APPROXIMATION / INTERPOLATION OF MULTIDIMENSIONAL DATA AND ITS USE IN SIMULATION PROGRAMS

Ing. Ivo Večeřa
Motorola Czech Systems Laboratories, Rožnov p. Radhoštěm

Abstract: The aim of this paper is the use of MATLAB (its SPLINE toolbox) for calculations when simulating circuits. It focuses on expressing multidimensional grided-data relations $z_i = f(x_i,y_i,...)$ in the form of a continuous piecewise polynomial (**pp**) function $z = f(x,y,...)$. Applicable properties of such a function, such as continuity of its derivatives between intervals of definition of each piece of the function, are also exploited. A model of a switching reluctance (**SR**) motor, simulated in the SABER environment, is used as an example.

INTRODUCTION

With increasing computing power of today's equipment, we have more opportunities to model increasingly complex systems. Since we already use virtual prototyping in our lab in Motorola, we have encountered the need for suitable mathematical descriptions of measured characteristics several times.

The reason why we looked at MATLAB capabilities was the following. Traditionally, in nonlinear modeling, a linear interpolation method among given knots is used (as in the case of our circuit simulator). However, the linear interpolation itself creates a convergence problem when the solution is close to any discontinuities. It results in an increased number of iteration loops per simulator step or (very probably) in a crash of the simulation. Moreover, when the derivative of such a function is needed, it is clear that such a function is discontinuous at breaks between linear pieces.

MOTIVATION

This paper uses the following example to demonstrate how to approach nonlinear behavior in modeling.

We have been given measured magnetization characteristics (which are unique) for a 2-phase SR motor $i_{i,j}=i(\psi_i,\theta_j)$ as shown in Fig.1.

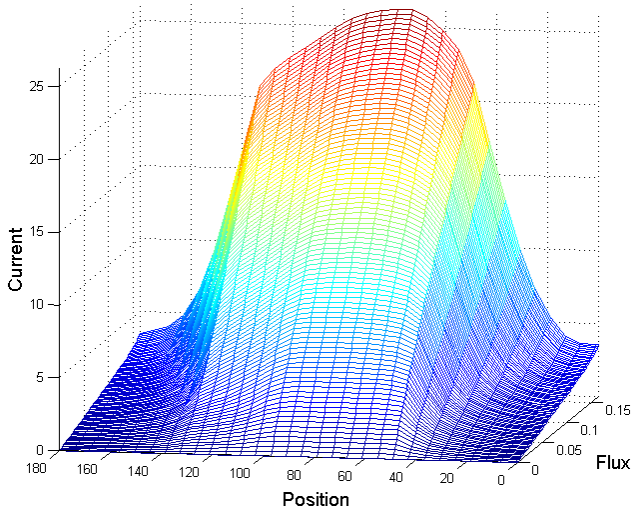

Fig. 1:
Grided data displayed as a 3-D mesh plot

The goal then was to create a SABER® model using equations describing the behavior of the motor, as follows:

$$v_k = r_k \cdot i_k(\psi_k,\theta_k) + \frac{\partial \psi_k}{\partial t} \tag{1}$$

$$T_e = -\int_0^{\psi_A}\left(\frac{\partial i_a(\psi_a,\theta_a)}{\partial \theta_a}\Big/_{\theta=\theta_A}\right)d\psi_a - \int_0^{\psi_B}\left(\frac{\partial i_b(\psi_b,\theta_b)}{\partial \theta_b}\Big/_{\theta=\theta_B}\right)d\psi_b \tag{2}$$

where:

$k = a, b$	phases of the SR motor,
v_k	stator voltage for individual phases,
r_k	stator phase resistance for individual phases,
i_k	stator current for individual phases,
ψ_k	stator magnetic flux for individual phases,
θ_k	rotor position,
T_e	electrical torque.

Conditions that must be satisfied, are:

i) zero current at $\psi = 0$, for all θ

ii) periodicity over position axis

iii) continuous function $i = i(\psi,\theta)$ and continuous partial derivatives $\frac{\partial i}{\partial \theta}$.

SABER BACKGROUND

Saber is a circuit simulator capable of simulating nonlinear characteristics. These nonlinearities, however, are represented in a piecewise-linear form.

