EDUCAÇÃO e TECNOLOGIA

Revista do Instituto Politécnico da Guarda

EDUCAÇÃO E TECNOLOGIA

Propriedade
Instituto Politécnico da Guarda

Director
João Bento Raimundo

Redacção
Serviços Centrais do IPG - Quinta do Zambito
6300 Guarda
telf. 222634 * telecópia 222690

Composição
Gabinete Editorial do IPG

Execução Gráfica e Impressão
Secção de Reprografia do IPG

Periodicidade
Semestral

Tiragem
1.000 ex.

Depósito Legal
nº 17.981/87

nº XI - Fevereiro de 1993

PARA ALÉM DA MEMÓRIA ...

Mercê de um esforço determinado, orientado desde o início por princípios de valorização do potencial humano e regional, foi possível, ao longo dos últimos seis anos, dotar a região com a realidade que é actualmente o Instituto Politécnico da Guarda.

Concretizámos igualmente a abertura do Pólo de Seia deste Instituto; para além do seu alcance cultural e social, ficou bem evidenciado que, quando há diálogo, empenho colectivo, preocupação pelos interesses da comunidade, o progresso resulta no tempo presente e imprime perspectivas de futuro. Partilhamos assim da opinião de João de Araújo Correia, "*não é preciso que os homens sejam anjos. Mas o tempo que a mesquinhez desperdiça em mesquinharias, se fosse aproveitado, cinquenta por cento que fosse, em elevadas missões, faria de meio fruste um grande meio e, de meio grande, um meio sublime*".

O Instituto Politécnico da Guarda — com toda a sua estrutura humana, técnica e administrativa — protagonizou a mudança, integrou-se na comunidade regional, assumiu-se como motor de desenvolvimento nas suas múltiplas facetas; caracterizou nesta interligação real, permanente, o símbolo do Portugal moderno e do papel de grande responsabilidade que incumbe ao ensino superior politécnico. Tal responsabilidade passa também por uma actividade editorial que seja incentivo constante a novos trabalhos, à reflexão e à investigação. É esse o desafio que a nossa Revista deixa em cada edição, entrelaçando-se na obra que está consubstanciada neste Instituto Politécnico.

Como escreveu o Padre António Vieira, "*as razões próprias nascem do entendimento, as alheias vão pegadas à memória, e os homens não se convencem pela memória, senão pelo entendimento*".

João Raimundo
Presidente da Comissão Instaladora
do Instituto Politécnico da Guarda

UMA APLICAÇÃO DO CÁLCULO SIMBÓLICO À TERMODINÂMICA

Álvaro Bento Leal*

RESUMO: *Apresenta-se uma solução baseada no processamento de símbolos matemáticos por computador, de um problema clássico da termodinâmica: a transformação das derivadas de 2ª ordem da equação fundamental de um sistema simples.*

A utilização do computador pelos físicos e engenheiros, costuma seguir um modelo padrão: quando as relações matemáticas atingem alguma complexidade, são transformadas em algoritmos numéricos equivalentes e, nessa forma, são utilizadas.

O procedimento é natural dado que o que se pretende é, regra geral, obter um conjunto de valores numéricos e, por isso, o momento em que se converte o modelo analítico em modelo numérico não é, em princípio, importante. É claro que isto acontece só em princípio, porque muitas vezes a transformação numérica das relações introduz erros não aceitaveis no processo de cálculo.

Sucede mesmo que o método pode tornar-se inviável devido ao conhecido problema da estabilidade numérica.

Os modelos numéricos possuem uma característica muito importante: são de aplicação mais geral, na medida que não estão dependentes da "arte" de trabalhar relações entre símbolos, como sucede nos modelos analíticos.

O aspecto interessante no domínio dos computadores é que a capacidade de trabalhar automaticamente símbolos, operações essas que aqui, talvez não muito adequadamente, se designarão por *cálculo simbólico*, foi reconhecida praticamente desde o início da sua utilização. No início da história moderna dos computadores o cálculo simbólico estava, e ainda está, muito associado à feitura dos compiladores e interpretadores.

* Professor Coordenador da ESTG.

O exemplo que aqui se vai tratar mostra que existem problemas no domínio da Física, cuja abordagem pelo cálculo simbólico, é perfeitamente viável e permite a obtenção da melhor solução.

O PROBLEMA TERMODINÂMICO

Seguindo o modelo termodinâmico apresentado em [1], os estados de equilíbrio de um sistema simples, que não permuta massa, são caracterizados pela função contínua, diferenciável duas vezes, de 1ªs derivadas não nulas num domínio finito das variáveis independentes:

$$u = u(s, v) \tag{1}$$

onde: u - energia interna específica por unidade de massa
s - entropia interna específica por unidade de massa
v - volume interno específico por unidade de massa

A diferenciação de (1) conduz a:

$$du = \left(\frac{\partial u}{\partial s}\right)_v ds + \left(\frac{\partial u}{\partial v}\right)_s dv = Tds - Pdv \tag{2}$$

onde: $T = T(s, v) = \left(\frac{\partial u}{\partial s}\right)_v$ é a temperatura

$P = P(s, v) = -\left(\frac{\partial u}{\partial v}\right)_s$ é a pressão

Dada a existência de (1), (2) é uma diferencial exacta, pelo que (relação de Maxwell):

$$\frac{\partial^2 u}{\partial s \partial v} = \frac{\partial^2 u}{\partial v \partial s} \Rightarrow \left(\frac{\partial T}{\partial v}\right)_s = -\left(\frac{\partial P}{\partial s}\right)_v \tag{3}$$

A pressuposta existência de (1) não significa o conhecimento da sua expressão analítica. Tal só acontece em sistemas conceptuais baseados em modelos muito simples. A previsão do comportamento de um sistema real tem de recorrer a dados obtidos em condições experimentais bem controladas.

A forma de sintetizar as medidas experimentais é diversa: gráficos, tabelas e coeficientes, sendo todos eles equivalentes do

ponto de vista da informação que fornecem. Actualmente, dada a utilização em grande escala do computador no cálculo, a preferência vai para a utilização dos coeficientes termodinâmicos.

