interp1

1-D data interpolation (table lookup)

If you pass nonuniformly spaced points and specify the `'v5cubic'` method, `interp1` issues a warning. In addition, note the status of the following syntaxes:

- The behavior of `interp1(...,'cubic')` will change in a future release.
- `pp = interp1(...,'pp')` is not recommended.

Syntax

```
vq = interp1(x,v,xq)
```
example

```
vq = interp1(x,v,xq,method)
```
example

```
vq = interp1(x,v,xq,method,extrapolation)
```
example

```
vq = interp1(v,xq)
```
example

```
vq = interp1(v,xq,method)
vq = interp1(v,xq,method,extrapolation)

pp = interp1(x,v,method,'pp')
```

Description

`vq = interp1(x,v,xq)` returns interpolated values of a 1-D function at specific query points using linear interpolation. Vector `x` contains the sample points, and `v` contains the corresponding values, *v*(*x*). Vector `xq` contains the coordinates of the query points. example

If you have multiple sets of data that are sampled at the same point coordinates, then you can pass `v` as an array. Each column of array `v` contains a different set of 1-D sample values.

`vq = interp1(x,v,xq,method)` specifies an alternative interpolation method: `'nearest'`, `'next'`, `'previous'`, `'linear'`, `'spline'`, `'pchip'`, or `'cubic'`. The default method is `'linear'`. example

`vq = interp1(x,v,xq,method,extrapolation)` specifies a strategy for evaluating points that lie outside the domain of `x`. Set `extrapolation` to `'extrap'` when you want to use the `method` algorithm for extrapolation. Alternatively, you can specify a scalar value, in which case, `interp1` returns that value for all points outside the domain of `x`. example

`vq = interp1(v,xq)` returns interpolated values and assumes a default set of sample point coordinates. The default points are the sequence of numbers from `1` to `n`, where `n` depends on the shape of `v`: example

- When `v` is a vector, the default points are `1:length(v)`.
- When `v` is an array, the default points are `1:size(v,1)`.

Use this syntax when you are not concerned about the absolute distances between points.

`vq = interp1(v,xq,method)` specifies any of the alternative interpolation methods and uses the default sample points.

`vq = interp1(v,xq,method,extrapolation)` specifies an extrapolation strategy and uses the default sample points.

`pp = interp1(x,v,method,'pp')` returns the piece-wise polynomial form of *v*(*x*) using the `method` algorithm.

Examples

collapse all

Interpolation of Coarsely Sampled Sine Function

Define the sample points, `x`, and corresponding sample values, `v`.