Saber also contains an implicit function `d_by_dt,` which calculates the time derivative of a given circuit variable.

Another property of Saber is that, in the phase of modeling, continuity of functions is strongly recommended to prevent simulations from failing due to analysis problems.

Summarizing all that and looking closer at motor equations, (1) can be easily handled by Saber if the function $i_k = i_k(\psi_k,\theta_k)$ can be mathematically expressed. Eq. (2) contains functions undefined in Saber.

On the other hand, if $i = i(\psi,\theta)$ is a continuous function and if partial derivatives $\frac{\partial i}{\partial \theta}$ exist and can be calculated, the goal is reached.

MATLAB AND INTERPOLATION/APPROXIMATION

The above paragraph shows that the choice of the right function $i = i(\psi,\theta)$ is a fundamental step in creating model equations. There are plenty of possible solutions, but let us look at piecewise polynomial functions as interpolation or approximation of the grided data. (All those MATLAB functions are based on tensor product functions.)

From a statistical point of view, the best choice seems to be least-squares approximation but this MATLAB solution (and even its other smoothing functions) is not able to handle defined boundary conditions i), ii). In [2], a discussion about various details concerning the approximation is held concluding that it is also possible to get a solution satisfying even these boundary conditions but this area is filled with mathematical and computational difficulties.

Therefore, the interpolation method had to be chosen for our data. The MATLAB SPLINE toolbox offers several functions for interpolation and the CSAPE m-file (cubic spline interpolation with end conditions) was the one that fit our criteria.

The biggest disadvantage of interpolation as the method is, however, that only some knots are chosen from the whole set of grided data leaving all the rest with no impact on the resulting function.

Here is the script that we applied to get the solution to create the model:

```
% given psi(i), theta(j), current(i,j), i=1..16, j=1..32
pick_psi=[1 9 16], pick_theta=[1 5 10 14 18 23 24 27 32]
x=psi(pick_psi);                        % We split flux axis into 2 pieces ...
y=theta(pick_theta);                    % ..., position axis into 8 pieces
z=current(pick_psi, pick_theta);        % and get corresponding z-values.

pp=csape({x,y},z,{'second','periodic'})  % These values are then proceeded through CSAPE
```

The resulting **pp** interpolating function is periodic over the position axis, and the second derivatives at the borders of the magnetic flux are zero:

```
pp =
      form: 'pp'
    breaks: {[0 0.0840 0.1650]  [0 19 49 73 97 127 133 151 180]}
     coefs: [1x8x32 double]
    pieces: [2 8]              % 3 breaks split interval into 2 pieces (9 br. into 8 p.)
     order: [4 4]              % cubic function contains 4 coefficients
```

We may visualize both the characteristic and interpolating functions in one graph, as shown in Fig. 2. Borders between the intervals of definition are marked in the graph as wider black curves.

```
mesh(x,y,z.'), view(280,10), axis([0 max(x) 0 180 0 max(max(z)) ]),
hold on, pause, fnplt(pp)           % plot of a pp function
```

The function is evaluated by calculating (All multivariable functions in SPLINE toolbox are held in tensor-product form)

$$i(x,y) = \sum_{i=1}^{4}\sum_{j=1}^{4}\left({}^{m,n}c_{i,j}\,x^{4-i}\,y^{4-j}\right) \tag{3}$$

where m, n locate in the characteristic (Fig. 2) the position of an actual point,
${}^{m,n}c_{i,j}$ is the matrix with 4×4 components, different for each of $m \times n$ (i.e. 2×8) pieces,
$x = \psi - break\_psi(\mathrm{m})$, $break\_psi(\mathrm{m}) \le \psi < break\_psi(\mathrm{m+1})$, (4)
$y = \theta - break\_theta(\mathrm{n})$, $break\_theta(\mathrm{n}) \le \theta < break\_theta(\mathrm{n+1})$. (5)
Break sequences, *break_psi*(m), *break_theta*(n), are assumed to be strictly increasing.

