Selected Solutions for Exercises in

Numerical Methods with MATLAB: Implementations and Applications

Gerald W. Recktenwald

Chapter 10

Interpolation

10.3 What are the condition numbers for the two Vandermonde matrices in Examples 10.4 and 10.5? How does the change in $\kappa(A)$ relate to the different results obtained in Example 10.5?

Numerical Answer: For Example 10.4, $\kappa(A) = 1.0 \times 10^{31}$. For Example 10.5, $\kappa(A) = 3.8 \times 10^{3}$.

$\diamondsuit$

10.3 (2+) In Example 10.4, the coefficients of the interpolating polynomial are evaluated and printed to five significant digits. Evaluate the gasoline prices and the errors in the interpolant using the truncated coefficients and the following definitions: Let (y_i, p_i) be the year and price in the given tabulated data. Let $\hat{p}_i$ be the price interpolated with the untruncated polynomial coefficients at the y_i, and $\tilde{p}_i$ be the price interpolated with the truncated coefficients at the y_i. In the absence of numerical errors we expect $p_i = \hat{p}_i = \tilde{p}_i$ from the definition of interpolation. Compute and print p_i, $\hat{p}_i - p_i$, $\|\hat{p} - p\|_2$, $\tilde{p}_i$, $\tilde{p}_i - p_i$, and $\|\tilde{p} - p\|_2$. How many digits of the c coefficients need to be retained in order to get $\|p - \tilde{p}\|_\infty$ comparable to $\|p - \hat{p}\|_\infty$? *Hint*: The `chop10` function in the `utils` directory of the NMM toolbox will be helpful.

Partial Solution: The original and chopped coefficients are:

```
      c(k)              chopped c(k)
 3.03815522e-03    3.03820000e-03
-3.02074596e+01   -3.02070000e+01
 1.20137182e+05    1.20140000e+05
-2.38896442e+08   -2.38900000e+08
 2.37525874e+11    2.37530000e+11
-9.44650495e+13   -9.44650000e+13
```

Evaluating the errors in the original polynomial (`error p`), and the polynomial obtained with chopped coefficients (`error pc`) gives

```
year     price      error p     error pc
1986    133.50     4.69e-02     2.48e+10
1988    132.20    -1.25e-02     2.49e+10
1990    138.70    -9.06e-02     2.50e+10
1992    141.50    -4.69e-02     2.51e+10
1994    137.60    -3.75e-02     2.51e+10
1996    144.20     9.69e-02     2.52e+10

L2 norm of price error with original coefficients =     1.53e-01
L2 norm of price error with chopped coefficients  =     6.13e+10
```

$\diamondsuit$

10.12 Manually compute the quadratic interpolation in Example 10.7 using instead the following support points.

(a) $T_1 = 20$, $T_2 = 30$, $T_3 = 40$

(b) $T_1 = 0$, $T_2 = 10$, $T_3 = 20$

Compare the results to the interpolant using $T_1 = 10$, $T_2 = 20$, and $T_3 = 30$.

Numerical Answer:

Support Points	$\mu(22)$	Difference
$T = 10, 20, 30$	1.202	0
$T = 20, 30, 40$	1.278	6.3%
$T = 0, 10, 20$	1.565	30%

◊

10.14 The `H2Osat.dat` file in the `data` directory of the NMM toolbox contains saturation data for water. Use this data and quadratic polynomial interpolation to manually compute p_{sat} in the range $30 \le T \le 35°\text{C}$.

(a) *Manually* construct the divided-difference table. Use the `divDiffTable` function to check your calculations.

(b) Extract the coefficients of the Newton interpolating polynomial from the divided difference table.

(c) Evaluate the interpolant at $T = 32$, 33, and 34 °C. Verify you calculations with `newtint`.

Partial Solution: Three support points are needed for quadratic interpolation. The data in `H2Osat.dat` is in increments of 5°C. Thus, reasonable support points for the interpolation are either $T = 25$, 30, and 35, or $T = 30$, 35, and 40. The table below summarizes the results of the interpolations.

support points	$p_{\text{sat}}(32)$	$p_{\text{sat}}(33)$	$p_{\text{sat}}(34)$
$T = 25, 30, 35$	0.04762	0.05039	0.05327
$T = 30, 35, 40$	0.04754	0.05030	0.05322

All pressures are in bar.