Os coeficientes termodinâmicos são expressões definidas em termos das 1ª e 2ª derivadas da equação fundamental (1). Atendendo a que só existem 3 derivadas independentes de 2ª ordem, de (1), conclui-se que, conhecidas as derivadas de 1ª ordem, o número de coeficientes termodinâmicos necessário e suficiente é 3; obviamente que os seus valores variam de estado de equilíbrio para estado de equilíbrio.

Tradicionalmente usam-se 4 coeficientes, embora esteja sempre presente que um pode determinar-se à custa dos restantes.

Assim:

$$(4)\quad \begin{cases} C_P = T\left(\frac{\partial s}{\partial T}\right)_P & \text{calor específico a pressão constante} \\ C_v = T\left(\frac{\partial s}{\partial T}\right)_v & \text{calor específico a volume constante} \\ \alpha_P = \frac{1}{v}\left(\frac{\partial v}{\partial T}\right)_P & \text{coeficiente de dilatação isobárica} \\ \kappa_T = -\frac{1}{v}\left(\frac{\partial v}{\partial P}\right)_T & \text{coeficiente de compressibilidade} \end{cases}$$

A relação existente entre eles é:

$$(5)\quad C_P - C_v = Tv\frac{\alpha_P^2}{\kappa_T}$$

Em termos práticos, o conhecimento experimental de (4) é usado da seguinte forma: numa evolução do sistema entre dois estados de equilíbrio próximos, as variáveis termodinâmicas x e y sofrem variações Δx e Δy; pretende-se conhecer a variação Δz duma variável $z\,(x, y)$. Ter-se-á:

$$(6)\quad \Delta z \simeq \left(\frac{\partial z}{\partial x}\right)_y \Delta x + \left(\frac{\partial z}{\partial y}\right)_x \Delta y$$

Se se puder traduzir $\left(\frac{\partial z}{\partial x}\right)_v$ e $\left(\frac{\partial z}{\partial y}\right)_x$ em termos dos coeficientes (4), supostos conhecidos, o problema fica resolvido.

A caracterização do sistema termodinâmico é feita em termos de variáveis às quais é possível atribuir um significado físico claro e relevante. Para além das mencionadas u, s, v, T e P, o estudo da termodinâmica conduziu a atribuir também significado especial às transformadas de Legendre da equação (1):

(7)

$$\begin{cases} f = f(T,v) = u - Ts & \text{potencial específico de Helmholtz} \\ h = h(s,P) = u + Pv & \text{entalpia livre específica} \\ g = g(T,P) = u - Ts + Pv & \text{energia livre específica de Gibbs} \end{cases}$$

As expressões diferenciais destas funções são:

(8)
$$\begin{cases} df = -sdT - Pdv \\ dh = Tds + vdP \\ dg = -sdT + vdP \end{cases}$$

O MÉTODO DOS JACOBIANOS

O problema proposto é calcular:

(9)
$$\left(\frac{\partial z}{\partial x}\right)_y = \frac{\partial(z,y)}{\partial(x,y)} = \begin{pmatrix} \left(\frac{\partial z}{\partial x}\right)_y & 0 \\ \left(\frac{\partial z}{\partial y}\right)_x & 1 \end{pmatrix}$$

onde $x, y, z \in [u, f, h, g, s, v, P, T]$, em termos de s, v, T, P e C_P, C_v, α_P, k_T, definidos em (4).

De acordo com o método apresentado em [2], considerem-se duas variáveis independentes α e β fixas que, por agora, não necessitam de particularização. Pode fazer-se (9) igual a:

(10)
$$\left(\frac{\partial z}{\partial x}\right)_y = \frac{\frac{\partial(z,y)}{\partial(\alpha,\beta)}}{\frac{\partial(x,y)}{\partial(\alpha,\beta)}}$$

Dado que α e β são fixas em tudo o que se segue, adopta-se para os Jacobianos relativos a essas variáveis, a convenção:

(11)
$$\mathbf{J}(z,y) \equiv \frac{\partial(z,y)}{\partial(\alpha,\beta)}$$

(10) Vem então:

$$\left(\frac{\partial z}{\partial x}\right)_y = \frac{\mathbf{J}(z,y)}{\mathbf{J}(x,y)} \tag{12}$$

Esta expressão serve de base a um método extremamente simples de resolver o problema proposto. A chave desse método reside no facto de em (12) o quociente de $J(z, y)$ por $J(x, y)$ não ser uma convenção simbólica de escrita, como acontecia com $\delta(z, y)$ e $\delta(x, y)$ em (9), mas sim ter o significado algébrico da divisão.

No seguimento apresentar-se-á o essencial do método para a solução do problema proposto; na referência [2] encontram-se outros desenvolvimentos.

Dada a definição (11), pode fazer-se:

$$\mathbf{J}(x,x) = 0 \tag{13}$$

$$\mathbf{J}(x,y) = -\mathbf{J}(y,x) \tag{14}$$

A aplicação de (12) e (14) a (3) conduz a:

$$\frac{\mathbf{J}(T,s)}{\mathbf{J}(v,s)} = \frac{\mathbf{J}(P,v)}{\mathbf{J}(s,v)} \Rightarrow \frac{\mathbf{J}(T,s)}{\mathbf{J}(v,s)} = \frac{\mathbf{J}(P,v)}{\mathbf{J}(v,s)}$$

$$\mathbf{J}(T,s) = \mathbf{J}(P,v) \tag{15}$$

As expressões diferenciais (2) e (8) permitem escrever, para qualquer variável x:

$$\left(\frac{\partial u}{\partial y}\right)_x = T\left(\frac{\partial s}{\partial y}\right)_x - P\left(\frac{\partial v}{\partial y}\right)_x$$

pelo que

$$\begin{cases} \mathbf{J}(u,x) = T\mathbf{J}(s,x) - P\mathbf{J}(v,x) \\ \mathbf{J}(f,x) = -s\mathbf{J}(T,x) - P\mathbf{J}(v,x) \\ \mathbf{J}(f,x) = T\mathbf{J}(s,x) + v\mathbf{J}(P,x) \\ \mathbf{J}(f,x) = -s\mathbf{J}(T,x) + v\mathbf{J}(P,x) \end{cases}$$

