Mathematica and the Glass Tempering Process

Jennifer Voitle

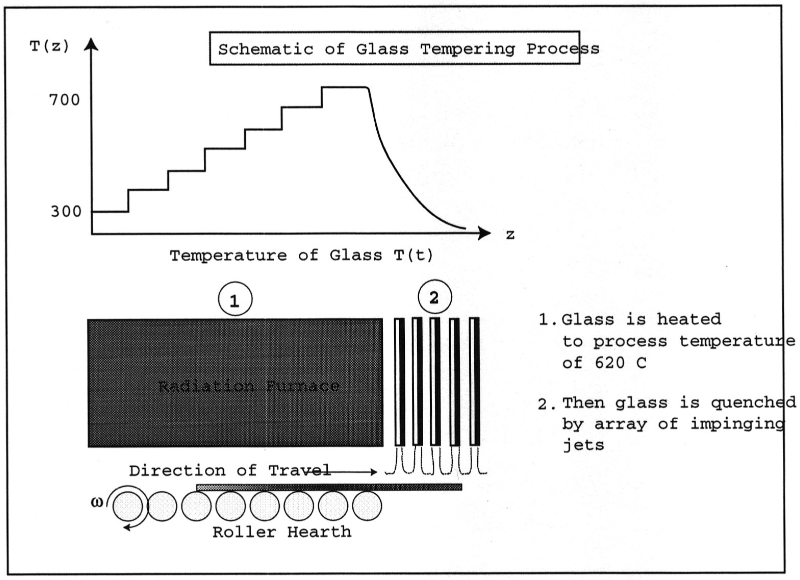
Introduction

Mathematica is being used in industry to solve difficult and highly complex problems, formerly insoluble or requiring the power of at least a mini-computer. With *Mathematica*, solutions to these problems are possible in a much shorter time-frame. Programs that in the past required months or years of effort are now capable of development in days or weeks. *Mathematica* is a valuable and indispensable tool for many reasons, not the least of which are the complete, powerful and self-contained programming environment; *Mathematica* notebooks which permit us to combine dynamic calculations, graphics, on-line help and examples, and provide a common environment in which non-specialists may use highly specialized technical results as long as they understand *Mathematica*; and the ease of interfacing *Mathematica* with a variety of other programs. In the following, two industrial applications in which the author has successfully applied *Mathematica* are described.

This notebook also includes examples of how *Mathematica* can be used with programs such as Microsoft Word (for equation writing); Adobe Illustrator and Aldus Freehand (for high-quality graphics production); Sigma Plot (for scientific plotting); and Microsoft Excel (for import/export of *Mathematica* data, and interface with *MathLink*.) Additionally, the author has combined *Mathematica* with LabTech Notebook, a data acquisition program, and Visual BASIC.

Description of the Tempering Process

Tempered (strengthened) glass for automotive and architectural uses (such as back and side-lites, patio doors and the like) is heated by radiation (or forced convection, or radiation and forced convection), to the processing temperature, then strengthened by quenching to handling temperature.