◊

10.18 The degree of the polynomial interpolant used by `lagrint` and `newtint` is determined by the length of the input vectors to these functions. Suppose one wished to specify the order of the interpolant, regardless of the length of the input data. In other words, suppose the objective was to perform a local interpolation of degree n in a table of length m, where $m > n$. This requires selecting an appropriate subset of the input data table and passing this subset to `lagrint` or `newtint`. For example, a quadratic interpolation could be performed with

```
x = ...       %  define tabular data
y = ...
xhat = ...    %  interpolate at this value
ibeg = ...    %  beginning index for support points of interpolant
yhat = newtint(x(ibeg:ibeg+2),y(ibeg:ibeg+2),xhat)
```

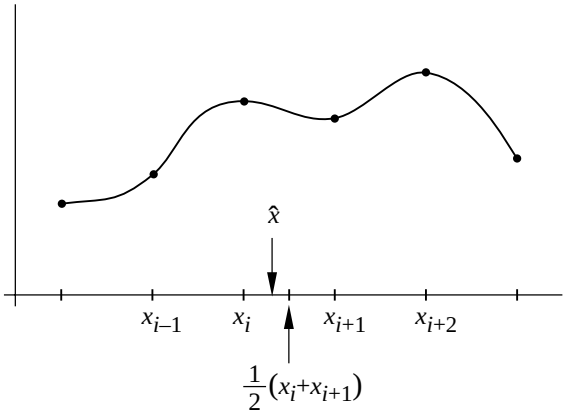
Write an m-file function called `quadinterp` that automatically selects an appropriate subset of the vectors `x` and `y`, and returns the value of the quadratic interpolant using those support points. The function definition statement for `quadinterp` is

```
function yhat = quadinterp(x,y,xhat)
```

The `quadinterp` function calls `newtint` to perform the interpolation. The `binSearch` function in Listing 10.6 will be useful. Use your `quadinterp` function to generate data for smooth plot of the glycerin viscosity data in Example 10.2.

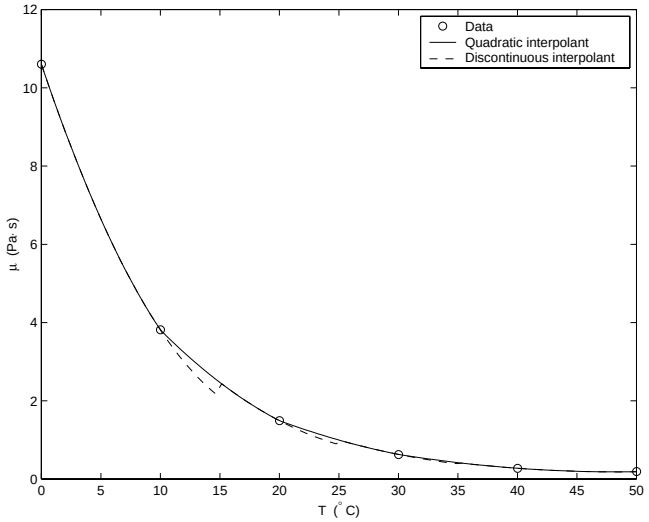
Partial Solution: The selection of support points can be automated with the `binSearch` function from Listing 10.6. The `binSearch` function finds the index `i` such that `x(i)` $\leq$ `xhat` $\leq$ `x(i+1)`, when the `x` vector contains monotonically increasing values. The question remains: "Which additional point is used to create the three support points needed to define the quadratic interpolant?" To avoid an index range error when `i` $= 1$, the three support pairs must be (`x(1)`,`y(1)`), (`x(2)`,`y(2)`), and (`x(3)`,`y(3)`). Similarly, when `i` $= n - 1$ the three support pairs must be (`x(n-2)`,`y(n-2)`), (`x(n-1)`,`y(n-1)`), and (`x(n)`,`y(n)`). How should we choose the support points when $2 \leq$ `i` $\leq n - 2$?

Suppose we an attempt to optimize the choice of support points. (This turns out to be a bad idea, but we'll pursue it anyway.) Assume, for simplicity, that the `x` data are equally spaced, and consider the case, as depicted in the following sketch, where `xhat` is closer to `x(i)` than `x(i+1)`



Exercise 10–18. Attempt at optimal selection of support Points

For this `xhat` a good choice of support points would seem to be (`x(i-1)`,`y(i-1)`), (`x(i)`,`y(i)`), and (`x(i+1)`,`y(i+1)`). On the other hand, if `xhat` $>$ (`x(i)` + `x(i + 1)`)/2 a good choice of support points would seem to be (`x(i)`,`y(i)`), (`x(i+1)`,`y(i+1)`), and (`x(i+2)`,`y(i+2)`). The `quadInterpBad` function listed on the next page here uses this logic The `demoQuadInterp` function uses `quadInterpBad` and `quadInterp` functions to interpolate the glycerin viscosity data. Runninng `demoQuadInterp` gives the following plot.



