A Web Service Infrastructure for Thermochemical Data

Christopher P. Paolini* and Subrata Bhattacharjee

Computational Science Research Center and the Department of Mechanical Engineering, San Diego State University, 5500 Campanile Drive, San Diego, California 92182

Received December 7, 2007

W3C standardized Web Services are becoming an increasingly popular middleware technology used to facilitate the open exchange of chemical data. While several projects in existence use Web Services to wrap existing commercial and open-source tools that mine chemical structure data, no Web Service infrastructure has yet been developed to compute thermochemical properties of substances. This work presents an infrastructure of Web Services for thermochemical data retrieval. Several examples are presented to demonstrate how our Web Services can be called from Java, through JavaScript using an AJAX methodology, and within commonly used commercial applications such as Microsoft Excel and MATLAB for use in computational work. We illustrate how a JANAF table, widely used by chemists and engineers, can be quickly reproduced through our Web Service infrastructure.

USE OF WEB SERVICES IN EXISTING RESEARCH

Recent efforts by Dong et al.[1] have shown the strength of using Web Service technology in chemoinformatics to facilitate the organization and retrieval of chemical data. The Basis Set Exchange (BSE) project by Schuchardt et al.[2] uses Web Services to facilitate collaboration among researchers involved in the development of functions that model molecular orbitals. BSE provides public-access Web Services that allow researchers to upload and download basis set data and query the names of contributed basis sets. Furthermore, the Blue Obelisk[3] movement has come about to promote the development of open-source, reusable software components for chemoinformatics research, and Guha et al.[3] have shown how Web Services can be used to provide distributed, programmatic access to CDK[4] functions for molecular similarity and descriptor calculation.

This work shows how Web Services can be used to provide an infrastructure for thermochemical property computation and retrieval and how such Web Services can be integrated with commercial applications to provide a synergistic environment for computational thermochemistry.

THERMOCHEMICAL PROPERTY COMPUTATION

Since the first edition of the International Critical Tables in 1926 by Washburn,[5] thermochemical data have been tabulated and published by various authors and institutions in printed form. Between 1955 and 1958, challenges in computing multiphase equilibrium in solid-fuel propellant systems led to the formation in 1958 of the Joint Army Navy Air Force (JANAF) Ad Hoc Panel on the Performance Calculation Methods of Thermodynamic Data. Later, in 1960, the first edition of the JANAF Thermochemical Tables[6] was published for military use only. These tables include data for over 1800 species and list, as a function of temperature, the standard state constant pressure specific heat $C_p°(T)$, standard state entropy,

$$S°(T) = \int_0^T \frac{C_p°(T)}{T}\,\mathrm{d}T \tag{1}$$

standard state sensible enthalpy (i.e., enthalpy increments or the energy required to bring a species from one temperature state to another),

$$H°(T) - H°(T_\mathrm{r}) \tag{2}$$

standard state enthalpy of formation, standard state Gibbs free energy of formation,

$$\Delta_f H°(T) = \Delta_f H°(T_r) + \underbrace{\left[H°(T) - H°(T_r)\right]}_{\text{compound}} - \sum_{\text{all elements}} \left[H°_{\text{element}}(T) - H°_{\text{element}}(T_r)\right] \tag{3}$$

and log values of the equilibrium constant of formation,

$$\Delta_f G°(T) = \Delta_f H°(T) - T\left[\underbrace{S°(T)}_{\text{compound}} - \sum_{\text{all elements}} S°(T)\right] \tag{4}$$

$$\ln K_\mathrm{f} = -\frac{\Delta_\mathrm{f} G°(T)}{RT} \tag{5}$$

where T_r is defined as the reference temperature 298.15 K. The JANAF tables list data for temperatures ranging from 100 to 6000 K. In 1963, thermochemical data became available to the public through the NASA publication of the *Thermodynamic Properties of Chemical Substances to 6000 K* by Gordon and McBride.[7] This database was generated using numerical methods of calculating thermochemical properties of ideal monatomic and diatomic gases, linear polyatomic molecules, and nonlinear polyatomic molecules using a rigid rotor, harmonic oscillator model. In addition, the NASA tables listed a new property for each species called the *absolute enthalpy*, given by

$$H°(T) = \Delta_\mathrm{f} H°(T_r) + [H°(T) - H°(T_r)] \tag{6}$$

Since 1963, several additional sources of thermodynamic data have been published. Most notable are the Thermodynamic Properties of Individual Substances by Gurvich[8] and the more recent 1995 Thermodynamic Data of Pure Sub-

* Corresponding author e-mail: paolini@engineering.sdsu.edu.

10.1021/ci700457p CCC: $40.75
Published on Web 06/11/2008

stances by Barin.[9] In addition to printed data, efforts have been made to make thermochemical data available online. For example, the National Institute of Standards and Technology (NIST) Webbook[10] offers a HTML Web form-type accessible database of chemical and physical properties for over 7000 compounds, and Burcat and Ruscic have made available their Thermodynamic Database for Combustion and Air-Pollution Use[11] (TDCAPU) in XML format via FTP.

The most efficient method used in computational thermodynamic software to compute properties such as those given by eqs 1–6 is through the use of polynomial functions of temperature. These polynomials are generated by least-squares-fitting a model power series function to a set of species data obtained from ab initio programs like Gaussian[12] and GAMESS[13] that compute electronic structure. To see why using a power series polynomial for property evaluation is more efficient than using a direct function derived from statistical thermodynamics, consider the equation for approximate statistical thermodynamic entropy for a gas at its standard state pressure an temperature T. For linear polyatomic species, statistical thermodynamic entropy is given by eq 7.

$$\begin{aligned}\overline{S}^{\circ}_{\text{statistical}} &= R\left(\frac{3}{2}\ln M + \frac{5}{2}\ln T\right) - 9.686 + R\ln\left(\frac{T}{\sigma\theta_{r,1}}\right) + R + \\ &\sum_{\nu=1}^{3N-5}\left[-R\ln(1-e^{-\Theta_\nu/T}) + R\frac{\Theta_\nu/T}{e^{\Theta_\nu/T}-1}\right] \\ &= R\left\{\frac{3}{2}\ln M + \frac{5}{2}\ln T + \ln\left(\frac{T}{\sigma\theta_{r,1}}\right) + \right. \\ &\left.\sum_{\nu=1}^{3N-5}\left[\frac{\Theta_\nu/T}{e^{\Theta_\nu/T}-1} - \ln(1-e^{-\Theta_\nu/T})\right] + 1\right\} - 9.686, \text{J/K mol}\end{aligned} \tag{7}$$

For nonlinear polyatomic species, there are three characteristic rotational temperatures, $\theta_{r,i}$, $i = 1$, 2, and 3, which are each a function of the molecule's respective principal moment of inertia. This difference, together with the symmetry number σ of a molecule, yields a slightly different function for statistical thermodynamic entropy of nonlinear species and is given by eq 8.

$$\begin{aligned}\overline{S}^{\circ}_{\text{statistical}} &= R\left(\frac{3}{2}\ln M + \frac{5}{2}\ln T\right) - 9.686 + \\ &R\ln\left(\frac{\sqrt{\pi}T^{3/2}}{\sigma\sqrt{\theta_{r,1},\theta_{r,2},\theta_{r,3}}}\right) + \frac{3}{2}R + \\ &\sum_{\nu=1}^{3N-6}\left[-R\ln(1-e^{-\Theta_\nu/T}) + R\frac{\Theta_\nu/T}{e^{\Theta_\nu/T}-1}\right] \\ &= R\left\{\frac{3}{2}\ln M + \frac{5}{2}\ln T + \ln\left(\frac{\sqrt{\pi}T^{3/2}}{\sigma\sqrt{\theta_{r,1}\theta_{r,2}\theta_{r,3}}}\right) + \right. \\ &\left.\sum_{\nu=1}^{3N-6}\left[\frac{\Theta_\nu/T}{e^{\Theta_\nu/T}-1} - \ln(1-e^{-\Theta_\nu/T})\right] + \frac{3}{2}\right\} - 9.686, \text{J/K mol}\end{aligned} \tag{8}$$

Table 1 shows a count of the number of transcendental, algebraic, and arithmetic operations required to evaluate eqs 7 and 8 as a function of the number of atoms n in a given molecule. One can see that the number of operations scales linearly with n.