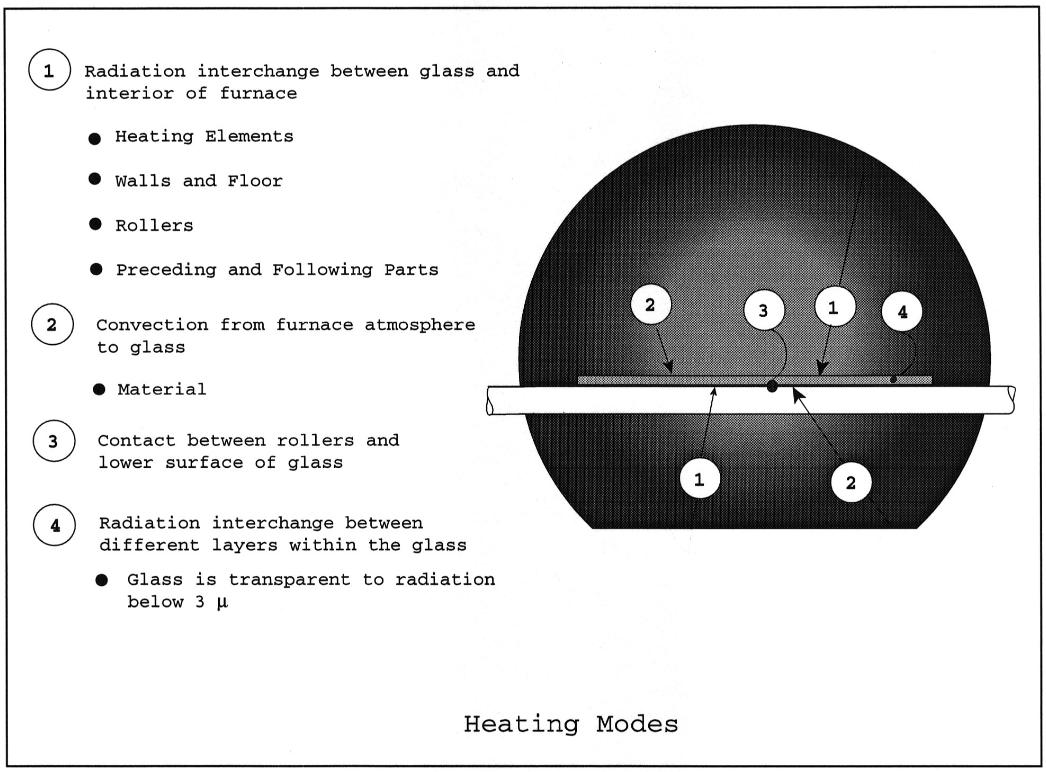


NOTE: above figure imported from Adobe Illustrator.

The Problem

We need to know the temperature-time and stress-time response of an arbitrary piece of glass under various operating conditions. We need to be able to predict the cooling power (convection heat transfer coefficient) required for "full temper", as well as the thermal response imposed by the furnace.

Description of Heat Transfer in Radiation Furnace



Mathematica is used in two major areas:
1) As a program for computing transient temperature and stress distributions in glass;
2) In prediction of the forced convection heat transfer coefficient in the quenching process.

Both of the above processes are summarized in the following.
Prior to u*sing Mathematica*, the calculations in (1) were carried out via a Unix workstation FORTRAN program, which has been converted *into Mathematica* for Macintosh and IBM, and is being extended to contain 2-D capability, general boundary conditions and improved convergence capability. The forced convection coefficient of (2) is empirically determined in an experimental setting w*here Mathematica* is used to compute and map temperature and convection profiles. Soon*, Mathematica*may be used in our data acquisition process as well (rather than LabTech Notebook.)

Advantages of using *Mathematica* in these applications include:

i. The *Mathematica* program runs on personal computers; it is friendly, documented and contains on-line help, program description and theory; built-in examples; definition of all required symbols (using **symbol::usage =" "** structure); is easily extensible and understandable; can interact with Excel and other prog*rams via* MathLink; has animation capability; can be dynamically inspected and modified during runtime via **Enter Dialog**.

ii. Results obtained from *Mathematica* enable us to better understand glass thermal history; predict parameters leading to successful tempering by better understanding the heat transfer process; design new processes.

iii. Many of these calculations would have been difficult if not impossible to carry out in anything but *Mathematica*, where they can usually be done in a line or so.

iv. These problems are typical of what one might face in industry, where some programming must be done to solve problems which have not been previously solved.

Calculation of Temperature Profiles in Glass

In analyzing the radiant heating of glass, we consider a one-dimensional transient model. The glass sheet is subdivided into *n* layers. An energy balance on a typical layer is:

$$\dot{E}_{in} + \dot{E}_{g} = \dot{E}_{st} + \dot{E}_{out}$$

where:

Ein is the energy in to the layer, which is the sum of radiation by reflection and absorption (Qr, Qa); and the energy conducted from a neighboring layer (Qc).

Eout is the energy out of the slab, which occurs by emission.

Est is the rate of change of storage of internal energy.

