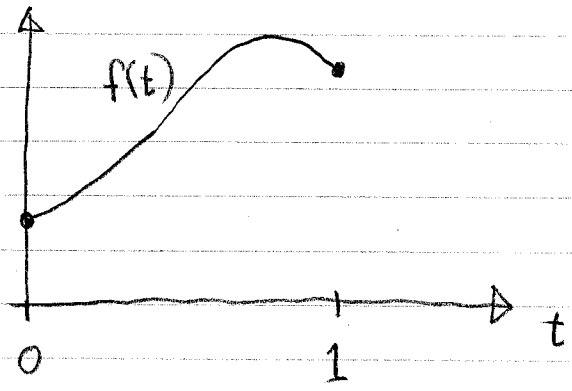
CUBIC INTERPOLATION

(NR §3.3 PROVIDES A SIMPLER BUT LESS GENERAL INTRODUCTION TO CUBIC SPLINES)

CONSIDER THE INTERVAL $0 \le t \le 1$. SUPPOSE $f(t)$ AND $f'(t)$ ARE KNOWN AT THE ENDPOINTS.



KNOWN

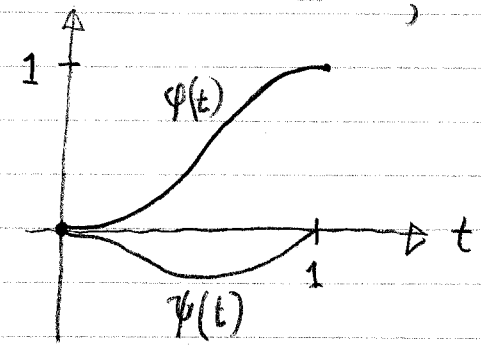
$f(0) = f_0$
$f(1) = f_1$
$f'(0) = d_0$
$f'(1) = d_1$

Q: WHAT KIND OF POLYNOMIAL DO I NEED TO FIT THE BOUNDARY CONDITIONS?

A: A CUBIC (4 COEFFICIENTS).

NOW CONSIDER THE "CUBIC HERMITE BASIS FUNCTIONS",

$$\varphi(t) = 3t^2 - 2t^3, \quad \psi(t) = t^3 - t^2.$$



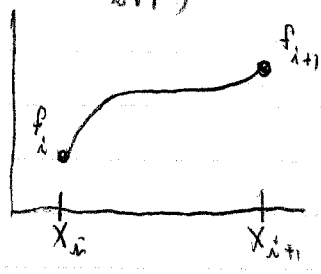
$f(t)$	$f(0)$	$f(1)$	$f'(0)$	$f'(1)$	$f''(0)$	$f''(1)$
$\varphi(t)$	0	1	0	0	6	-6
$\varphi(1-t)$	1	0	0	0	-6	6
$\psi(t)$	0	0	0	1	-2	4
$\psi(1-t)$	0	0	-1	0	4	-2

$$f(t) \approx p(t) \equiv f_0\,\varphi(1-t) + f_1\,\varphi(t) - d_0\,\psi(1-t) + d_1\,\psi(t)$$

NOW CONSIDER THE INTERVAL $X_i \le X \le X_{i+1}$)

GIVEN

$$f(x_i) = f_i$$
$$f'(x_i) = d_i$$
$$x_{i+1} - x_i = h_i$$



OUR INTERPOLATING FUNCTION OVER THE INTERVAL WILL NOW BE

$$p_i(x) = f_i \, \varphi((x_{i+1} - x)/h_i) + f_{i+1} \, \varphi((x - x_i)/h_i) - d_i \, h_i \, \psi((x_{i+1} - x)/h_i) + d_{i+1} \, h_i \, \psi((x - x_i)/h_i) \quad (1)$$

DO THE ARGUMENTS MAKE SENSE?

$$t = \frac{x - x_i}{h_i}$$

$$1 - t = \frac{x_{i+1} - x}{h_i} = 1 - \frac{x - x_i}{h_i}$$

DOES THE SIGN MAKE SENSE?

DO THESE MAKE SENSE?

[DISCUSS]

THIS MACHINERY CAN BE USED TO GENERATE ANY SORT OF PIECEWISE-CONTINUOUS CUBIC INTERPOLATION.

① f_i ARE SIMPLY THE DATA

② d_i ARE FREE PARAMETERS.

