STUDIES ON VAPOR ADSORPTION SYSTEMS

N. Shamsundar

M. Ramotowski

Department of Mechanical Engineering

University of Houston

Houston, Texas

November 1998

Contract NAG-9-853

Project Technical Report

Johnson Space Center

Houston, Texas

Project Manager: J.R. Cornwell

Abstract

This report provides an overview and summary of the work performed in the Mechanical Engineering Department of the University of Houston under contract to the Johnson Space Center of NASA, Contract No. NAG-9-853. The project consisted of performing experiments on single and dual bed vapor adsorption systems, thermodynamic cycle optimization, and thermal modeling.

The work was described in a technical paper that appeared in conference proceedings and a Master's thesis, which were previously submitted to NASA. The present report describes some additional thermal modeling work done subsequently, and includes listings of computer codes developed during the project. Recommendations for future work are provided.

Contents

Chapter 1

Project Overview and Summary

This is a report on work done at the University of Houston on vapor-adsorption heat pump systems. In these systems, one or more adsorbent beds are alternately heated and cooled to move the refrigerant through the system, in lieu of the ubiquitous vapor compressor. The tasks in the project were (i) completing the fabrication of a two-bed test setup at the Johnson Space Center (JSC) premises, (ii) running thermal tests at JSC using the apparatus, and (iii) constructing analytical models for the performance of vapor-adsorption systems.

The project was started in early 1996, and the initial work on familiarizing ourselves with the topic was started soon after. During the summer of that year, M. Ramotowski worked at the JSC premises with the objective of completing the assembly and fabrication of a two-bed adsorption system, building upon the spaced spiral tubes and canisters that had already been fabricated by Lovchik and others at JSC. There were insurmountable delays in obtaining shop support and little progress was made as the summer was about to end and Ramotowski had to return to the University of Houston (UH) for her course work and thesis research.

report are provided on an IBM PC-DOS formatted disk. Listings are to be found at the end of Chapter 2 of this report. The Fortran codes will require the use of a nonlinear constrained optimization library package, as the code used by us cannot be distributed outside UH. They will run on any computer with a Fortran-77 compiler and suitable libraries.

Chapter 2

Single Bed Adsorption/Desorption Model

2.1 Scope of Chapter

This chapter contains a partial report on work done under contract to the Johnson Space Center on a vapor-adsorption heat pump system. This chapter, together with a conference technical paper [1] (April 1997) and an M.S. thesis [2] (January 1998) given to JSC earlier are the technical outcomes of the project. An overview of the whole project and the recommendations for future work were presented in the first chapter.

Ramotowski's M.S. Thesis contains the description of a single-bed electrically heated canister test and an analytical model against which to compare the test results. She ignored the heat capacity of the adsorbed refrigerant and used an explicit finite-difference scheme. The objective of the subsequent work, described here, is to include the ignored heat capacity and to describe an implicit finite-difference scheme, and to display some of the results from it.

The complete project involved three tasks, including (i) a thermodynamic model for a two-bed system with or without regenerative heat exchange, counterflow or cocurrent flow regenerators, (ii) design and construction of the single-bed test rig alluded to above and (iii) completion of a test apparatus involving two vapor-adsorbent beds, using canisters previously constructed at JSC. The first two tasks were completed successfully. The two-bed system was assembled and some tests run, but leaks in the canisters and, less importantly, in other parts of the system, prevented the obtaining of accurate data. The small amount of test results obtained indicated that the canister design as well as the design of the tubes for the heat transfer fluid require significant modifications to work properly.

2.2 Problem Description

The apparatus and the earlier analytical model are described in Ramotowski's thesis [2]. Consider one-dimensional transient heat transfer in an adsorbent bed. The adsorbent, for example, zeolite, is contained between two electrical heaters in the region $0 \leq x \leq 2L$. The two heaters have identical heat inputs, $2q_e$ each. The temperature, T, and the adsorbed fraction, ω, defined as the ratio of adsorbed fluid mass to *dry* adsorbent mass, are functions of position and time, that is, of (x, t). By symmetry, we may treat the equivalent problem in the region $0 \leq x \leq L$, with heat input q_e at $x = 0$ and zero heat flux at the symmetry line, $x = L$.

