ELSEVIER

Journal of Hazardous Materials B:63 (1998) 145–162

Journal of Hazardous Materials

An object-oriented approach to numerical methods: the *Regula Falsi* method for solving equations with tight tolerances for environmental applications

J.B. Phillips *, G. Price, S. Fry, T. Arcziscewski, S. DeMonsabert, A.S. Menawat

Engineering Development Institute, 7925-A.N. Oracle Rd., Suite 155, Tucson, AZ 85704, USA

Received 1 December 1997; revised 17 June 1998; accepted 18 June 1998

Abstract

The *Regula Falsi* method for numerical solution of a system of algebraic equations containing a single equation and a single unknown is implemented with an object-oriented methodology. The method is applied to the solution of equations predicting atmospheric emissions from combustion point sources. Two physical approaches are taken in modeling the combustion phenomenon: a kinetic approach and an equilibrium approach. Emissions from the kinetic approach are found to be significantly smaller than those from the equilibrium approach, indicating marked potential for overall industrial emissions reduction through improvements in the design of burners for boilers, incinerators, and furnaces. The object-oriented methodology displays significant benefits in terms of modularity, code reuse, and future upgrades.

Keywords: Numerical methods; Air emissions; Object-oriented programming; *Regula Falsi* method; Phillips number; Combustion; Reaction kinetics; Chemical equilibrium

1. Introduction

Recent federal and state environmental regulations have resulted in increased interest in technology for accurately calculating emissions of all types of air pollutants. Much of this activity was spurred by the Clean Air Act Amendments of 1990, which resulted shortly thereafter in the United States Environmental Protection Agency's (EPA)

* Corresponding author. Tel.: +1 504 825 7046.

0304-3894/98/$ - see front matter
PII: S0304-3894(98)00180-0

promulgation of the Benzene NESHAP (National Emission Standards for Hazardous Air Pollutants) Regulation [1] and the HON (Hazardous Organic NESHAP) Rule [2]. The Benzene NESHAP Regulation applies to a wide variety of emissions of benzene, a known carcinogen, from industrial and commercial sources; the HON Rule applies to emissions of 189 toxic or carcinogenic substances, including a number of halogenated organics that are suspected carcinogens. More recently, the Title V emission inventory program [3,4] has been enacted at the federal level and is being enforced by the states. Under Title V, all industrial operations must be analyzed for their effects on air quality. Specifically, an emission inventory for all major pollutant categories (i.e. particulate matter, carbon dioxide, carbon monoxide, volatile organics, and oxides of nitrogen and sulfur) must be prepared.

One of the most common methods of preparing emission inventories is to utilize emission factors compiled by the State of California and subsequently adopted by the United States EPA. These emission factors, commonly referred to as the AP-42 emission factors [5,6], were compiled as a result of measuring emissions from many common industrial and commercial operations. Point sources contained in the AP-42 documentation include boilers, incinerators, furnaces, flares, internal combustion engines, rail car loading stations, tank truck loading stations, chemical mixing/blending operations, compressors, storage tanks, service stations, motor vehicle tanks, barges, FCC units, cokers, fugitive emissions, and other industrial equipment items. Emission factors were measured through stack tests and equivalent measurement methods, and accordingly are considered to be among the most accurate and reliable methods for estimating emissions from a wide range of industrial operations.

The methodology for employing an emission factor to estimate emissions is to multiply the pertinent emission factor by the throughput for a given industrial operation. For example, the emission factors for combustion of natural gas are reported in the AP-42 documentation in units of kilograms of emissions (speciated as above into particulate matter, sulfur dioxide, etc.) per million cubic meters of fired material. Since it generally is easier to measure the input to an industrial operation than it is to measure the output, especially in the case of a combustion process, the emission factor methodology can simplify accurate determination of emissions.

However, there are some drawbacks to using an emission factor methodology. It must be recognized that the use of emission factors will estimate emissions *on average* for various categories of industrial operations. Thus, there is no guarantee that using an emission factor will provide the emissions for any specific equipment item or unit operation. This is significant for a number of reasons. First, since the emission factors were compiled in 1985, recent improvements in equipment design have resulted in the manufacture of industrial processing equipment that emits less than similar equipment did several years ago. Therefore, operators of industrial facilities may well be over-reporting their emissions if they choose to calculate their emission inventories by using emission factors. Typically, since facility operators have a certain degree of latitude in deciding which method to use in calculating emission inventories for their facilities (as long as the method they select is reasonable and defensible from an engineering perspective), they will select the method that yields the lowest calculated emissions. This generally results in facilities with equipment emitting less than the emission factors

would predict (i.e. better than average equipment) calculating emissions via stack tests or other chemical analyses, and facilities with equipment emitting more than the emission factors would predict (i.e. worse than average equipment) calculating emissions using the emission factors. The net result is an overall under-reporting of emissions.

Other disadvantages inherent to the emission factor approach include the lack of data pertaining to emissions from many unusual industrial operations, and the inability of emission factors to predict emissions from combustion of non-traditional fuels.

New designs of equipment items that are more efficient and emit less pollutants have resulted in an increased interest in calculating the emissions of pollutants based on a methodology other than the use of emission factors. A recent approach based on the kinetics of reaction has suggested that estimating the residence time of the combusted material in the flame would allow a more accurate prediction of effluent levels from said process equipment [7].

One area of improvement along these lines is the prediction of emissions from process equipment containing a combustion process. Typically, such a process will involve feeding fuel (or other combustible materials) into a piece of process equipment, then allowing a combustion process to occur, and routing the combustion products through a stack for discharge into the atmosphere (or into a stack gas scrubbing unit). Some of the recent design improvements involve pre-mixing combustible materials with air (or oxygen) prior to exposure to an ignition source, and improving residence time in the combustion unit. It is well established that time, temperature, and turbulence are the major factors affecting combustion efficiency [14].