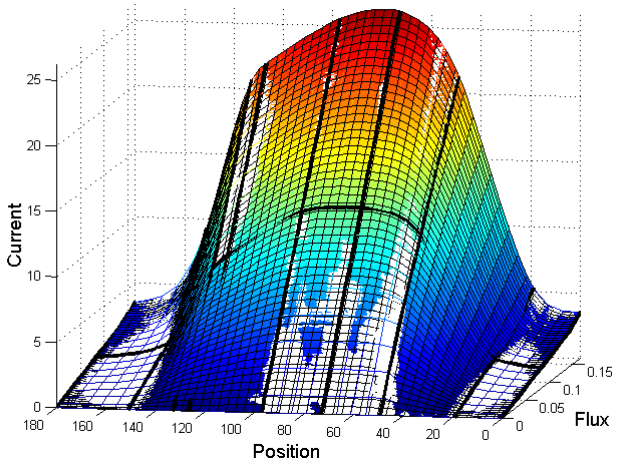

Fig. 2:
Obtained interpolation function displayed as a surface plot (interleaved with generic mesh plot)

Note here that the credibility of interpolation is strongly dependent on the right choice of knots (i.e. breaks).

Evaluation of the torque is made using Eq. (2) directly, where the integral has to be calculated part by part individually for each definition interval. In the case of the situation shown in Fig. 3, for example, the calculation is as follows:

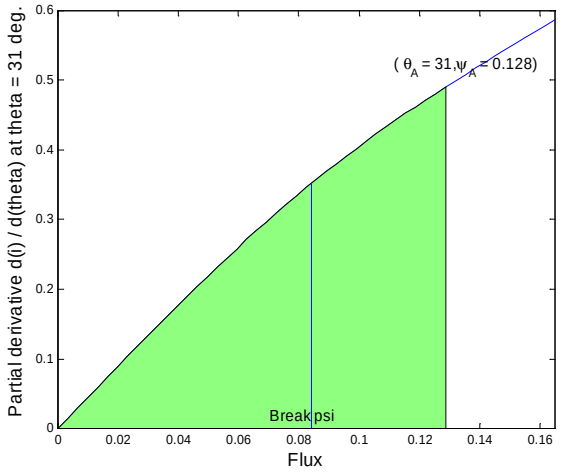

Fig. 3
Calculation of contribution to T_e as integral of partial derivative $\left.\frac{\partial i_a(\psi_a,\theta_a)}{\partial\theta_a}\right/_{\theta=\theta_A}$

$$\int_0^{\psi_A}\left(\left.\frac{\partial i_a(\psi_a,\theta_a)}{\partial\theta_a}\right/_{\theta=\theta_A}\right)d\psi_a = \int_0^{break\_psi}\left(\left.\frac{\partial}{\partial\theta_a}\left({}^{2,1}i_a(\psi_a,\theta_a)\right)\right/_{\theta=\theta_A}\right)d\psi_a + \int_{break\_psi}^{\psi_A}\left(\left.\frac{\partial}{\partial\theta_a}\left({}^{2,2}i_a(\psi_a,\theta_a)\right)\right/_{\theta=\theta_A}\right)d\psi_a$$

and after substitution (4), (5) we get

$$\int_0^{\psi_A}\left(\left.\frac{\partial i_a(\psi_a,\theta_a)}{\partial\theta_a}\right/_{\theta=\theta_A}\right)d\psi_a = \int_0^{break\_psi}\left(\left.\frac{\partial}{\partial y}\left({}^{2,1}i_a(x,y)\right)\right/_{y=y_A}\right)dx + \int_0^{x_A}\left(\left.\frac{\partial}{\partial y}\left({}^{2,2}i_a(x,y)\right)\right/_{y=y_A}\right)dx = \ldots = (3) = \ldots =$$
$$= \sum_{i=1}^{4}\sum_{j=1}^{3}\left(\frac{4-j}{4-i+1}\cdot{}^{2,1}c_{i,j}\cdot break\_psi^{4-i+1}\cdot {y_A}^{4-j}\right) + \sum_{i=1}^{4}\sum_{j=1}^{3}\left(\frac{4-j}{4-i+1}\cdot{}^{2,2}c_{i,j}\cdot {x_A}^{4-i+1}\cdot {y_A}^{4-j}\right) \quad (6)$$