Open This Example

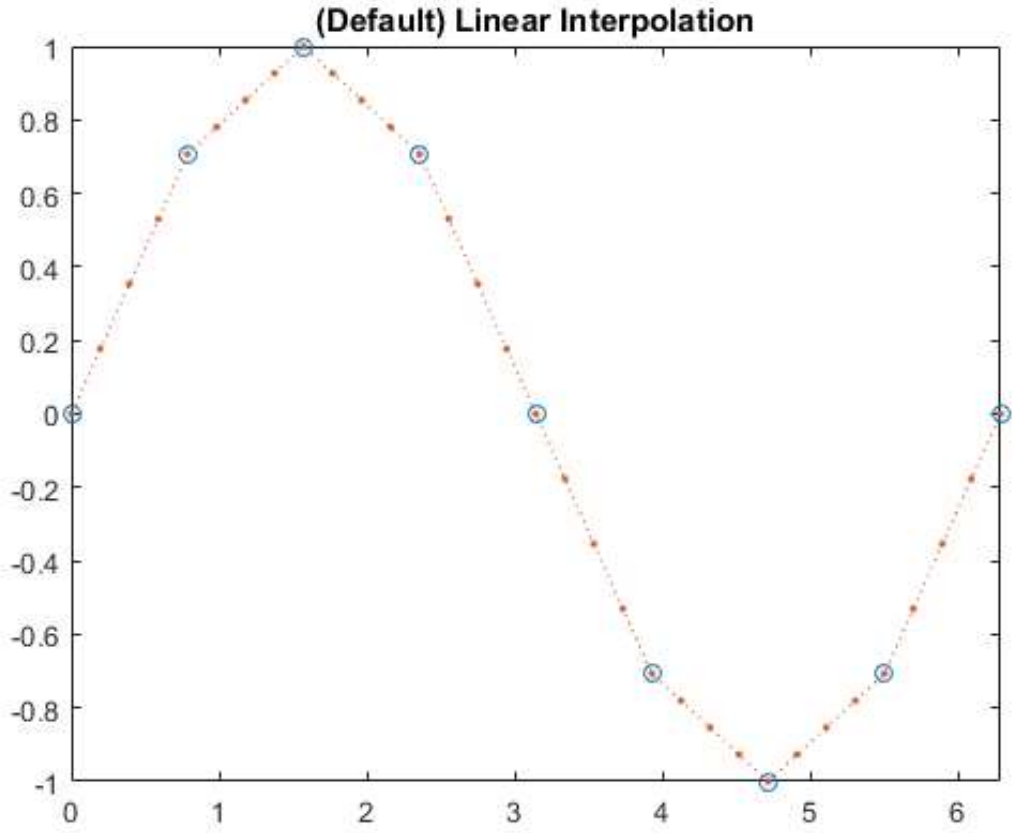
```
x = 0:pi/4:2*pi;
v = sin(x);
```

Define the query points to be a finer sampling over the range of `x`.

```
xq = 0:pi/16:2*pi;
```

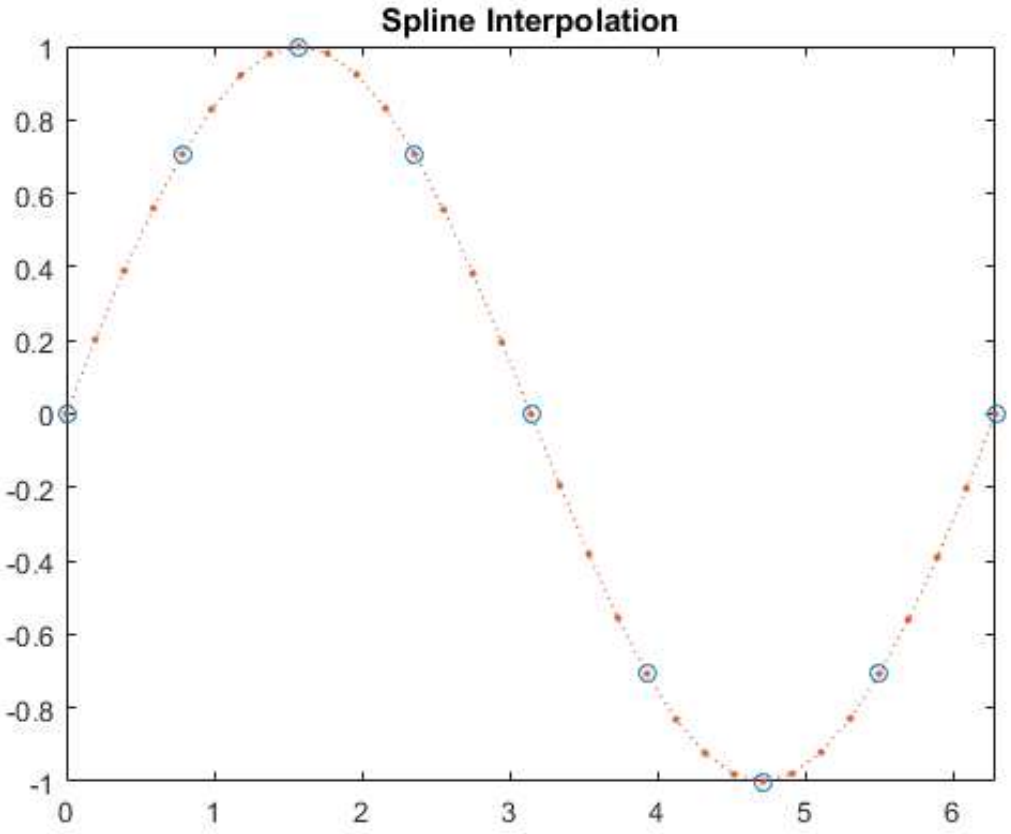
Interpolate the function at the query points and plot the result.

```
figure
vq1 = interp1(x,v,xq);
plot(x,v,'o',xq,vq1,':.');
xlim([0 2*pi]);
title('(Default) Linear Interpolation');
```



Now evaluate `v` at the same points using the `'spline'` method.

```
figure
vq2 = interp1(x,v,xq,'spline');
plot(x,v,'o',xq,vq2,':.');
xlim([0 2*pi]);
title('Spline Interpolation');
```



Interpolation Without Specifying Points

Define a set of function values.