On the other hand, once a power series polynomial of finite degree has been fit to experimental or statistically derived thermodynamic data, the number of transcendental, algebraic, and arithmetic operations required to evaluate the polynomial is fixed and not dependent on n. For example, the NASA nine-term polynomial for standard state entropy given by eq 16 can be parenthesized to minimize the number of multiplications required for evaluation. A fully parenthesized expression for 16 is given by

$$S^{\circ} = R\left(-\frac{1}{T}\left[\frac{a_1}{2T} + a_2\right] + a_3\ln T + T\left\{a_4 + T\left[\frac{a_5}{2} + T\left(\frac{a_6}{3} + T\frac{a_7}{4}\right)\right]\right\} + a_9\right), \text{J/K mol} \tag{9}$$

and the corresponding operation counts are given in Table 2. Consider also that $\overline{S}^{\circ}_{\text{statistical}}$ is a function of $3n - 1$ variables for linear polyatomic species and $3n$ variables for nonlinear species, while the power series polynomial is a function of just one variable—temperature.

The same model function is used to fit data for all species, and what is then stored in a database are the unique coefficients obtained from each fit. This method of representing thermochemical information greatly reduces the amount of data that must be stored to compute thermodynamic properties in software during runtime. Though many model functions have been proposed and used,[14,15] the three most prevalent models used in equilibrium calculations are the NASA seven- and nine-term polynomials and the Shomate polynomial.

The NASA seven-term polynomials and associated species data were published by McBride and Gordon in 1967[16] and provided a means to compute, in dimensionless form, the standard state specific molar heat capacity, enthalpy, and entropy in the 200−6000 K temperature range. These polynomials are as follows:

$$\frac{C_p^{\circ}(T)}{R} = a_1 + a_2T + a_3T^2 + a_4T^3 + a_5T^4 \tag{10}$$

$$\frac{H^{\circ}(T)}{RT} = a_1 + \frac{a_2}{2}T + \frac{a_3}{3}T^2 + \frac{a_4}{4}T^3 + \frac{a_5}{5}T^4 + \frac{a_6}{T} \tag{11}$$

where $H^{\circ}(T)$ in eq 11 is the absolute enthalpy,

$$H^{\circ}(T) = \Delta_f H^{\circ}_{298.15\text{K}} + \int_{298.15}^{T} C_p^{\circ}\,dT \tag{12}$$

commonly used in engineering applications, and

$$\frac{S^{\circ}(T)}{R} = a_1\ln(T) + a_2T + \frac{a_3}{2}T^2 + \frac{a_4}{3}T^3 + \frac{a_5}{4}T^4 + a_7 \tag{13}$$

Polynomials 10, 11, and 13 reproduce fitted data within an error between 1×10^{-2} and 1×10^{-3}. Two sets of coefficients, a_1,..., a_7, are provided in the published database for each species: one set is used to compute properties for $200 \le T \le 1000$ and the other set for $1000 < T \le 6000$.

In 1987, McBride and Gordon published a nine-term polynomial database to solve shuttle re-entry problems that were valid for the extended temperature range 200−20 000 K. The nine-term polynomials reproduce fitted data within an error between 1×10^{-4} and 1×10^{-5}. These polynomials are given by

$$\frac{C_p^\circ(T)}{R} = a_1T^{-2} + a_2T^{-1} + a_3 + a_4T + a_5T^2 + a_6T^3 + a_7T^4 \quad (14)$$

$$\frac{H^\circ(T)}{RT} = -a_1T^{-2} + a_2T^{-1}\ln T + a_3 + a_4\frac{T}{2} + a_5\frac{T^2}{3} + a_6\frac{T^3}{4} + a_7\frac{T^4}{5} + \frac{a_8}{T} \quad (15)$$

and

$$\frac{S^\circ(T)}{R} = -a_1\frac{T^{-2}}{2} - a_2T^{-1} + a_3\ln T + a_4T + a_5\frac{T^2}{2} + a_6\frac{T^3}{3} + a_7\frac{T^4}{4} + a_9 \quad (16)$$

An alternative set of polynomials was adopted by the NIST called the Showmate polynomials. These functions were developed in the early 1940s using the method of Shomate.[17] NIST publishes species coefficients of the Showmate polynomials for temperatures in the range 200−6000 K. The Showmate polynomials for standard state heat capacity, enthalpy, and entropy are given as

$$C_p^\circ(t) = a_1 + a_2t + a_3t^2 + a_4t^3 + \frac{a_5}{t^2}\ \text{[kJ/kmol•K]} \quad (17)$$

$$H^\circ(t) - H^\circ_{298.15}(t) = a_1t + a_2\frac{t^2}{2} + a_3\frac{t^3}{3} + a_4\frac{t^4}{4} + \frac{a_5}{t} + a_6 - a_8\ \text{[kJ/mol]} \quad (18)$$

and

$$S^\circ(t) = a_1\ln(t) + a_2t + a_3\frac{t^2}{2} + a_4\frac{t^3}{3} - \frac{a_5}{(2t^2)} + a_7\ \text{[kJ/mol•K]} \quad (19)$$

where

$$t = \frac{T}{1000}\ \text{[K]} \quad (20)$$

THERMOCHEMICAL DATABASE

Using custom software tools that we developed, we extracted and stored the coefficients of all species from the NIST, NASA, TDCAPU, and CHEMKIN[18] thermochemical databases in relational form using a MySQL RDBMS.[19] The database we developed does not simply store polynomial coefficients, however. The database is used to actually compute thermodynamic properties by way of MySQL *stored procedures*.[20] To illustrate this concept, suppose one wishes to compute the standard state entropy of oxygen gas at 298.15 K using the NASA nine-term polynomial given by eq 16. From the MySQL command line processor, one would simply execute

```
mysql > CALL calculateEntropy
                (298.15, 'O2', 0, @h)(21)
```

and then

```
mysql> select @h;
+------------------+
| @h               |
+------------------+
| 205.14852976789 |
+------------------+
```

would display the value for entropy in units of J K^{-1} mol^{-1}. The third argument in expression 21 is an integer used to select a particular phase where integers from the set {0,1,2} select gas-, liquid-, or solid-phase species data, respectively. The use of stored procedures to perform polynomial evaluation relieves the calling client application or Web Service operation from the responsibility of performing necessary arithmetic. Since the database has direct access to the coefficients needed to evaluate a polynomial at a given temperature and an ability to perform arithmetic computation including support for transcendental function evaluation, property computation by the database can reduce the Web application server load that would otherwise be incurred when the calling Web Service operation retrieves coefficients from the database and then constructs and evaluates the polynomial in application code. However, this technique does not necessarily result in faster runtime performance, as it was observed that thermodynamic property polynomial evaluation using stored procedures is significantly slower than having the calling Web Service operation utilize the database just for coefficient retrieval. Figure 1 shows the result of profiling the execution of two Java methods named *storedProcedures()* and *selfEvaluation()* using a Sun Microsystems SunBlade 1000 running the Glassfish Web Application server. The storedProcedures() method computes the standard state specific molar Gibbs free energy of 2,2,4-trimethylpentane (isooctane) from 200−6000 K in increments of 1 K by calling Web Service operations *getH()* and *getS()* to compute $h - Ts$ where both Web Service operations rely on stored procedures to compute enthalpy and entropy. The selfEvaluation() method does the same computation with the exception that the corresponding *getH()* and *getS()* Web Service operations retrieve polynomial coefficients from the database and then evaluate the respective polynomial in

Table 1. Number of Operations Required to Evaluate $\bar{S}_{\text{statistical}}^\circ$ for Linear and Nonlinear Polyatomic Species Having n Atoms

	logarithm and exponential function calls	additions and subtractions	multiplications and divisions	square root function calls
linear species	$6n - 7$	$12n - 16$	$9n - 10$	0
nonlinear species	$6n - 9$	$12n - 20$	$9n - 8$	1

Table 2. Number of Operations Required to Evaluate the NASA 9-Term Polynomial for $\bar{S}^\circ$ for Both Linear and Nonlinear Species[a]

	logarithm and exponential function calls	additions and subtractions	multiplications and divisions	square root function calls
linear and nonlinear species	1	7	14	0

[a] The number of operations does not depend on n.