It is assumed that the bed is packed sufficiently loosely that the pressure, p, in the bed is uniform, but can vary in time. The assumption of uniform pressure is reasonable if no sudden and large pressure or temperature changes are imposed on the bed, forcing high refrigerant velocities to occur in the porous bed. We also assume that the thermal conductivity, k, of the zeolite is not significantly affected by the adsorption of refrigerant. Since adsorption occurs at the surface of zeolite granules and the interior of the granules is dry, this may be reasonable. However, if measured values of k in the partial adsorption state become available, the modification required in the equations below is minimal. The specific heats of dry zeolite, c_z, and of adsorbate, c_r, are assumed to be independent of temperature, as well. Operating pressures in heat pumps are usually so low (below atmospheric, often) that the mass of the vapor refrigerant filling the packing space between the zeolite granules is negligible.

2.3 Governing Equations

With these assumptions, it is possible to write the heat conduction equation in the bed as

$$\rho_z \frac{\partial h}{\partial t} = k \frac{\partial^2 T}{\partial x^2} \tag{1}$$

where h is the enthalpy of the bed per unit mass of dry zeolite, with the boundary conditions

$$-k \frac{\partial T}{\partial x} = q_e, \qquad x = 0 \tag{2}$$

$$-k \frac{\partial T}{\partial x} = 0, \qquad x = L \tag{3}$$

The thermodynamics of adsorption (see [1] for details) gives us the relation

$$h - h_{\text{ref}} = (c_z + \omega c_r)(T - T_{\text{ref}}) + \Delta h_a \tag{4}$$

where Δh_a is the heat of adsorption, which is related to the adsoption pressure by

$$\Delta h_a = -R \int_0^{\omega} b(\omega)\, d\omega \tag{5}$$

and $b(\omega)$ occurs in the adsorption equilibrium equation,

$$\ln p = a(\omega) + \frac{b(\omega)}{T} \tag{6}$$

where p and T are absolute pressure and temperature and R is the gas constant of the adsorbate. The functions $a(\omega)$ and $b(\omega)$ are experimentally obtained functions for each adsorbent/adsorbate pair.

2.4 Solution Technique

We use the effective specific heat concept, which is also used in freezing-melting problems, to convert eq. (4) to the usual heat equation. Thus,

$$c_{\text{eff}} \equiv \left(\frac{\partial h}{\partial T}\right)_p = c_z + \omega c_r - R\, b(\omega) \left(\frac{\partial \omega}{\partial T}\right)_p \tag{7}$$

and

$$\frac{\partial T}{\partial t} = \alpha_{\text{eff}} \frac{\partial^2 T}{\partial x^2} \tag{8}$$

where $\alpha_{\text{eff}} = k/\rho_z c_{\text{eff}}$ is the effective diffusivity.

A uniform grid with nodes numbered 0 (at the heater location) to N (at the symmetry line) is used, and the heat equation and boundary conditions are discretized

using central differencing for derivatives with respect to x and the backward implicit difference for derivatives with respect to t. The resulting equations resemble the usual tridiagonal equations of one-dimensional transient heat conduction, but they are nonlinear, and need to be solved together with the adsorption thermodynamics equations and in keeping with the system constraints.

The method for advancing the solution from t to $t + \Delta t$ will now be described, for two specific modes of bed operation.

For *constant pressure heating*, using the current values of ω at the nodes, compute the valued of c_{eff}. Note that $\partial\omega/\partial T$ can be obtained from eq. (6) using the relation

$$\frac{\partial \omega}{\partial T} = -\left(\frac{\partial \ln p}{\partial T}\right) / \left(\frac{\partial \ln p}{\partial \omega}\right) \tag{9}$$

so that

$$c_{\text{eff}} = c_z + \omega c_r + R \frac{[b(\omega)/T]^2}{[a'(\omega) + b'(\omega)/T]} \tag{10}$$

where primes indicate derivatives with respect to ω.

Substitute these values into the tridiagonal equations and solve to obtain a new tentative set of improved temperatures, T_1. Insert these improved temperatures in eq. (6) with the specified pressure and solve for improved values of ω at each node, using the Newton-Raphson algorithm. If any values of ω are less than zero, replace them by zeros. Using these improved values of ω, recalculate c_{eff} at all nodes and re-solve the tridiagonal equations to obtain the temperature distribution at $t + \Delta t$. By integrating ω over the domain, obtain the total adsorbent fraction, $\bar{\omega} \equiv m_r/m_z$.

For *heating at constant adsorbed fraction*, that is, heating with no refrigerant allowed to enter or leave the canister, calculate tentative improved temperatures T_1 as in the previous case. Using these temperatures and a *trial* value of the new pressure, compute the ω values at all nodes using eq. (6), and integrate to obtain $\bar{\omega}$ as in the previous case. Recalculate $\bar{\omega}$ using a slightly different pressure, $p + \Delta p$, and obtain an improved p using the secant method. Repeat until p converges to the value at which the $\bar{\omega}$ matches the specified adsorbed fraction. Note that the heat equation has only been solved once in the current time-step. Using the ω distribution that permits the $\bar{\omega}$ to match the specified value, recalculate c_{eff} over the grid and re-solve the tridiagonal equations to obtain the temperature distribution at $t + \Delta t$.

