# MATLAB Programs for Converting Thermistor and Thermocouple Measurements to Temperature

Gerald Recktenwald*

April 7, 2006

## 1 Overview

This document describes MATLAB functions for converting thermocouple and thermistor measurements to temperature. The codes can be downloaded from the web site for ME 449, *Thermal Management Measurements*: `www.me.pdx.edu/~gerry/class/ME449/codes/`.

## 2 Thermistors

The temperature versus resistance data for thermistors can be fit with

$$T = \frac{1}{c_1 + c_2 \ln R + c_3 (\ln R)^3} \tag{1}$$

where $R$ is the resistance of the thermistor, and $T$ is the absolute temperature. A least squares fit of Equation (1) to the calibration data for YSI model 44006 thermistors in the range $0 \leq T \leq 50°\text{C}$ gives[1]

$$\begin{aligned} c_1 &= 1.025\ 227\ 462\ 259\ 867 \times 10^{-3} \\ c_2 &= 2.397\ 895\ 314\ 112\ 997 \times 10^{-4} \\ c_3 &= 1.539\ 983\ 937\ 555\ 444 \times 10^{-7} \end{aligned} \tag{2}$$

where $R$ is in ohms and $T$ in kelvin. Data for the curve fit was downloaded from the YSI web site, `www.ysitemperature.com`.

The `thermistorT` function in Listing 1 evaluates Equation (1) for resistance values in the range $3890 \leq R \leq 28490\,\Omega$. The input to `thermistorT` is the resistance in $\Omega$, and the output is the temperature in °C. For example, if the resistance is $12000\,\Omega$, the temperature of the thermistor is

```
>> T = thermistorT(12000)
T =
   20.5272
```

*Mechanical and Materials Engineering Department, Portland State University, Portland, Oregon, `gerry@me.pdx.edu`

[1]Spaces between digits are intended to aid in transcription.

```
function T = thermistorT(r)
% thermistorT  Temperature of YSI model 44006 thermistor as function of resistance.
%
% Synopsis:  T = thermistorT(r)
%
% Input:  R = resistance in ohms
%
% Output:  T = temperature in C
%
% Note:  Curve fit T = f(r)  (T in C, r in ohms) is valid only for 0 <= T <= 50,
%        which corresponds to 29490 >= r >= 3893

% --- Range check:  correlation is valid only for 3890 <= r <= 29490
ibad = [find(r<3890), find(r>29490)];
if ~isempty(ibad)
  error(sprintf('r = %g is out of range:  3890 <= r <= 29490 Ohm\n',r(ibad)));
end
c = [ 1.025227462259867e-03  2.397895314112997e-04  1.539983937555444e-07];
lnr = log(r);
T = 1./(c(1) + c(2)*lnr + c(3)*lnr.^3) - 273.15;
```

Listing 1: The `thermistorT` function converts thermistor resistance to temperature

Note that `thermistorT` is vectorized.

```
>> R = 11000:1000:14000
R =
       11000       12000       13000       14000

>> T = thermistorT(R)
T =
   22.6449   20.5272   18.6008   16.8354
```

# 3 Thermocouples

Table 1 lists MATLAB m-files for thermocouple data conversion. The table identifies scalar and vectorized routines. Vectorized routines allow the input arguments to be vectors instead of scalars, which provides a convenient way to convert multiple thermocouples with the same reference junction. Unfortunately, the vectorized code can be hard to read for individuals not familiar with MATLAB. To keep this presentation simple, only the scalar versions of the code are discussed here. For routine use, the vectorized codes are recommended. Although the vectorized codes are a little harder to *read*, they can be *used* with the same input and output arguments as the scalar codes.

The routines in Table 1 provide conversions for J, K and T type thermocouples. The polynomials for the EMF to temperature, and temperature to EMF conversions are taken from the ASTM Thermocouple Manual [1]. Developing versions for other types of thermocouples is straightforward. Usually one only needs to change the values of the constants in the polynomial curve fits, and the limits on the acceptable ranges of input values.