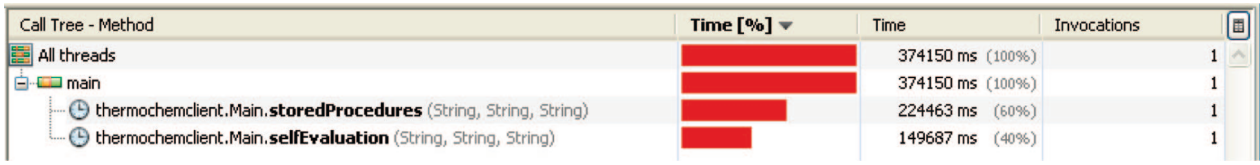

Figure 1. Profile showing a runtime performance comparison between using MySQL stored procedures and a Web Service operation which evaluates the NASA nine-term polynomials for enthalpy and entropy. Profiling was performed on a Sun Microsystems SunBlade 1000 running the Glassfish Web Application server, version b58g.

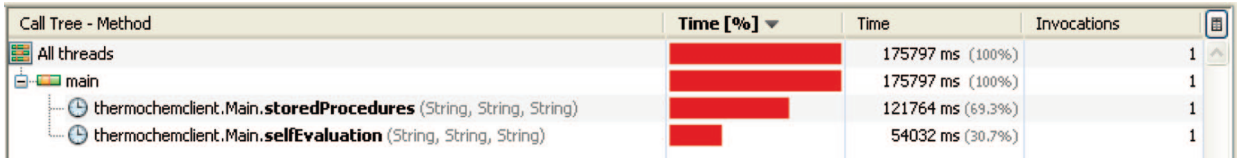

Figure 2. Performance comparison where profiling was performed on a Sun Microsystems SunFire v210 running the Glassfish Web Application server, version b58-rc1.

application code. Figure 2 shows a profile of the same test using Glassfish running on a SunFire v210. In this second test, one can see that using stored procedures to evaluate the NASA nine-term polynomials takes more than twice as much time than evaluating the corresponding polynomial within the Web Service operation's Java code.

Figure 3 shows a schema of our thermochemical relational database. Sets of coefficient data are uniquely identified through the use of three primary keys that store species formula and temperature range. We also maintain a table that is populated with Chemical Abstracts Service (CAS) registry numbers using SciFinder Scholar to allow property retrieval of different isomers that have the same molecular formula given in the polynomial database but a different structural formula. We are currently working on adding the ability to specify species in our Web Service operations using the OpenSMILES specification developed by the Blue Obelisk[3] open-source chemistry community and, additionally, using an IUPAC International Chemical Identifier,[21] or InChI, specification.

WEB SERVICES

As discussed in Dong et al.,[1] the use of W3C standardized Web Services[22] in chemoinformatics applications has led to new methods of aggregating and integrating chemical data. Web Services are a type of middleware that provide a platform-independent method of facilitating machine-to-machine communication. This communication is accomplished through the use of Web application servers that deliver software services to client computers over a computer network. Web application servers are different than traditional Web servers in that an application server will manage and invoke user-supplied code when requested to do so from a client system. Application servers publish a set of publicly available or *exposed* operations available to calling client applications using a standardized interface language called Web Services Description Language (WSDL).[23] A WSDL file is a structured XML document, often publicly accessible, that client applications can access via a URL to determine the exact name, argument specification, and return type of a particular operation. The interface of a Web Service is separate from its actual implementation, and the practice of separating the interface from implementation is a core characteristic of all Web Services. While the interface of every Web Service is specified in standardized WSDL format, the implementation can be in any programming language. This separation allows Web Services to be platform-independent and provide transparent access to legacy code. For example, one could make an old FORTRAN subroutine network accessible by wrapping it a Web Service and publishing its interface in WSDL.

A typical WSDL document that adheres to W3C standards[23] has the format shown in Figure 4. Without going into too much detail, the important elements that deserve attention in the WSDL shown in Figure 4 are the service name, port name, and the message definitions. The *ServiceName* specified in the name attribute of the service element gets mapped by the client proxy code into a class that is instantiated by the client and used to obtain a port. Port objects contain the network address of the remote application server that is hosting the Web Service. Figure 5 shows an excerpt of the WDSL document for the thermochemical data Web Service we developed.

Instead of specifying the number and type of parameters in the message element for each operation as shown in Figure 4, an XML Schema Definition is imported by our WSDL that defines the data types of the parameters required by each exposed operation. For example, Figure 6 shows that the *getCp* operation for retrieving heat capacity data requires four parameters: the chemical species name (formula) as a string, temperature as a double precision real number, phase as a string, and the database name as a string.

On the basis of the WSDL shown in Figure 5, it is a simple matter to instantiate a port and call a Web Service operation. The Java code shown in Figure 7 illustrates this simplicity by showing how one could invoke the getCp operation to obtain the standard state specific heat of nitrogen dioxide gas at 3000 K using data from the NASA nine-term thermochemical database. The fourth argument selects a database to query. In addition to the NASA nine-term database, three other databases (described later) are available for property retrieval. A particular database must be specified in a Web Service operation invocation because differences will exist between $\bar{c}_p^\circ$ values calculated from coefficients in each database. To illustrate such differences, Table 3 shows values of $\bar{c}_p^\circ$ at 3000 K for NO_2 computed from each of the four currently supported databases.