The procedure for constant $\bar{\omega}$ is slightly more time consuming than that for constant pressure. Modification of the two procedures to handle cases where q_e, p and $\bar{\omega}$ are specified to vary with time in known ways is trivial.

2.5 Sample Results

The results to be presented illustrate the application of the calculation method just described. They pertain to Zeolite-NaX/water, and will be presented in nondimensional terms to the extent possible. Temperature cannot be non-dimensionalized since the adsorption properties depend on it in a definitely nonlinear fashion.

Figure 2.1 shows the results for a 51-point grid, for operation with constant moisture. The bed is initially at 300 K and $\omega = 0.1$ throughout. The heating rate is such that $q_e L/k = 1000$K, and the temperature profiles show a constant slope at $\bar{x} = 0$, as expected. The resemblance to the temperature profiles in a semi-infinite region that is encountered in single-phase heat conduction is evident in the time interval chosen. Given that the temperature at $\bar{x} = 0$ is already over the safe temperature for zeolite, we are led to the conclusion that the heater spacing could be reduced, saving both space and zeolite. The calculation used $\Delta\tau = 0.0005$, which is over twice the stability limit in the explicit method described in Ramotowski's thesis.

The corresponding adsorbed fraction profiles are shown in Figure 2.2. The migration of the water from the heated end to the opposite end is very evident. The portion near the heater does go completely dry after $\tau = 0.06$, and the moisture driven off accumulates at the other end. In our model, it is assumed that the migration velocities are low enough that the vapor stays in temperature equilibrium with the bed as it moves. Ramotowski observed that when the velocities were increased by turning on a vacuum pump, the thermocouples at the cold end showed a sudden increase in temperature because of the heat of adsorption released there. The analytical model does not account for such effects.

Next, we present results for heating at constant pressure, which requires that the desorbed vapor be allowed to leave the adsorption bed. Figure 2.3 shows the temperature distributions, and it is noted that the hot end gets heated slightly faster than

