```
>> type interp1

function vq = interp1(x,v,xq,method,extrapVal) %#codegen
%INTERP1 Interpolacja 1-D na datetimes (wyszukiwanie w tabeli)
% Copyright 2020 The MathWorks, Inc.
narginchk(3,5); % interp1(V,Xq) nie jest obsługiwane
if nargin < 4
   method = 'linear';
elseif isa(method,'datetime')
   % Pozwól interp1 obsłużyć często popełniany błąd, interp1(X,V,Xq,EXTRAPVAL)
   method = char(method);
end
needMeanCenter = ~any(strcmp(method, {'previous' 'next' 'nearest'}));
% Jeśli X i Xq są datetimes, przekonwertuj je na numeryczne po upewnieniu się, że są
% zgodne. Obie mogą być ciągami datetime, jeśli druga z nich jest datetimes.
if isa(x,'datetime') || isa(xq,'datetime')
   [xProcessed,xqProcessed] = datetime.compareUtil(x,xq);


   % Jeśli X lub Xq mają część niskiego rzędu, lepiej jest wyśrodkować, ponieważ
   % część wysokiego rzędu nie ma wystarczającej precyzji, aby wykonać
   % dokładnej interpolacji. Jeśli nie mają one części niskiego rzędu, nie jest
   % konieczne jest średnie wyśrodkowanie dla metod "następna", "najbliższa" lub "poprzednia", ponieważ operacja
średniego wyśrodkowania nie jest konieczna.
   % ponieważ operacja średniego centrowania spowoduje zaokrąglenie.
   haveLowOrderX = ~isreal(xProcessed) && nnz(imag(xProcessed)) > 0;
   haveLowOrderXq = ~isreal(xqProcessed) && nnz(imag(xqProcessed)) > 0;
   if (needMeanCenter || haveLowOrderX || haveLowOrderXq)
       % Przekształć na podwójną precyzję przesunięcia od średniej lokalizacji x
       [xData,xqData] = dd2d(xProcessed,xqProcessed);
   else
       xDData = real(xProcessed);
       xqData = real(xqProcessed);
   end
else
   xData = x;
```

```
   xqData = xq;
end
% Jeśli V (i extrapVal, jeśli jest podana jako wartość) są datetimes, należy je przekonwertować na
% numeryczne po upewnieniu się, że są zgodne. Oba mogą być datetime
% jedno z nich jest datetimes, to drugie może być (=stringiem) ciągiem znaków.
timey = isa(v,'datetime') || (nargin > 4 && isa(extrapVal,'datetime'));
if timey
   if nargin < 5 || strcmpi(extrapVal, "extrap")
       vqDT = matlab.internal.coder.datetime(matlab.internal.coder.datatypes.uninitialized); % użyj v jako
szablonu dla danych wyjściowych
       vqDT.tz = v.tz;
       vqDT.fmt = v.fmt;
       if (needMeanCenter)
           v0 = matlab.internal.coder.datetime.datetimeMean(v.data(:),1,false); % v0 = mean(v(:),'includenan')
           % Zaokrąglamy średnią do najbliższej liczby całkowitej, aby uniknąć wprowadzenia nowych
           % błędów zaokrąglenia do wartości wyśrodkowanych.
           v0 = round(real(v0));
           vData = matlab.internal.coder.doubledouble.minus(v.data,v0,true); % v = milisekundy(v - v0), pełna
precyzja
       else
           vData = v.data;
       end
       haveLowOrder = ~isreal(vData) && nnz(imag(vData)) > 0;
       vHi = real(vData);
       vLow = imag(vData);
       if nargin > 4
           extrapValHi = extrapVal;
           extrapValLow = 0;
       end

       if haveLowOrder
           vLow = imag(vData);
           extrapValLow = 0; % Ekstrapoluj używając zera dla interp1(...,'extrap')
       end
   else
```

```
        [vData,extrapValData,vqDT] = datetime.compareUtil(v,extrapVal);

        if (needMeanCenter)
            v0 = matlab.internal.coder.datetime.datetimeMean(vData(:),1,false); % v0 = mean(v(:),'includenan')
            % Zaokrąglamy średnią do najbliższej liczby całkowitej, aby uniknąć wprowadzenia nowych
            % błędów zaokrąglenia do wartości wyśrodkowanych.