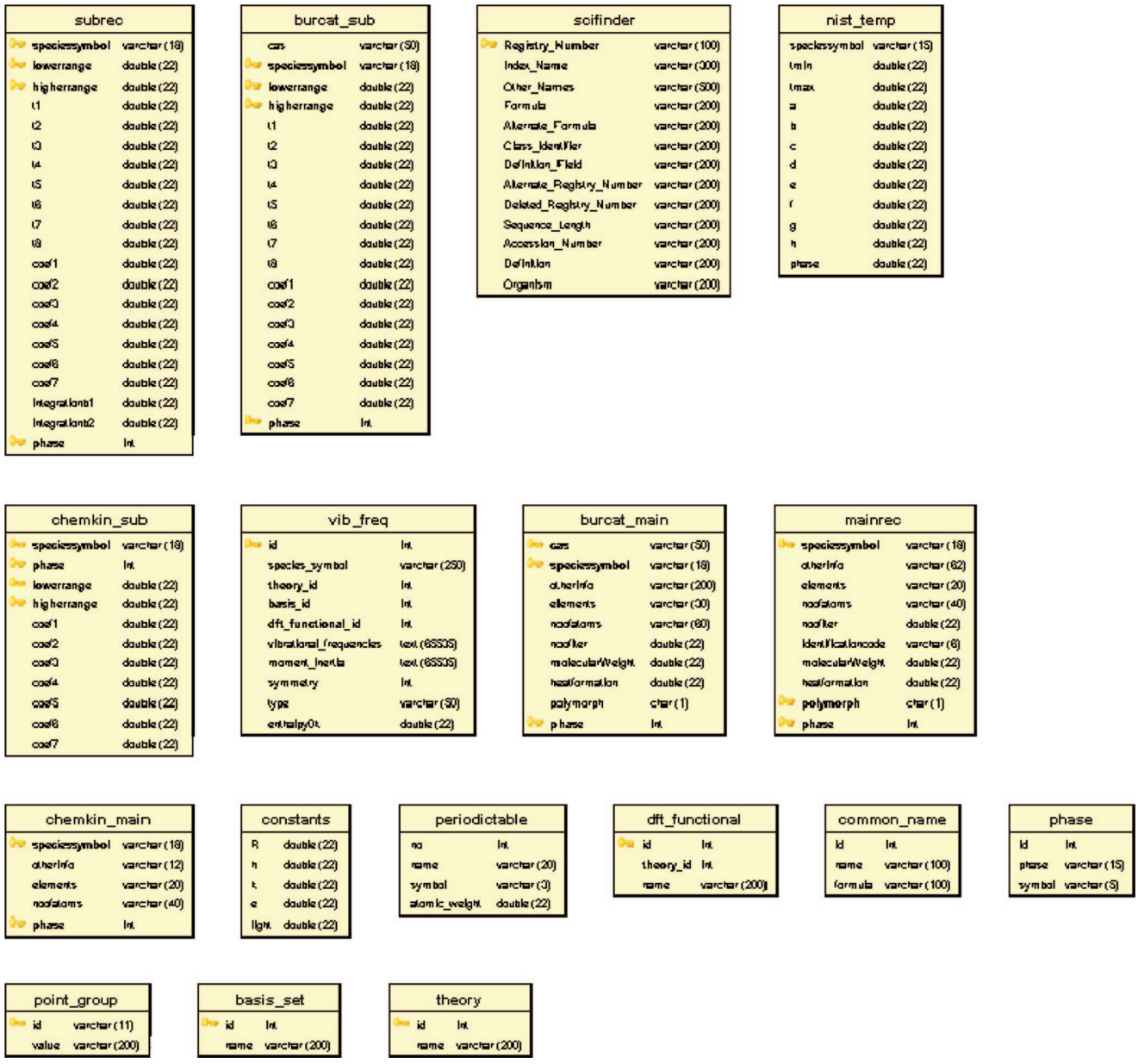

Figure 3. Schema of our thermochemical database showing table and column names. Primary keys are indicated by a yellow key glyph.

THERMOCHEMICAL WEB SERVICES

We developed a thermochemical Web Service that exposes a family of operations to calculate and return thermodynamic properties given temperature, species name or CAS number,[24] phase, and database identifier. The four core thermochemical Web Service operations currently accessible to the public are as follows:

```
double getMw(string speciesNameOrCAS)
```

The getMw operation returns the molecular weight of a species in kilograms per kilomole given the species molecular formula or CAS number.

```
double getCp(string speciesNameOrCAS,
double T, string phase, string database)
```

The getCp operation returns the standard state specific molar heat capacity of a species in kilojoules per kilomole·Kelvin given the species molecular formula or CAS number, temperature T in Kelvin, phase name (one of "gas", "liquid", or "solid"), and a database identifier (one of "NASA", "NIST", "BURCAT", or "CHEMKIN" to select the NASA nine-term coefficient database, Shomate coefficient database, Burcat and Ruscic NASA seven-term coefficient database, or the NASA seven-term coefficient database used by CHEMKIN, respectively).

```
double getH(string speciesNameOrCAS,
double T, string phase, string database
```

The getH operation returns the absolute specific molar enthalpy of a species in kilojoules per kilomole. The required arguments are the same as those specified in the getCp() operation. This operation computes the absolute specific molar enthalpy, $\bar{h}_i$, for species i, by first computing the sensible enthalpy change, $\Delta\bar{h}_i$, from the standard reference temperature, 298.15 K, to the user-supplied value of T using a polynomial, and then adding this value to the species' enthalpy of formation at 298.15 K, $\bar{h}f,i°$, which is retrieved from a table in our relational database. Thus,

$$\bar{h}_i(T) = \Delta\bar{h}_i(T) + \bar{h}^{\circ}_{f,i}(298.15\ \mathrm{K}) \quad (22)$$

```
double getS(string speciesNameOrCAS,
double T, string phase, string database)
```

Finally, the getS operation returns the absolute standard state specific molar entropy of a species in kilojoules per kilomole·Kelvin. The required arguments are the same as those specified in the getCp() operation. The WSDL associated with our thermochemical Web Service is given in Figure 8. Currently, any one of the following strings may be supplied to select data from a particular database: {"NASA", "NIST", "CHEMKIN", "BURCAT"}; we are, however, adding support for other databases.

In addition to the aforementioned core Web Service operations, a number of other auxiliary operations are provided. Table 4 presents a summary of all exposed Web Service operations available to the public. For those who need access to "raw" data such as sets of polynomial coefficients by temperature range, the *getData()* and *getCoefficients()* operations will return coefficient data in XML format.

THERMOCHEMICAL DATA COMPARISON

Because our thermochemical Web Service currently supports four different polynomial coefficient databases from which to compute properties, we found it useful to be able to visually compare data obtained from these four sources. To do so, we developed an AJAX-based[25] visualization tool[26] written in JavaScript that allows a user to select any one of several thousand species for which we have incorporated data into our relational database and dynamically generate plots for standard state heat capacity, enthalpy, and entropy for a given supported temperature range. As new polynomial formulations are added to our database, or new species data for existing formulations are added, one will be able to obtain a qualitative view of how the new species data compares with existing data. Our AJAX-based visualization tool is accessible online via the *Tools* section of http://cheqs.sdsu.edu/. The visualizer tool invokes our data Web Services to produce a plot of the standard state specific heat, enthalpy, and entropy for a given species. These plots can be used to obtain a qualitative understanding of a species' thermodynamic properties. Moreover, different thermochemical data sets can be compared against one another and against data obtained using different theories and basis sets in ab initio methods. To illustrate the use of the visualizer to compare data, Figure 9 shows a plot of the standard state heat capacity of nitrogen dioxide (NO_2) based on data from the NIST, NASA, CHEMKIN, and TDCAPU databases. Notice that the values for heat capacity begin to diverge after about 1000 K. At 4816

```
<definitions targetNamespace="namespace URI" name="ServiceName">
  <!-- Message definitions -->
  <message name="operationName">
    <part name="parameter1" type="dataType"/>
    <part name="parameter2" type="dataType"/>
    .
    .
  </message>
  <message name="operationNameResponse">
    <part name="parameter" element="dataType"/>
  </message>
  <!-- PortTypes that refer to message definitions -->
    <portType name="PortName">
      <operation name="operationName">
        <input message="operationName"/>
        <output message="operationNameResponse"/>
      </operation>
    </portType>
  <!-- Bindings for each operation defined in portType -->
    <binding name="PortNamePortBinding" type="PortName">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
             style="document"/>
       <operation name="operationName">
        <soap:operation soapAction=""/>
           <input>
            <soap:body use="literal"/>
          </input>
           <output>
            <soap:body use="literal"/>
          </output>
      </operation>
      </binding>
  <!-- Services specify the port address for each binding -->
    <service name="ServiceName">
      <port name="PortNamePort" binding="PortNamePortBinding">
        <soap:address location="Service URI"/>
      </port>
    </service>
</definitions>
```

Figure 4. A generic WSDL document.

```
<definitions targetNamespace="http://thermochem/" name="ThermochemicalService">
  <types>
    <xsd:schema>
      <xsd:import                                    namespace="http://thermochem/"
schemaLocation="http://romulus.sdsu.edu:8080/Thermochem/ThermochemicalService?x
sd=1"/>
    </xsd:schema>
  </types>
  .
  .
  <message name="getCp">
    <part name="parameters" element="tns:getCp"/>
  </message>
  .
  .
  <portType name="ThermoChem">
    <operation name="getCp">
      <input message="tns:getCp"/>
      <output message="tns:getCpResponse"/>
    </operation>
  .
  .
  <binding name="ThermoChemPortBinding" type="tns:ThermoChem">
    <soap:binding              transport="http://schemas.xmlsoap.org/soap/http"
style="document"/>
      <operation name="getCp">
        <soap:operation soapAction=""/>
       <input>
          <soap:body use="literal"/>
        </input>
        <output>
          <soap:body use="literal"/>
        </output>
      </operation>
.
.
  <service name="ThermochemicalService">
    <port name="ThermoChemPort" binding="tns:ThermoChemPortBinding">
      <soap:address
location="http://romulus.sdsu.edu:8080/Thermochem/ThermochemicalService"/>
    </port>
  </service>
</definitions>
```

Figure 5. Excerpt from the WSDL of our thermochemical data Web Services showing elements relevant to the *getCp* operation for retrieving heat capacity data.

```
<xs:complexType name="getCp">
  <xs:sequence>
    <xs:element name="species" type="xs:string" minOccurs="0"/>
    <xs:element name="temperature" type="xs:double"/>
    <xs:element name="phase" type="xs:string" minOccurs="0"/>
    <xs:element name="source" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Figure 6. Excerpt from the XML Schema Definition imported by the WSDL shown in Figure 5. The schema shows the number and types of parameters required by the *getCp* operation for retrieving heat capacity data.

