, 2D functions, differential operators, and graphical user interface, respectively. The other classes support functionality including representation of functions in 1D with or without singularities, on bounded or unbounded intervals, periodic or nonperiodic; delta functions; representation of functions in 2D; automatic spectral discretization of differential and integral operators; problems with block structure; runtime preferences; and automatic differentiation to enable Newton iteration in solution of nonlinear differential and integral equations. The class diagram can be found at `github.com/chebfun/chebfun/wiki/Class-diagram`. Chebfun also has extensive collections of test programs (about 200 at present) and example programs (about 150).