A aplicação de (13), (14), (15), e (16) e a identificação de α com P e β com T em (11), permitem exprimir qualquer derivada (9) em termos dos seguintes Jacobianos:

$$(17)\quad \begin{cases} a & = & \mathbf{J}(v,T) & & \\ b & = & \mathbf{J}(P,v) & = & \mathbf{J}(T,s) \\ c & = & \mathbf{J}(P,s) & & \\ d & = & \mathbf{J}(v,s) & & \\ \mathbf{J}(P,T) & = & 1 & & \left(\mathbf{J}(P,T) = \frac{\partial(P,T)}{\partial(P,T)} = 1\right) \end{cases}$$

Os coeficientes termodinâmicos (4) vêm:

$$(18)\quad C_P = Tc;\ C_v = T\frac{d}{a};\ \alpha_P = \frac{b}{v};\ \kappa_T = -\frac{a}{v}$$

pelo que os valores de a, b, c e d são dados por:

$$(19)\quad \begin{cases} a & = & -v\kappa_T \\ b & = & v\alpha_P \\ c & = & C_P/T \\ d & = & -v\kappa_T C_v/T \end{cases}$$

(5) conduz à relação entre a, b, c : $\qquad d = b^2 + ac \cdot d = 0$

Usando (19) na expressão da derivada em termos de (17), fica o problema resolvido[1].

O PROGRAMA DE CÁLCULO SIMBÓLICO

Quando se pensa na automatização das operações, a abordagem imediata é a do cálculo numérico. Para tal as derivadas serão aproximadas por quocientes de diferenças, ou expressões equivalentes, e o cálculo desenvolver-se-á a partir dessas expressões.

Acontece que a derivação é um dos "casos difíceis" da Análise Numérica pelo facto de as operações envolverem

(1) Como se verifica o método é extremamente simples, no entanto, na literatura didáctica da Termodinâmica, não tem havido a percepção deste facto sendo, na maioria dos casos, apresentadas soluções mais complexas. O método foi introduzido nas aulas de Termodinâmica do Prof. D. Domingos, no I.S.T., há cerca de 25 anos. Os interessados pela Termodinâmica encontram vários exemplos de aplicações na referência [3].

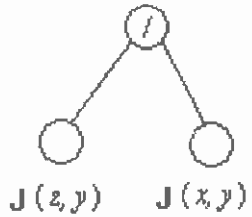
diferenças dos números muito próximos que ocasionam erros de arredondamento importantes. Daí a vantagem em levar o tratamento analítico, isto é, a manipulação dos símbolos, tão longe quanto possível.

No problema posto atrás, da obtenção das expressões das derivadas em termos dos coeficientes termodinâmicos, é sempre possível chegar à expressão "simbólica" final de uma forma automática.

Represente-se a derivada

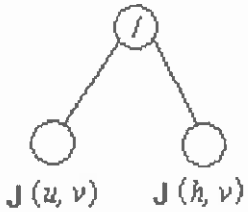
$$\left(\frac{\partial z}{\partial x}\right)_y = \frac{\mathbf{J}(z,y)}{\mathbf{J}(x,y)}$$

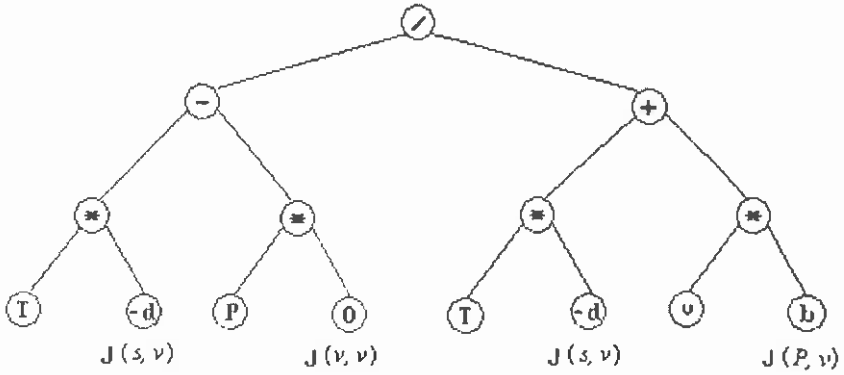
como uma árvore binária



A aplicação da (16) aos nós onde figuram u, f, h, ou g, cria novos nós. Por exemplo:

$$\frac{\partial(u,v)}{\partial(h,v)} = \frac{\mathbf{J}(u,v)}{\mathbf{J}(h,v)}$$





vem

A aplicação sucessiva de (16), usando (13) e (14) conduz a uma árvore onde todos os **J** (...,...) pertencem a (17); pode então fazer-se a transformação da árvore numa expressão dos símbolos das variáveis e *a, b, c, d* de (17). No exemplo

(19) (((T * (-d) - (P * 0)) / ((T * (-d) + (v * b)).

O processo é o inverso do desenvolvimento da árvore: caminha-se dos nós terminais para o nó raiz, uma estratégia que em [4] é definida por visita "postorder traversal".

Em (19) os símbolos ***a, b, c*** e ***d*** são substituídos pelas expressões (18) e o problema proposto fica resolvido.

No programa concreto, escrito em Quick - Basic, cuja listagem se apresenta em Apêndice, representa-se cada nó da árvore por uma lista (3 "arrays")

— **0$ ( )** indica uma operação aritmética (/* + -) ou um **J** se trata de um Jacobiano ou **j** se for um Jacobiano afectado por -, ou ainda **V** no caso de o nó representar uma variável do conjunto *[u, f, h, g, s, v, P, T]*

— **p ( )** indica o índice, na lista que representa a árvore, do nó "filho à esquerda" no caso de **0$ ( )** ser uma operação.
No caso de **0$ ( )** ser **J** ou **j**, **p ( )** representa o índice do nó onde reside a 1ª variável do Jacobiano.
Se for **0$ ( ) = "V"**, **p ( )** é o número de ordem da variável representada pelo nó.

