Calculating Specialty Materials Usage Accurately: an Improved Algorithm and a Regression Series
Mapping Voltage to Flow Rates

Rachel Barcklay
Berkeley High School, Summer 2015
Presentation Agenda

- programming project
  - overview
  - implementation
- lab work
  - objectives
  - results
RUMS Usage Method Overview

INPUT - equip. ID, start time, and end time

FUNCTION - calculates the amount consumed within the two time constraints

OUTPUT - returns amounts of each material used
RUMS Usage Method Implementation

- uses parameterized queries (vs. string concatenation)

Parameterized Query
- checked input
- prevents most SQL injection attacks

String Concatenation
- unchecked input
- less secure than prepared statements

SELECT id FROM equipment WHERE name = "? DROP TABLE equipment"
What next?

- gets back information about all sensors associated with the equipment
  - sensor names & ID's
  - which gases the sensors monitor
  - sensor types
Which sensors are connected to the AMATEPI?

Germane_flow_Epi & Dichlorosilane

```sql
SELECT value FROM resources, properties
WHERE resources.id = ?
AND id= object AND name = 'USAGE' AND value!=''
```
- queries RUMS (parameters are sensor ID, start time, end time)
- statement returns data points for the sensor between start and end times
Totalizing sensor

Integrating sensor
SENSOR
«abstract»

INTEGRATING SENSOR
SUM CONSUMED()

TOTALIZING SENSOR
SUM CONSUMED()
/** Totalizing sensor type
 * @author Rachel */

public class TotalizingS extends Sensor {

    private TotalizingS()
    {
        super();
    }

    public TotalizingS(String name, String invenID){
        super(name, invenID);
    }

    @Override
    public BigDecimal sumConsumed(DataSet data){
        if(data.isEmpty()){
            return new BigDecimal("0");
        } else{
            BigDecimal[] x = data.getRates();
            return x[x.length -1].subtract(x[0]);
        }
    }
}

public class IntegratingS extends Sensor {

    private IntegratingS()
    {
        super();
    }

    public IntegratingS(String name, String invenID){
        super(name, invenID);
    }

    @Override
    public BigDecimal sumConsumed(DataSet data){
        BigDecimal areaSum = new BigDecimal("0");
        BigDecimal[] t = data.getTimes();
        BigDecimal[] r = data.getRates();

        for (int i = 0; i < r.length; i++) {
            if (t[i].compareTo(this.getBound()) < 0) {
                t[i] = new BigDecimal("0");
            }
        }

        if(t.length == 1){
            System.out.println("only one data point.");
        }
    }
}
sumConsumed() Algorithms

Totalizing Sensor

Integrating Sensor
Lab Work Objectives

- debug software
- map voltage to flow rates using a regression series
- test GeH₄ flow at different rates
- test Ge deposition properties
  - stress using flexus
First Run on the AMATEPI

- ran at 1, 5, 10, 20, 30, 40, 50, 75, 100 sccm
- program crashed
  - incomplete parameters
- RUMS nano charting was inaccurate
Improving Data Accuracy

- sensors calibrated
- mapped voltage to actual flow rates rather than programmed rates
- 30.4 sccm: now 29.99 rather than 27.5
- \( r^2 \) value of .998 for flow rates up to 100 sccm
What I learned In the Process

- dynamic programs
- choice of variable and list types
- communication
- failure during testing
Thank you…

- Olek Proskurowski and Richelieu Hemphill for mentoring me
- Marilyn, Jayss, and Cheryl for taking us to Semicon and helping me in the lab
- Bill Flounders for giving me the opportunity to intern this summer