There are (at least) two significant methodologies to calculating emission from combustion point sources. The first of these involves estimating the kinetics of reaction and the residence time in the combustion locus (interpreted in this work to refer to the flame, and not the plenum where combustion materials reside at a temperature typically much lower than that existing inside a flame), in an effort to determine the fraction of uncombusted starting materials. The second of these involves conducting an equilibrium calculation to determine the maximum theoretical combustion efficiency that can be attained.

Consider a combustion reaction of the form

$$\nu_{\text{fuel}}\text{fuel} + \nu_{\text{O}_2}\text{O}_2 \rightarrow \nu_{\text{CO}_2}\text{CO}_2 + \nu_{\text{H}_2\text{O}}\text{H}_2\text{O} \tag{1a}$$

(Generally, the stoichiometric coefficient for fuel, ν_{fuel}, will be set to unity.) In order to conduct the kinetic equation, the rate of reaction equation must be considered:

$$r = -\frac{\text{d}[\text{fuel}]}{\text{d}t} \tag{1b}$$

where

$$\frac{\text{d}[\text{fuel}]}{\text{d}t} = k[\text{fuel}]^a[\text{O}_2]^b \tag{2}$$

Initial concentrations of fuel and oxygen fed into the system are known (or can be calculated with data on hand). Eq. (2) can be rearranged to

$$\frac{\mathrm{d}[\mathrm{fuel}]}{k[\mathrm{fuel}]^{a}[\mathrm{O}_2]^{b}} = \mathrm{d}t \tag{3}$$

$[\mathrm{O}_2]$ can be expressed as $[\mathrm{O}_2]_{\mathrm{initial}} - \nu_{\mathrm{O}_2}([\mathrm{fuel}]_{\mathrm{initial}} - [\mathrm{fuel}])$. Integration of Eq. (3) yields:

$$\int \left[\frac{\mathrm{d}[\mathrm{fuel}]}{k[\mathrm{fuel}]^{a}\left([\mathrm{O}_2]_{\mathrm{initial}} - \left(\nu_{\mathrm{O}_2}([\mathrm{fuel}]_{\mathrm{initial}} - [\mathrm{fuel}])\right)\right)^{b}} \right] = t \tag{4}$$

In general, the initial oxygen concentration can be expressed as a function of fuel concentration:

$$[\mathrm{O}_2]_{\mathrm{initial}} = (1 + (\%\mathrm{XS}/100))\,\nu_{\mathrm{O}_2}[\mathrm{fuel}]_{\mathrm{initial}} \tag{5}$$

where %XS is the percent excess oxygen used to accomplish the combustion process. If a theoretical amount of oxygen is used, %XS = 0. Thus, Eq. (4) can be expressed as

$$\int \left[\frac{\mathrm{d}[\mathrm{fuel}]}{k[\mathrm{fuel}]^{a}\left((1 + (\%\mathrm{XS}/100))\,\nu_{\mathrm{O}_2}[\mathrm{fuel}]_{\mathrm{initial}}] - ([\mathrm{fuel}]_{\mathrm{initial}} - [\mathrm{fuel}])\right)^{b}} \right] - t = 0 \tag{6a}$$

or, when a theoretical amount of oxygen is used,

$$\int \left[\frac{\mathrm{d}[\mathrm{fuel}]}{k[\mathrm{fuel}]^{a}\left(\nu_{\mathrm{O}_2}[\mathrm{fuel}]\right)^{b}} \right] - t = 0 \tag{6b}$$

The initial concentrations of fuel and oxygen are known. Also, the overall rate constant for combustion, k, generally is known [9]. Moreover, for typical combustion reactions the residence time can be found through a correlation that is the subject of recent work [7]. Therefore, for a given residence time in the reactor (combustion unit), this leads to a system of one equation and one unknown (i.e. the final fuel concentration), which can be solved using a variety of numerical methods. Since the final reaction conversion is high (and, correspondingly, the final concentrations of oxygen and fuel in the denominator are low), extremely tight tolerances are needed in order to solve this equation numerically. This is especially true because of the small term(s) in the denominator of Eq. (3).

To solve the problem of maximum theoretical combustion efficiency, consider the combustion reaction for an aliphatic hydrocarbon:

$$\mathrm{C}_n\mathrm{H}_{2n+2} + (1.5n + 0.5)\mathrm{O}_2 \rightarrow n\mathrm{CO}_2 + (n+1)\mathrm{H}_2\mathrm{O} \tag{7}$$

If one were to find the maximum theoretical conversion of hydrocarbon to carbon dioxide and water, this would occur at the state of chemical equilibrium. The standard

enthalpy of reaction for the combustion reaction may be found by summing the products of the standard enthalpies of formation and the stoichiometric coefficients:

$$\Delta H^{o}_{Rx} = \Sigma\left(\nu_i \Delta H^{o}_{f,i}\right) \tag{8}$$

The standard enthalpies of formation for a wide variety of substances (i.e., reactants and products in this instance) have been compiled in the JANAF (Joint Army, Navy, Air Force) tables [8]. For the combustion of methane and butane, the relevant data are given in Table 1.

