A Smart Burn and Spill Proof “SAFE” Microwave that spares the Salad: Novel Application of Levenberg-Marquardt Algorithms in Bayesian Analysis for Real-Time Numerical Thermodynamic Modeling

By Muhammad Shahir Rahman
Problem Statements

• The use of heat is essential in household applications
• Many cooking apparatus, such as a microwave and oven are not intelligent and can often cause fires
• In fact, 42% of household fires are caused by cooking devices
• The goal is to create an intelligent microwave that can cook food to the correct adaptable target temperature profile with minimal user input.
Intelligent Microwave

A thermopile sensor array at the top of the microwave will detect food temperatures.

User places food of any kind inside the microwave.

The internet connected computer sends data to the user wirelessly.

IOT solutions provide remote control and full telemetry. The user is able to set the target temperature to his/her own preferences.
Cooking Apparatus

- Cooking Device Body
- Infrared Sensor Array
- Food Platter
1. Start by compiling the sensor data into an array
2. Rotate & Smooth the data
3. Take the food temperature data & compare it to the virtual graph
4. Iterate through each cell and calculate the viscosity
5. And then calculate the target temperature

Extensive programming (with Python) work for computation has brought this idea to reality!
Data

Liquid Data

Each line represents a pixel in the sensor array.
Data from liquid food
Platter data

Solid Data

Each line represents a pixel in the sensor array.
Data from solid food
Platter data

Semisolid Data

Each line represents a pixel in the sensor array.
Data from solid food
Platter data
Intelligent Cooking

Programming Strategy:

- **Create** a virtual thermal profile in the microwave
- **Optimize** the total squared residual between the virtual profile and the thermopile data to minimum
- **By Changing** the viscosity, amount, initial temperature and/or other properties in the virtual food profile

\[
R^2 = \sum_{t=0}^{\infty} (T_{data} - T_{virtual})^2
\]

Start warming and record initial temperature \\
Solve for Viscosity and Amount by optimizing Total Squared Residual \\
Calculate Target Time using the Viscosity and Amount from the virtual graph \\
Stop Microwave

Target Time Reached

Yes

No
The electromagnetic waves generated by the magnetron severely interferes with the sensor’s data acquisition and the sensor cannot even survive for a second. At the same time, recovery of the sensor while it is ON takes too long. This difficult problem was solved by using a mutually exclusive duty cycle. In addition to safeguard the sensor from a high voltage drop ($V_L = L \frac{\partial I}{\partial t}$) from magnetron inductance the duty cycle was carefully controlled as shown below.
Internet of Things Microwave

- Sensor
- Lamp and Motor for Turntable
- Magnetron
- Socket Programming

Internet Cloud
- Temperature Data
- Sever Calculations
- Calculations
- Target Temperature

Cloud Computer
- Algorithm
- Machine Learning
- Temperature Profiling Database

User Preferences
- Device Signature

Android Device
- User Detection
- Heat Map

User Customization
- Profile Settings
- Target Temperature
Targeted Pixel Based Heating (TPBH) is one of the new innovations. Each temperature cell in the sensor array is interpreted as an individual food particle with its own separate properties. By targeting only specific pixels that need warming, we can all the food in the platter to the perfect temperate, warming the chicken and sparing the salad.
# State of Art Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Current State of Art</th>
<th>My Innovation</th>
</tr>
</thead>
<tbody>
<tr>
<td># of User Input Parameters</td>
<td>1-4</td>
<td>0 required</td>
</tr>
<tr>
<td>Fire Safety</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Unattended Cooking</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Cooking Accuracy</td>
<td>60-90% <em>(human error)</em></td>
<td>99%</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>Time is guessed, magnetron: usually stays on longer than needed</td>
<td>Magnetron only on as required (saves lot of power)</td>
</tr>
<tr>
<td>Temperature Preference</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Personalization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Food Auto Detection</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Integration in to Cloud</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Remotely Controller through</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smartphone</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Cost</td>
<td>$50-400</td>
<td>Less than 17% increase for a high end microwave</td>
</tr>
</tbody>
</table>
Intelligent Microwave

A thermopile sensor array at the top of the microwave will detect food temperatures.