Eg is the rate of internal energy generation (0 here). Then the energy balance becomes:

The temperatures are updated from previous values in accordance with the scheme

$$2n^2(Q_a - Q_e + Q_r) + kQ_c = \rho c_p \frac{dT}{dt}$$

In the above equation, the value of dT/dt is computed from the energy balance. The temperatures are stored as a list **Temperatures** and the above calculation may be implemented as:

$$T_{i+1} = T_i + \frac{dT}{dt}\Delta t$$

```
newtemps  =   Drop[Temperatures,-1] +
    Actist dTdt;
Temperatures =
    AppendTo[newtemps,newtemps[[Nd2]] ];
```

Also, n is the glass index of refraction; Qa, Qe and Qr are radiation functions computed as:

```
 Qa =
     Sum[Gamm[ktlist[[#]],j] Wh[j] zeta[ktlist[[#]],j,#],
```

```
        {j,1,mm1}]& /@ Range[Nh1];
   Teab = Te + 273;
Qe =
     2 Sum[Gamm[ktlist[[#]],j] Wbb[j,Teab[[#]] ],
       {j,1,mm1}]& /@ Range[Nh1];

Klist = Take[Transpose[{Teab,RotateLeft[Teab]}],Nh1-1];

Qr = Sum[Gamm[ktlist[[#]],j]*
      Table[psi[ktlist[[#]],j,#,i],{i,Nh1-1}].
Apply[Plus,Wbb[j,Klist],1],
              {j,1,mm1}]& /@ Range[Nh1];

fact =  (1-Reflm) Wh[m]-
         Emism Wbb[m,Tsura]+h (Tamb-Tsur)-Tcon[Temperatures[[1]] ]*
              (Tsur-Temperatures[[2]])/deltx;) (* end of If *)];
```

and Qc is given as

$$Q_c = \frac{k(T_i)}{\rho(T_i)c_p(T_i)}\left(\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2}\right)$$

where we use a Laplace Operator for this (below from Shaw and Tigg, *Applied Mathematica*: Getting it Started, Getting it Done).

```
Laplace[data_List] :=
   RotateRight[data] - 2 data +
      RotateLeft[data]

LaplaceInterior[data_List] :=
   Drop[Rest[Laplace[data]],-1]
```

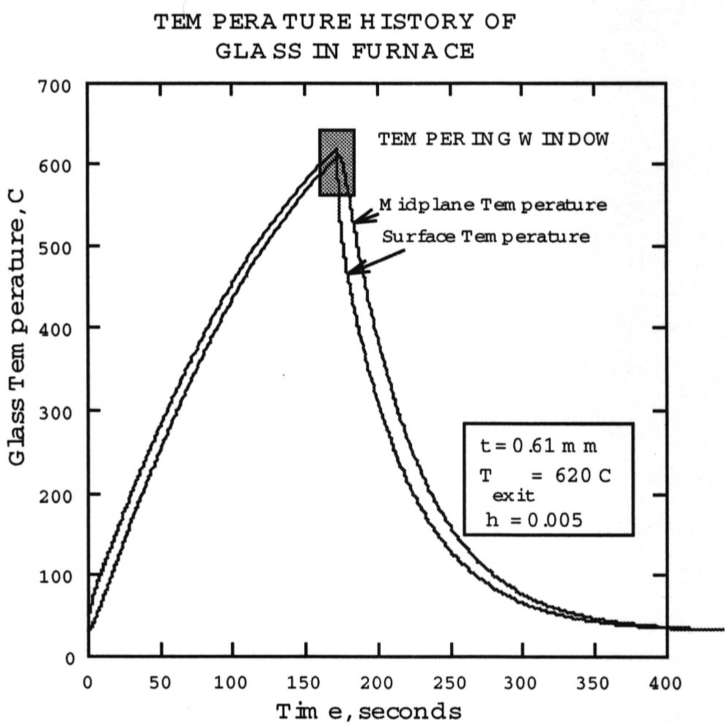
- **Example:**

```
Laplace[Array[k,{10}] ]
```