Enthalpies of formation are provided in the JANAF tables at standard conditions of 1 atm and 298 K. Therefore, the energetics of reaction (or, in this case, combustion) computed via the stoichiometrically weighted sums of the energetics (i.e. free energy, enthalpy, entropy, or Gibbs free energy) are provided at those standard conditions as well. Naturally, most reactions do not occur at standard conditions, especially combustion reactions, which occur at or near the adiabatic flame temperature. Therefore, through the use of the energetics of combustion data from Table 1, along with heat capacity data for chemical species involved in combustion reactions, adiabatic flame temperatures can be calculated for combusted species. An assumption must be made whether the combustion is conducted with pure oxygen (or, in an unusual instance, an alternative oxidant), or with air (and if so, at what percent excess theoretical air). Ideal gas heat capacities can be calculated using data of the form

$$C_P = a + bT + cT^2 + dT^3 + eT^4 \tag{9}$$

Specific values used in this work for the heat capacity data are listed in Table 2. Care should be taken when employing the heat capacity data as contained in Table 2. The data are valid only at temperatures up to 1500 K. Thus, if a potential solution procedure would integrate to a temperature significantly above 1500 K, a physically unrealistic solution would be attained due to the negative coefficient in the quartic term for nitrogen. Hence, the adiabatic flame temperature can be calculated by evaluating the heat capacity at 1500 K and using this 'average' heat capacity, i.e.

$$T_{ad} = T_{in} - \frac{\Delta H^{o}_{Rx}}{C_{P_{av}}} \tag{10}$$

In this calculation, the heat capacity pertains to the flue gas, which specifically consists of the mole-weighted average of gases produced by the combustion process of the fuel with air.

Table 1
Standard enthalpies of formation and combustion pertaining to combustion of methane and butane

	Standard enthalpy of formation (kcal/mol)	Standard enthalpy of combustion (kcal/mol)
Methane	−17.889	−191.759
Butane	−29.812	−635.385
Carbon dioxide	−94.052	NA
Water	−57.7979	NA
Oxygen	0 (by definition)	NA

Table 2
Ideal heat capacity data used in the calculation of adiabatic flame temperatures (C_p, J/mol, K)

	Oxygen	Water	Carbon dioxide	Nitrogen
a	29.8832	34.0471	19.0223	29.4119
b	$-1.13842*10^{-2}$	$-9.65064*10^{-3}$	$7.796291*10^{-3}$	$-3.00681*10^{-3}$
c	$4.33779*10^{-5}$	$3.29983*10^{-5}$	$-7.37067*10^{-5}$	$5.45064*10^{-6}$
d	$-3.70082*10^{-8}$	$-2.047467*10^{-8}$	$3.74572*10^{-8}$	$5.13186*10^{-9}$
e	$1.01006*10^{-11}$	$4.30228*10^{-12}$	$-8.13304*10^{-12}$	$-4.30228*10^{-12}$

Although combustion reactions generally are presumed to go to completion, the measurements on which the AP-42 emission factors are based demonstrated that a finite fraction of uncombusted fuel remains in the stack gas. While it has been documented that combustion reactions do not proceed to equilibrium [7], this would represent the minimum theoretical emission level that could be expected from a combustion point source. Accordingly, if combustion equipment is found to produce emissions approaching the theoretical minimum, then additional modifications to equipment design will yield little or no reduction in emissions.

The maximum theoretical conversion that could be attained in a combustion process would be at the state of chemical equilibrium. The adiabatic flame temperature, along with the enthalpy of combustion and Gibbs free energy of combustion, can be employed to calculate the equilibrium constant at the adiabatic flame temperature. The equilibrium constant at standard conditions is found by the relation

$$K^{\circ} = \exp\left(\frac{-\Delta G^{\circ}_{\mathrm{Rx}}}{RT}\right) \tag{11}$$

The standard Gibbs free energy of combustion can be determined in a manner analogous to that used for the standard enthalpy of combustion, i.e.,

$$\Delta G^{\circ}_{\mathrm{Rx}} = \Sigma\left(\nu_i \Delta G^{\circ}_{f,i}\right) \tag{12}$$

Data pertaining to the energetics of methane and butane combustion are contained in Table 3.

The Gibbs–Helmholtz equation can be employed to calculate the equilibrium constant at the adiabatic flame temperature by assuming that the enthalpies of reaction are

Table 3
Standard Gibbs free energies of formation and combustion pertaining to combustion of methane and butane

Species	Standard Gibbs free energy of formation (kcal/mol)	Standard Gibbs free energy of combustion (kcal/mol)
Methane	−12.14	−199.234
Butane	−4.111	−646.105
Oxygen	0 (by definition)	NA
Carbon dioxide	−94.26	NA
Water	−54.6351	NA

Table 4
Calculated parameters pertaining to the combustion of methane and butane

Species	T_{ad}, K	K°	$K_{T_{ad}}$
Methane	2245	$1.343 * 10^{146}$	$6.146 * 10^{32}$
Butane	2439	$4.435 * 10^{473}$	$8.983 * 10^{64}$

not nearly as dependent on temperature as the Gibbs free energies of reaction [10]. This gives the following relation for the equilibrium constant at the adiabatic flame temperature:

$$K_{T_{ad}} = \exp\left[\left(\frac{\Delta G^{o}_{Rx}}{RT^{o}}\right) + \left(\frac{\Delta H^{o}_{Rx}}{R}\right)\left[\left(\frac{1}{T_{ad}}\right) - \left(\frac{1}{T^{o}}\right)\right]\right] \tag{13}$$

The results of the calculations represented by Eqs. (11)–(13) are shown in Table 4 for the combustion of methane and butane.

Knowledge of the inlet conditions (i.e. air:fuel ratio) allows calculation of the final composition that would be attained at equilibrium through the use of the equilibrium constant. Typical process conditions would set the combustion pressure at 1 atm, so the equilibrium expression may be written as follows:

$$K_{T_{ad}} = \Pi C_i^{\nu i} = \Pi P_i^{\nu i} \tag{14}$$

For combustion of methane, the equilibrium equation is:

$$K_{T_{ad}} = \frac{[CO_2][H_2O]^2}{[CH_4][O_2]^2} = \frac{P_{CO_2} P^2_{H_2O}}{P_{CH_4} P^2_{O_2}} \tag{15}$$

And for combustion of butane, the equilibrium equation is:

$$K_{T_{ad}} = \frac{[CO_2]^4[H_2O]^5}{[C_4H_{10}][O_2]^{6.5}} = \frac{P^4_{CO_2} P^5_{H_2O}}{P_{C_4H_{10}} P^{6.5}_{O_2}} \tag{16}$$