As seen in (3), (6), all calculations (for i and T_e, respectively) involve only matrixes ${}^{m,n}c_{i,j}$ and arrays defining breaks (*break_psi* and *break_theta*, respectively), although obtaining new coefficients for calculating torque is also possible using command FNDER:

```
torque=fnder(pp,[-1,1])          % [-1, 1] means integration with respect to flux and
                                 %         derivation by position
```

Its **pp** representation and plot of this function follows (with its plot shown in Fig. 4):

```
torque =
      form: 'pp'
    breaks: {[0 0.0840 0.1650]  [0 19 49 73 97 127 133 151 180]}
     coefs: [1x10x24 double]
    pieces: [2 8]                % again the same breaks thus the same pieces ...
     order: [5 3]                % but different order of function
```

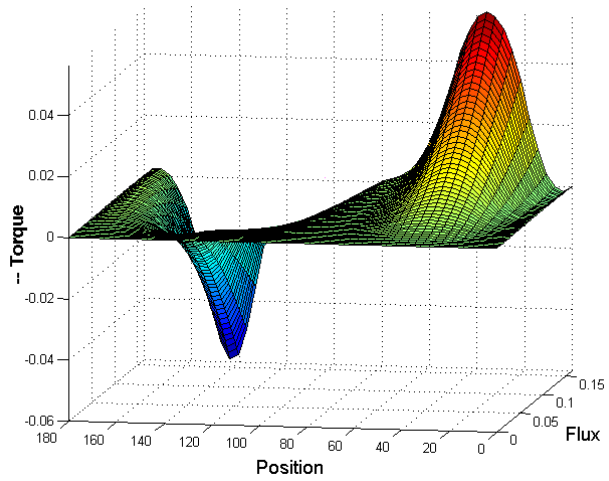

Fig. 4:
Plot of contribution of one phase of the SR motor (as the function of position and magnetic flux) to electrical torque

Finally, just for comparison, the following script produces an approximation (without end conditions; the function is smooth but not periodic, neither zero at $\psi = 0$; view plot in Fig. 5) by using SPAP2 (spline approximation) m-file:

```
k_theta=4, kn_theta = augknt(theta([1 4 8 11 14 18 23 24 28 32]), k_theta)
k_psi=3, kn_psi = augknt([0 0.07 0.14 max(psi)], k_psi)
current_app=spap2({kn_psi,kn_theta},[k_psi,k_theta],{psi,theta},proud)
```

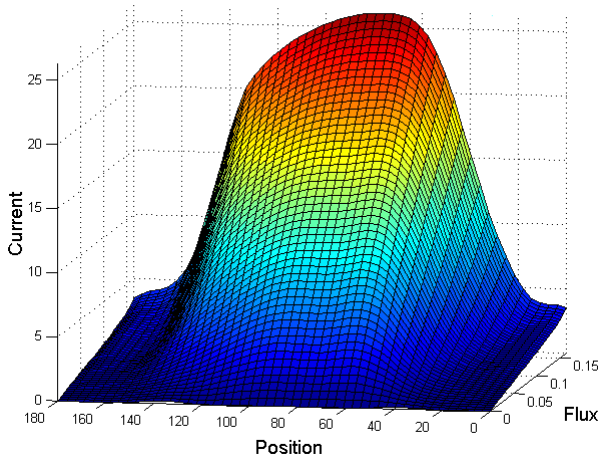

Fig. 5:
Plot of the approximation of the given characteristic

CONCLUSION

This paper presents the modeling of characteristics described as piecewise polynomial (**pp)** functions among discrete values of a mesh grid. This is fully applicable when linearization methods cannot be used. Two methods are compared and different aspects are discussed. This **pp** form of description is also preferred in situations when the multiply-accumulate structure of a processor is used. Further work will probably be done in this field to accommodate boundary conditions, nevertheless this model is successfully implemented by calling a 'C' function from the Saber template of an SR motor so that the algorithm to calculate current and torque is realized in the 'C' language.

REFERENCES

[1] "MATLAB 5 Documentation", "SPLINE Toolbox". The MathWorks, Inc.
[2] Carl de Boor, "A Practical Guide to Splines". Applied Mathematical Sciences, Vol. 27, Springer-Verlag New York, 1978
[3] "Saber MAST reference manual". Analogy, Inc., March 1999