— **s ( )** no caso de **0$ ( ) = "V"** não tem significado, para os restantes casos é o mesmo que **p ( )** mas para o nó "filho à direita" e para a 2ª variável do Jacobiano,

A inicialização da árvore faz-se por:

0$ (i) = "V", **p** (i) = i (i = 1, 2 ... 8)

isto é, colocam-se nos 8 primeiros elementos da lista os nós correspondentes às variáveis.

No nó 9 coloca-se a operação / e definem-se como "filhos" os nós 10 e 11.

0$ (9) = " / ", **p** (9) = 10, **s** (9) = 11

nos nós 10 e 11 colocam-se, respectivamente, os Jacobianos numerador e denominador da expressão (11) da derivada que se pretende transformar.

0$ (10) = "J", **p (10)** = nº da variável numerador da derivada
s(10) = nº da variável "constante" da derivada
0$ (11) = "J", **p (11)** = nº da variável denominador da derivada
s (11) = s (10)

As relações (16) utilizadas no desenvolvimento da árvore são aplicadas no programa na forma expressa pelos "arrays": **01 ( ) m_1 ( ), v_1 ( ), 02 ( ), m2 ( ) e v2 ( )**, cujo significado é representado simbolicamente, por

J (i, x) = 01 (i) m1(i) * J (v1 (i), x) + 02 (i) m2 (i) * J (v2 (ij, x)

onde **01 ( )** e **02 ( )** são **1** ou **-1 m1 ( ), m2 ( ) v1 ( ) e v2 ( )**, são os números das variáveis. Por exemplo, a 1ª relação de (16)

J (u, x) = T. J (s, x) - P. J (v, x)

traduz-se por (i = 1, u é a 1ª variável):

01 (1) = 1 ;	**m1 (1) = 8;**	**v1 (1) = 5**
+	T	s
02 (1) = -1;	**m2 (1) = 7;**	**v2 (1) = 6**
—	P	v

Inicializada a árvore o desenvolvimento processa-se, por aplicação sucessiva das relações (16), até deixarem de figurar as variáveis **u, f, h** e **g** nos nós que representam Jacobianos.

No programa a aplicação de (16), faz-se pela chamada do subprograma recursivo **slegen (j)**, onde na chamada inicial **j ≡ 9** (nó raiz da árvore). No desenvolvimento novos nós são

adicionados à lista, esta operação é controlada pela variável ***na*** que indica o índice do último elemento da lista. Por esta razão e dada a recursividade de **slegen ( )**, a variável ***na*** tem de ser externa ao subprograma.

No subprograma **slegen ( )** a execução inicia-se pela averiguação dos nós, 1° o nó filho à esquerda depois o nó filho à direita; só se analisam nós Jacobianos (**0$ ( )** com valor "**J**" ou "**j**"), com o valor de **p ( )** menor que **5** (variáveis **u, f, h,** e **g**). Os casos em que nos novos Jacobianos surge um par de variáveis iguais são tratados separadamente, isto é, não são introduzidos os nós onde essa condição se verifica, dada a propriedade (13).

Devido às cracterísticas de **slegen ( )**, todos os nós da árvore são visitados, inclusive os que entretanto forem criados, pelo que, após a execução, existe a garantia de as variáveis **u, f, h,** e **g**, terem desaparecido das posições de 1ªs variáveis dos Jacobianos. Se uma, ou mais, dessas variáveis ocupar a 2ª posição do Jacobiano, ela não desaparecerá dessa 2ª posição nos nós onde tal se verifica. O problema resolve-se pela chamada de **permuta ( )**, cuja função é permutar as variáveis nos nós com **0$ ( )** igual a **J** ou **j** a seguir faz-se nova chamada de **slegen ( )** e as variáveis **u, f, h, g,** desaparecem então da árvore.

Atingida esta fase, passa-se à transformação inversa da árvore, isto é, à tradução das operações representadas pela árvore numa cadeia de caracteres (texto) com o mesmo significado. Esta transformação é realizada pelo subprograma **conver$ (u$)** que recorre à função **cjacob$ ( )** que realiza a substituição dos Jacobianos em termos de **a, b, c, d** e **1**, ver (17) com eventual recorrência à relação (14).

O subprograma **conver (u $)** é chamado inicialmente com **u$** dado pela cadeia de caracteres

(? x x) / (? y y)

xx são os dois bytes da memória onde reside o valor endereçado por **p(9)**

yy o mesmo que **xx** para o valor **s (9)**

A lógica do subprograma **convert$ (u $)** é a seguinte:

— Se em **u$** não existir o caracter **?** a análise está completa.

— Procura-se um **?**; o valor representado nos dois caracteres que se lhe seguem, é o número do nó que se passa a analisar.

— Se o nó a analisar tiver índice **p ( ) ≤ 8**, trata-se de uma variável pelo que os caracteres **? xx** (ou **? yy**), substituem-se pelo texto (caracter) que representa a variável.

— Se o nó for um Jacobiano (**0 $ ( )** igual a "**J**" ou "**j**"), usa-se a função **cjacob$( )**, que fornece os caracteres que substituem **? xx** na cadeia de caracteres.

— No caso de se tratar de outra operação, os caracteres? **xx**, são substituidos pelo texto

(? zz) O (? ww)

onde **O** é o símbolo que representa a operação e **zz** e **ww** são os nós "filhos" de **xx**.

Notar a necessidade dos parênteses para garantir o respeito pelas regras de prioridade das operações aritméticas.

— O subprograma é chamado recursivamente.

Obtida a expressão final substitui-se "a", "b", "c" e "d" por **"-v * X", "v * L", "v * L", "y * I"** e **"(-v * H * X * I)"**, respectivamente, onde $X = k_t$, $L = \alpha_p$, $H = C_v$, $Y = C_p$ e $I = 1/T$.

Para efeitos de apresentação os valores de **X, L, H, Y,** e **I** passam a **Kt, Ap, Cv, Cp, e 1/T,** no final.