## 3.1 Temperature to EMF Conversion

The EMF of thermocouples with a reference junction at 0 °C are computed with `emfJtype`, `emfKtype` and `emfTtype`. For example, the `emfJtype` function can be

Table 1: MATLAB functions for thermocouple conversion. Note that all of the routines for converting EMF to temperature allow a second, optional input to specify the temperature reference junction. The routines that convert temperature to EMF only compute the EMF for an ice-point reference temperature.

| **Scalar Only** | |
|---|---|
| `emfTtypes` | Convert temperature to the EMF of a T-type thermocouple with an ice-point reference junction |
| `Jtemps` | Convert EMF to temperature for a J-type thermocouple with an ice-point reference junction |
| `Ttemps` | Convert EMF to temperature for a T-type thermocouple with an ice-point reference junction |

| **Vectorized** | |
|---|---|
| `emfJtype` | Convert temperature to the EMF of a J-type thermocouple with an ice-point reference junction. |
| `emfKtype` | Convert temperature to the EMF of a K-type thermocouple with an ice-point reference junction. |
| `emfTtype` | Convert temperature to the EMF of a T-type thermocouple with an ice-point reference junction. |
| `Jtemp` | Convert EMF to temperature for a J-type thermocouple. |
| `Ktemp` | Convert EMF to temperature for a K-type thermocouple. |
| `Ttemp` | Convert EMF to temperature for a T-type thermocouple. |

used the EMF of a J-type, ice-point reference thermocouple with its measuring junction at 21.5 °C. Given a temperature in °C, `emfJtype` returns the EMF in volts.

```
>> Jtemp(1.4482e-3)
ans =
   28.2622
>> e = emfJtype(21.5)
e =
    0.0011
>> mv = e*1000
mv =
    1.0965
```

Multiplying the output of `emfJtype` gives the EMF in millivolt.

## 3.2 EMF to Temperature Conversion

The reverse process — converting EMF to temperature — is performed by the `Jtemp`, `Ktemp` and `Ttemp` functions. For example, to reverse the preceding calculation (using the value stored in `e` from the preceding MATLAB commands)

```
>> T = Jtemp(e)
T =
   21.4653
```

The original temperature is not obtained because of approximation errors in the polynomial curve fits, and to a lesser degree the rounding errors in the evaluations of the polynomials for $E = F(T)$ and $T = G(E)$.

```
>> T - 21.5
ans =
   -0.0347
```

This error in the round-trip calculation provides one indication on the accuracy limit for thermocouple calculations using the standard polynomials. One could obtain better results by using curve fits for limited ranges of temperature (and therefore EMF), or by calibrating the thermocouple wire against a primary or secondary standard.

Note that when using the EMF to temperature conversion functions, the EMF value must be entered in volts, not millivolts. The temperature-to-EMF and EMF-to-temperature conversion functions check the range of input arguments and stop execution if there is an error

```
>> Jtemp(1.0965e-3)
ans =
   21.4661
>> Jtemp(1.0965)
Warning: e = 1.0965e+06 value outside of range -5603 <= e <= 20872 microV

> In Jtemp>TofJtype at 54
  In Jtemp at 14
ans =
     0
>>
```

The `Jtemp`, `Ktemp` and `Ttemp` functions provide a second input for specifying the temperature of the reference junction. For example, the temperature of a J-type thermocouple with an output of 1.223 mV and having a reference junction at 15 °C is

```
>> Jtemps(1.223e-3,15)
ans =
   38.5661
```

By default, the reference junction is at 0 °C, so if only one argument is supplied to `Jtemp`, `Ktemp` or `Ttemp`, the reference junction is assumed to be at 0 °C.

## 3.3 Thermocouples with Zone Box Reference Junctions

The second input to `Jtemp`, `Ktemp` and `Ttemp` makes it easy to convert the EMF from thermocouples having reference junctions in a zone box. If the zone box temperature is measured with a YSI model 44006 thermistor, the `thermistorT` function is first used to convert the thermistor resistance to temperature. The zone box temperature is then used as the reference junction temperature for the `Jtemp`, `Ktemp` or `Ttemp` function.