All of the partial pressures in Eq. (15) can be expressed in terms of the partial pressure of methane, and all of the partial pressures in Eq. (16) can be expressed in terms of the partial pressure of butane, i.e.

$$P_{CO_2} = P_{CO_2,max}\left(P_{CH_4,o} - P_{CH_4}\right) \tag{15a}$$

$$P_{H_2O} = 2\,P_{CO_2,max}\left(P_{CH_4,o} - P_{CH_4}\right) \tag{15b}$$

$$P_{O_2} = P_{O_2,o} - 2\left(P_{CH_4,o} - P_{CH_4}\right) \tag{15c}$$

$$P_{CO_2} = P_{CO_2,max}\left(P_{C_4H_{10},o} - P_{C_4H_{10}}\right) \tag{16a}$$

$$P_{H_2O} = 1.25 P_{CO_2,max}\left(P_{C_4H_{10},o} P_{C_4H_{10}}\right) \tag{16b}$$

$$P_{O_2} = P_{O_2,o} - 6.5\left(P_{C_4H_{10},o} - P_{C_4H_{10}}\right) \tag{16c}$$

$P_{CO_2,max}$ is the maximum partial pressure that could be observed if all of the fuel (methane or butane) were converted to carbon dioxide and water. Substituting into Eqs. (15) and (16) yields the following results:

$$K_{T_{ad}} = \frac{\left(a - bP_{CH_4}\right)\left(2a - 2bP_{CH_4}\right)^2}{P_{CH_4}\left(c + 2P_{CH_4}\right)^2} \tag{17}$$

for methane, and

$$K_{T_{ad}} = \frac{\left(d - eP_{C_4H_{10}}\right)^4\left(1.25d - 1.25eP_{C_4H_{10}}\right)^5}{P_{C_4H_{10}}\left(f + 6.5P_{C_4H_{10}}\right)^{6.5}} \tag{18}$$

for butane. The constants a through f are based on the inlet composition and reaction stoichiometry. Therefore, Eqs. (17) and (18) each represent a system of one equation and one unknown, the equilibrium pressure of the fuel species. Rearrangement of Eqs. (17) and (18) yield

$$K_{T_{ad}} - \frac{\left(a - bP_{CH_4}\right)\left(2a - 2bP_{CH_4}\right)^2}{P_{CH_4}\left(c + 2P_{CH_4}\right)^2} = 0 \tag{19}$$

$$K_{T_{ad}} - \frac{\left(d - eP_{C_3H_8}\right)^4\left(1.25d - 1.25eP_{C_4H_{10}}\right)^5}{P_{C_4H_{10}}\left(f + 6.5P_{C_4H_{10}}\right)^{6.5}} = 0 \tag{20}$$

which are in the form $f(P) = 0$. Attempts to solve Eq. (19) using Lotus, MathCAD, and Qbasic were not successful, due to the extremely small term in the denominator and the large equilibrium constant. However, the reverse reactions can be examined,

$$CO_2 + 2H_2O \rightarrow CH_4 + 2O_2 \tag{21}$$

$$4CO_2 + 5H_2O \rightarrow C_4H_{10} + 6.5O_2 \tag{22}$$

in which carbon dioxide and water form methane (or butane) and oxygen. This is equivalent to rearranging Eqs. (17) and (18), with the equilibrium constant for the reverse reaction (formation of fuel) being the inverse of the equilibrium constant for the forward reaction (combustion of fuel), i.e.

$$K_{rev} = K_{T_{ad}}^{-1} \tag{23}$$

which gives

$$K_{rev} - \frac{P_{CH_4}\left(c + 2P_{CH_4}\right)^2}{\left(a - bP_{CH_4}\right)\left(2a - 2bP_{CH_4}\right)^2} = 0 \tag{24a}$$

$$K_{rev} - \frac{P_{C_4H_{10}}\left(f + 6.5P_{C_4H_{10}}\right)^{6.5}}{\left(d - eP_{C_4H_{10}}\right)^4\left(1.25d - 1.25eP_{C_4H_{10}}\right)^5} = 0 \tag{24b}$$

Although the details are not shown, an analogous procedure was followed to determine equilibrium emissions of heptane as well. Solution of Eqs. (6a), (6b), (24a) and (24b) are the focus of the remainder of this work.

2. The Regula Falsi method

A number of numerical methods exist for the solution of systems of one equation and one unknown of the form $f(x) = 0$, with f: ♦› ♦(or, $f(P) = 0$, in the situation at hand). Successive substitution, Newton's method, the secant method, and the *Regula Falsi* method are among the most common of these. Each has its own distinct set of advantages and disadvantages. Newton's Method is a reliable predictor of the root of such equations, but it requires that the derivative of the function be known and calculated at each step in the procedure. The Secant Method is easier to implement, because it relies only on function values instead of requiring both a function evaluation and a derivative evaluation at each step, but it can be unstable and has the potential to diverge from the root. Successive substitution is particularly known for its potential to fail to converge on solutions to stiff equations, or when the requirements for tolerances are tight.