Open This Example

```
v = [0  1.41  2  1.41  0  -1.41  -2  -1.41 0];
```

Define a set of query points that fall between the default points, `1:9`. In this case, the default points are `1:9` because `v` contains `9` values.

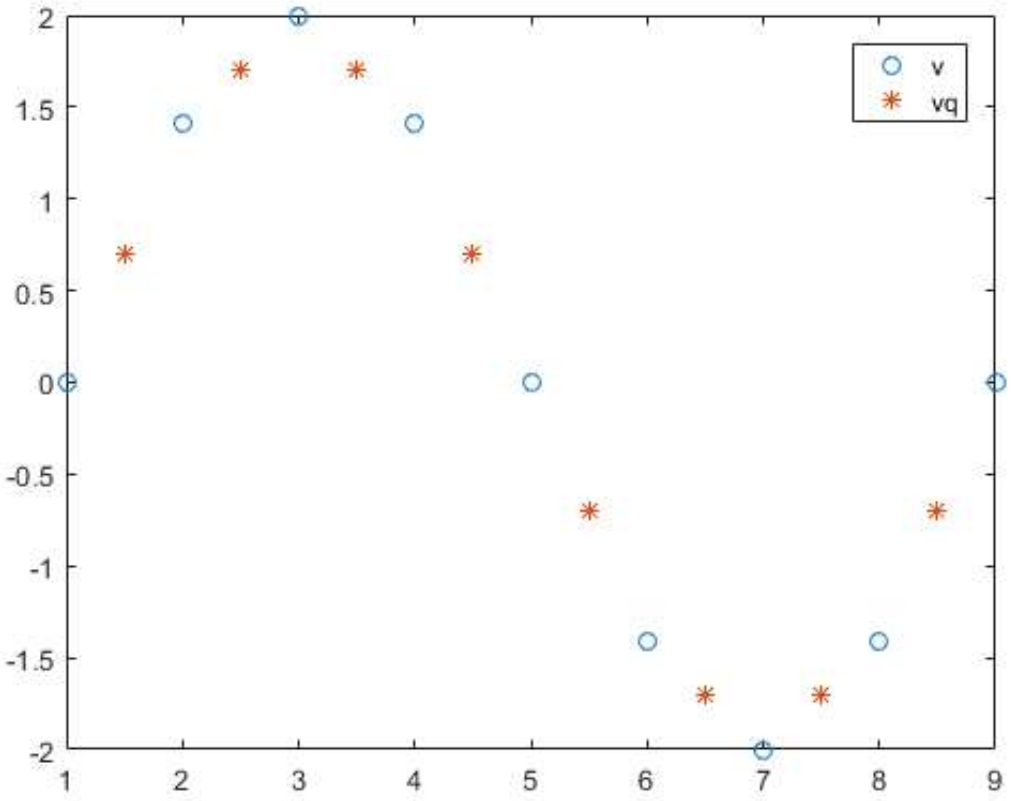
```
xq = 1.5:8.5;
```

Evaluate `v` at `xq`.

```
vq = interp1(v,xq);
```

Plot the result.

```
figure
plot((1:9),v,'o',xq,vq,'*');
legend('v','vq');
```



Interpolation of Complex Values

Define a set of sample points.

Open This Example

```
x = 1:10;
```

Define the values of the function, $v(x) = 5x + x^2 i$, at the sample points.

```
v = (5*x)+(x.^2*1i);
```

Define the query points to be a finer sampling over the range of x.

