Mathematical Physics 212C

Numerical Methods
Dennis Silverman

Some of the examples are taken from "Guide to Standard Mathematica Packages", Version 2.2, Wolfram Research

■ Interpolation

There are two ways to input data :
(1) If equally spaced xi integers starting at 1, only need to give data points fi .
(2) If at various xi, have to give set of pairs -
{{x1, f1}, {x2, f2}, ...}
Ex . Least squares fit in x using given functions 1, x, x^2

```
data = Table[n^2, {n, 1, 7}]
```

```
{1, 4, 9, 16, 25, 36, 49}
```

```
Fit[data, {1, x, x^2}, x]
```

$3.55271\times 10^{-15} - 7.10543\times 10^{-15}\, x + 1.\, x^2$

```
?? Fit
```

```
Fit[data, funs, vars] finds a least-squares fit to a list of data as a linear combination
  of the functions funs of variables vars. The data can have the form {{x1, y1, ... ,
  f1}, {x2, y2, ... , f2}, ... }, where the number of coordinates x, y, ...  is equal
  to the number of variables in the list vars. The data can also be of the form {f1,
  f2, ... }, with a single coordinate assumed to take values 1, 2, ... . The argument
  funs can be any list of functions that depend only on the objects vars.
```

```
Attributes[Fit] = {Protected}
```

```
InterpolatingPolynomial[{{1, 4}, {2, 5}, {4, 6}}, x]
```

$4 + \left(1 + \frac{2 - x}{6}\right) (-1 + x)$

```
?? InterpolatingPolynomial
```

```
InterpolatingPolynomial[data, var] gives a polynomial in the variable var which provides an exact
  fit to a list of data. The data can have the forms {{x1, f1}, {x2, f2}, ... } or {f1,
  f2, ... }, where in the second case, the xi are taken to have values 1, 2, ... . The
  fi can be replaced by {fi, dfi, ddfi, ... }, specifying derivatives at the points xi.

Attributes[InterpolatingPolynomial] = {Protected}
```

```
?? Interpolation
```

```
Interpolation[data] constructs an InterpolatingFunction object which represents an approximate
  function that interpolates the data. The data can have the forms {{x1, f1}, {x2,
  f2}, ... } or {f1, f2, ... }, where in the second case, the xi are taken to have values 1, 2, ... .

Attributes[Interpolation] = {Protected}

Options[Interpolation] = {InterpolationOrder -> 3}
```

Rational Interpolation or Pade Approximant with numerator of degree m and denominator of degree k.

```
<< NumericalMath`Approximations`
```

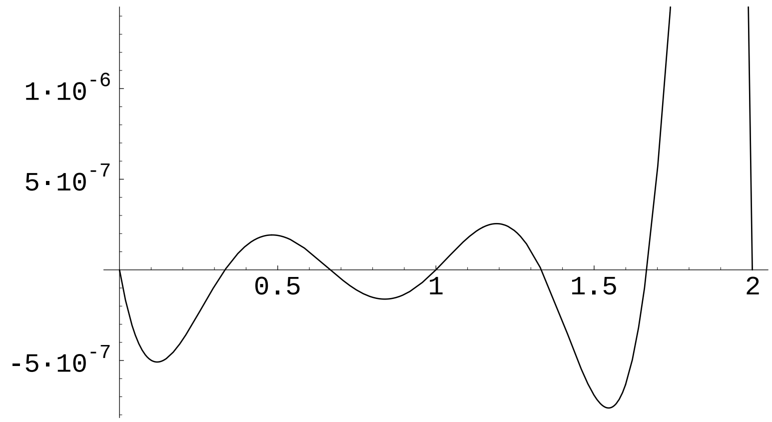
```
?? RationalInterpolation
```

```
RationalInterpolation[func, {x, m, k}, {x1, x2, ..., xn}, (opts)], (n = m+k+1),
  gives the rational interpolant to func (a function of the variable x), where m  and
  k are the degrees of the numerator and denominator, respectively, and {x1, x2, ...,
  xn} is a list of m+k+1 abscissas of the interpolation points. An alternative form is
 RationalInterpolation[func, {x, m, k}, {x, x0, x1}, (opts)], which specifies the list of abscissas
  implicitly: the abscissas come from the interval (x0,x1).  The function func must be Listable.
```

```
ri1 = RationalInterpolation[Exp[x], {x, 2, 4}, {0, 1/3, 2/3, 1, 4/3, 5/3, 2}]
```

