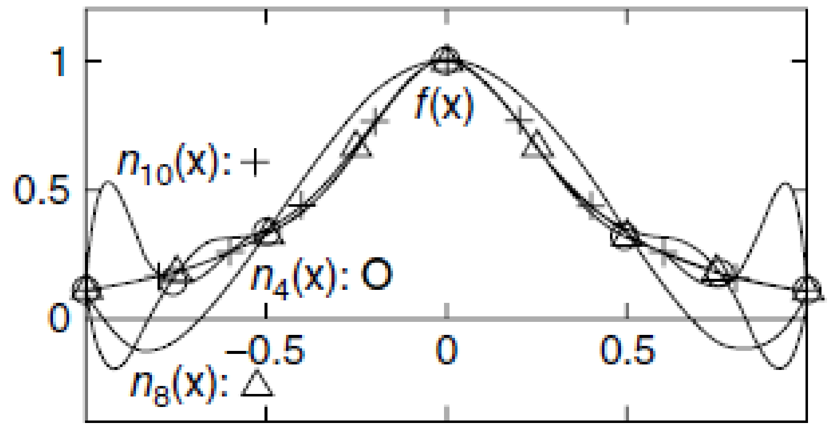
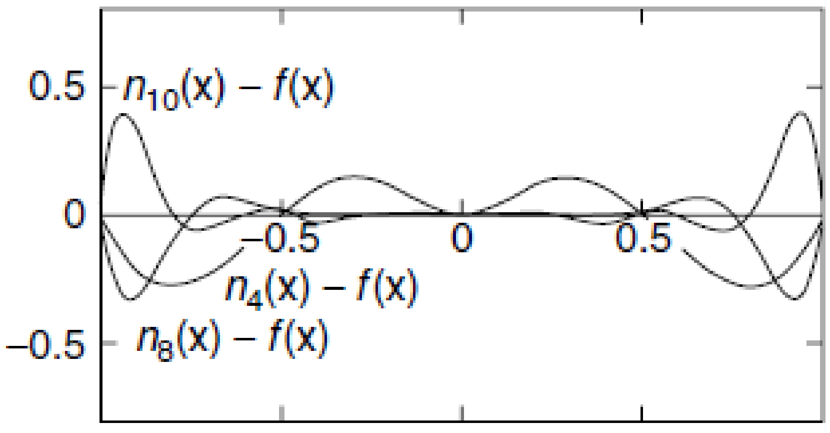
Wiggle/Oscillation: Runge Phenomenon