```
ThermochemicalService service = new ThermochemicalService();
ThermoChem port = service.getThermoChemPort();
double cp = port.getCp("NO2", 3000.0, "gas", "NASA");
```

Figure 7. Calling the getCp operation of the thermochemical data Web Service to obtain the specific heat capacity of nitrogen dioxide gas at 3000.0 K.

K, there is a 5.43% difference between the NASA and CHEMKIN values.

We have recently begun adding data computed using ab initio methods to our database to compare properties derived

Table 3. Differences between Standard State Specific Molar Heat Values of Nitrogen Dioxide Computed Using Four Different Thermochemical Polynomial Databases

database	$\bar{c}_p^\circ$ [J/mol]
NASA	60.9911
NIST	57.3944
CHEMKIN	57.3971
BURCAT	61.1094

from quantum chemistry methods against properties computed using coefficients from the four aforementioned databases. Figure 10 shows a plot of the standard state enthalpy of methane (CH_4) gas generated using our data visualizer. The brown line is derived from data obtained by running GAMESS[13] using a 3-21g split-valence Pople basis set and second-order Møller−Plesset perturbation theory. In Figure 10, one can see a fairly close agreement between enthalpy values derived from the NIST and CHEMKIN databases and an ab initio analysis. However, after about 3000 K, values of enthalpy obtained using the NASA nine-term database begin to deviate from the other three data sets'.

To facilitate thermodynamic property computation using ab initio methods, we have implemented a preliminary data extraction tool[27] for public use which is accessible from the tools section of the CHEQS Web site http://cheqs.sdsu.edu/. This tool allows one to upload a MDL[28] (.mol) or PDB[29] chemical format file and extract thermochemical properties, which are then permanently stored in our data repository for Web-Service-based retrieval. Figure 11 shows the user interface of this Web application. Once a user uploads a.mol or.pdb file for a particular species, a Web Service is invoked which first converts the incoming chemical format file into a GAMESS input configuration using OpenBabel.[3] GAMESS is then executed on one of our servers and, upon computation of the energy Hessian, the species' moments of inertia, symmetry number, and vibrational frequencies are extracted from the GAMESS output and stored in our thermochemical database. Thermodynamic properties derived from ab initio methods can then be queried using a Web Service or visually compared against properties derived from any of the polynomial-based databases using the visualization tool.

Currently, our data extraction tool only supports restricted Hartree−Fock (RHF) computation with a second-order Møller−Plesset perturbation theory level (MP2), a coupled-cluster calculation for the ground state (CCSD), or CCSD with noniterative triples corrections (CCSD(T)). Additionally, users may select DFT from the *Theory* pull down menu to perform a density functional theory computation instead of an ab initio computation. At this time, DFT is only supported using either the Becke exchange with the Lee/Yang/Parr (LYP) correlation (BLYP) functional or the hybrid HF/Becke/LYP using VWN formula 5 (B3LYP) functional. Computation of the energy Hessian using GAMESS can be a time-consuming process, and we are currently constructing a cluster of hosts that can run GAMESS in a distributed-parallel mode to reduce the time a user must wait for jobs to complete that are submitted using the data extractor tool.

INTEGRATION OF THERMOCHEMICAL DATA WEB SERVICES WITH THIRD PARTY APPLICATIONS

One of the primary advantages of standardized Web Services is platform-independent accessibility. This means one should be able to invoke a Web Service as if it were a third-party software component. To demonstrate this concept, we have developed a Microsoft Excel macro package[30] and a MATLAB toolbox, both downloadable via the *Tools* section of http://cheqs.sdsu.edu/, that access our thermochemical data Web Service to generate a JANAF formatted thermochemical table. Figure 12 shows a screenshot of running our Excel macro package to generate a JANAF table for methane (CH_4) gas.

The macro package uses the Microsoft Office 2003 Web Services toolkit to invoke our thermochemical Web Service for each temperature in the table. Property values are displayed in respective cells so they can be further referenced in a custom calculation. We anticipate researchers who frequently use Excel for chemical thermodynamic work will find this macro package quite useful.

MATLAB JANAF THERMOCHEMICAL TABLE GENERATOR

In addition to Microsoft Excel, the numerical computing package MATLAB[31] is often used by researchers to perform computational work. MATLAB possesses a powerful interpretive scripting engine which allows researchers to develop computational codes with relative ease, compared to development using a compiled language such as FORTRAN or C. Introduced in version 7, MATLAB contains built in functionality to generate SOAP messages and invoke Web Services. This capability allows researchers who use MATLAB to solve chemical thermodynamics problems to interface with our thermochemical data Web Services to dynamically acquire data during runtime. To illustrate, Figure 13 shows an example using the MATLAB thermochemical toolbox we developed to generate a JANAF formatted table from the MATLAB command line. The toolbox is available via the *Tools* section of http://cheqs.sdsu.edu/. To use the toolbox, one simply downloads the janaf.zip archive, extracts the contents to a local drive, and sets the MATLAB working directory to the folder created during extraction. To generate a JANAF formatted table, simply call the *janaf(species, phase, database)* function. For example, to generate a JANAF table for carbon dioxide (CO_2) gas using data from the NASA Glenn Research center, one simply types

```
janaf('CO2','gas','NASA')
```

in the MATLAB command window or within an M-code script. The output of the command above is shown in Figure 13. The following databases may be specified as the third argument to the janaf function: "NASA", "NIST", "CHEMKIN", or "BURCAT". Phase can be any one of "gas", "liquid", or "solid". The toolbox consists of a set of wrapper functions which invoke our thermochemical Web Service and unmarshalls the response into the MATLAB variable space.