Atingido este ponto o problema inicialmente proposto está resolvido.

A SIMPLIFICAÇÃO

A expressão final obtida cumpre em tudo os requisitos exigidos para a finalidade a que destina, o cálculo de valores. Sob este aspecto pode mesmo dizer-se que a transformação da árvore em texto é uma tarefa inútil dado que a árvore pode utilizar-se directamente para o cálculo de valores; para tal basta substituir os valores numéricos das variáveis e Jacobianos (17) nos nós e processar as operações pela lógica seguida atrás: visita "postorder traversal".

A expressão analítica obtida apresenta um aspecto que do ponto de vista "estético" não é agradável, pois contem pares de parênteses supérfluos e, em alguns casos, factores comuns no numerador a denominador, pelo que é possível dar-lhe uma forma mais compacta. Este é o problema da simplificação da expressão.

Pode começar por dizer-se que a natureza do problema da simplificação é de grau de dificuldade superior ao da obtenção da expressão analítica que foi tratado atrás.

Uma das dificuldades com que se depara na abordagem deste problema reside na própria definição de simplicidade da expressão. Note-se que o sentido que se dá ao termo, é resultado de hábitos de aprendizagem e não tem, exactamente, o significado de simplicidade no sentido computacional. Por exemplo a expressão: a $x^2 + b\,x + c$ é considerada habitualmente como o estado final de simplificação em detrimento de **c + x (b + a x)** (forma de Horner) que como se sabe é computacionalmente mais simples.

No caso presente, embora de uma forma discutível, pode definir-se como a forma mais simples da expressão, a que cumpra as condições: inclua o máximo de 4 parênteses, não inclua

factores comuns ao numerador e ao denominador, não contenha expressões na forma **x/x**, que não comporte o valor 1 com factor associado a outros factores, não exista a parcela **0** e não haja ocorrência do prefixo +.

Esta definição é equivalente a dizer: a expressão mais simples é a que é invariante relativamente a um conjunto de transformações de equivalência.

No caso presente verifica-se que a expressão é um quociente de duas expressões **n$ / d$** e que em **n$ e d$**, à parte **1** e **-1** não ocorrem outras constantes. A operação / aparece somente associada à variável **T** na forma **/T**. Também a associação de termos semelhantes não necessita de ser considerada, porque o método de geração seguido não os cria.

Tendo em vista estas particularidades, entende-se como suficiente a aplicação às expressões do numerador e do denominador tomadas como independentes, excepto no que se refere à eliminação de factores comuns, da seguinte sequência de transformações:

— Retiram-se (função **factor $ (n$)**) todos os parênteses da expressão. A função **factor $ (n$)** consta da aplicação sucessiva da propriedade distributiva e eliminação (função **tirap $ (n$)** dos pares de parênteses supérfluos.

Em cada parcela os factores dispõem-se numa ordem pre-estabelecida (função **ordvar ($)**)

— Substitui-se o texto "**T * T**" por 1 (notar que **I = 1/T**). A operação de substituição é realizada pela função **substr $ (u$, a$, b$)**

— Determinam-se os factores comuns a todas as parcelas (subprograma **comuns (u$, n, z $ ( )**).

— Retiram-se do numerador e denominador os factores comuns (subprograma **cancel (...)**)

— Eliminam-se as ocorrências supérfluas associadas ao factor **1**, por exemplo: "**- 1***" ≡ " - ", * **1 +** " ≡ " +", "**1***" no início pode eliminar-se, etc.

— Averigua-se, no caso de o denominador não ser **1** a necessidade dos parênteses a encerrar o numerador, existência de "**+**" **ou** "**-**", sem ser como 1º caracter, ou "**/**"

— O mesmo para o denominador; neste caso verifica-se a existência ou não "**+**", "**-**", "*" e "**/**" em qualquer posição.

— Se o denominador for "1", elimina-se.

— O caso particular da expressão **1/(1/T)** transforma-se em T.

Alguns exemplos de resultados obtidos, são apresentados a seguir:

```
( D h / D u )      = Cp / (Cp-Ap*v*P)
               P

( D v / D s )      = 1 / (P/T)
               u

( D g / D f )      = (Ap*s*T-s-Cp) / (Ap*s*T-s-Kt*Cv*P-Ap*v*P)
               h
```

No apêndice não foram incluidos os subprogramas utilizados no processo de simplificação, por serem longos e sem características especiais a não ser o tratamento sistemático de inúmeros casos particulares dentro de determinada linha lógica. É importante ter sempre presente a ocorrência de efeitos colaterais nas substituições de simplificação que se utilizam.

CONCLUSÕES

O problema proposto para o qual se apresentou uma resolução concreta automatizada, poderia resolver-se também de uma forma mais prosaica: o número de derivadas distintas do tipo (9) na Termodinâmica, excluindo os casos degenerados de 2 variáveis envolvidas serem iguais e tendo em atenção que $(\partial z/\partial x)_y = 1/(\partial x/\partial z)_y$, é igual a 168, pelo que é viável a construção normal de uma tabela de fórmulas.

O problema serviu no entanto para tornar claro o interesse da utilização do tratamento de símbolos na execução das tarefas de cálculo.

Mostrou-se o suficiente para concluir que a tarefa de simplificação das expressões merece uma atenção especial, dado existirem artifícios muito difíceis de tratar de forma global, por exemplo, os casos notáveis da multiplicação.

No aspecto da simplificação uma sugestão parece prometedora, que é a de desenvolver o processo com a expressão representada na forma de árvore, isto é, realizar a "poda da árvore" antes da transformação inversa.

Tornou-se claro que as palavras chave do tratamento simbólico do cálculo são: árvores, recursividade (ou "pilhas") e tratamento de caracteres.