(a) 4/8/10th -degree polynomial approximation

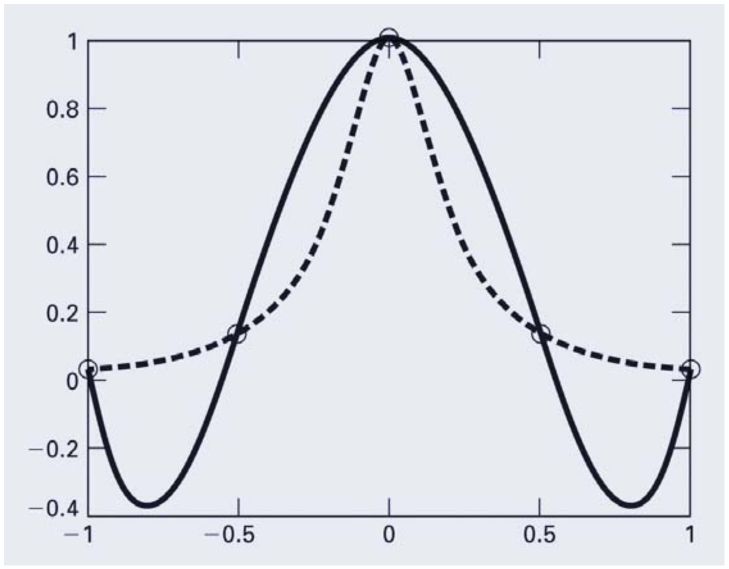


(b) The error between the approximating polynomial and the true function

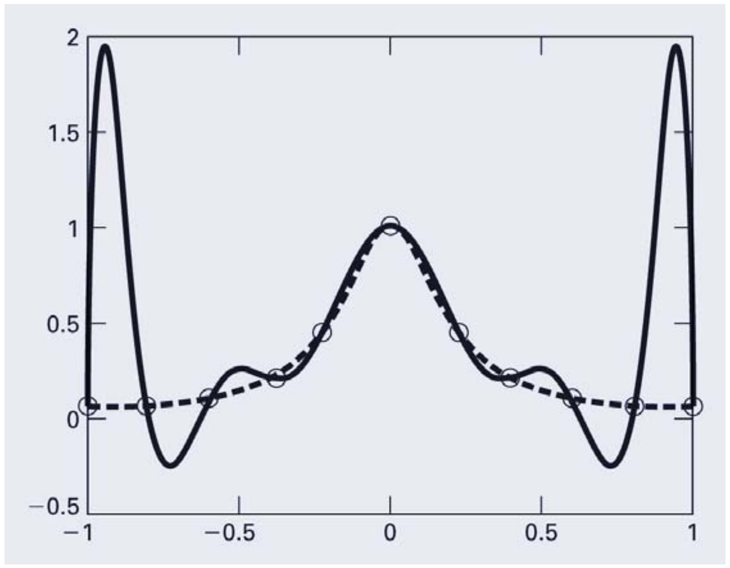
Polynomial Wiggle and Runge Phenomenon. Here is one thing to note. Strangely, increasing the degree of polynomial contributes little to reducing the approximation error. Rather contrary to our usual expectation, it tends to make the oscillation strikingly large, which is called the polynomial wiggle and the error gets bigger in the parts close to both ends as can be seen in the above Figure, which is called the Runge phenomenon. That is why polynomials of degree 5 or above are seldom used for the purpose of interpolation, unless they are sure to fit the data.

Wiggle: More example

- Higher-order polynomials can not only lead to round-off errors due to ill-conditioning, but can also introduce oscillations to an interpolation or fit where they should not be.
- In the figures below, the dashed line represents an function, the circles represent samples of the function, and the solid line represents the results of a polynomial interpolation:



Using 4th order Polynomial



Using 10th order Polynomial

Spline

- apply lower-order polynomials in a piecewise fashion to data points.
- These connecting polynomials are called *spline functions*.
- Splines minimize oscillations and reduce round-off error

  due to their lower-order nature.

- Splines eliminate oscillations by using small subsets of points for each interval rather than every point. This is especially useful when there are jumps in the data:

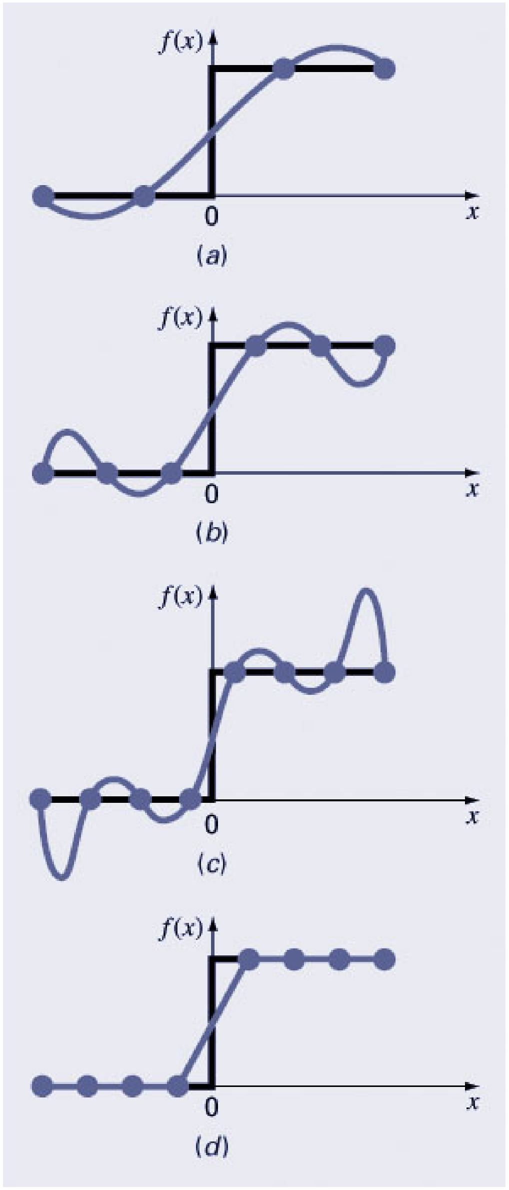
  a) 3rd order polynomial

  b) 5th order polynomial

  c) 7th order polynomial

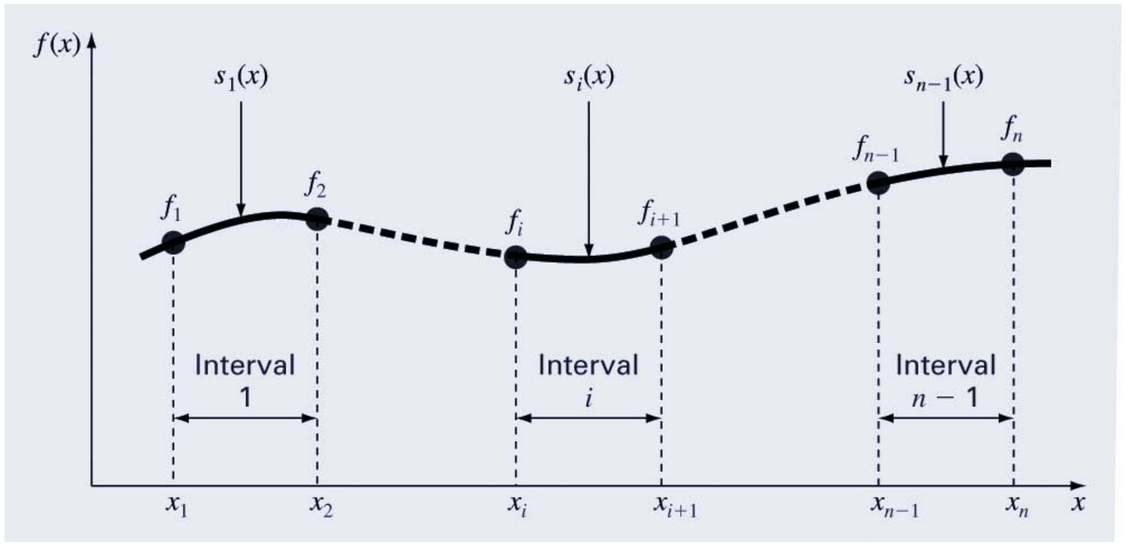
  d) Linear spline

  - seven 1st order polynomials generated by using pairs of points at a time



Spline

- Spline function ($s_i(x)$)coefficients are calculated for each interval of a data set.
- The number of data points (f_i) used for each spline function depends on the order of the spline function.