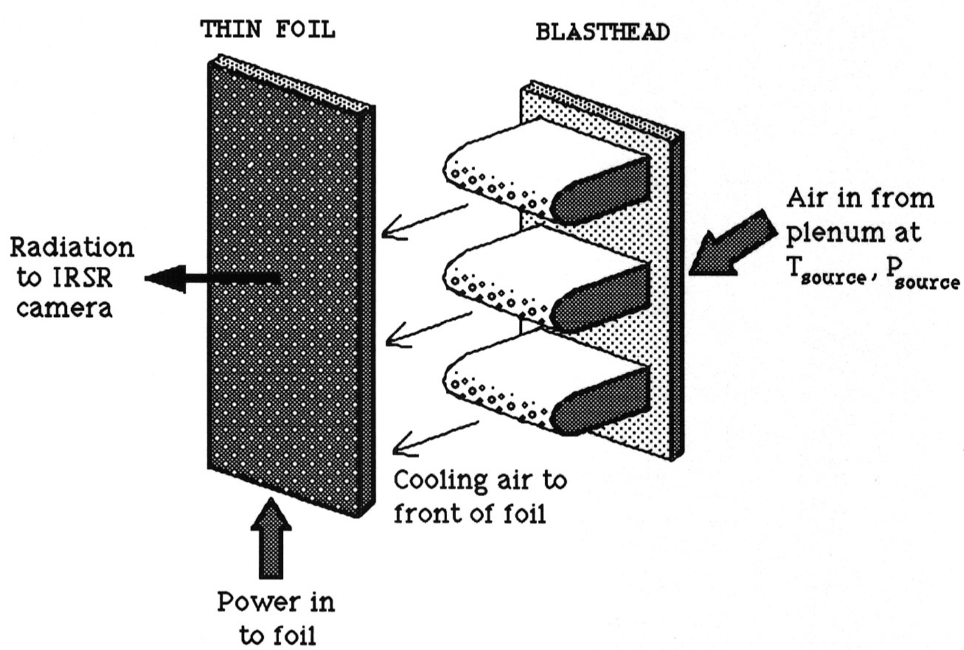
```
{-2 k[1] + k[2] + k[10], k[1] - 2 k[2] + k[3], k[2] - 2 k[3] + k[4],
 k[3] - 2 k[4] + k[5], k[4] - 2 k[5] + k[6], k[5] - 2 k[6] + k[7], k[6] - 2 k[7] + k[8],
 k[7] - 2 k[8] + k[9], k[8] - 2 k[9] + k[10], k[1] + k[9] - 2 k[10]}
```

Empirical Calculation of Forced Convection Coefficient

Glass is tempered by rapidly cooling it from the process temperature of 620 C to the strain temperature of about 520 C. Glass may break during tempering if the transient stresses exceed the yield stresses. Glass may break if it is too hot or too cool during quenching. There is a "tempering window" illustrated below. (This curve was rendered in Jandel Scientific's Sigma Plot with tab-delimited output from the *Mathematica* glass tempering program.)



To predict the heat transfer coefficient, an experimental test rig is used consisting of a heated thin foil, an Inframetrics Radiation Scanning Radiometer (IRSR), a data acquisition system and *Mathematica*. The set up is shown schematically as:



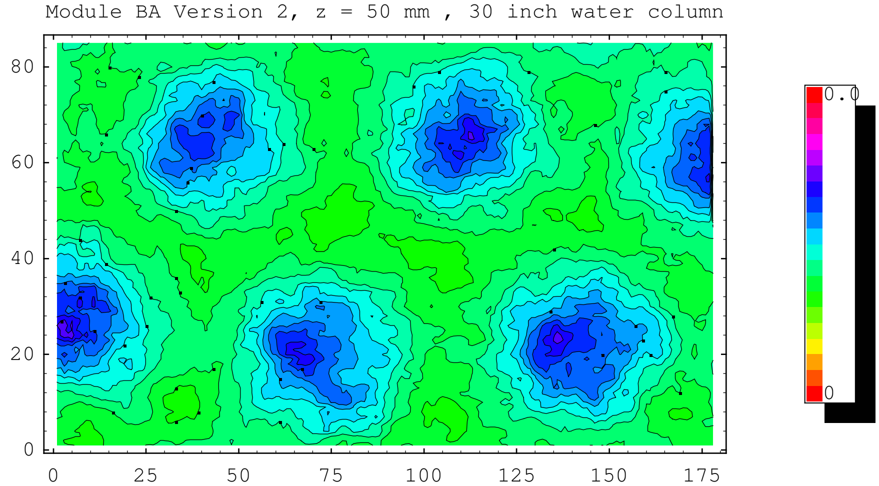
The IRSR camera senses radiant emission from the foil. Based on calibration tables, these can be converted into temperatures. *Mathematica* is used with Microsoft EXCEL, Visual Basic and LabTech Notebook to analyze the data, print a summary report including contour plots. Animations, curve fitting, statistical and trend analysis are some of the desired outcomes. A sample report is generated by *Mathematica* as:

Report

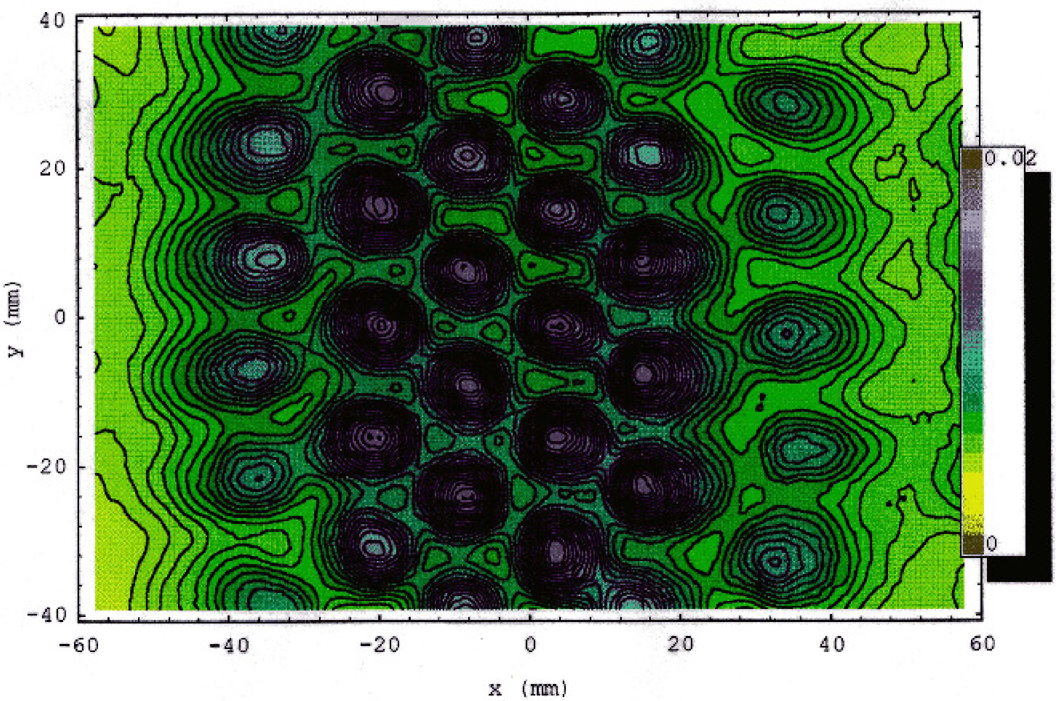
```
RESULTS FOR MODULE BA Version 2, z = 50 mm , 30 inch water column
The time is 21:39:54.31.
The date is  4-08-1994.
Nozzle-to-foil distance = 50 mm.
Camera-to-foil distance = 250 mm.
    Horizontal, Vertical Spatial Resolution = 1.949 , 0.738
Ambient Temperature = 21.4708 C.
Plenum Temperature = 23.3908 C.
Plenum Pressure = 30.6058 in.w.c.
Foil Power = 123.16 W.
Heat Flux = 779.003 W/m^2.
Foil Surface Temperature = 47.1194 C.
The minimum and maximum h's are: 0.00654722 , 0.0143878 cal/cm^2-s-C.
Average Convection Coefficient = 0.00895142 cal/cm^2-s-C.
                                                  6
RMS Value of Convection Coefficient = 3.59667 10

Data were read from file Quadra 800 HD:glasstech:L108:L108\

   Images:ba_50.dat
Results were stored in file: Quadra 800 HD:glasstech:L108:L108\

   Images:0004
```