```
http://romulus.sdsu.edu:8080/Thermochem/ThermochemicalService?wsdl
```

Figure 8. WSDL of our publicly accessible thermochemical data Web Service.

Table 4. Summary of Exposed Thermochemical Data and Computation Web Service Operations[a]

operation	arguments	purpose
getData	none	returns the entire NASA 9-term coefficient database in XML
getDeltaFormationEnthalpy	*S*,*p*,*db*	returns $\Delta_f^- h_{298.15K}^\circ$ of species *S*from *db* in kJ/kmol
getMw	*S*	returns the molecular weight of *S*in kg/mol
getCp	*S*,*T*,*p*,*db*	returns $\bar{c}_p^\circ(T)$ of *S* from *db* in kJ/(kmol K)
getH	*S*,*T*,*p*,*db*	returns $\bar{h}^\circ(T)$ of *S* from *db* in kJ/kmol
getS	*S*,*T*,*p*,*db*	returns $\bar{s}^\circ(T)$ of *S* from *db* in kJ/(kmol K)
getSpecies	none	returns the molecular formula names of all supported species in XML
getSpeciesCSV	none	returns the molecular formula names of all supported species in comma separated values format
getCoefficients	*S*,*p*,*db*	returns the coefficient data set for *S* from *db* in XML

[a] In the arguments column, *S* is a string containing a species' molecular formula or CAS registry number, *T* is a double precision value for temperature in units K, *p* is a string containing one of {"gas", "liquid", or "solid"} to indicate phase, and *db* is a string identifying the database to query.

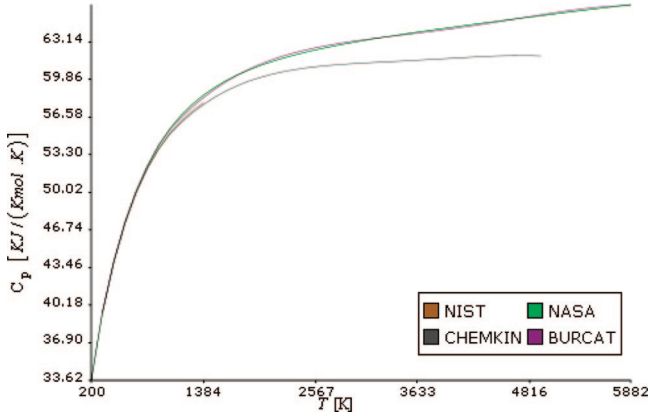

Figure 9. Plot of the standard state heat capacity of nitrogen dioxide (NO_2) showing a 5.43% difference between NASA and CHEMKIN values at 4816 K.

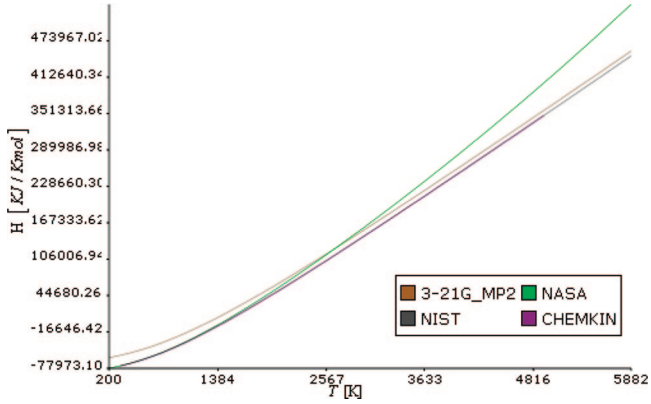

Figure 10. Plot of the standard state enthalpy of methane (CH_4) showing a comparison of ab initio data with that produced using the NASA nine-term and Shomate polynomials.

USING THE MATLAB THERMOCHEMICAL DATA TOOLBOX

The functions provided by the thermochemical data toolbox can be invoked within MATLAB scripts (m-code) to solve a variety of problems. For example, in Figure 14, we see a plot of ln *K* as a function of temperature for each reaction involved in the gasification of solid coal (C(s)) to produce methane gas (CH_4). From the figure, we see that the initial gasification reaction is endothermic as solid coal

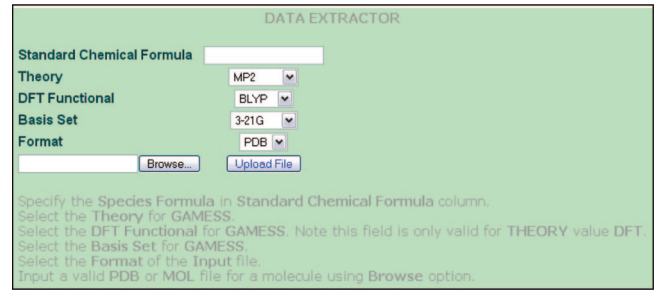

Figure 11. User interface of our Web-based thermochemical data extraction tool.

is heated with steam to produce carbon monoxide and hydrogen via the water gas reaction:

$$C\,(s) + H_2O\,(g) \rightarrow CO\,(g) + H_2\,(g) \tag{23}$$

Reaction 23 is accompanied by the exothermic water gas shift reaction

$$CO\,(g) + H_2O\,(g) \rightleftharpoons CO_2 + H_2\,(g) \tag{24}$$

which balances the concentration of carbon monoxide, water vapor, carbon dioxide, and hydrogen. Methane is then formed from the exothermic reaction of carbon monoxide and hydrogen given by

$$CO\,(g) + 3H_2\,(g) \rightarrow H_2O\,(g) + CH_4\,(g) \tag{25}$$

In order to show ln *K* as a function of *T*, the van't Hoff equation is numerically integrated using MATLAB by first considering the enthalpy of each reaction, ΔH°, as a constant and then, subsequently, as a function of temperature. This example demonstrates how easy it is to solve thermodynamic problems using the MATLAB thermochemical data toolbox. Recall that the van't Hoff equation is given by

$$\left(\frac{d \ln K}{dT}\right)_P = \frac{\Delta H^\circ}{RT^2} \tag{26}$$

If ΔH° is assumed to be constant with respect to temperature, the definite integral of eq 26 from the standard reference temperature $T_1 = 298.15$ K to T_2 is given by

$$\ln\left(\frac{K_{T_2}}{K_{T_1}}\right) = \frac{-\Delta H^\circ}{R}\left(\frac{1}{T_2} - \frac{1}{T_1}\right) \tag{27}$$

The solid lines in Figure 14 show ln K_2 as a function of T computed from eq 27 using MATLAB. The enthalpy of each reaction is computed once using the thermochemical

	A	B	C	D	E	F	G	H	I
1	Species	Mol. W.	Phase	Database	T(K)	C°p[J/(K mol)]	S°[J/(K mol)]	-[G°-H°]/T [J/(K mol)]	H°-H°$_{298}$(kJ/mol)
2	CH4	16.0425	gas	Nasa	0	0.000	0.000	INFINITE	74.600
3					100.00	0.000	0.000	-745.997	74.600
4					200.00	33.506	172.675	189.542	-3.373
5					250.00	34.267	180.224	186.949	-1.681
6					298.15	35.691	186.370	186.370	0.000
7					300.00	35.760	186.591	186.371	0.066
8					350.00	37.956	192.261	186.814	1.907
9					400.00	40.618	197.499	187.826	3.869
10					450.00	43.541	202.450	189.178	5.973
11					500.00	46.585	207.195	190.744	8.226
12					600.00	52.691	216.230	194.246	13.190
13					700.00	58.543	224.796	198.003	18.755
14					800.00	64.012	232.975	201.867	24.886
15					900.00	69.065	240.810	205.762	31.543
16		Generate JANAF Table			1000.00	73.676	248.330	209.645	38.684
17					1100.00	77.849	255.551	213.493	46.264
18					1200.00	81.627	262.489	217.288	54.241
65					5900.00	145.027	439.738	335.915	612.555
66					6000.00	146.214	442.185	337.666	627.117
67									
68					Reported values are for a standard state pressure of 1 bar = 100000 Pa				

Figure 12. Generating a JANAF table for methane gas (CH_4) using Microsoft Excel.

```
>> janaf('CO2','gas','NASA')