Exercise 10–18. Two versions of piecewise quadratic interpolation

The problem with our attempt to optimize the choice of the support points is that the support points change in the middle of an interval, which causes the interpolant to be discontinuous. For this problem it is better to change the support points only at the ends of the intervals *at* the support points. The code to do this is actually simpler than the code in `quadInterpBad`. A better implementation of `quadInterp` involves choosing `ibeg=i` for all cases except when `i=length(x)-1`. Converting the code in `quadInterpBad` is left for the reader. On the next page are listings of the `demoQuadInterp` and `quadInterpBad` functions.

An explicit loop is used to return a vector of interpolant values (`yhat`) if the input `xhat` is a vector. This is necessary since the `binSearch` function cannot be vectorized. The `scalarQuadInterp` *subfunction* performs the quadratic interpolation for a single value of `xhat`.

```
function demoQuadInterp
% demoQuadInterp  Piecewise quadratic interpolation of glycerin data
%                 Exercise 10-18

% --- Read data from glycerin.dat
[t,D] = loadColData('glycerin.dat',5,7);
mu = D(:,2);   %  viscosity is in second column of file

% --- Evaluate interpolants
ti = linspace(min(t),max(t));
mui = quadInterp(t,mu,ti);
mu2 = quadInterpBad(t,mu,ti);

plot(t,mu,'ko',ti,mui,'b-',ti,mu2,'r--')
legend('Data','Quadratic interpolant','Discontinuous interpolant');
xlabel('T  (^\circ C)');    ylabel('\mu  (Pa\cdot s)');
```

```
function yhat = quadInterpBad(x,y,xhat)
% quadInterpBad  Discontinuous piecewise quadratic interpolation
%                in a table of (x,y) data.   Exercise 10-18
%
% Synposis:  yhat = quadInterpBad(x,y,xhat)
%
% Input:     x,y = vectors containing the tabulated data
%            xi  = value of x at which function y = f(x) is desired
%
% Output:    yi = value of y at xi obtained by quadratic interpolation

% --- Allow vector of inputs:  use explicit loop over all elements of xhat
yhat = zeros(size(xhat));
for k=1:length(xhat)
   yhat(k) = scalarQuadInterp(x,y,xhat(k));
end

% =============
function yhat = scalarQuadInterp(x,y,xhat)
% scalarQuadInterp   Piecewise quadratic interpolation at a single point
%
%  Use hueristic choice of three support points.  Let i be the index
%  returned from binSearch, i.e. x(i) <= xhat <= x(i+1).  If i is in
%  the range 2 <= i <= n-1, and if xhat is less than the midpoint of
%  [x(i), x(i+1)], then choose support points [i-1, i, i+1].  Otherwise
%  choose support points [i, i+1, i+2].
i = binSearch(x,xhat);       %  Find appropriate data pair
if i==1
  ibeg=1;
elseif i==length(x)-1
  ibeg=length(x)-2;                     %  Avoid index range error
elseif xhat < (x(i)+0.5*(x(i+1)-x(i)))   %  hueristic choice of support points
  ibeg = i-1;
else
  ibeg = i;
end
yhat = newtint(x(ibeg:ibeg+2),y(ibeg:ibeg+2),xhat);  %  Evaluate interpolant
```

10.21 The classic example of polynomial wiggle is the so-called *Runge* function

$$r(x) = \frac{1}{1 + 25x^2},$$

named after German mathematician Carl Runge (1856–1927), who used this function to study the behavior of high-degree polynomial interpolation. Write a function m-file called `runge` to perform the following tasks.

(a) Compute n equally spaced x_k values ($k = 1, \ldots, n$) on the interval $-1 \le x \le 1$. Let n be an input parameter to `runge`.

(b) Evaluate $r(x_k)$ from Equation (10.55) for $k = 1, \ldots, n$.