Graphs

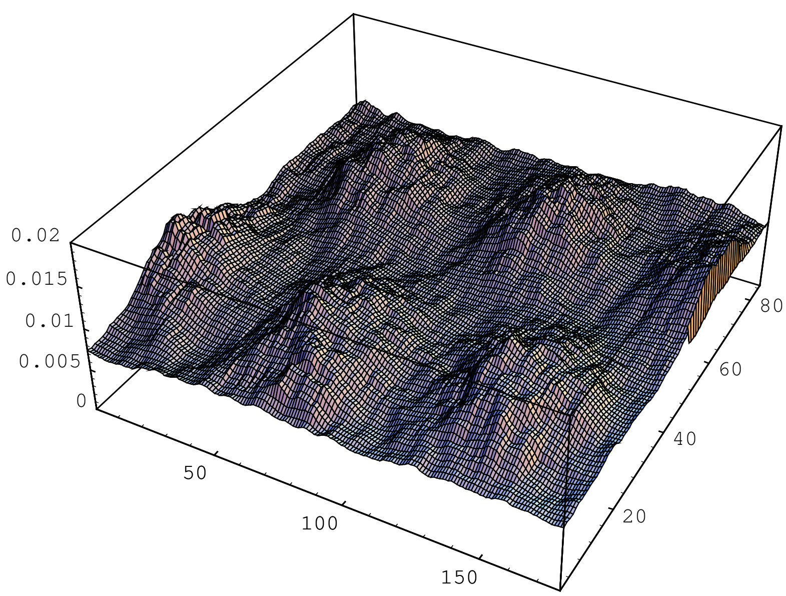
Module BA Version 2, z = 50 mm , 30 inch water column

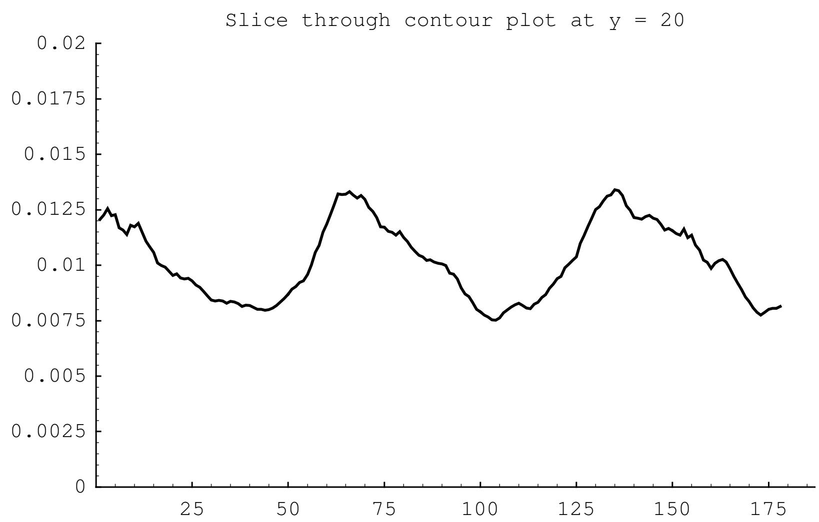


HEAT TRANSFER COEFFICIENT (cal/ cm^2 -s-C)
Module O , 40 in.w.c, z = 10 mm
The date is 6-09-1995.
Average Convection Coefficient = 0.00728538 cal/cm^2-s-C
Minimum h = 0.00432626 at {-57.7093 mm, -39.2057 mm}
Maximum h = 0.0127622 at {13.7099 mm, -9.25078 mm}



Module BA Version 2, z = 50 mm , 30 inch water column





Coloring the Contour Plot

Mathematica has a built-in function, ListContourPlot, which can work with arrays of heights to render a contour plot. The data may be colored with greys or the function Hue. One problem is that the unmodified Hue is for periodic data: in ListContourPlot[data], the extreme values of data are mapped to Hue[0] and Hue[1], which are both red. We didn't want this so had to come up with our own coloring function, shown below.