[1] H. B. Callen: "Thermodynamic", Wiley 1960.
[2] N. Show: "The Derivation of Thermodynamical Relations for Simple Systems", Prod. Transf. Roy. Soc. London 1935.
[3] A. Bento Leal, M. Rosário Machado: "Prática de Termodinâmica", AEIST, 1967-68.
[4] D. E. Knuth: Vol 1 / Fundamental Algorithms ", Addison - Wesley, 1968.

```
' Programa principal

DEFINT A-Z

DECLARE FUNCTION ordvar$ (u$, nv, v$())
DECLARE FUNCTION tirap$ (ui$)
DECLARE FUNCTION substr$ (u$, e$, n$)
DECLARE SUB cancel (n$, nn, an$(), d$, nd, ad$())
DECLARE FUNCTION factor$ (u$)
DECLARE SUB comuns (u$, zn, z$())
DECLARE SUB siegen (j)
DECLARE SUB permut (j)
DECLARE FUNCTION ojacob$ (j)
DECLARE SUB conver (u$)
DECLARE FUNCTION delopd$ (u$)
DECLARE FUNCTION deluns$ (u$)
DECLARE FUNCTION rectbar$ (u$)

DIM o$(100), p(100), s(100)              ' representação da árvore binária

DIM o1(4), m1(4), v1(4), o2(4), m2(4), v2(4)

DIM zn$(10), zd$(10)
DIM ov$(10)

v$ = "ufhgsvPT": vm$ = "UFHGSVPT"     ' símbolos das variáveis termodinâmicas

            ' relações resultantes dos diferenciais de:  u, f, h, g
o1(1) = 1: m1(1) = 8: v1(1) = 5: o2(1) = -1: m2(1) = 7: v2(1) = 6
o1(2) = -1: m1(2) = 5: v1(2) = 8: o2(2) = -1: m2(2) = 7: v2(2) = 6
o1(3) = 1: m1(3) = 8. v1(3) = 5. o2(3) = 1: m2(3) = 6. v2(3) = 7
o1(4) = -1: m1(4) = 5  v1(4) = 8  o2(4) = 1  m2(4) = 6: v2(4) = 7

            ' ov$() estabelece a sequência da ordenação dos símbolos
ov$(1) = "L": ov$(2) = "K": ov$(3) = "H"  ov$(4) = "C"
ov$(5) = "s"  ov$(6) = "v". ov$(7) = "P". ov$(8) = "T"  ov$(9) = "I"

FOR i = 1 TO 8: o$(i) = "V": p(i) = i  NEXT       'inicial  nós das variáveis
```

```
CLS
li = 1
DO                                          ' Input das variáveis
 IF li > 22 THEN
  SLEEP
  CLS : li = 1
 END IF
 LOCATE li, 1: PRINT "(D?/D?)",
 LOCATE li + 1, 1: PRINT "      ?",
 DO
  LOCATE li, 3, 1: COLOR 0, 7: PRINT "?"; : LOCATE li, 3
  DO: y$ = UCASE$(INKEY$): LOOP WHILE y$ = ""
  IF y$ = CHR$(27) THEN
   COLOR 7, 0: END
  END IF
 LOOP WHILE INSTR(vm$, y$) = 0
 IF INSTR(vm$, y$) < 7 THEN y$ = LCASE$(y$)
 COLOR 7, 0: LOCATE li, 3: PRINT y$
DO
 LOCATE li, 6: COLOR 0, 7: PRINT "?"; : LOCATE li, 6
 DO: x$ = UCASE$(INKEY$): LOOP WHILE x$ = ""
LOOP WHILE INSTR(vm$, x$) = 0
IF INSTR(vm$, x$) < 7 THEN x$ = LCASE$(x$)
COLOR 7, 0: LOCATE li, 6: PRINT x$,
DO
 LOCATE li + 1, 8: COLOR 0, 7: PRINT "?"; : LOCATE li + 1, 8
 DO: z$ = UCASE$(INKEY$): LOOP WHILE z$ = ""
LOOP WHILE INSTR(vm$, z$) = 0
IF INSTR(vm$, z$) < 7 THEN z$ = LCASE$(z$)
COLOR 7, 0: LOCATE li + 1, 8: PRINT z$,
LOCATE li, 9: PRINT " = ",
IF y$ = z$ THEN
 PRINT "0"                          ' var numerador = var constante
 li = li + 3
ELSEIF x$ = y$ THEN
 PRINT "1"                          ' var denominador = var numerador
 li = li + 3
ELSEIF x$ = z$ THEN
 PRINT "INFINITO";                  ' var denominador = var constante
 li = li + 3
ELSE                                ' caso geral

                                    ' inicialização da árvore
 na = 11
 o$(9) = "/": p(9) = 10: s(9) = 11
 o$(10) = "J": p(10) = INSTR(v$, y$): s(10) = INSTR(v$, z$)
 o$(11) = "J": p(11) = INSTR(v$, x$): s(11) = s(10)

 CALL siegen(9)                     ' desenvolvimento da árvore
 CALL permut(9)                     ' permuta de variáveis nos Jacobianos
                                    ' no caso da 2ª variável ser u,f,h,g
 CALL siegen(9)                     ' repetição do desenvolvimento da árvore

      ' início da transformação da representação em árvore para texto
 u$ = "(?" + MKI$(p(9)) + ")" + o$(9) + "(?" + MKI$(s(9)) + ")"
```

```
CALL conver(u$)       ' transformação da representação em árvore em texto

        ' substituição dos Jacobianos pelos coeficentes termidinâmicos
u$ = substr$(u$, "a", "(-v*X)")          ' X - coef  compress  isotérmico
u$ = substr$(u$, "b", "v*L")             ' L - coef  dilatação isobárica
u$ = substr$(u$, "c", "Y*I")             ' Y - calor espec  pressão cons.
u$ = substr$(u$, "d", "(-v*H*X*I)")      ' I - 1/T
                                ' H - calor espec  volume cons

                              ' simplificação da expressão final
j = INSTR(u$, "/")                    ' se j=0 não existe denominador
                        ' averiguação de o denominador ser = "1"
IF j = LEN(u$) - 1 AND MID$(u$, LEN(u$), 1) = "1" THEN
  u$ = MID$(u$, 1, j - 1)