③ THE RESULTING INTERPOLANTS FORM A FUNCTION

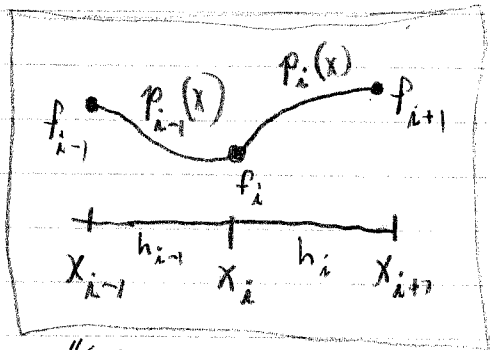
$$F(x) = p_i(x), \quad x_i \le x \le x_{i+1},$$

NOTE THAT F AND F' ARE CONTINUOUS $\forall x_i$.
(THIS HAPPENS AUTOMATICALLY!)

THERE ARE MANY WAYS TO CHOOSE THE d_i.

CUBIC SPLINE — CHOOSE d_i SUCH THAT F'' IS CONTINUOUS $\forall x_i$.
THE CUBIC SPLINE IS VERY WIDELY USED, SO IT'S WORTH UNDERSTANDING HOW IT WORKS.

WANT: $\underline{\underline{p_{i-1}''(x_i) = p_i''(x_i)}}$



NOW, $$p_i''(x_i) = \frac{f_i}{h_i^2}\underbrace{\varphi''(1)}_{-6} + \frac{f_{i+1}}{h_i^2}\underbrace{\varphi''(0)}_{6} - \frac{d_i}{h_i}\underbrace{\psi''(1)}_{4} + \frac{d_{i+1}}{h_i}\underbrace{\psi''(0)}_{-2}$$

AND

$$p_{i-1}''(x_i) = \frac{f_{i-1}}{h_{i-1}^2}\underbrace{\varphi''(0)}_{6} + \frac{f_i}{h_{i-1}^2}\underbrace{\varphi''(1)}_{-6} - \frac{d_{i-1}}{h_{i-1}}\underbrace{\psi''(0)}_{-2} + \frac{d_i}{h_{i-1}}\underbrace{\psi''(1)}_{4}$$

SETTING THESE EQUAL,

$$\frac{6}{h_{i-1}^2} f_{i-1} - \frac{6}{h_{i-1}^2} f_i + \frac{2}{h_{i-1}} d_{i-1} + \frac{4}{h_{i-1}} d_i$$

$$= -\frac{6}{h_i^2} f_i + \frac{6}{h_i^2} f_{i+1} - \frac{4}{h_i} d_i - \frac{2}{h_i} d_{i+1}$$

CONT. →

WE WISH TO ISOLATE THE d_i's (WHICH ARE UNKNOWN)...

$$\frac{2}{h_{i-1}} d_{i-1} + \left(\frac{4}{h_{i-1}} + \frac{4}{h_i}\right) d_i + \frac{2}{h_i} d_{i+1}$$

$$= \frac{-6}{h_{i-1}^2} f_{i-1} + \left(\frac{6}{h_{i-1}^2} - \frac{6}{h_i^2}\right) f_i + \frac{6}{h_i^2} f_{i+1} \quad (2)$$

SUPPOSE WE HAVE N DATA POINTS $(i = 1, 2, \ldots N)$.

Q: DOES (2) GIVE US N EQUATIONS TO SOLVE FOR THE N UNKNOWNS, d_i ?

A: NO...

IF THERE ARE N DATA POINTS $(i = 1, 2, \ldots N)$, THEN EQUATION (2) IS OBVIOUSLY VALID ONLY FOR $i = 2 \ldots N-1$.

WE NEED TWO MORE EQUATIONS. THERE ARE SEVERAL WIDELY USED OPTIONS:

① "CLAMPED" SPLINE: LET d_1 AND d_N BE SPECIFIED EXPLICITLY.

② "NATURAL" SPLINE: $P_1''(X_1) = P_{N-1}''(X_N) = 0$.

③ "NOT A KNOT": $P_1'''(X_2) = P_2'''(X_2)$,
$P_{N-2}'''(X_{N-1}) = P_{N-1}'''(X_{N-1})$

IN OTHER WORDS, THE LAST TWO INTERVALS AT EACH END SHARE THE SAME CUBIC POLYNOMIAL.

NOTE: MATLAB'S SPLINE FUNCTION IMPLEMENTS ① & ③ BUT NOT ②.