Suppose the resistance of a YSI 44006 thermistor in the zone box is 11240 Ω, and that a J-type thermocouple with a reference junction in the zone box has an output of 0.7892 mV. The temperature of the thermocouple is obtained with the following statements.

```
>> Rz = 11240;
>> Tz = thermistorT(Rz)
Tz =
   22.1173
```

```
>> T = Jtemp(0.7892e-3,Tz)
T =
   37.2704
```

The thermocouple conversion calculations can also be combined into one line

```
>> Rz = 11240;
>> T = Jtemp(0.7892e-3,thermistorT(Rz))
```

## 3.4 Vectorized Calculations

The `Jtemp`, `Ktemp` and `Ttemp` functions should be used for routine thermocouple conversions. These functions perform the same conversions as `Jtemps`, `Ktemps` and `Ttemps`, but `Jtemp` and `Ttemp` accept scalar or vectors of EMF values.

Suppose that five T-type thermocouples have reference junctions in a zone box. The temperature of the zone box is measured with a YSI 44006 thermistor, and the output of the thermistor is 11075 Ω. The EMFs of the five thermocouples are 1.070, 1.899, −0.022, 0.0809, and 0.7702 mV. The conversion calculations are as follows.

```
>> Rz = 11075;
>> Tz = thermistorT(Rz)
Tz =
   22.4787

>> e = [1.070, 1.899, -0.022, 0.0809, 0.7702] * 1e-3;
>> T = Ttemp(emf,Tz)
T =
   48.2093   67.2767   21.9537   24.4893   41.1388
```

The thermocouple conversion calculations can be combined into one line

```
>> e = [1.070, 1.899, -0.022, 0.0809, 0.7702] * 1e-3;
>> T = Ttemp(e,thermistorT(11075))
T =
   48.2093   67.2767   21.9537   24.4893   41.1388
```

# References

[1] American Society for Testing and Materials. *A Manual on the Use of Thermocouples in Temperature Measurement*. ASTM, Philadelphia, New York, fourth edition, 1993.

```
function e = emfJtype(T)
% emfJtype  EMF of J type thermocouple as function of temperature
%
% Synopsis:  e = emfJtype(T)
%
% Input:  T = temperature, degrees C
%
% Output: e = thermocouple emf in Volts

% --- Range check:  correlation is valid only for -210 <= T < 760
ibad = [find(T<-210), find(T>760)];
if ~isempty(ibad)
  error(sprintf('T = %g is out of range:  -210 <= T <= 760 C\n',T(ibad)));
end
% --- Assign constants and calculate emf in volts
c = [1.5631725697e-20  -1.2538395336e-16  2.0948090697e-13   -1.7052958337e-10 ...
     1.3228095295e-7  -8.5681065720e-5   3.0475836930e-2  50.381187815  0];
e = polyval(c,T)*1e-6;
```

Listing 2: The `emfJtype` converts temperature to EMF for a J-type thermocouple with an ice-point reference junction. The input can be a scalar or vector of $T$ values.

```
function e = emfTtypes(T)
% emfTtypes  EMF of T type thermocouple as function of temperature; scalar version
%
% Synopsis:  e = emfTtypes(T)
%
% Input:  T = temperature, degrees C,  -200 <= T <= 400 C
%
% Output: e = thermocouple emf in Volts

% --- Assign constants based on value of T
if 0<=T && T<=400
  c = [-2.7512901673e-17  4.5479135290e-14  -3.0815758772e-11   1.0996880928e-8 ...
       -2.1882256846e-6   2.0618243404e-4    3.3292227880e-2   38.748106364  0];
elseif -270<=T && T<0
  c = [ 7.9795153927e-28  1.3945027062e-24  1.0795539270e-21  4.8768662286e-19 ...
        1.4251594779e-16  2.8213521925e-14  3.8493939883e-12  3.6071154205e-10 ...
        2.2651156593e-8   9.0138019559e-7   2.0032973554e-5   1.1844323105e-4 ...
        4.1944343347e-2  38.748106364  0];
else
  error(sprintf('T = %g value outside of range -270 <= T <= 400 C\n',T));
end
e = polyval(c,T)*1e-6;   %  compute emf and convert microvolts to volts
```