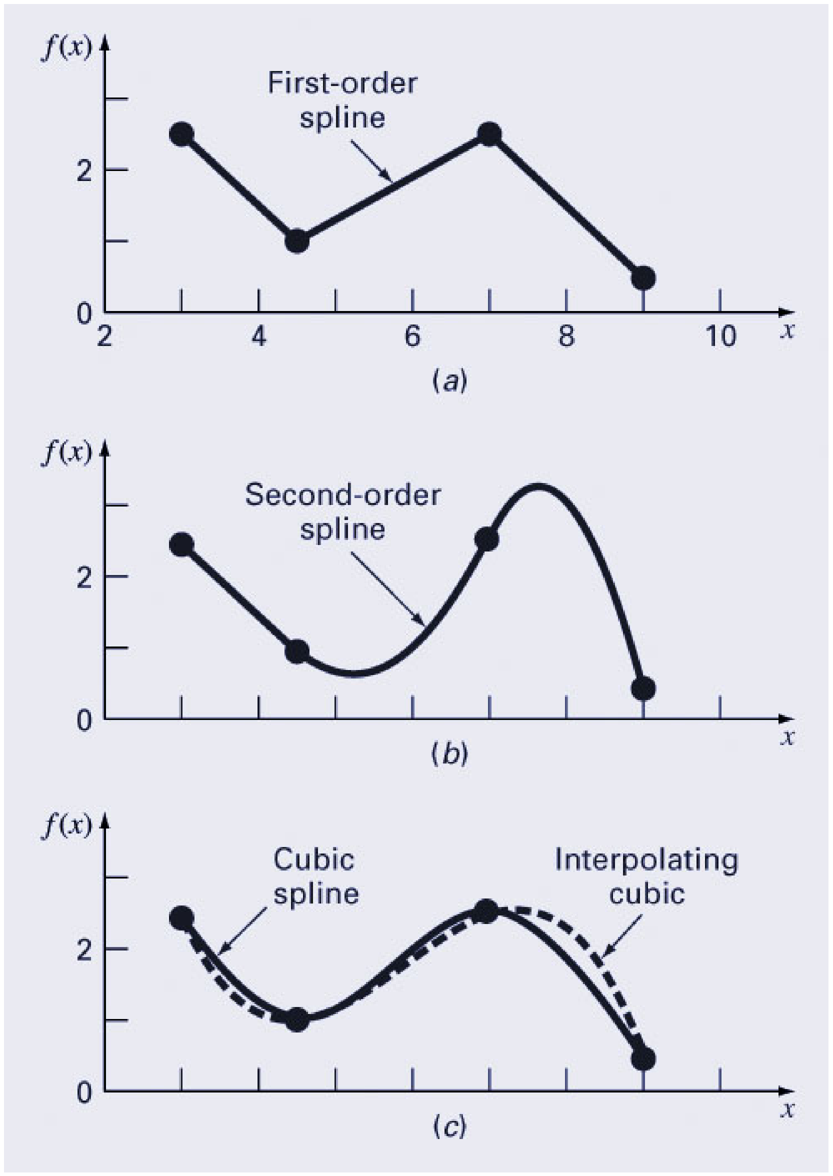
Spline

a) First-order splines find straight-line equations between each pair of points that
   - Go through the points

b) Second-order splines find quadratic equations between each pair of points that
   - Go through the points
   - Match first derivatives at the interior points

c) Third-order splines find cubic equations between each pair of points that
   - Go through the points
   - Match first and second derivatives at the interior points

- Note that the results of cubic spline interpolation are different from the results of an interpolating cubic.



Cubic Spline

- While data of a particular size presents many options for the order of spline functions, cubic splines are preferred because they provide the simplest representation that exhibits the desired appearance of smoothness.
  - Linear splines have discontinuous first derivatives
  - Quadratic splines have discontinuous second derivatives and require setting the second derivative at some point to a pre-determined value
    but
  - Quartic or higher-order splines tend to exhibit the instabilities inherent in higher order polynomials (ill-conditioning or oscillations)

- In general, the *i*th spline function for a cubic spline can be written as:

$$s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

- For *n* data points, there are *n*-1 intervals

  - thus 4(n-1) unknowns to evaluate to solve all the spline function coefficients.

Spline in MATLAB

- MATLAB has a built-in function to implement spline interpolation.

  ```
  yy=spline(x, y, xx)
  ```

  This performs cubic spline interpolation, generally using not-a-knot conditions. If y contains two more values than x has entries, then the first and last value in y are used as the derivatives at the end points (i.e. clamped)

MATLAB's interp1 Function

- While spline can only perform cubic splines, MATLAB's interp1 function can perform several different kinds of interpolation:

```
yi = interp1(x, y, xi, 'method')
```

  - `x` & `y` contain the original data
  - `xi` contains the points at which to interpolate
  - `'method'` is a string containing the desired method:
    - `'nearest'` - nearest neighbor interpolation
    - `'linear'` - connects the points with straight lines
    - `'spline'` - not-a-knot cubic spline interpolation
    - `'pchip'` or `'cubic'` - piecewise cubic Hermite interpolation