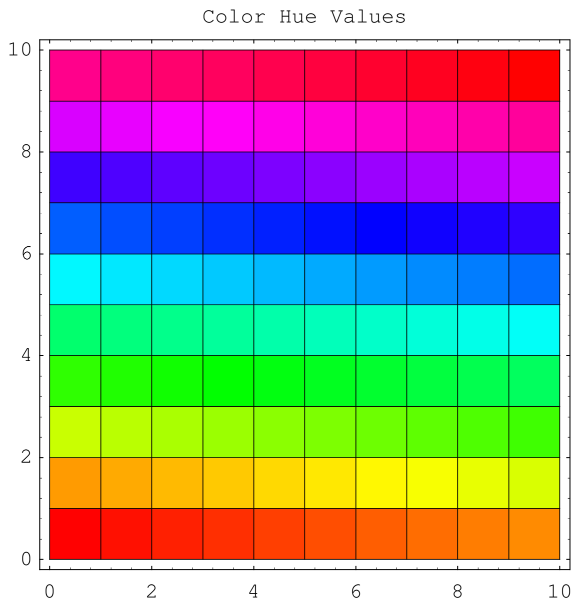
```
ShowLegend[
ListContourPlot[hData,
   DisplayFunction->Identity,
   ColorFunction->
      (Hue[Min[hData]/200+(Hue[#] /.
         Hue[x_] ->
    (Max[hData]-Min[hData])/200 x)]&),
PlotLabel->plottitle,
AspectRatio->200/326],
  {Hue[1-#]&,20,"0.020", "0",
      LegendPosition->{1.1,-.4}}]
```

■ Redefining Hue

The default coloring is exhibited with an array of data:

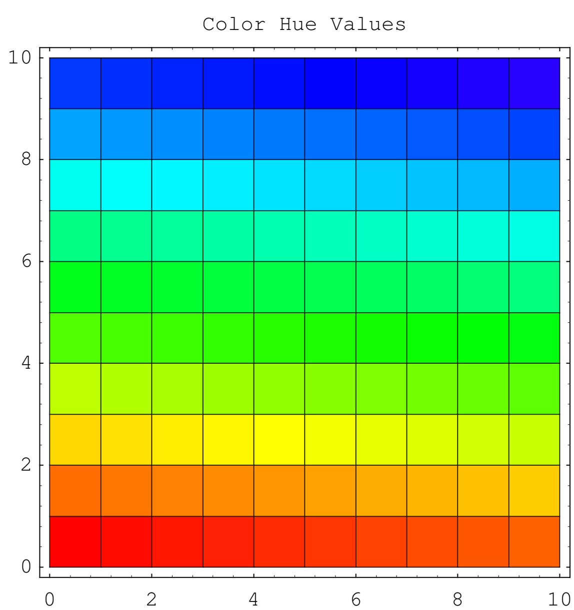
```
TableForm[huedata=
  Transpose[Table[j+i,{i,0,.9,.1},{j,0,9,1}]]/10//N,
  TableSpacing->2*Table[1,{10}]];
```

```
ListDensityPlot[huedata,ColorFunction->Hue,
PlotLabel->"Color Hue Values"];
```



The default **Hue**, then, is periodic. Suppose that we want to have the maximum value mapped to Hue[0.7]. Since Hue is a linear map, we can define a linear function having the properties that we seek. Thus Max[ourdata] -> Hue[0.7], Min[ourdata] -> Hue[0]. Then our function is defined by

```
ListDensityPlot[huedata, ColorFunction →
  (Hue[Min[huedata] 0.7` + (Hue[#1] /. Hue[x_] → (Max[huedata] - Min[huedata]) 0.7` x)] &),
 PlotLabel → "Color Hue Values"]
```



More details of the programming involved will be covered in another talk, "Report from the Trenches: Tips, Tricks and Traps for *Mathematica* Programmers."