JANAF Thermochemical Table for CO2 Mw: 44.01

Standard State Pressure = 1 bar

      T          Cp           S     -[G-H]/T    H-H298

      K   J/(K mol)   J/(K mol)   J/(K mol)    kJ/mol

 200.00      32.361     199.968     217.031    -3.412

 250.00      34.828     207.452     214.385    -1.733

 298.15      37.135     213.786     213.786     0.000

 300.00      37.220     214.016     213.787     0.069

 350.00      39.387     219.920     214.248     1.985

 400.00      41.325     225.308     215.299     4.004

 450.00      43.061     230.277     216.691     6.114

 500.00      44.624     234.897     218.283     8.307

 600.00      47.322     243.279     221.765    12.909

 700.00      49.561     250.747     225.381    17.756

 800.00      51.432     257.491     228.980    22.809

 900.00      52.998     263.642     232.495    28.032

1000.00      54.308     269.296     235.896    33.400

1100.00      55.420     274.526     239.173    38.888

1200.00      56.347     279.389     242.324    44.477

5900.00      66.478     377.231     319.914   338.174

6000.00      66.767     378.351     320.878   344.837
```

Figure 13. Generating a JANAF formatted thermochemical table for carbon dioxide gas using data from the NASA Glenn Research center.

data toolbox since it is assumed constant. The solution to eq 26 where $\Delta H°$ is treated more realistically as a function of temperature is shown in the figure by the dotted lines. Integration of the right-hand side of eq 26 is a computationally intensive process that is made substantially easier using the toolbox. Equation 26 was numerically integrated using the MATLAB adaptive Simpson quadrature function quad() and a wrapper function that implements the right-hand side of eq 26. The wrapper function computes the $\Delta H°$ of a reaction by calling the toolbox getH() function to compute the individual enthalpies of each species and then using

$$\Delta H^{\circ}_{\mathrm{rxn}} = H^{\circ}_{\mathrm{products}} - H^{\circ}_{\mathrm{reactants}} \tag{28}$$

Running on a Fujitsu S Series Lifebook (Intel Pentium M, 1.73 GHz, 1 GB RAM), approximately 1 h and 19 min was required to generate the dotted data shown in Figure 14. The M-code used to solve eq 26 is given in the examples directory found in the janaf.zip archive file available from our Web site. Table 5 shows the number of function evaluations that MATLAB performed to integrate the van't Hoff equation for the three different reactions. We used MATLAB's adaptive Simpson quadrature function, quad(), to perform the integration within an error of 10^{-6}. Each function evaluation causes four invocations of the getH() Web Service operation to compute the enthalpy of reaction at a particular temperature. Thus, the total number of invocations of getH() is 18 604, which translates to approximately four Web-Service-based computations of absolute enthalpy per second within the MATLAB scripting environment.

DYNAMIC INVOCATION OF THERMOCHEMICAL DATA WEB SERVICES

Web Services can be either statically or dynamically invoked. To statically invoke a Web Service, one uses a *Service* class created by a tool that generates portable artifacts. For example, in the Java environment the wsimport tool distributed with the Java API for XML Web Services

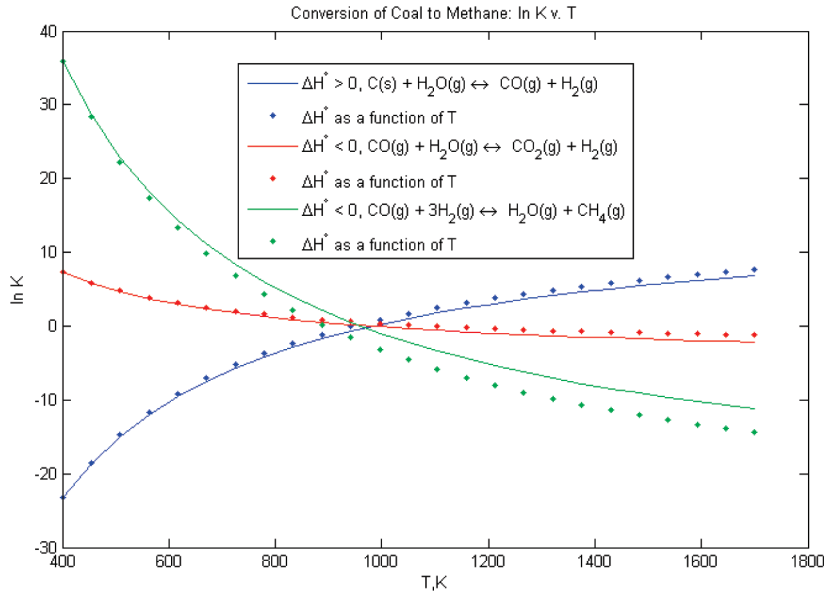

Figure 14. The MATLAB Thermochemical Data Toolbox used to plot ln *K* as a function of temperature for each reaction involved in the conversion of solid coal (C(s)) to methane gas (CH_4).

Table 5. Number of Function Evaluations and Corresponding getH() Operations Required to Integrate the van't Hoff Equation Using MATLAB's Recursive Adaptive Simpson Quadrature Function within an Error of 10^{-6}

reaction	function evaluations	getH() invocations
water gas	1605	6420
water gas shift	1285	5140
methane formation	1761	7044
total	4651	18604

(JAX-WS) generates a Service class from a Web Service description. The Service class is used to invoke a Web Service through its end point and is compiled and linked together with the main application. An alternative approach to Web Service invocation uses the dynamic dispatch interface provided by JAX-WS. The practice of designing dynamic software that self-constructs at runtime through automated service discovery offers the computational chemistry community exciting opportunities for the development of novel and innovative applications. Dynamic applications operate by discovering the URLs of Web Service WSDLs by searching a registry of published descriptions using keywords or taxonomic identifiers. The two most popular types of registries used to publish Web Services are the Universal Description, Discovery, and Integration (UDDI) registry and the Electronic Business XML (ebXML) registry. To circumvent the differences between using either standard, one typically uses an API that encapsulates the technical details involved in locating a Web Service that provides some needed capability. In the Java environment, the Java API for XML Registries[33] (JAXR) provides a standard API for querying both ebXML and UDDI registries. Using JAXR in conjunction with the dynamic dispatch interface provided