HOW DO WE SOLVE FOR d_i?

NOTE THAT (2) IS A MATRIX EQUATION:

$$M \vec{d} = \vec{b}$$

KNOWN MATRIX — UNKNOWN — KNOWN

AND M IS A TRIDIAGONAL MATRIX:

$$\begin{bmatrix} M_{11} & M_{12} & 0 & & \\ 2/h_1 & (\frac{4}{h_1} + \frac{4}{h_2}) & 2/h_2 & 0 & \cdots \\ 0 & 2/h_2 & (4/h_2 + 4/h_3) & 2/h_3 & 0 \cdots \\ & & \cdots & & \\ & & 0 & 2/h_{N-1} & (4/h_{N-1} + 4/h_N) \quad 2/h_N \\ & & & 0 & M_{N,N-1} \quad M_{NN} \end{bmatrix}$$

5.4

Tridiagonal Systems

Charles Kankelborg

Rev. January 15, 2009

1 The Problem

A matrix equation $M\mathbf{u} = \mathbf{r}$ is just a way of writing N linear, algebraic equations. The ith equation is:

$$\sum_j M_{ij}\, u_j = r_i. \tag{1}$$

The standard problem is to solve for $\mathbf{u}$ given M and $\mathbf{r}$. A square matrix is said to be *tridiagonal* when

$$M_{ij} = 0, \quad |i - j| > 1. \tag{2}$$

For example,

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix}. \tag{3}$$

2 Algorithm

Have a look *Numerical Recipes* Section 2.4. The *Numerical Recipes* function `tridag` is easily understood without LU decomposition.

To solve the 4×4 system in equation 3 for $\mathbf{u}$, we begin by using row 1 to eliminate a_2 from row 2. Similarly, we eliminate all the a_i row by row.

Here is how the `tridag` implementation works. The first row stands for the equation:

5.5

$$b_1\, u_1 + c_1\, u_2 = r_1.$$

$$\text{Let} \quad \beta_1 = b_1, \tag{4}$$

$$v_1 = \frac{r_1}{\beta_1}, \tag{5}$$

$$\gamma_2 = \frac{c_1}{\beta_1} \tag{6}$$

If we divide row 1 by β_1, its equation becomes

$$u_1 + \gamma_2\, u_2 = v_1. \tag{7}$$

The equation for the second row is:

$$a_2\, u_1 + b_2\, u_2 + c_2\, u_3 = r_2.$$

$$\text{Let} \quad \beta_2 = b_2 - a_2\, \gamma_2,$$

$$v_2 = \frac{(r_2 - a_2\, v_1)}{\beta_2},$$

$$\gamma_3 = \frac{c_2}{\beta_2}.$$

We now subtract a_2 times row 1 from row 2. Dividing the result by β_2, our new row 2 is written:

$$u_2 + \gamma_3\, u_3 = v_2. \tag{8}$$

This process may be repeated over the range $i = 2, 3, \ldots N$, using the recursion formulas

$$\gamma_i = \frac{c_{i-1}}{\beta_{i-1}}, \tag{9}$$

$$\beta_i = b_i - a_i\, \gamma_i, \tag{10}$$

$$v_i = \frac{r_i - a_i\, v_{i-1}}{\beta_i}. \tag{11}$$

5.6

The result is that the matrix equation is rewritten in uper triangular form. For our 4×4 example,

$$\begin{pmatrix} 1 & \gamma_2 & 0 & 0 \\ 0 & 1 & \gamma_3 & 0 \\ 0 & 0 & 1 & \gamma_4 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}. \tag{12}$$

This problem is trivially solved by starting from the bottom and back substituting as you go up:

$$u_N = v_N, \tag{13}$$

$$u_i = v_i - \gamma_{i+1}\, u_{i+1}. \tag{14}$$

3 How to Break the Algorithm

If $\beta_i = 0$, for some i, the program quits to avoid a division by zero. The authors of *Numerical Recipes* assure us that this doesn't happen very often. For example, it can be shown that $\beta_i = 0$ cannot occur in the calculation of y_i'' for the cubic spline.