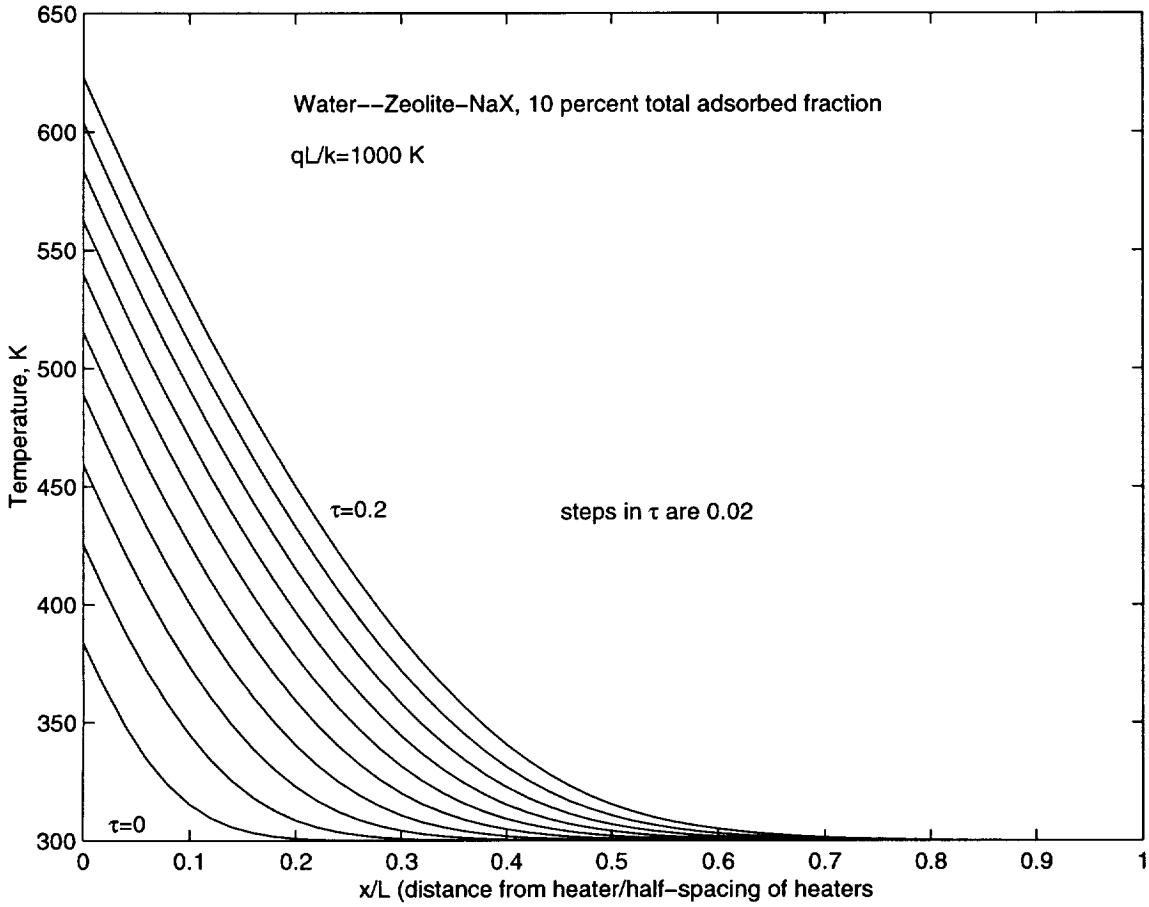

Figure 2.1: Temperature Distributions, Constant Moisture Heating

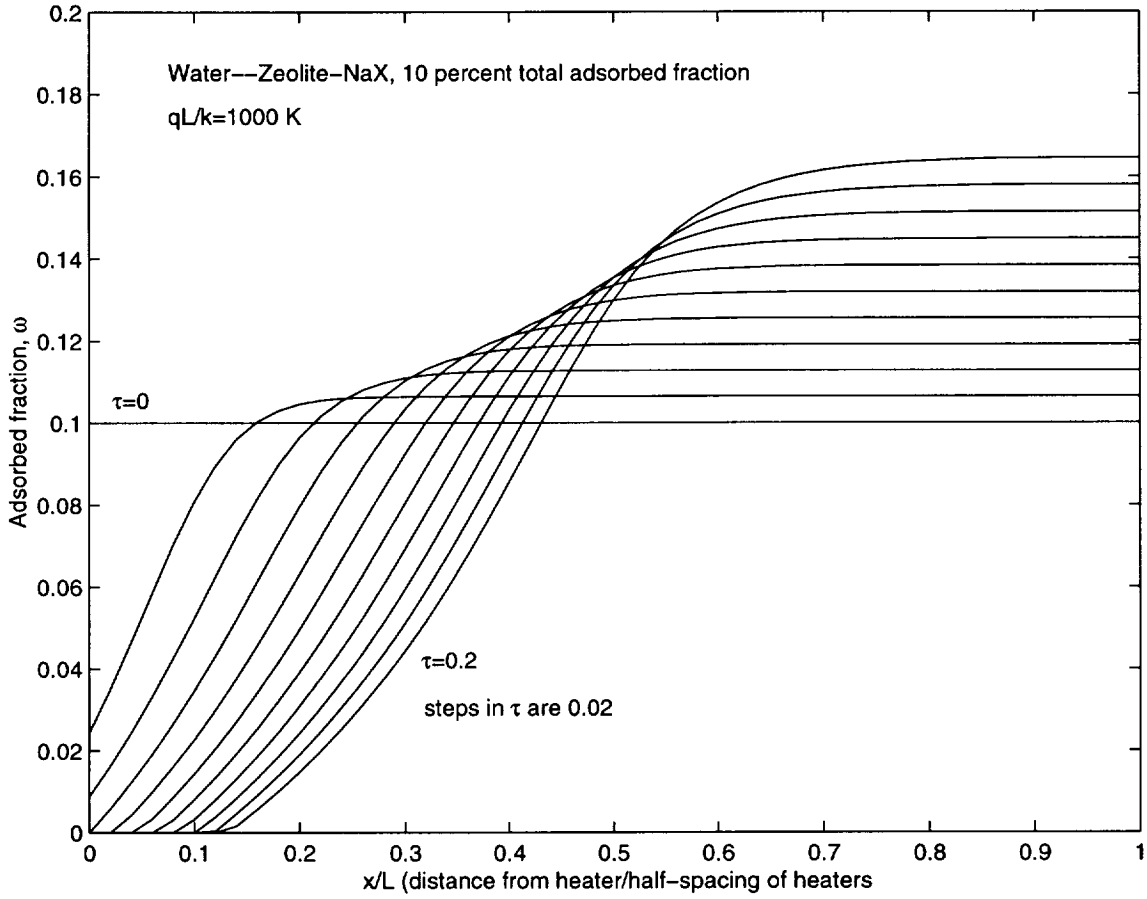

Figure 2.2: Adsorbed fraction Distributions, Constant Moisture Heating

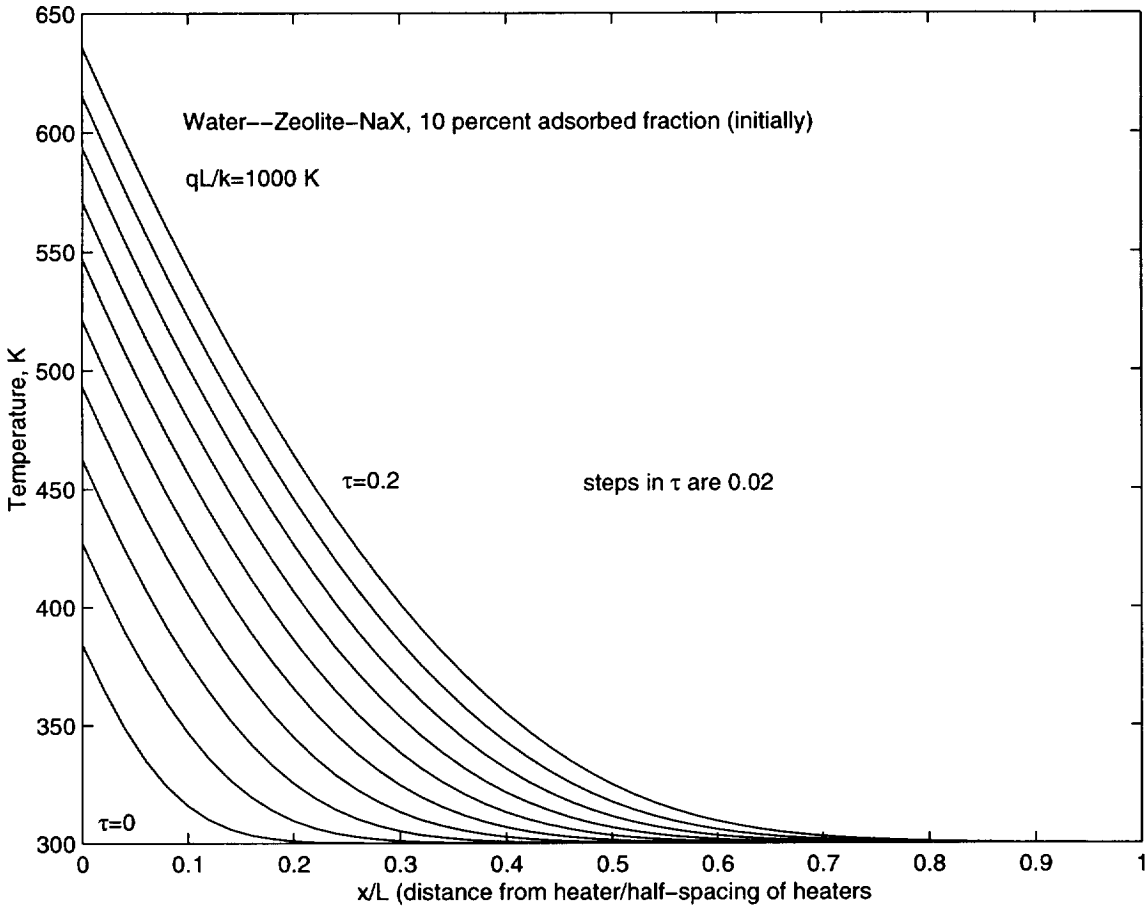

Figure 2.3: Temperature Distributions, Constant Pressure Heating

in the constant moisture case described above. Figure 2.4 contains the corresponding ω-distributions, from which it is observed that the hot end dries out faster when desorption is allowed from the system. This reduces the effective heat capacity of the bed locally, and causes its temperature to rise faster.

Code was written in Fortran-77 to implement the solution method and used in producing the results just displayed. This code is provided at the end of this chapter.