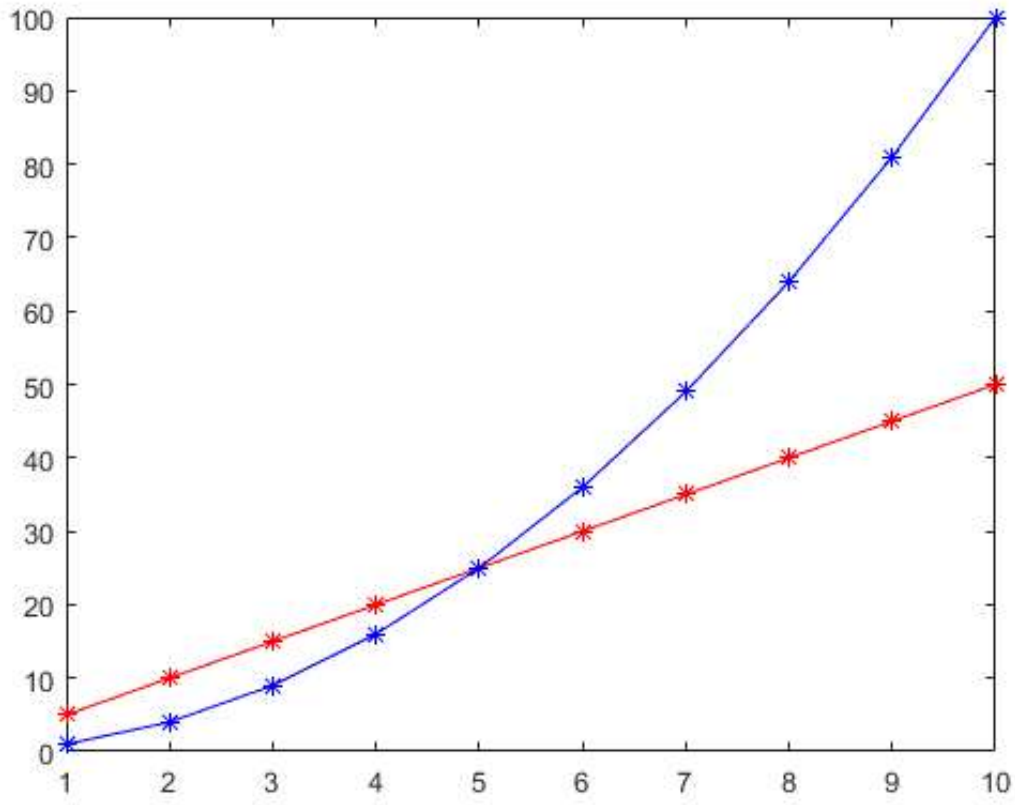
```
xq = 1:0.25:10;
```

Interpolate v at the query points.

```
vq = interp1(x,v,xq);
```

Plot the real part of the result in red and the imaginary part in blue.

```
figure
plot(x,real(v),'*r',xq,real(vq),'-r');
hold on
plot(x,imag(v),'*b',xq,imag(vq),'-b');
```



Extrapolation Using Two Different Methods

Define the sample points, x, and corresponding sample values, v.

Open This Example

```
x = [1 2 3 4 5];
v = [12 16 31 10 6];
```

Specify the query points, xq, that extend beyond the domain of x.

```
xq = [0 0.5 1.5 5.5 6];
```

Evaluate v at xq using the 'pchip' method.

```
vq1 = interp1(x,v,xq,'pchip')
```

```
vq1 =

   19.3684   13.6316   13.2105    7.4800   12.5600
```

Next, evaluate v at xq using the 'linear' method.

```
vq2 = interp1(x,v,xq,'linear')
```

```
vq2 =

   NaN   NaN    14   NaN   NaN
```

Now, use the `'linear'` method with the `'extrap'` option.

```
vq3 = interp1(x,v,xq,'linear','extrap')
```

```
vq3 =

     8    10    14     4     2
```

`'pchip'` extrapolates by default, but `'linear'` does not.

Designate Constant Value for All Queries Outside the Domain of x

Define the sample points, `x`, and corresponding sample values, `v`.

Open This Example

```
x = [-3 -2 -1 0 1 2 3];
v = 3*x.^2;
```

Specify the query points, `xq`, that extend beyond the domain of `x`.

```
xq = [-4 -2.5 -0.5 0.5 2.5 4];
```

Now evaluate `v` at `xq` using the `'pchip'` method and assign any values outside the domain of `x` to the value, `27`.

```
vq = interp1(x,v,xq,'pchip',27)
```

```
vq =

   27.0000   18.6562    0.9375    0.9375   18.6562   27.0000
```

Interpolate Multiple Sets of Data in One Pass

Define the sample points.