4 Applications

Tridiagonal matrices come up in a variety of contexts:

- Cubic splines (*NR* § 3.3)
- A concluding step in diagonalization of matrices (*NR* §§ 11.0, 11.2)
- Implicit differencing schemes for diffusive PDEs (*NR* § 19.2)
- Cyclic reduction methods for PDE BV problems (*NR* § 19.4)

5 MATLAB and Octave

In MATLAB or Octave, the code `u = M\r` will solve the matrix equation for **u**. The backslash operator is smart enough (as of MATLAB 7.5, and Octave 2.9) to test whether M is tridiagonal. If so, then a fast algorithm like the one described above is used. This all requires that your matrix is stored as a sparse matrix; see, e.g., `spdiags`.

PCHIP — PIECEWISE CUBIC HERMITE INTERPOLATING POLYNOMIALS

CUBIC SPLINES ARE NOT NORMALLY DERIVED IN TERMS OF THE CUBIC HERMITE BASIS FUNCTIONS. (COMPARE THE DERIVATION IN NUMERICAL RECIPES 2nd ED., § 3.3). I DID IT THIS WAY TO EMPHASIZE THE RELATIONSHIP TO ANOTHER CATEGORY OF CUBIC INTERPOLATION SCHEMES.

PCHIP IS THE NAME OF A WELL-KNOWN INTERPOLATION ALGORITHM (AVAILABLE IN FORTRAN & MATLAB) THAT IS DESIGNED TO AVOID OVERSHOOTS.

THE BASIC FRAMEWORK IS GIVEN BY EQUATION (1).

RATHER THAN REQUIRING CONTINUITY OF THE 2nd DERIVATIVES, PCHIP SETS THE d_i ACCORDING TO A 2-STEP PROCEDURE:

① $$d_i = \frac{f_{i+1} - f_{i-1}}{x_{i+1} - x_{i-1}}, \qquad 2 \le i \le N-1$$

$$d_1 = \frac{f_2 - f_1}{x_2 - x_1}, \qquad d_N = \frac{f_N - f_{N-1}}{x_N - x_{N-1}}$$

} IF WE STOP HERE THIS IS MY "SIMPLE CUBIC" ALGORITHM USED IN SUBSEQUENT EXAMPLES.

② FOR EACH INTERVAL i, IF $P_i(x)$ IS NOT MONOTONIC, THEN MODIFY d_i AND d_{i+1} SO THAT IT IS.

THE DETAILS OF STEP ② ARE IN FRITSCH & CARLSON, "MONOTONE PIECEWISE CUBIC INTERPOLATION", SIAM JOURNAL OF NUMERICAL ANALYSIS, 17 : 238-246 (1980).

SUMMARY

★ THE PROBLEM:

- GIVEN N DATA POINTS (x_i, f_i)
- RECONSTRUCT $f(x)$, EXACTLY SATISFYING THE DATA.

★ APPROACHES: ("SMOOTH" TO VARYING DEGREES)

- LINEAR INTERPOLATION (C_0)
- PCHIP AND "SIMPLE CUBIC" (C_1)
- CUBIC SPLINE (C_2) [NR § 3.3]
  - → VARIOUS END CONDITIONS
- POLYNOMIAL OR RATIONAL ("C_∞") [NR § 3.2-3.3]

WE SKIPPED THIS.

HOW DO I CHOOSE AN INTERPOLATION METHOD?
→ EXAMPLES FORTHCOMING

★ IMPORTANT POINT: FITTING DATA WITH SIGNIFICANT MEASUREMENT NOISE IS A DIFFERENT PROBLEM. INTERPOLATION IN THAT CASE IS NOT HELPFUL!

★ WHERE DO WE GO FROM HERE?

- EXAMPLES ⇒ ADVANTAGES & DISADVANTAGES OF EACH METHOD
- INTERPOLATION OF UNIFORMLY GRIDDED DATA (MOST TIME SERIES, DIGITAL IMAGES, ETC.)
  - → FOURIER TRANSFORM INTERPOLATION (FFT)
  - → CONVOLUTIONAL INTERPOLATION

EXAMPLES FOLLOW. THE MAIN PROGRAM IS interpolation.m. A SIMPLE CUBIC INTERPOLATION, USING JUST STEP 1 OF THE PCHIP ALGORITHM, IS IMPLEMENTED IN simple_cubic_setup.m.

Q: WHAT ARE THE ADVANTAGES & DISADVANTAGES OF EACH OF THE FOLLOWING:

- LINEAR INTERPOLATION
- CUBIC SPLINE
- "SIMPLE" CUBIC
- PCHIP

[DISCUSS]