$$(1.000000000000000 + 0.379961505998214\,x + 0.0469527572648759\,x^2) / (1 - 0.620028516690566\,x + 0.1669139144430911\,x^2 - 0.02340576618306169\,x^3 + 0.001452790199322340\,x^4)$$

```
Plot[ri1 - Exp[x], {x, 0, 2}]
```



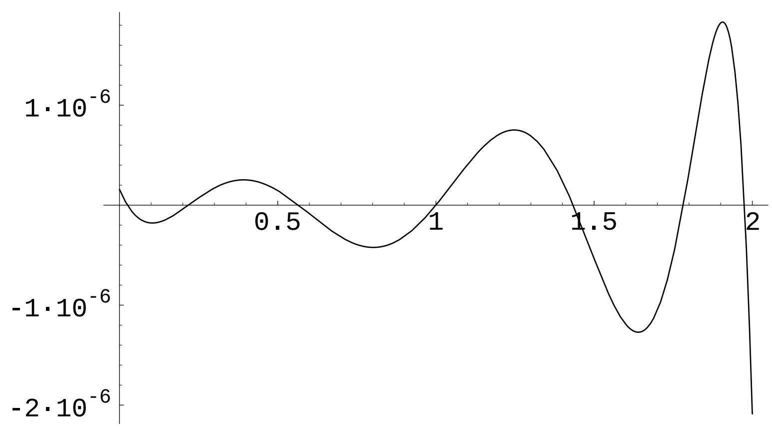
- Graphics -

Instead, we can let Mathematica choose the interpolation points.

```
ri2 = RationalInterpolation[Exp[x], {x, 2, 4}, {x, 0, 2}]
```

$(1.000000157557967 + 0.379826643610590\, x + 0.0468693215807399\, x^2) \,/$
$(1 - 0.620165730391327\, x + 0.1669778816332341\, x^2 - 0.0234118949793 0664\, x^3 +$
$0.001451916095874726\, x^4)$

```
Plot[ri2 - Exp[x], {x, 0, 2}]
```



- Graphics -

To minimize the maximum error over the entire range of interpolation, use

? MiniMaxApproximation

```
MiniMaxApproximation[func, {x, {x0, x1}, m, k}, (opts)] finds the mini-max approximation
  to func (a function of the variable x) on the interval (x0, x1), where m and k are
  the degrees of the numerator and denominator, respectively. The answer returned is
  {AbscissaList, {Approximation, MaxError}}, where AbscissaList is a list of the abscissas
  where the maximum error occurs, Approximation is the rational approximation desired,
  and MaxError is the value of the mini-max error.  The function func must be Listable.
 MiniMaxApproximation[f, approx, {x, {x0, x1}, m, k}, (opts)] is a form that allows the
  user to start the iteration from a known approximation.  Here approx must be in the
  form of an answer returned by MiniMaxApproximation.
```

mmlist = MiniMaxApproximation[Exp[x], {x, {0, 2}, 2, 4}]