Listing 3: The `emfTtypes` converts temperature to EMF for a T-type thermocouple with an ice-point reference junction. This is a scalar function, meaning that input values of EMF must be a scalar, not a vector.

```
function T = Jtemps(e,Tref)
% Jtemps  Temperature of J-type thermocouple;  scalar version
%
% Synopsis:  T = Jtemps(e)
%            T = Jtemps(e,Tref)
%
% Input:  e = (scalar) emf of thermocouple junction in Volts
%         Tref = (optional) temperature of reference junction
%                Default:  Tref = 0 C (ice point)
%
% Output:  T = temperature, degrees C

if length(e)>1
  warning('e must be a scalar:  only e(1) is converted');
end
if nargin<2
  T = TofJtypes(e(1));  return;
elseif nargin==2
  eref = emfJtype(Tref);     %  emf of ice-point referenced T-couple at Tref
  T = TofJtypes(e(1)+eref);  %  adjust for nonzero reference temperature
  return;
else
  error('only one or two inputs allowed');
end

% ============================================================
function T = TofJtypes(e)
% TofJtypes  Temperature of J type thermocouple with ice-point compensation
%            Scalar version:  input e must be a scalar
%
% Synopsis:  T = TofJtypes(e)
%
% Input:   e = (vector or scalar) emf of thermocouple junction in Volts
%              T-couple is assumed to have a reference junction at 0 C
%
% Output:  T = temperature, degrees C

em = e*1e6;     %  emf in microvolts, must be a scalar
% --- Polynomial constants are determined by value of em
if em>=-8095 && em<=0
  c = [-8.3823321e-29  -2.3963370e-24  -2.8131513e-20  -1.7256713e-16 ...
       -5.9086933e-13  -1.0752178e-9  -1.2286185e-6  1.9528268e-2  0];
elseif em>=0 && em<=42919
  c = [5.099890e-31  -5.344285e-26  3.585153e-21  -2.549687e-16  1.036969e-11 ...
       -2.001204e-7  1.978425e-2  0];
else
  error(sprintf('e = %g value outside of range -5603 <= e <= 20872 microV\n',em));
end
T = polyval(c,em);
```

Listing 4: The `Jtemps` converts EMF to temperature for a J-type thermocouple with an ice-point reference junction. This is a scalar function, meaning that input values of EMF must be a scalar, not a vector.

```
function T = Ttemps(e,Tref)
% Ttemps  Temperature of T-type thermocouple;  scalar version
%
% Synopsis:  T = Ttemps(e)
%            T = Ttemps(e,Tref)
%
% Input:  e = (scalar) emf of thermocouple junction in Volts
%         Tref = (optional) temperature of reference junction
%                Default:  Tref = 0 C (ice point)
%
% Output:  T = temperature, degrees C

if length(e)>1
  warning('e must be a scalar:  only e(1) is converted');
end
if nargin<2
  T = TofTtypes(e(1));  return;
elseif nargin==2
  eref = emfTtypes(Tref);    %  emf of ice-point referenced T-couple at Tref
  T = TofTtypes(e(1)+eref);  %  adjust for nonzero reference temperature
  return;
else
  error('only one or two inputs allowed');
end

% ============================================================
function T = TofTtypes(e)
% TofTtypes  Temperature of T type thermocouple with ice-point compensation
%            Scalar version:  input e must be a scalar
%
% Synopsis:  T = TofTtypes(e)
%
% Input:   e = (vector or scalar) emf of thermocouple junction in Volts
%              T-couple is assumed to have a reference junction at 0 C
%
% Output:  T = temperature, degrees C

em = e*1e3;   %  emf in mV, must be a scalar
% --- Polynomial constants are determined by value of em
if 0<=em && em<=20.872
  c = [-7.293422e-7  6.048144e-5  -2.165394e-3  4.637791e-2 ...
       -0.7602961 25.92800  0];
elseif -5.603<=em && em<0
  c = [1.2668171e-3   2.0241446e-2  0.13304473  0.42527777 ...
       0.79018692  -0.21316967  25.949192 0];
else
  error(sprintf('e = %g value outside of range -5.603 <= e <= 20.872 mV\n',em));
end
T = polyval(c,em);
```

Listing 5: The `Ttemps` converts EMF to temperature for a T-type thermocouple with an ice-point reference junction. This is a scalar function, meaning that input values of EMF must be a scalar, not a vector.