The *Regula Falsi* Method is not without its disadvantages, in that it requires some advance knowledge about the behavior of the function and where the root lies. However, once the solution is bounded (i.e. with the knowledge that $\exists\ x_1 \ni f(x_1) < 0$ and $\exists\ x_2 \ni f(x_2) > 0$), then convergence upon the root through the use of the *Regula Falsi* Method is extremely reliable. Theoretically, solution to within nearly any desired tolerance may be obtained, with the limitation generally being that of machine precision. Subsequent iterations on x are obtained through the relation

$$x_{i+1} = x_i - f_i \frac{(x_i - x_o)}{(f_i - f_o)} \tag{25}$$

The *Regula Falsi* method is shown graphically in Fig. 1. The graphical representation of the *Regula Falsi* method consists of drawing a line connecting the first two points, one

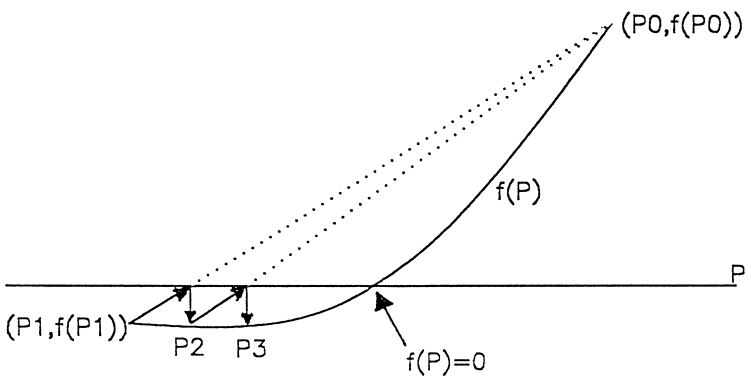

Fig. 1. The *Regula Falsi* method.

known to be above the solution $(P_0,\ f(P_0))$ and one known to be below the solution $(P_1,\ f(P_1))$. This line crosses the P axis at P_2. The function then is evaluated at P_2, and a line is drawn from $(P_0,\ f(P_0))$ to $(P_2,\ f(P_2))$. Again, the line drawn crosses the P axis, this time at P_3. In this manner the solution is converged upon until $f(P)$ is within a desired level of tolerance. It is a near certainty that, in contrast to Aitken's Δ^2 method [21,22], the *Regula Falsi* method will not yield an exact solution. However, a solution to nearly any desired level of tolerance may be specified. Thus, the *Regula Falsi* method may be less than satisfying from a mathematician's perspective, but it generally can more than adequate when applied to a physical situation.

There have been a number of recent developments in the environmental field that have necessitated the ability to calculate concentrations of pollutants to within extremely tight tolerances. Part of this need has been driven by advances in analytical chemistry, and the concomitant capacity to analyze picomolar, femtomolar, or even attomolar concentrations of trace substances within a sample. Accordingly, a version of the *Regula Falsi* Method has been employed recently in an effort to meet this need [15].

Refinements to the *Regula Falsi* method have been driven by the fact that its order of convergence is unity (linear convergence) [16]. The order of convergence is a parameter that describes how efficiently and quickly the root is approached. Linear convergence, as in the case of the *Regula Falsi* method, corresponds to a larger number of iterations required to approach the root. In contrast, the Newton–Raphson method has an order of convergence of 2 (quadratic convergence). As mentioned previously (vide supra), the Newton–Raphson method requires function derivatives at each iteration, and can become numerically cumbersome when the derivative formulae become involved. However, the question remains whether a numerical method of cubic or higher order of convergence should be employed. Unfortunately, numerical methods displaying more rapid convergence result in rapidly increasingly complexity of iterative formulae [17]. Thus, a need persists to improve upon numerical methods such as the *Regula Falsi* method in an effort to preserve its features of near-certain convergence to very tight tolerances and straightforward numerical implementation while reducing the number of iterations.

Simultaneously, advances in approaches to computing in high-level languages have resulted in a move away from procedural methodologies and toward object-oriented methodologies. C + + [11], Visual Basic [12] and Java [13] are among the most widely used object-oriented programming languages, and their utility suggests that many processes that previously were accomplished via procedural methodologies can be conducted more efficiently by using an object-oriented approach. It is recognized that this approach is more straightforward and easier to implement than the approaches taken by Werner [18] and Della Doro and Di Crescenzo [19]. Other refinements to the *Regula Falsi* method have applied the concept of approaching the root simultaneously from both sides of the domain. This is referred to as the modified false position method [20].

An object-oriented school of thought has led to the development of a variety of methodologies for implementing object-oriented programming, object-oriented design, and object-oriented analysis of scientific and engineering problems. Booch, Rational–Rose, and Rumbaugh have been among the most prominent of the object-oriented methodologies, and have revolutionized many approaches to complex problems and their

solutions [24]. However, the use of numerical methods for the solution of engineering problems represents virgin territory for an object-oriented approach, and is the subject of this work. An object-oriented approach is applied to the *Regula Falsi* method for the purpose of solving equations with extremely tight tolerances, as are required for many environmental calculations.

3. Object-oriented programming (OOP)

The major difference between a procedural methodology and an object-oriented methodology is that an object-oriented methodology allows a program to designate objects that are members of a pre-defined class, and then uses class member functions to manipulate the objects [23]. In contrast, a procedural methodology calls procedures to manipulate numbers in a variety of functions, but does not automatically associate a set of variables with each other (or with an object, as in the case of object-oriented programming).

An object-oriented approach to programming focuses on data and behavior that relates to the data. Data and functions processing the data are considered classes whose instances are objects. Objects are variables belonging to a class (generally a user-defined class). Related to this is the concept of an Abstract Data Type (ADT), which can be considered as a user-defined extension to the base data structures provided by a high level language package. An ADT comprises a set of values and a group of functions for which the data represent the domain of said functions.

OOP embodies a number of properties that are designed to promote facile implementation of ADT's. One of the most useful of these is the concept of inheritance, in which a new data structure can be derived from an existing one. Chemical nomenclature is in many respects similar to this mechanism. For example, organics and aromatics are types of chemical species. If the information encoding organic substances applies to all instances of that class, then creating the class 'aromatic' from the class 'organic' (i.e. creation of a subclass) represents an avoidance of duplication of effort.

In OOP, classes need to define not only data structures (objects, in particular), but also the operations that can be carried out on such objects. For example, if a polynomial is an ADT, then user-defined operations must be encoded for functions such as polynomial addition, subtraction, and multiplication. These are referred to as class member functions, in that they are specific to objects, or instances of a given class. Programming should be provided during class definition for any operations that objects could be expected to experience.