User places food of any kind inside the microwave.

The internet connected computer sends data to the user wirelessly.

IOT solutions provide remote control and full telemetry. The user is able to set the target temperature to his/her own preferences.

User places food of any kind inside the microwave.

The internet connected computer sends data to the user wirelessly.

IOT solutions provide remote control and full telemetry. The user is able to set the target temperature to his/her own preferences.
References

Scan this QR code for more Information!

Smart Microwave

Thank You
How to skip the Salad!

Stepper motor allows optimum control of rotation at high heat points for denser food sections.

Asymmetric high heat areas in cabin

Asymmetric density of Food sections

Low viscosity
High viscosity

3.75° Step angle for extra precision rotation

FULL STEP
Two phases at a time
Strongest Torque

Stepper motor can step through its rotation at any given multiple of step (3.75 degrees) and turn in any direction.
Bayesian Food Characteristics Algorithm

Using a specific viscosity and amount from the algorithm, the matching temperature profile is generated by interpolation of the profiles in the database as in a machine learning technique.

### Algorithms Studied

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Timing Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nelder-Mead</td>
<td>10 loops, best of 3: 37.8 ms per loop</td>
</tr>
<tr>
<td>Powell</td>
<td>10 loops, best of 3: 26.5 ms per loop</td>
</tr>
<tr>
<td>CG</td>
<td>10 loops, best of 3: 404 ms per loop</td>
</tr>
<tr>
<td>BFGS</td>
<td>10 loops, best of 3: 48.6 ms per loop</td>
</tr>
<tr>
<td>L-BFGS-B</td>
<td>10 loops, best of 3: 46.1 ms per loop</td>
</tr>
<tr>
<td>TNC</td>
<td>10 loops, best of 3: 28.3 ms per loop</td>
</tr>
<tr>
<td>SLSQP</td>
<td>10 loops, best of 3: 23.2 ms per loop</td>
</tr>
<tr>
<td>COBYLA</td>
<td>100 loops, best of 3: 16.1 ms per loop</td>
</tr>
<tr>
<td>Differential Evolution</td>
<td>10 loops, best of 3: 275 ms per loop</td>
</tr>
<tr>
<td>LM-LSQ</td>
<td>100 loops, best of 3: 7.82 ms per loop</td>
</tr>
</tbody>
</table>

LM-LSQ is Levenberg Marquardt Least Squares Method

Data from a single pixel of sensor

Profile from the data (blue) is compared to that of the algorithm (red)

Algorithm Profile

Real-time Heat Map from Sensor

Profile from the data (blue) is compared to that of the algorithm (red)
The Cheese Experiment

Normal standing heat pattern

Oven inverted heat pattern
Data collection involves four main components:
Sensor, Edison, Network socket, & Algorithm with Python code

**Edison Thread 1**
- Data Acquisition
- Dynamic Filtering
- Asynchronous Data Buffering

**Edison Thread 2**
- Bidirectional Socket Data Send
- Server ACK
- Cloud Server Task Delegation
-Recv. Calculations from Server
- Stop Microwave if Needed

**Server Thread 1**
- Threaded/Pipeline Mgmt.
- Asynchronous DataRecv.
- Relay Calculations back to Edison

**Server Thread 2**
- Bayesian Predictive Model
- LM-LSQ Optimization
- Target Determination
- Deviate Results using User Settings

**Server Thread 3**
- Generate Heat Map Image
- Broadcast data for Smartphone app
- Receive Phone Signature
- Receive User Target Settings
- Pipeline Mgmt. for Algorithm
Journey to the Current Platform

• Started with Galileo
  – Inadequate and non-native display
  – computation speed

• Minnow board was suggested for better computation
  – Same problems as Galileo

• At NDG the Edison was a better choice, however challenges remain:
  – PySerial
  – HID device and PyUSB
  – Yocto linux
  – I2C voltage level shifting
  – New linux command line
  – Limited documentation

• With all of these platforms I had to:
  – Start from scratch
  – Port code base
  – Deal with device specific driver issues
Sensor Installation

First prototype was built with Melexis 16x4 sensor array EVB. However the aspect ratio was not optimal and communication was difficult and its radiation tolerance was very poor.