Open This Example

```
x = (-5:5)';
```

Sample three different parabolic functions at the points defined in `x`.

```
v1 = x.^2;
v2 = 2*x.^2 + 2;
v3 = 3*x.^2 + 4;
```

Create matrix `v`, whose columns are the vectors, `v1`, `v2`, and `v3`.

```
v = [v1 v2 v3];
```

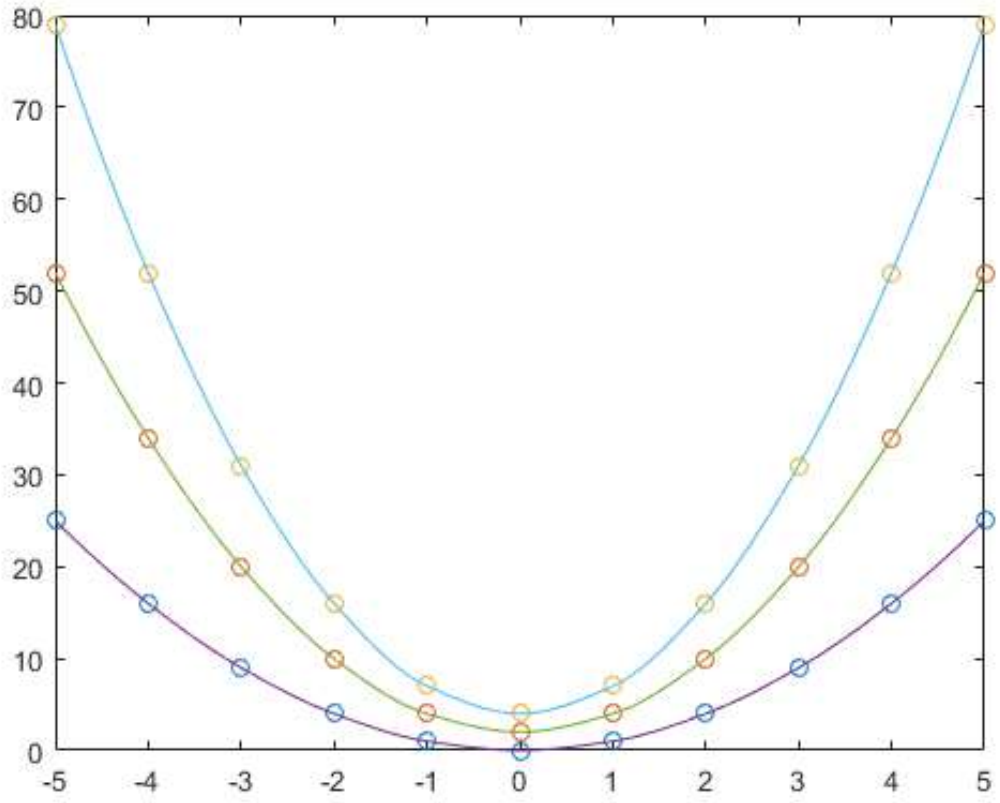
Define a set of query points, `xq`, to be a finer sampling over the range of `x`.

```
xq = -5:0.1:5;
```

Evaluate all three functions at `xq` and plot the results.

```
vq = interp1(x,v,xq,'pchip');
figure
plot(x,v,'o',xq,vq);

h = gca;
h.XTick = -5:5;
```



The circles in the plot represent v, and the solid lines represent vq.

Input Arguments

collapse all

x — Sample points
vector

Sample points, specified as a row or column vector of real numbers. The values in x must be distinct. The length of x must conform to one of the following requirements:

- If v is a vector, then `length(x)` must equal `length(v)`.
- If v is an array, then `length(x)` must equal `size(v,1)`.

Example: `[1 2 3 4 5 6 7 8 9 10]`

Example: `1:10`

Example: `[3 7 11 15 19 23 27 31]'`

Data Types: `single` | `double` | `duration` | `datetime`

`v` — Sample values
vector | matrix | array

Sample values, specified as a vector, matrix, or array of real or complex numbers. If `v` is a matrix or an array, then each column contains a separate set of 1-D values.

Example: `rand(1,10)`

Example: `rand(10,1)`

Example: `rand(10,3)`

Data Types: `single` | `double` | `duration` | `datetime`
Complex Number Support: Yes

`xq` — Query points
scalar | vector | matrix | array

Query points, specified as a scalar, vector, matrix, or array of real numbers.