Another feature of OOP is the concept of encapsulation. Some operations are unavailable to objects or other data structures outside a particular class. Encapsulation includes the internal implementation details of a particular type, as well as the externally available operations and functions that can act on objects of that type. Details regarding the execution of certain operations can be hidden from user-provided code that implements the data type. For example, if a program to calculate emissions from point sources uses a class member function to convert units from concentration of emitted species (in a stream of given mass flow rate) to tons per year of pollutants, changing the internal

workings of the unit conversion code should have no effect on the accuracy of the calculation. In short, the implementation of the unit conversion is hidden from its clients.

OOP can require a more sophisticated understanding of programming procedures than top-down programming or procedural methodologies. In other programming methodologies objects generally are implicitly created, and subsequently destroyed after being used. In contrast, C + + offers user control of dynamic memory allocation. Selection and definition of objects in C + + allows memory to be apportioned for the object and its associated data. (If a data structure or object is not used at some point in the program, memory can at that juncture be deallocated as well.) Thus, selection of objects is a key to a number of facets in an OOP approach to problem-solving.

In the case of the *Regula Falsi* method, the obvious choice of an object is the line that connects the two points that bound the solution. The approach taken in this work involves manipulating the object through the use of class member functions that refine one endpoint of the line until the function value is within the specified tolerance.

4. Results and discussion

The results from the emission calculations are contained in Table 5. In agreement with previous work [9], the emissions of heavier hydrocarbons are in lower partial pressures than emissions of lighter hydrocarbons. This is not to state that overall emissions, including emissions of partially-combusted species, would be lower for heavier compounds than for lighter ones, but rather that emissions of unreacted fuel species is lower for heavier hydrocarbons than for lighter ones.

As expected, emissions obtained from the kinetic approach are lower than those obtained from the equilibrium approach. This verifies previous work that combustion does not proceed to completion, and moreover, indicates that there is significant room for improvement in the field of equipment design to promote more complete combustion in industrial boilers, incinerators, and furnaces. The fact that there is over a three order of magnitude difference between the equilibrium and kinetic approaches suggests that marked reduction in overall industrial emissions could accrue from additional work in this area.

Appendix A contains the complete C + + code for the program. Lines 2–4 include the standard C + + functions for input/output, mathematical functions, and the standard library (which contains the exit function, among others). Lines 6–11 define the line class. All data elements and functions of this class are public, which means that they can

Table 5
Results of emissions calculations for kinetic and equilibrium approaches

Species	Approach	Emission concentration (mol/l)	Emission partial pressure (atm)
Methane	Equilibrium Eqs. (24a) and (24b)	$4.804 * 10^{-13}$	$1.174 * 10^{-11}$
Butane	Equilibrium Eqs. (24a) and (24b)	$2.225 * 10^{-15}$	$5.436 * 10^{-14}$
Butane	Kinetic Eqs. (6a) and (6b)	$8.202 * 10^{-12}$	$2.004 * 10^{-10}$
Heptane	Equilibrium Eqs. (24a) and (24b)	$3.812 * 10^{-16}$	$9.315 * 10^{-15}$

be accessed from the main body of the function. Line 8 specifies the data which define a line, specifically the two x values (x_1 and x_2), as well as the two y values, alternatively referred to as function values (f_1 and f_2). Lines 9 and 10 apply the *Regula Falsi* iteration formula, Eq. (25). In an object-oriented analysis, these two functions modify the line by determining a new endpoint for the line. Examination of Fig. 1 clarifies why two such functions are required. If the initial points are $(P_o,\ f(P_o))$ and $(P_1,\ f(P_1))$, then when P_2 is selected, a line will need to be drawn either from $(P_o,\ f(P_o))$ or $(P_1,\ f(P_1))$ to $(P_2,\ f(P_2))$. If $f(P_2) < 0$, then $(P_o,\ f(P_o))$ remains as an endpoint for the domain of the calculation. Otherwise, $(P_1,\ f(P_1))$ is retained as an endpoint for the domain and $(P_o,\ f(P_o))$ is discarded. Accordingly, line 63 tests whether the new function value, referred to as ftemp in the code in Appendix A (or, $f(P_2)$ in Fig. 1) is greater than zero or less than zero. Lines 61 and 71 call the two *Regula Falsi* iterative formulae, where one is based on retaining the point at the upper end of the domain in the graphical representation, and the other is based on retaining the point at the lower end of the domain.

Lines 13 through 25 implement the function that is to be optimized. Appendix A shows the function for Eqs. (6a) and (6b), but this could be modified to optimize any function of interest. Line 27 is the beginning of the main body of the program. Lines 29 and 30 declare variables, as follows:

fabs = the absolute value of the function;
tol = the tolerance specified in line 40;
x_{temp} = the new function value (corresponding to P_2 in Fig. 1);
f_{temp} = the function value corresponding to x_{temp};
i = the number of iterations, which is set to one in line 34 and incremented in line 60;
i_{max} = the maximum number of iterations
manipx = a Boolean variable set to zero or unity depending on whether the new function value (f_{temp}) is greater or less than zero.

Line 31 declares the object, line l1. Line 37 sets the maximum number of iterations, arbitrarily chosen to be 100 for this example. Line 40 sets the tolerance, which was chosen to be 10^{-15} for the solution of Eqs. (6a) and (6b). Since the solution of Eqs. (6a) and (6b) yielded a concentration on the order of 10^{-12}, a tolerance of approximately three orders of magnitude smaller was chosen in order to make the calculation meaningful. Lines 43 through 53 solicit input from the user regarding the initial guesses for the solution. As stated previously (vide supra), one of the initial guesses must be below the solution, and one must be above the solution.