(c) Use the n pairs of $(x_k, r(x_k))$ values to define a degree $n-1$ polynomial interpolant, P_{n-1}. Evaluate the interpolant at $\hat{x}_j$, $j = 1, \ldots, 100$ points in the interval $-1 \le x \le 1$.

(d) Plot a comparison of the original data, $(x_k, r(x_k))$, the interpolant, $(\hat{x}_j, P_{n-1}(\hat{x}_j)$, and the true value of the function at the interpolating points $(\hat{x}_j, r(\hat{x}_j)$. Use open circles for $r(x_k)$, a dashed line for $P_{n-1}(\hat{x}_j)$, and a solid line for $r(\hat{x}_j)$.

(e) Print the value of $\|r(\hat{x}) - P_{n-1}(\hat{x})\|_2$.

Run your `runge` function for `n=5:2:15`, and discuss the behavior of $\|r(\hat{x}) - P_{n-1}(\hat{x})\|_2$ as `n` is increased.

Partial Solution: The `runge` function is listed below. Running the function and interpretting the results are left to the reader.

```
function e = runge(n)
% runge  Uniformly spaced interpolation of the Runge function.  Exercise 10-21
%
% Synopsis:  runge(n)
%
% Input:  n = number of points to define the polynomial interpolant.
%             The polynomial is of degree n-1
%
% Output: e = L2 norm of error for the interpolant evaluated at
%             100 points in the interval -1 <= x <= 1

x = linspace(-1,1,n);        %  Define n support points (x,r(x))
r = 1./(1+25*x.^2);          %  Exact value of Runge function

xhat = linspace(-1,1);       %  100 equally-spaced points
yhat = newtint(x,r,xhat);    %  Polynomial interpolant of degree n-1
rhat = 1./(1+25*xhat.^2);    %  Exact r(x) at xhat

plot(x,r,'o',xhat,yhat,'--',xhat,rhat,'-');
e = norm(yhat-rhat);
```

◊

10.24 The `stdatm.dat` in the `data` directory of the NMM toolbox gives the properties of the so-called standard atmosphere as a function of elevation above sea level. Write a routine that uses piecewise linear interpolation to return the values of T, p, and ρ at any elevation in the range of the table. Write your m-file function so that it does not read the data from this file when it is executed; that is, store the data in matrix variables in your m-file. What are the values of T, p, and ρ at $z = 5500\,\text{m}$ and $z = 9023\,\text{m}$? Plot the variation of T and ρ in the range $0 \le z \le 15\,\text{km}$.

Numerical Answer:

z (m)	T (°C)	p (Pa)	ρ (kg/m^3)
5500	−17.47	54050	0.7364
9023	−43.56	30701	0.4659

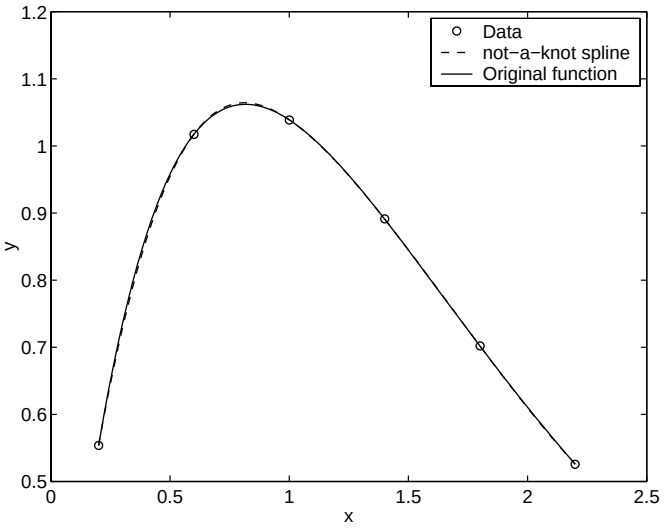
◊

10.32 Plot the spline interpolant of the following data:

x_i	0.2	0.6	1.0	1.4	1.8	2.2
y_i	0.5535	1.0173	1.0389	0.8911	0.7020	0.5257

On the same plot, compare the spline interpolant to $y = \sqrt{12.5}\,x \exp(-\sqrt{1.5}\,x)$, which was used to create the data in the table.

Partial Solution: Below is a plot of the sample data, the spline interpolant, and the function used to create the sample data. Lacking any information about the slope at the endpoints, the not-a-knot end conditions were used to define the spline.



Plot of solution to **Exercise 10–32**.

◊