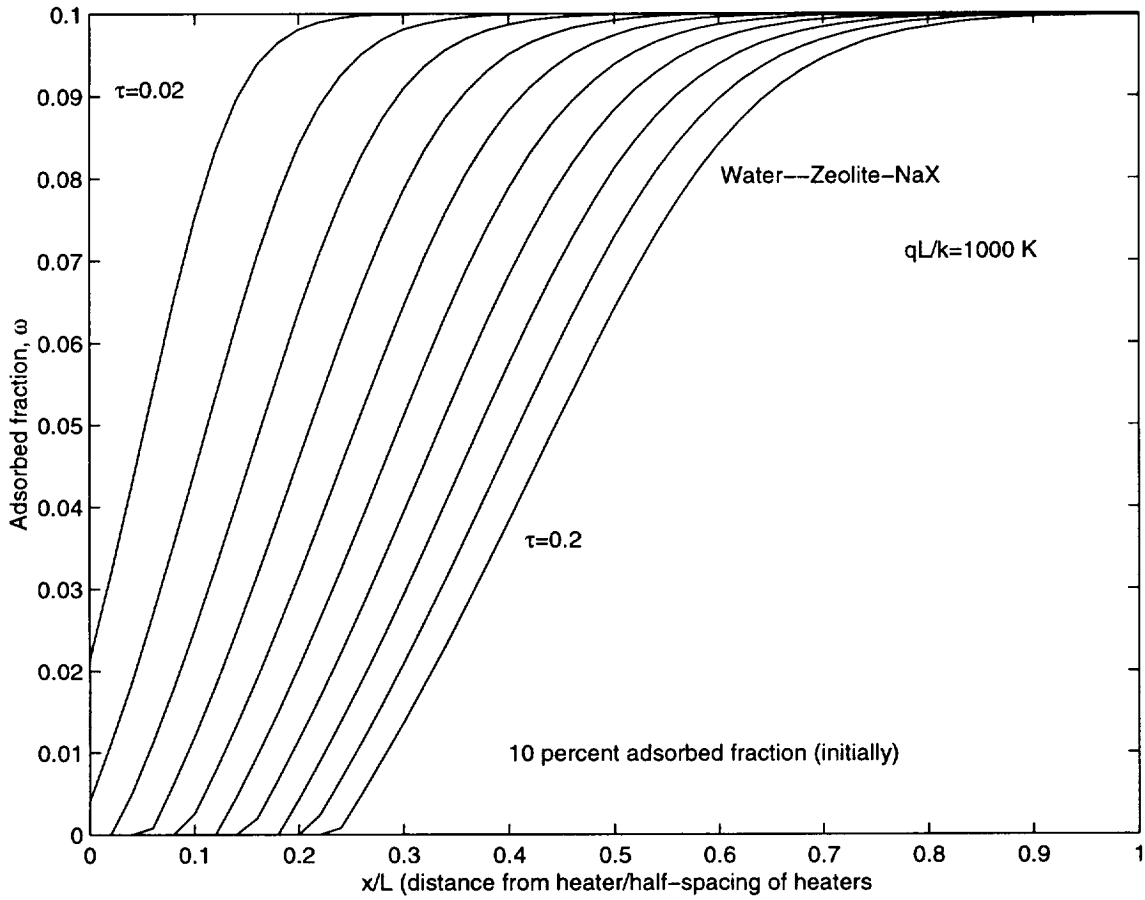

Figure 2.4: Adsorbed fraction Distributions, Constant Pressure Heating

The code takes 6 seconds to run the constant moisture case and 3 seconds to run the constant pressure case on a 66 MHz AMD-486 personal computer. These timing data indicate that it should certainly be feasible to extend the code to two-dimensional transients. This extension will be rather straightforward for regular geometries such as shell-and-tube heat exchangers, but not for helical or spirally wound tubes.

If JSC contemplates further development work in this topic, it is recommended that a two-dimensional version of the code be developed, and that the code be then used to design a test apparatus with specific performance objectives.

2.6 Nomenclature

$a(\omega)$	function in adsorption pressure equation
$b(\omega)$	function in adsorption pressure equation
c_r	specific heat of adsorbed fluid
c_z	specific heat of adsorbent
h	enthalpy of adsorbent and adsorbate, per unit mass of adsorbent
k	thermal conductivity of adsorbent bed
L	half-spacing between heaters
m_r	total mass of refrigerant
m_z	total mass of adsorbent
N	number of mesh divisions
p	absolute pressure
q_e	half of electrical heat output rate per unit area of heater
R	gas constant of refrigerant
t	time
T	absolute temperature
x	distance from heater
$\bar{x}$	x/L

α_{eff} effective thermal diffusivity of bed

α_z thermal diffusivity of adsorbent

Δh_a enthalpy of adsorption

Δt time step

ρ_z density of zeolite in bed

τ non-dimensional time, $\alpha_z t/L^2$

ω mass of refrigerant per unit mass of adsorbent

eff effective value

ref reference state

2.7 References

1. Ramotowski, M.R., and Shamsundar, N., 1997, "Optimization of a simplified vapor adsorption cycle", *Proceedings of the ASME Advanced Energy Systems Division, 1997 ASME International Mechanical Engineering Congress and Exposition,* AES-Vol. 37, pp. 93–102.