Lines 57 through 59 give output regarding the x and y values for each iteration, as well as the iteration number, in order to provide the user with an idea of whether convergence on a solution is progressing. Line 61 applies the *Regula Falsi* method in determining a new estimate for the solution, and line 62 provides the function value corresponding to the new estimate. If the new function value is less than zero, then lines 64 through 68 assign the endpoint x_1 in line l1 to that new estimate for the solution, and assign f_1 in line l1 to the function value corresponding to that new estimate. The Boolean variable manipx1 is set to unity to denote that endpoint x_1 was manipulated by the *Regula Falsi* method. If the new function value xtemp is greater than zero, lines 70 through 75 perform a converse set of operations.

Lines 77 through 80 assign the absolute value of the new function value to the variable fabs. Line 82 tests whether the iteration conditions have been met (specifically, whether the new estimate of the solution has yielded a function value within the specified tolerance, or whether the maximum number of iterations have been conducted.

Lines 84 through 87 provide output for the final iteration, with a test of whether the solution is on the lower or upper end of the domain. Lines 89 through 92 tell the user whether the iteration was stopped by reaching a solution that was within levels of tolerance, or whether the maximum number of iterations were reached. Line 94 uses the EXIT_SUCCESS function in stdlib.h to denote whether the function terminated successfully.

It is believed that this work represents virgin territory in the sense that an object-oriented approach is applied to numerical methods for the solution of stiff equations. Benefits are expected to accrue from the features of object-oriented methodologies, particularly encapsulation, inheritance, abstraction, and data sharing. Moreover, this work is expected to facilitate the process of upgrading software and extending to related applications.

One major requirement of the *Regula Falsi* method, as stated previously (vide supra), is that there exist some knowledge regarding the approximate domain within which the root of the equation lies. The quality of the initial estimates for the root impact the number of iterations significantly. As indicated in Table 5, the solution to Eqs. (6a) and (6b) is at a concentration of $8.20148 * 10^{-12}$ mol/l. The number of iterations required to converge upon this solution is a function of the quality of the two initial estimates. This is shown in Fig. 2, in which the number of iterations required is shown as a function of the two initial estimates. As each of the initial estimates approaches the root, the number of iterations required decreases monotonically.

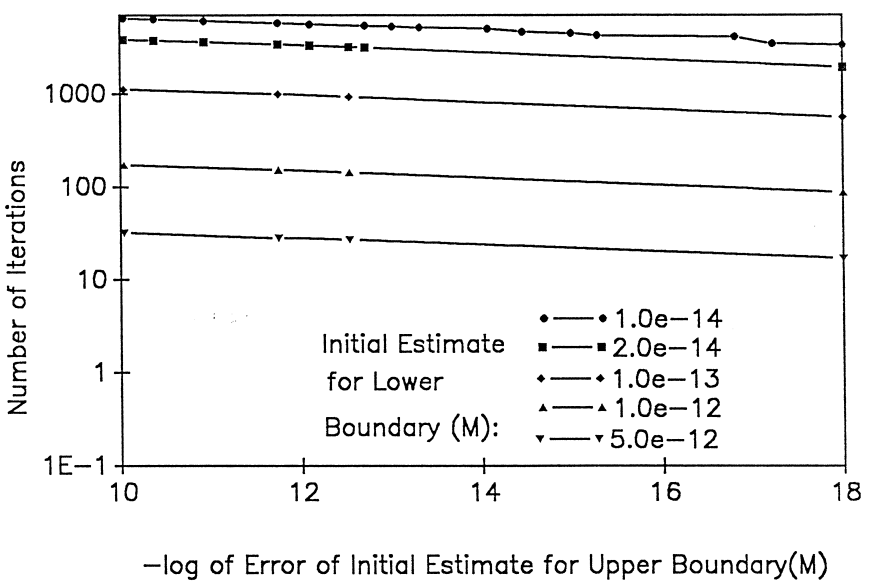

Fig. 2. Number of iterations required for convergence as a function of quality of initial estimates.

One method to bracket the solution between two known function values is by trial and error. Once the program accepts initial estimates that define the domain of the solution space, using either endpoint too distant from the root results in excessive numbers of iterations required. However, by setting the maximum number of iterations to be a relatively low number initially, the program can assist in narrowing the domain, so a subsequent run can be initiated with more accurate estimates that define the solution space. Although this procedure must be conducted with user input at present, automating this procedure with the use of artificial intelligence will be the subject of future work.

5. Summary and conclusions

This work has applied principles of object-oriented programming to the solution of algebraic equations using numerical methods. An object-oriented methodology has been applied to the *Regula Falsi* method for the solution of systems of a single equation with a single unknown. This approach has facilitated the solution of equations pertaining to emissions of pollutant species from industrial point sources for the purpose of facilitating preparation of Title V operating permits. For point sources containing a combustion process, emissions using a kinetic model were compared with those using an equilibrium model. Emissions were found to be significantly lower for the kinetic model than for the equilibrium model, suggesting that significant reductions in overall industrial emissions can be realized through improvements in burner design.

A major advantage in applying an object-oriented methodology is that programs become more modular and reusable. Accordingly, future work is anticipated to build upon this treatise in improving the method described herein to accelerate convergence and provide other desirable program features.