Example: `5`

Example: `1:0.05:10`

Example: `(1:0.05:10)'`

Example: `[0 1 2 7.5 10]`

Data Types: `single` | `double` | `duration` | `datetime`

`method` — Interpolation method
`'linear'` (default) | `'nearest'` | `'next'` | `'previous'` | `'spline'` | `'pchip'` | `'cubic'`

Interpolation method, specified as a value from the table below.

Method	Description	Continuity	Comments
`'linear'`	Linear interpolation. The interpolated value at a query point is based on linear interpolation of the values at neighboring grid points in each respective dimension. This is the default interpolation method.	C^0	• Requires at least 2 points. • Requires more memory and computation time than nearest neighbor.
`'nearest'`	Nearest neighbor interpolation. The interpolated value at a query point is the value at the nearest sample grid point.	Discontinuous	• Requires at least 2 points. • Modest memory requirements • Fastest computation time
`'next'`	Next neighbor interpolation. The interpolated value at a query point is the value at the next sample grid point.	Discontinuous	• Requires at least 2 points. • Same memory requirements and computation time as `'nearest'`.
`'previous'`	Previous neighbor interpolation. The interpolated value at a query point is the value at the previous sample grid point.	Discontinuous	• Requires at least 2 points. • Same memory requirements and computation time as `'nearest'`.

`'pchip'`	Shape-preserving piecewise cubic interpolation. The interpolated value at a query point is based on a shape-preserving piecewise cubic interpolation of the values at neighboring grid points.	C^1	• Requires at least 4 points. • Requires more memory and computation time than linear.
`'cubic'`	Same as `'pchip'`.	C^1	This method currently returns the same result as `'pchip'`. In a future release, this method will perform cubic convolution.
`'v5cubic'`	Cubic convolution used in MATLAB® 5.	C^1	Points must be uniformly spaced. `'cubic'` will replace `'v5cubic'` in a future release.
`'spline'`	Spline interpolation using not-a-knot end conditions. The interpolated value at a query point is based on a cubic interpolation of the values at neighboring grid points in each respective dimension.	C^2	• Requires at least 4 points. • Requires more memory and computation time than `'pchip'`.

`extrapolation` — Extrapolation strategy
`'extrap'` | scalar value

Extrapolation strategy, specified as `'extrap'` or a real scalar value.

- Specify `'extrap'` when you want `interp1` to evaluate points outside the domain using the same method it uses for interpolation.
- Specify a scalar value when you want `interp1` to return a specific constant value for points outside the domain.

The default behavior depends on the input arguments:

- If you specify the `'pchip'` or `'spline'` interpolation methods, then the default behavior is `'extrap'`.
- All other interpolation methods return `NaN` by default for query points outside the domain.

Example: `'extrap'`

Example: `5`

Data Types: `char` | `single` | `double`

Output Arguments

collapse all

`vq` — Interpolated values
scalar | vector | matrix | array

Interpolated values, returned as a scalar, vector, matrix, or array. The size of `vq` depends on the shape of `v` and `xq`.

Shape of v	Shape of xq	Size of Vq	Example
Vector	Vector	`size(xq)`	If `size(v) = [1 100]` and `size(xq) = [1 500]`, then `size(vq) = [1 500]`.
Vector	Matrix or N-D Array	`size(xq)`	If `size(v) = [1 100]` and `size(xq) = [50 30]`,

			then `size(vq) = [50 30]`.
Matrix or N-D Array	Vector	`[length(xq)` `size(v,2),...,size(v,n)]`	If `size(v) = [100 3]` and `size(xq) = [1 500]`, then `size(vq) = [500 3]`.
Matrix or N-D Array	Matrix or N-D Array	`[size(xq,1),...,size(xq,n),...` `size(v,2),...,size(v,m)]`	If `size(v) = [4 5 6]` and `size(xq) = [2 3 7]`, then `size(vq) = [2 3 7 5 6]`.

`pp` — Piecewise polynomial
structure

Piecewise polynomial, returned as a structure that you can pass to the `ppval` function for evaluation.

See Also

`griddedInterpolant` | `interp2` | `interp3` | `interpn`

Introduced before R2006a