2. Ramotowski, M.R., "Optimization and Testing of a Two-bed Regenerative Adsorption Heat Pump", M.S.M.E. Thesis, University of Houston, 1997.

2.8 Computer code for Single-bed system

```
      program snglbed
c     ************************************************************
c     Simulation of single zeolite-water bed with electric
c     heating at x=0 and zero heat flux at x=L (symmetry line).
c     See Chapter 4 of M. Ramotowski, M.S.M.E. Thesis,
c     University of Houston, 1998 for description of experiment
c     and experimental results.
c
c     Variables : xbar=x/L, tau=\alpha_z t/L^2
c     Output : Temperature and adsorbed fraction profiles
c              for plotting  in files t.dat and w.dat
c
c     Author : N. Shamsundar, University of Houston
c     Date   : August, 1998
c     ************************************************************
      implicit real*8(a-h,o-z)
      real*8 mw
      integer tout,wout
      parameter (NN=50)
      dimension T(0:NN),TN(0:NN),w(0:NN)
      common /zeo/cz,cr,Rref
      logical constm
c
      cz=1406d0           ! specific heat of zeolite
      cr=4200d0           ! specific heat of adsorbed water
      Rref=461d0          ! gas constant of water vapor
      mw=0.1d0            ! mass of water/mass of zeolite
      n=50                ! number of spatial steps
      Ti=300d0            ! initial temperature, K
      qLbyk=1000d0        ! (heat input) X L /k, K
      dt=0.0005d0         ! step in Fourier number
      nt=40               ! number of time steps between plots
      nplt=10             ! number of plots
      constm=.true.       ! true if heating with valves closed
c                         ! false if heating at constant pressure
      tout=7
      wout=8
      open(unit=tout,file='T.dat',status='unknown')
      open(unit=wout,file='w.dat',status='unknown')
```

```
      do i=0,n
        T(i)=Ti
        w(i)=mw
      end do
      write(tout,'(1x,51f7.2)')(T(i),i=0,n)
      write(wout,'(1x,51f7.4)')(w(i),i=0,n)
c
      p=4                          ! trial value
      tau=0
      call findp(n,mw,T,p,w)       ! find initial pressure
      s=0
      do i=0,n
        s=s+T(i)
      end do
      Tmean=(s-0.5*(T(0)+T(n)))/n
      if(constm)then
        write(*,'(1x,A)')
     +      '    Tau    p(Pa)  T(x=0)  T_mean  Grd chk.'
        write(*,'(1x,2f8.4,3f8.2)') tau,p,T(0),Tmean,(T(0)-T(1))*n
      else
        write(*,'(1x,A)')
     +      '    Tau     mw    T(x=0)  T_mean  Grd chk.'
        write(*,'(1x,2f8.4,3f8.2)') tau,mw,T(0),Tmean,(T(0)-T(1))*n
      endif
c
      do iplt=1,nplt
        do it=1,nt
           call onestep(n,dt,T,w,qLbyk,Tn)
           if(constm)then
             call findp(n,mw,Tn,p,w)
           else
             call totmoist(n,p,Tn,mw,w)
           endif
           call onestep(n,dt,T,w,qlbyk,Tn)
           tau=tau+dt
           do i=0,n
             T(i)=Tn(i)
           end do
        end do
        s=0
        do i=0,n
          s=s+T(i)
```

```
        end do
        Tmean=s/n
        if(constm)then
          write(*,'(1x,2f8.4,3f8.2)') tau,p,T(0),Tmean,(T(0)-T(1))*n
        else
          write(*,'(1x,2f8.4,3f8.2)') tau,mw,T(0),Tmean,(T(0)-T(1))*n
        endif
        write(tout,'(1x,51f7.2)')(T(i),i=0,n)
        write(wout,'(1x,51f7.4)')(w(i),i=0,n)
      end do
      stop
      end
c
      subroutine findp(n,mw,T,p,w)
c
c     Given a temperature distribution,
c     find a (uniform) pressure such that the total adsobed mass
c             = mw X zeolite mass