  j = 0
END IF
      ' substr$(u$, a$, b$)   substitui a$ por b$ em u$
      ' ordvar$(u$, n, o$())  ordena em cada parcela os n simbolos o$()
      ' factor$(u$)           retira parentisis
un$ = substr$(ordvar$(factor$(MID$(u$, 1, j - 1)), 9, ov$()), "T*I", "1")

IF j > 0 THEN
  ud$ = substr$(ordvar$(factor$(MID$(u$, j + 2, LEN(u$) - j - 2)), 9, ov$()), "T*I", "1")
END IF

IF j > 0 THEN
              ' communs(u$, n, z$())   dá os n símbolos z$() comuns

CALL comuns(un$, nn, zn$())             ' a todas as parcelas de u$
CALL comuns(ud$, nd, zd$())

  ' cancel(un$, nn, zn$(), ud$, nd, zd$()) ' elimina os símbolos comuns
CALL cancel(un$, nn, zn$(), ud$, nd, zd$()) ' a un$ e ud$

IF ud$ = "" THEN
  j = 0
ELSE
  IF MID$(ud$, 1, 1) = "-" THEN ' se o denominador tiver um - na 1ª par-
    i = 1                       ' cela multiplicam-se (simbolicamente) o
    DO                          ' nummerador e denominador por -1
      k1 = INSTR(i, un$, "-")
      k2 = INSTR(i, un$, "+")
      k = k1
      IF (k2 > 0 AND k = 0) OR (k2 > 0 AND k2 < k) THEN k = k2
      IF k > 0 AND k = k1 THEN MID$(un$, k, 1) = "+"
      IF k > 0 AND k = k2 THEN MID$(un$, k, 1) = "-"
      i = k + 1
    LOOP UNTIL k = 0
    IF MID$(un$, 1, 1) <> "-" AND MID$(un$, 1, 1) <> "+" THEN un$ = "-" + un$

    i = 1                       ' multiplicação do denominador por -1
    DO
      k1 = INSTR(i, ud$, "-")
      k2 = INSTR(i, ud$, "+")
      k = k1
      IF (k2 > 0 AND k = 0) OR (k2 > 0 AND k2 < k) THEN k = k2
      IF k > 0 AND k = k1 THEN MID$(ud$, k, 1) = "+"
      IF k > 0 AND k = k2 THEN MID$(ud$, k, 1) = "-"
```

```
      i = k + 1
    LOOP UNTIL k = 0
   END IF
  END IF
END IF
                              ' o sinal + no início é supérfulo
IF MID$(un$, 1, 1) = "+" THEN un$ = MID$(un$, 2, LEN(un$) - 1)
IF j > 0 AND MID$(ud$, 1, 1) = "+" THEN ud$ = MID$(ud$, 2, LEN(ud$) - 1)

IF ud$ = "1" THEN j = 0          ' averiguação de o denominador ser = "1"

un$ = substr$(un$, "I", "/T ")
un$ = rectbar$(un$)       ' elimina efeitos colaterais da substi anterior
un$ = deluns(un$)                          ' elimina "1" supérfulos
un$ = delopd$(un$)        ' corrige efeitos colaterais da substi anterior
                          ' substituições por simbolos mais usuais
un$ = substr$(un$, "v*v", "v^2")
un$ = substr$(un$, "Y", "Cp")
un$ = substr$(un$, "H", "Cv")
un$ = substr$(un$, "L", "Ap")
un$ = substr$(un$, "X", "Kt")
 IF j > 0 THEN
   ud$ = substr$(ud$, "I", "/T")
   ud$ = rectbar$(ud$)
   ud$ = deluns(ud$)
   ud$ = delopd$(ud$)
   ud$ = substr$(ud$, "v*v", "v^2")
   ud$ = substr$(ud$, "Y", "Cp")
   ud$ = substr$(ud$, "H", "Cv")
   ud$ = substr$(ud$, "L", "Ap")
   ud$ = substr$(ud$, "X", "Kt")
 END IF

      ' averiguação da necessidade de encerrar o numerador por parentisis
k = INSTR(2, un$, "+")
IF k = 0 THEN k = INSTR(2, un$, "-")
IF k = 0 THEN k = INSTR(2, un$, "/")
IF j = 0 THEN k = 0

IF ud$ = "1/T" THEN              ' caso em que o denominador é igual a 1/T
  IF un$ = "1" THEN
    un$ = "T"
  ELSE
    un$ = un$ + "*T"
  END IF
 j = 0
END IF
                          ' impressão do numerador (resultado final)
IF k > 0 THEN PRINT "(";
PRINT un$;
IF k > 0 THEN PRINT ")";

IF j > 0 THEN
    ' averiguação da necessidade de encerrar o denominador por parentisis
 k = INSTR(ud$, "+")
```

```
      IF k = 0 THEN k = INSTR(ud$, "-")
      IF k = 0 THEN k = INSTR(ud$, "*")
      IF k = 0 THEN k = INSTR(ud$, "/")
                        ' impressão do denominador (resultado final)
      PRINT " / ";
      IF k > 0 THEN PRINT "(";
      PRINT ud$;
      IF k > 0 THEN PRINT ")";
    END IF
    li = li + 3
  END IF
LOOP

END

FUNCTION cjacob$ (j)

' Substituição dos Jacobianos

  SHARED u$, na, o$(), p(), s()

    cc = 0
    IF p(j) = 6 AND s(j) = 8 THEN cc = 1                    ' J(v,T)
    IF p(j) = 7 AND s(j) = 6 THEN cc = 2                    ' J(P,v)
    IF p(j) = 8 AND s(j) = 5 THEN cc = 2                    ' J(T,s)
    IF p(j) = 7 AND s(j) = 5 THEN cc = 3                    ' J(P,s)
    IF p(j) = 7 AND s(j) = 8 THEN cc = 4                    ' J(P,T)
    IF p(j) = 6 AND s(j) = 5 THEN cc = 5                    ' J(v,s)
                          ' se não foi enncontrado o Jacobiano
    IF cc = 0 THEN                ' tenta-se com as variáveis permutadas