Appendix A. Complete C + + code

```
//Object-Oriented Regula Ralsi Method
 include <iostream.h>
# include <math.h>
# include <stdlib.h>

class line {
public:
   double x1, x2, f1, f2;
   double Regfalsl(line *pt) {return ((*pt).xt - (f1 * ((x2 - x1)/(f2 - f1))));}
   double Regfals2(line *pt) {return ((*pt).x2 - (f2 * ((x2 - x1)/(f2 - f1))));}
};

double f (double x)
```

```
{
   double k = 8.56e8; /* mol/l-sec */
   double NuO2 = 6.5;
   double Finitial = 0.00117; /* mol/l */
   double t = (1.0/82.91);
   double a = 0.15;
   double b = 1.6;
   double factor, parens;
   factor = (1/(k*pow(NuO2,b)));
   parens = (pow(x, (1.0 - a - b)) - pow(Finitial, (1.0 - a - b)));
   return factor * parens - t;
}

inst main ()
{
double fabs, tol, xtemp, ftemp;
int i, imax, manipx1;
line l1;

//Initialize counter
i = 1;

//Set maximum iterations
imax = 100;

//Set tolerance
tol = 0.000000000000001;

//Initialize guesses
do {
   cout < "Input an estimate of x below the root (i.e., f(x)) < 0)\n";
   cin >> l1.x1;
   l1.f1 = f(l1.fl > 0.0);
} while (l1.f2 < 0.0);

do {
   cout < "Input an estimate of x above the root (i.e., f(x) > 0)\n";
   cin >> l1.x2;
   11.f2 = f(l1.x2);
} while (l1.f2 < 0.0);

do {
   cout < "\n\nx1 = " <l1.x1 < " f1 = " < l1.f1 < " x2 = " < l1.x2
      < " f2 = " < l1.f2;
   cout < " i = " < i;
   ++i;
   xtemp = l1.Regfalsl(&l1);
   if (ftemp < 0.0)
```

```
    {
    l1.x1 = xtemp;
    l1.f1 = f(l1.x1);
    manipx1 = 1;
    }
  else
    {
    xtemp = l1.Regfals2(&l1);
    l1.x2 = xtemp;
    l1.f2 = f(l1.x2);
    manipx1 = 0;
    }
  if (ftemp > 0.0)
    fabs = ftemp;
  else
    fabs = -ftemp;
} while ((fabs > tol) && (i < max));

if (manipx1 == 0)
  cout << "\nSolution is x = " << l1.x2 << " f2 = " << l1.f2 << " i = " << i;
else
  cout << "\nSolution is x = " << l1.x1 << " f1 = " << l1.f1 << " i = " << i;

if (i < imax)
  cout << "\nMaximum iteration not reached." ;
else
  cout << "\nMaximum iteration reached." ;

return (EXIT_SUCCESS);
}
```

References

[1] U.S. Environmental Protection Agency, National emission standards for hazardous air pollutants; benzene waste operations: final rule, Federal Register, Washington, DC (1993) pp. 3072–3105.

[2] U.S. Environmental Protection Agency, Regulation of emissions of certain organic halogenated air pollutants and revision of test methods: proposed rules, Federal Register, Washington, DC (1992) pp. 62608–62806.

[3] S. Shepard, Clean Air Permitting Manual, McGraw-Hill, New York, 1995, p. 1ff.

[4] R.L. Burke, Permitting for Clean Air, Air and Waste Management Assoc., Pittsburgh, 1992, p. 1ff.

[5] U.S. Environmental Protection Agency, Compilation of air pollutant emission factors, Vol. 1, Stationary point and area sources, Federal Register, Washington, DC, 1985, p. 1.4–3.

[6] U.S. Environmental Protection Agency (Off. Air Qual. Plann. Stand., Research Triangle Park, NC), Compilation of air pollutant emission factors supplement number 7, U.S. NTIS, PB Rep. PB-270281, from Gov. Rep. Announce. Index, U.S. Government Printing Office, 77(22), 1977, pp. 156ff.

[7] J.B. Phillips, Prediction of air emission factors for combustion point sources, Ind. Eng. Chem. Res. 35 (1996) 2007–2014.

[8] R.C. Reid, J.M. Prausnitz, T.K. Sherwood, The Properties of Gases and Liquids, McGraw-Hill, New York, 1977.
[9] F.L. Dryer, The phenomenology of modeling combustion chemistry, in: W. Bartok, A.F. Sarofim, (Eds.), Fossil Fuel Combustion, A Source Book, Wiley, New York, 1991, pp. 121–213.
[10] J.M. Smith, H.C. Van Ness, Introduction to Chemical Engineering Thermodynamics, McGraw-Hill, New York, 1975.
[11] R. Sedgewick, Algorithms in C++, Addison Wesley, New York, 1992, p. 1ff.
[12] P.G. Aitken, Visual Basic for Windows 95 Insider, Wiley, New York, 1996, p. 1ff.
[13] A.E. Walsh, Java™, IDG Books Worldwide, Foster City, CA, 1996, p. 1ff.
[14] M.L. Tapper, The non-steady burning of liquid fuel droplets in a diesel-like environment, West. States Sect., Combust. Inst., [Pap.] (1976) 76–79.
[15] J.B. Phillips, A.S. Menawat, S.R. Carden, Modification of the *Regula Falsi* method to accelerate system convergence in the prediction of trace quantities of atmospheric pollutants, J. Haz. Materials 44 (1995) 25–35.
[16] G. Dahlquist, A. Bjorck, Numerical Methods, Prentice-Hall, Englewood Cliffs, NJ, 1974, p. 218ff.
[17] P. Jarratt, A review of methods for solving nonlinear algebraic equations in one variable, in: P. Rabinowitz (Ed.), Numerical Methods for Nonlinear Algebraic Equations, Gordon and Breach, London, 1970, p. 1ff.
[18] W. Werner, Some efficient algorithms for the solution of a single nonlinear equation, J. Comput. Math. 9 (B) (1981) 141–149.
[19] J. Della Doro, C. Di Crescenzo, Approximants de Pade-Hermite, 1'ere partie: theorie, Numer. Math. 43 (1984) 23–39.
[20] R.W. Hamming, Numerical Methods for Scientists and Engineers, 2nd edn., McGraw-Hill, New York, 1973, p. 59ff.
[21] A.C. Aitken, On Bernoulli's numerical solution of algebraic equations, Proc. R. Soc. Edinburgh 46 (1926) 289–305.
[22] C. Brezinski, Survey on convergence acceleration methods in numerical analysis, Math. Student 46 (1) (1978) 28–41.
[23] B. Stroustrup, The C++ Programming Language, 2nd edn., Addison-Wesley, Reading, MA, 1991, p. 1ff.
[24] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, Englewood Cliffs, NJ, 1991, pp. 7–10.