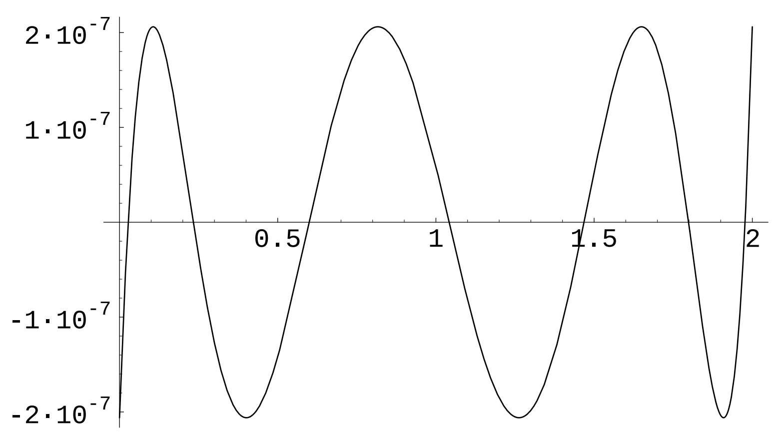
```
{{0, 0.1063486487562547, 0.400915308263480, 0.816636147986609,
  1.262697793261276, 1.649749456495247, 1.909120496974660, 2.000000000000000},
 {(1.000000206052106 + 0.380881473298916 x + 0.0472394925260850 x^2) /
   (1 - 0.619109229762559 x +
     0.1662828794271723 x^2 - 0.02323044886220643 x^3 + 0.001433248923483058 x^4),
  -2.060521062645246 × 10^-7}}
```

To extract just the rational approximation

mmfunc = mmlist[[2, 1]]

```
(1.000000206052106 + 0.380881473298916 x + 0.0472394925260850 x^2) /
 (1 - 0.619109229762559 x + 0.1662828794271723 x^2 - 0.02323044886220643 x^3 +
   0.001433248923483058 x^4)
```

Plot[1 - mmfunc / Exp[x], {x, 0, 2}]



- Graphics -

■ Roots

```
?FindRoot
```

```
FindRoot[lhs==rhs, {x, x0}] searches for a numerical solution to the equation lhs==rhs,
  starting with x=x0.
```

```
FindRoot[gam == Sinh[gam], {gam, 1}]
```

```
{gam → 0.0178633}
```

```
FindRoot[gam == 0.5 Cosh[gam], {gam, 0}]
```

```
{gam → 0.589388}
```

■ Integrating Numerical Data

```
<< NumericalMath`ListIntegrate`
```

```
?ListIntegrate
```

```
ListIntegrate[{y0, y1, ..., yn}, h, k] uses an InterpolatingFunction object to give an
  approximation to the integral of a function with values equal to  y0,...,yn at points
  equally spaced a distance h apart.  If k is odd it is increased by 1.  The default
  value of k is 4. ListIntegrate[{{x0,y0}, {x1,y1}, ..., {xn,yn}}, k] can be used for
  variable stepsize data.  If the data are known to contain errors, you may be better
  off performing Integrate on the result of Fit applied to the data.
```

This integrates a collection of interpolating polynomials of degree k .

```
data = Table[n^2, {n, 0, 7}]
```

```
{0, 1, 4, 9, 16, 25, 36, 49}
```

```
ListIntegrate[data, 1]
```

$\frac{343}{3}$

```
ListIntegrate[data, 1, 2]
```

$\frac{231}{2}$

```
ListIntegrate[data, 1, 3]
```

$\frac{343}{3}$

```
Integrate[x^2, {x, 0, 7}]
```

$\frac{343}{3}$

■ Gaussian Quadrature

```
<< NumericalMath`GaussianQuadrature`
```

```
?GaussianQuadratureWeights
```

GaussianQuadratureWeights[n, a, b, prec] gives a list of the pairs {abscissa, weight} to prec digits precision for the elementary n-point Gaussian quadrature formula for quadrature on the interval a to b. The argument prec is optional.

```
GaussianQuadratureWeights[2, -1, 1]
```

```
{{-0.57735, 1.}, {0.57735, 1.}}
```

```
GaussianQuadratureError[2, f, -1, 1]
```

$-0.00740741\, f^{(4)}$