Eventually the Panasonic Grid-EYE, an 8 by 8 thermopile array with high accuracy and low cost was found. Communication with was also simple (both I2C & USB) and this is the current solution in the microwave as demonstrated.

Numerous low-cost, low-resolution, high accuracy thermopile sensor were investigated for to put inside the microwave.
In a microwave oven, standing waves emitted from the magnetron are emitted horizontally which creates uneven heating. To correct this the platter is rotated for homogenous heating. However, if the rotation of the microwave was controlled intelligently, this unevenness could be utilized for selective heating as explained in the TPBH section.
Microwave Edison Setup

11 Digital Pins + 2 I2C Data Pins

Sensor

Scipy
Matplotlib
Numpy

Computational Libraries

Yocto Poky Linux on Edison

Socket
Pyserial
Libmraa

I/O Libraries

Microwave Programs

Python Interpreter

Lamp and Turntable

5 Analog Pins

Magnetron
Mathematical Concepts

The Levenberg-Marquardt algorithm at its core is a trust-region algorithm that interpolates between the gradient descent method and the Gauss-Newton method.

**Gradient Descent**

The Gradient Descent Method is based on the observation that if the multi-variable function \( f(x) \) is defined at differentiable in the neighborhood of a point at a certain point, then \( f(x) \) decreases fastest in the direction of the negative gradient \( -\nabla f(x) \). Therefore, it follows that with an initial guess \( x_0 \), an iterative method can be produced for the next point to minimize \( f(x) \):

\[
x_{n+1} = x_n - \gamma \nabla f(x_n)
\]

(assuming \( \gamma \) is a small value such that \( \gamma > 0 \))

![Gradient Descent](image)

The gradient (a multidimensional derivative) of the function \( f(x,y) = -(\cos^2x + \cos^2y)^2 \) depicted as a projected vector field on the bottom plane.

**Gauss-Newton**

The Gauss-Newton Method on the other hand tries to minimize the sum of squared residuals. The algorithm is given \( m \) functions (residuals) \( \mathbf{r} = (r_1, r_2, \ldots r_m) \) of \( n \) variables \( \mathbf{x} = (x_1, x_2, \ldots x_m) \) where \( m \geq n \), and the sum of squares is minimized:

\[
\| \mathbf{r}(\mathbf{x}) \|^2 = \sum_{i=0}^{m} r_i^2(\mathbf{x})
\]

Rather than calculating second-order derivatives or the Hessian matrix, the method uses \( \mathbf{r}(\mathbf{x}) \), and its derivative, the Jacobian \( (J_r)_{ij}(\mathbf{x}) = \frac{\partial r_i(x)}{\partial x_j} \). Using the Jacobian, \( r(\mathbf{x}) \) is linearized at current iteration \( n \) given an initial guess \( x_0 \):

\[
\begin{align*}
    r(\mathbf{x}) & \approx r(x_n) + J_r(x_n)(\mathbf{x} - x_n) \\
    &= A_n(x_n)x - b_n(x_n) \\
    A_n(x_n) &= J_r(x_n), \\
    b_n(x_n) &= J_r(x_n)x_n - r(x_n)
\end{align*}
\]

The next iteration is found by minimizing the sum of squares

\[
\| r(x_{n+1}) \|^2 \approx \| A_n(x_n) - b_n(x_n) \|^2
\]
Rotation Algorithm

As the plate rotates, the location of each of the food pixels relative to the initial grid changes, and that is tracked through each data collection.

- Each line is a pixel in the sensor array.
- Raw data as is. Spikes exist because in some positions, the sensor senses food, in other times it senses the plate.
- The rotational tracking is applied to help compensate. However, some errors still do occur.
- Finally smoothing is applied, eliminating all of the rest of the spikes.