            v0 = round(real(v0));
            vData = matlab.internal.coder.doubledouble.minus(vData,v0,true); % v = milisekundy(v - v0), pełna
precyzja
            extrapValData = matlab.internal.coder.doubledouble.minus(extrapValData,v0,true); % extrapVal =
milisekundy(extrapVal - v0)
        end

        haveLowOrder = (~isreal(vData) && nnz(imag(vData)) > 0) ...
            | (~isreal(extrapValData) && nnz(imag(extrapValData))) > 0);
        vHi = real(vData);
        vLow = imag(vData);
        extrapValHi = real(extrapValData);
        extrapValLow = imag(extrapValData);
        if haveLowOrder
            vLow = imag(vData);
            extrapValLow = imag(extrapValData);
        end
    end
else
    if isa(v,'duration')
        vHi = v;
    else
        vHi = real(v);
        vLow = imag(v);
    end
    if nargin > 4
        extrapValHi = extrapVal;
        extrapValLow = 0;
    end
end
```

```
% Wykonaj interpolację na (numerycznych) ms od 1970 roku. Jeśli istnieje część niskiego rzędu,
% wykonaj interpolację na niej osobno, aby później ją dodać. Dzięki temu
% zapytanie o oryginalne dane x zwraca dokładnie te same oryginalne dane v.
if nargin < 5
   vqHi = interp1(xData,vHi,xqData,metoda);
   if timey && haveLowOrder
       % Ekstrapoluj część niskiego rzędu, używając zera dla interp1(x,v,xq,method),
       % w przeciwnym razie skończyłoby się to jako NaN dla 'linear' i innych.
       vqLow = interp1(xData,vLow,xqData,metoda,0);
   else
       vqLow = imag(vqHi);
   end
else % interp1(...,'extrap') lub interp1(...,extrapVal)
   vqHi = interp1(xData,vHi,xqData,method,extrapValHi);
   if timey && haveLowOrder
       vqLow = interp1(xData,vLow,xqData,method,extrapValLow);
   else
       if isa(vqHi,'duration')
           vqLow = 0;
       else
           vqLow = imag(vqHi);
       end
   end
end
% Konwertowanie danych wyjściowych na czas przeszły.
if timey
   if haveLowOrder
       vqData = matlab.internal.coder.doubledouble.plus(vqHi,vqLow); % vq = milliseconds(vq + vqLow)
   else
       vqData = vqHi;
   end
   % Dodaj z powrotem datetime "origin"
   if (needMeanCenter)
       vqDT.data = matlab.internal.coder.doubledouble.plus(v0,vqData); % vq = v0 + milisekundy(vq)
   else
       vqDT.data = vqData;
```

```
    end
    vq = vqDT;

else
    vq = vqHi;
end
%-------------------------------------------------------------------------
funkcja [xc,xqc] = dd2d(x,xq)
% Konwertuj wartości podwójne na podwójne przesunięcia od średniej.
x0 = matlab.internal.coder.datetime.datetimeMean(x(:),1,false); % x0 = mean(x,'includenan')
% Zaokrąglamy średnią do najbliższej liczby całkowitej, aby uniknąć wprowadzenia nowych
% błędów zaokrąglenia do wartości wyśrodkowanych.
x0 = round(real(x0));
xc = matlab.internal.coder.doubledouble.minus(x,x0,false); % x = milisekundy(x - x0)
xqc = matlab.internal.coder.doubledouble.minus(xq,x0,false); % xq = milisekundy(xq - x0)
% Wartości double-double x, które były różne, mogły stać się identycznymi wartościami double
% precyzji xc jako artefakt konwersji. Nie można po prostu usunąć
% duplikatów, ponieważ mogą one mieć różne wartości y.
if any((diff(xc(:)) == 0) & (diff(x(:)) ~= 0)) % wywołuje złożoną funkcję diff, to jest OK
    xdt = datetime.fromMillis(x(:));
    range = max(xdt) - min(xdt);
    minDiff = min(diff(xdt));
    rangeStr = sprintf('%.5g',seconds(range));
    minDiffStr = sprintf('%.5g',seconds(minDiff));
    coder.internal.error('MATLAB:datetime:interp1:GridPointMinimumDifference',rangeStr,minDiffStr);
end
>>
```