```
Org name: San Diego State University

Org description: Thermochemical Data Web Services

Org key id: urn:uuid:19b105e8-d62d-43f4-9b45-8e824072e768

 Contact name: Christopher Paolini

  Phone number: (619) 594-7159

 Service name: San Diego State University:thermochemical

 Service description: Thermochemical Data Web Services

  Binding Description: Thermochemical Data Web Services

  Access URI: http://130.191.166.202:8080/Thermochem/ThermochemicalService?wsdl

 ---

  URI: http://130.191.166.202:8080/Thermochem/ThermochemicalService

  Specific Heat of CO2 at 298.15 K: 37.135217955903
```

Figure 15. Output from running the JAXRFindHeatCapacity example code.

by JAX-WS, one can build truly dynamic chemical informatics applications that span networks of interconnected systems through service discovery and Web Service operation execution.

By using the dynamic dispatch interface[32] provided by JAX-WS, one can invoke our thermochemical Web Service without having to use the Java wsimport tool to generate JAX-WS portable artifacts. The output shown in Figure 15 illustrates how software can dynamically "discover" Web Services using JAXR.[33] The example shown in the figure uses the JAXR API to query our UDDI[34] registry, available for public use at http://uddi.sdsu.edu/, to find the thermochemical data Web Service and instantiate a dynamic invocation object to call the *getCp()* operation and compute the standard state specific molar heat capacity of carbon dioxide gas at 298.15 K.

The code used to produce the output shown in Figure 15 can be downloaded from the *Tools* section of http://cheqs.sdsu.edu/.

CONCLUSION AND FUTURE DIRECTIONS

Electronic access to thermochemical data frequently presents difficulties for developers of computational chemistry software. Many thermodynamic databases exist in print and electronically in a flat-file format, and most comprehensive databases are privately sold, which makes access costly for academic researchers. Furthermore, subtle differences exist among databases in the polynomial models and coefficients used to express equations of thermodynamic properties. This work has shown how standardized Web Services provide a platform-independent way in which researchers can obtain accurate thermochemical data, free of charge, in a variety of ways for use in their own computational research. The codes presented throughout this paper are publicly available from the http://cheqs.sdsu.edu/ Web site. Computational results and data provided by our thermochemical Web Service are in the public domain and may be used for any purpose without requiring any special license. The NASA database is updated whenever a new version of *NASA Chemical Equilibrium with Applications* (CEA)[35,36] is released by the NASA Glenn Research Center. The NIST database is updated using a script that extracts coefficient data from the online NIST Chemistry *Webbook*.[10] The *Third Millennium Thermodynamic Database for Combustion and Air-Pollution Use with Updates from Active Thermochemical Tables* is updated automatically from http://garfield.chem.elte.hu/Burcat/BURCAT.THR on a regular basis. The CHEMKIN NASA seven-term database is updated each time a new version of CHEMKIN[18] is released by Reaction Design Inc.

CheQS is an acronym for Chemical Equilibrium Services and, though not discussed in this paper, offers Web Services for complex chemical equilibrium computation in addition to thermochemical data retrieval. We will continue to improve our thermochemical data Web Service by adding support for additional databases as well as data obtained through ab initio methods. We will soon be implementing the ability to invoke operations such as *getCp()*, *getH()*, and *getS()*, where the respective thermodynamic property is computed using statistical thermodynamics with data (moments of inertia, vibrational frequencies, and symmetry numbers) obtained from ab initio and semiempirical quantum chemistry packages and permanently stored in our thermochemical relational database.

ACKNOWLEDGMENT

This work was supported by NSF Office of CyberInfrastructure CI-TEAM Grant 0753283.

REFERENCES AND NOTES

(1) Dong, X.; Gilbert, K. E.; Guha, R.; Heiland, R.; Kim, J.; Pierce, M. E.; Fox, G. C.; Wild, D. J. Web Service Infrastructure for Chemoinformatics. *J. Chem. Inf. Model.* **2007**, *47*, 1303–1307.

(2) Schuchardt, K. L.; Didier, B. T.; Elsethagen, T.; Sun, L.; Gurumoorthi, V.; Chase, J.; Li, J.; Windus, T. L. Basis Set Exchange: A Community Database for Computational Sciences. *J. Chem. Inf. Model.* **2007**, *47*, 1045–1052.

(3) Guha, R.; Howard, M. T.; Hutchison, G. R.; Murray-Rust, P.; Rzepa, H.; Steinbeck, C.; Wegner, J.; Willighagen, E. L. The Blue Obelisk−Interoperability in Chemical Informatics. *J. Chem. Inf. Model.* **2006**, *46*, 991–998.

(4) Steinbeck, C.; Han, Y.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 493–500.

(5) *International Critical Tables*; Washburn, E., Ed.; McGraw-Hill: New York, 1926−1930; Vols. *I*−*VII*.

(6) Chase, M. W., Jr. NIST-JANAF Thermochemical Tables, 4th ed. *J. Phys. Chem. Ref. Data* **1998**, *9*, 1–1951.

(7) Gordon, S.; McBride, B. J. *Thermodynamic Properties of Chemical Substances to 6000 K*; NASA Report SP-3001, NASA Glenn Research Center: Cleveland, OH, 1963.

(8) Gurvich, L. V. In *Thermodynamic Properties of Individual Substances (TPIS)*, 3rd ed.; Nauka: Moscow, Russia, 1978; 1979; 1981; 1982; Vols. *1*−*4* (in Russian).

(9) Barin, I. In *Thermodynamic Data of Pure Substances*, 3rd ed.; VCH: Weinheim, Germany, 1995.

(10) NIST Chemistry WebBook, NIST Standard Reference Database Number 69, National Institute of Standards and Technology, June 2005 Release. http://webbook.nist.gov/chemistry/ (accessed Mar 9, 2008).

(11) Third Millennium Thermodynamic Database for Combustion and Air-Pollution Use with updates from Active Thermochemical Tables. ftp://ftp.technion.ac.il/pub/supported/aetdd/thermodynamics/ (accessed Mar 12, 2007);http://garfield.chem.elte.hu/Burcat/burcat.html (accessed Mar 12, 2007).

(12) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Montgomery, J. A., Jr.; Vreven, T.; Kudin, K. N.; Burant, J. C.; Millam, J. M.; Iyengar, S. S.; Tomasi, J.; Barone, V.; Mennucci, B.; Cossi, M.; Scalmani, G.; Rega, N.; Petersson, G. A.; Nakatsuji, H.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Klene, M.; Li, X.; Knox, J. E.; Hratchian, H. P.; Cross, J. B.; Bakken, V.; Adamo, C.; Jaramillo, J.; Gomperts, R. Stratmann, R. E.; Yazyev, O.; Austin, A. J.; Cammi, R.; Pomelli, C.; Ochterski, J. W.; Ayala, P. Y.; Morokuma, K.; Voth, G. A.; Salvador, P.; Dannenberg, J. J.; Zakrzewski, V. G.; Dapprich, S.; Daniels, A. D.; Strain, M. C.; Farkas, O.; Malick, D. K.; Rabuck, A. D.; Raghavachari, K.; Foresman, J. B.; Ortiz, J. V.; Cui, Q.; Baboul, A. G.; Clifford, S.; Cioslowski, J.; Stefanov, B. B.; Liu, G.; Liashenko, A.; Piskorz, P.; Komaromi, I.; Martin, R. L.; Fox, D. J.; Keith, T.; Al-Laham, M. A.; Peng, C. Y.; Nanayakkara, A.; Challacombe, M.; Gill, P. M. W.; Johnson, B.; Chen, W.; Wong, M. W.; Gonzalez, C.; Pople, J. A. *Gaussian 03*, Revision C.02; Gaussian, Inc.: Wallingford CT, 2004.

(13) Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. GAMESS (General Atomic and Molecular Electronic Structure System). *J. Comput. Chem.* **1993**, *14*, 1347.

(14) Wilhoit, R. C. *Ideal Gas Thermodynamic Functions*; Thermodynamics Research Center Current Data News, NIST: Boulder, CO, 1975; Vol. 3, No. 2.

(15) Lanzfame, R.; Messina, M. V. Order Logarithmic Polynomials for Thermodynamic Calculations. In *Progress in SI and Diesel Engine Modeling*; Society of Automotive Engineers (SAE) Inc.: Warrendale, PA, 2001.

(16) McBride, B. J.; Gordon, S. *FORTRAN IV Program for Calculation of Thermodynamic Data; NASA TN-D 4097*; NASA Glenn Research Center: Cleveland, OH, 1967.

(17) Shomate, C. H. High-temperature Heat Contents of Magnesium Nitrate, Calcium Nitrate and Barium Nitrate. *J. Am. Chem. Soc.* **1944**, *66*, 928–929.

(18) Kee, R. J.; Rupley, F. M.; Miller, J. A.; Coltrin, M. E.; Grcar, J. F.; Meeks, E.; Moffat, H. K.; Lutz, A. E.; Dixon-Lewis, G.; Smooke, M. D.; Warnatz, J.; Evans, G. H.; Larson, R. S.; Mitchell, R. E.; Petzold, L. R.; Reynolds, W. C.; Caracotsios, M.; Stewart, W. E.; Glarborg, P.; Wang, C.; McLellan, C. L.; Adigun, O.; Houf, W. G.; Chou, C. P.; Miller, S. F.; Ho, P.; Young, P. D.; Young, D. J.; Hodgson, D. W.; Petrova, M. V.; Puduppakkam, K. V. *CHEMKIN*, release 4.1.1; Reaction Design: San Diego, CA, 2007.

(19) *The MySQL Relational Database*, version 5.0 Community Server; Sun Microsystems: Santa Clara, CA, 2008.

(20) Stored Procedures in MySQL 5.0. http://dev.mysql.com/tech-resources/articles/mysql-storedproc.html (accessed May 24, 2007).

(21) McNaught, A. The IUPAC International Chemical Identifier: InChl. *Chem. Int.* **2006**, *28*, 12–15.

(22) W3C Web Services Activity. http://www.w3.org/2002/ws/ (accessed Dec 4, 2007).

(23) Christensen, E.; Curbera, F.; Meredith, G.; Weerawarana, S. Web Services Description Language (WSDL) 1.1; W3C Recommendation. http://www.w3.org/TR/wsdl (accessed Dec 4, 2007).

(24) Chemical Abstracts Service (CAS) Registry. http://www.cas.org/ (accessed May 26, 2007).

(25) Garrett, J. J. Ajax: A New Approach to Web Applications; Adaptive Path, LLC, Feb 18, 2005. http://www.adaptivepath.com/publications/essays/archives/000385.php (accessed Dec 5, 2007).

(26) Devalia, B. V. *Preliminary Implementation of Thermochemical Data Web Services*, Master's Thesis, San Diego State University, San Diego, CA, 2006.

(27) Jain, H. B. *A Web Service Based Testing Framework for Thermodynamic Data*, Master's Thesis, San Diego State University, San Diego, CA, 2008.

(28) Dalby, A.; Nourse, J. G.; Hounshell, W. D.; Gushurst, A. K. I.; Grier, D. L. Description of Several Chemical Structure File Formats used by Computer Programs Developed at Molecular Design Limited. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 244–255.

(29) Berman, H. M.; Henrick, K.; Nakamura, H. Announcing the Worldwide Protein Data Bank. *Nat. Struct. Biol.* **2003**, *10*, 980.

(30) Chan, W. W. *A Service-Oriented Architecture (SOA) Model for Performing Chemical Equilibrium Analysis in a Distributed Framework by Consuming Java-based Equilibrium Web Services*, Master's Thesis, San Diego State University, San Diego, CA, 2007.

(31) *MATLAB*, version R2006a; The MathWorks Inc.: Natick, MA, 2007.

(32) Butek, R.; Gallardo, N. Web Services Hints and Tips: JAX-RPC versus JAX-WS, Part 4 The Dynamic Invocation Interfaces. http://www.ibm.com/developerworks/library/ws-tip-jaxwsrpc4/index.html (accessed Nov 28, 2007).

(33) Java API for XML Registries (JAXR). http://java.sun.com/webservices/jaxr/ (accessed Dec 6, 2007).

(34) UDDI. http://www.uddi.org/ (accessed Dec 6, 2007).

(35) Gordon, S.; McBride, B. J. *Computer Program for the Calculation of Complex Chemical Equilibrium Compositions and Applications - I. Analysis*; NASA Reference Publication 1311, NASA Glenn Research Center: Cleveland, OH, October 1994.

(36) McBride, B. J.; Gordon, S. *Computer Program for the Calculation of Complex Chemical Equilibrium Compositions and Applications - II. Users Manual and Program Description*; NASA Reference Publication 1311; NASA Glenn Research Center: Cleveland, OH, June 1996.

CI700457P