      IF o$(j) = "J" THEN o$(j) = "j" ELSE o$(j) = "J"
      au = p(j): p(j) = s(j)  s(j) = au
      IF p(j) = 6 AND s(j) = 8 THEN cc = 1
      IF p(j) = 7 AND s(j) = 6 THEN cc = 2
      IF p(j) = 8 AND s(j) = 5 THEN cc = 2
      IF p(j) = 7 AND s(j) = 5 THEN cc = 3
      IF p(j) = 7 AND s(j) = 8 THEN cc = 4
      IF p(j) = 6 AND s(j) = 5 THEN cc = 5
    END IF
    IF o$(j) = "j" THEN sa$ = "-" ELSE sa$ = ""
    SELECT CASE cc
      CASE 1: sa$ = sa$ + "a"
      CASE 2: sa$ = sa$ + "b"
      CASE 3: sa$ = sa$ + "c"
      CASE 4: sa$ = sa$ + "1"
      CASE 5: sa$ = sa$ + "d"
    END SELECT
    cjacob$ = sa$
END FUNCTION


SUB conver (u$)

' Conversão da representação em árvore para texto

  SHARED v$, na, o$(), p(), s()
```

```
 k = INSTR(u$, "?")
 IF k = 0 THEN EXIT SUB              ' quando não houver ? em u$ o processo
                                     ' está concluido
 j = CVI(MID$(u$, k + 1, 2))

 IF j <= 8 THEN                      ' caso em que ao nó corresponde uma variável
   u$ = MID$(u$, 1, k - 1) + MID$(v$, j, 1) + MID$(u$, k + 3, LEN(u$) - 3)

 ELSEIF o$(j) = "J" OR o$(j) = "j" THEN      'caso de um nó com um Jacobiano
   u$ = MID$(u$, 1, k - 1) + cjacob$(j) + MID$(u$, k + 3, LEN(u$) - 3)

 ELSE                                ' uma operação envolvendo 2 nós
   u$ = MID$(u$, 1, k - 1) + "(?" + MKI$(p(j)) + ")" + o$(j) + "(?" + MKI$(s(j)) + ")" + MID$(u$, k
+ 3, LEN(u$) - 3)
 END IF
 CALL conver(u$)                     ' recursividade
END SUB

 SUB permut (j)

 ' Permuta as variáveis em J(. ,. ) quando a 2ª é. u, f, h ou g

 SHARED na, o$(), p(), s()

  k = p(j)                           ' tratamento do nó esquerdo
  IF k > 8 THEN
   IF o$(k) = "J" OR o$(k) = "j" THEN
    IF s(k) < 5 THEN
     IF o$(k) = "J" THEN o$(k) = "j" ELSE o$(k) = "J"
     u = p(k): p(k) = s(k): s(k) = u
    END IF
   ELSE
    CALL permut(k)                   ' recursividade
   END IF
  END IF

  k = s(j)                           ' tratamento do nó direito
  IF k > 8 THEN
   IF o$(k) = "J" OR o$(k) = "j" THEN
    IF s(k) < 5 THEN
     IF o$(k) = "J" THEN o$(k) = "j" ELSE o$(k) = "J"
     u = p(k): p(k) = s(k): s(k) = u
    END IF
   ELSE
    CALL permut(k)
   END IF
  END IF
 END SUB


 SUB slegen (j)

' Desenvolvimento da árvore
' Substituição da variáveis u, f, h, g, nos Jacobianos

 SHARED na, o$(), p(), s()
 SHARED o1(), m1(), v1(), o2(), m2(), v2()
```

```
  k = p(j)
  FOR r = 1 TO 2
   IF k > 8 THEN
    IF o$(k) = "J" OR o$(k) = "j" THEN
                              ' as relações resultantes dos difer-
     IF p(k) < 5 THEN                ' ciais só se aplicam para as variá-
                              ' veis 1, 2, 3 e 4
                             ' caso em que o 1º J( , .) na relação
      IF s(k) = v1(p(k)) THEN      ' é 0 por serem iguais as variáveis
       na = na + 1                 ' só é criado um novo nó
       IF o2(p(k)) < 0 THEN
        IF o$(k) = "J" THEN o$(na) = "j" ELSE o$(na) = "J"
       ELSE
        o$(na) = o$(k)
       END IF
       p(na) = v2(p(k))  s(na) = s(k)
       o$(k) = "*"  p(k) = m2(p(k))  s(k) = na
                             ' caso em que o 2º J(.., .) na relação
      ELSEIF s(k) = v2(p(k)) THEN   ' é 0 por serem iguais as variáveis
       na = na + 1                 ' só é gerado um novo nó
       IF o1(p(k)) < 0 THEN
        IF o$(k) = "J" THEN o$(na) = "j" ELSE o$(na) = "J"
       ELSE
        o$(na) = o$(k)
       END IF
       p(na) = v1(p(k))  s(na) = s(k)
       o$(k) = "*"  p(k) = m1(p(k))  s(k) = na

      ELSE                         ' caso geral, são criados 4 novos nós
       na = na + 1: n1 = na
       o$(na) = "*"  p(na) = m1(p(k))  s(na) = na + 1
       na = na + 1
       IF o1(p(k)) < 0 THEN
        IF o$(k) = "J" THEN o$(na) = "j" ELSE o$(na) = "J"
       ELSE
        o$(na) = o$(k)
       END IF
       p(na) = v1(p(k))  s(na) = s(k)
       na = na + 1: n2 = na
       o$(na) = "*"  p(na) = m2(p(k))  s(na) = na + 1
       na = na + 1
       IF o2(p(k)) < 0 THEN
        IF o$(k) = "J" THEN o$(na) = "j" ELSE o$(na) = "J"
       ELSE
        o$(na) = o$(k)
       END IF
       p(na) = v2(p(k))  s(na) = s(k)
       o$(k) = "+"  p(k) = n1: s(k) = n2
      END IF
     END IF
    ELSE
     CALL siegen(k)                          ' recursividade
    END IF
   END IF
   k = s(j)
  NEXT
END SUB
```