c     Method: secant method
c     Input parameters  : n, mw, T(0:n), guess for p
c     Output parameters : solution p, corresponding w(0:n)
c
      implicit real*8(a-h,o-z)
      parameter (ITMAX=10)
      real*8 mw,mw1,mw0,T(0:n),p,w(0:n)
      dp=0.001*p
      call totmoist(n,p,T,mw0,w)
      do it=1,ITMAX
        p=p+dp
        call totmoist(n,p,T,mw1,w)
        dp=(mw-mw1)/(mw1-mw0)*dp
        mw0=mw1
        if(abs(dp/p).lt.1d-4)exit
      end do
      if(it.eq.ITMAX)stop 'Failed to converge in FindP'
      return
      end
c
      subroutine totmoist(n,p,T,mw,w)
c
c     Given a temperature distribution and a total pressure (uniform),
c     calculate the corresponding adsorbed fraction distribution and
```

```
c     the adsorbed mass/zeolite mass ratio for the whole bed
c
c     Input parameters : n,p,T(0:n)
c     Output parameters: mw, w(0:n)
c     Method: Newton-Raphson
c
      implicit real*8(a-h,o-z)
      parameter (ITMAX=10)
      real*8 mw,T(0:n),p,w(0:n),lp
c
c     The following functions are for Zeolite-NaX + Water
c
      afun(a)=(((1.6448d3*a-7.3176d2)*a+1.10854d2)*a+
     +           13.4244d0+4.660517d0)
      bfun(a)=((-3.4867d3*a+0.564447d4)*a+67.2292d2)*a-7373.78d0
      dafun(a)=(3*1.6448d3*a-2*7.3176d2)*a+1.10854d2
      dbfun(a)=(-3*3.4867d3*a+2*0.564447d4)*a+67.2292d2
      a0=13.4244d0+4.660517d0
      b0=-7373.78d0
      a1=1.10854d2
      b1=67.2292d2
c
      lp=log(p)
      mw=0
      do i=0,n
        omeg=(lp-a0-b0/T(i))/(a1+b1/T(i))    ! trial solution
        do j=1,ITMAX
          domeg=(lp-afun(omeg)-bfun(omeg)/T(i))/
     +          (dafun(omeg)+dbfun(omeg)/T(i))
          omeg=omeg+domeg
          if(abs(domeg/omeg).lt.1d-5)exit
        end do
        if(j.eq.ITMAX)stop 'Did not converge in TotMoist'
        w(i)=max(omeg,0d0)                   ! no negative ads. fract.
        mw=mw+w(i)
      end do
      mw=(mw-0.5d0*(w(0)+w(n)))/n
      return
      end
c
      subroutine onestep(n,dt,T,w,qlbyk,Tn)
c
```

```
c     Do one step of implicit finite-difference integration of
c     one-dimensional transient heat equation, using effective
c     heat capacity technique
c
c     Input parameters   : n,dt,T(0:n),w(0:n),qL/k
c     Output parameter   : Tn(0:n) (temperatures at tau+dt)
c
      implicit real*8(a-h,o-z)
      real*8 T(0:n),w(0:n),Tn(0:n)
      common /zeo/cz,cr,Rref
      parameter (NN=50)
      dimension A(0:NN),B(0:NN),C(0:NN)
c
c     The following functions are for Zeolite-NaX + Water
c
      afun(a)=(((1.6448d3*a-7.3176d2)*a+1.10854d2)*a+
     +           13.4244d0+4.660517d0)
      bfun(a)=((-3.4867d3*a+0.564447d4)*a+67.2292d2)*a-7373.78d0
      dafun(a)=(3*1.6448d3*a-2*7.3176d2)*a+1.10854d2
      dbfun(a)=(-3*3.4867d3*a+2*0.564447d4)*a+67.2292d2

c
      if(n.gt.NN)stop 'Array overrun in OneStep'
      do i=0,n
        bb=bfun(w(i))
        da=dafun(w(i))
        db=dbfun(w(i))
        ceff=cz+w(i)*cr+bb*bb*Rref/(T(i)*(da*T(i)+db))
        r=dt*n*n*cz/ceff
        A(i)=-r
        B(i)=1+2*r
        C(i)=-r
        Tn(i)=T(i)
        if(i.eq.0)then
          C(i)=C(i)-r
          Tn(i)=Tn(i)+2*qLbyk*r/n
        else if(i.eq.n)then
          A(i)=A(i)-r
        endif
      end do
      call tridiag(n,A,B,C,Tn)
      return
```

```
      end
c
      subroutine tridiag(n,A,B,C,x)
c
c     Solve tridiagonal equations A(i)x(i-1)+B(i)x(i)+C(i)x(i+1)=D(i)
c     Input parameters : n, A(0:n), B(0:n), C(0:n), D(0:n) in x
c     Output parameter : x(0:n), with A,B,C destroyed
c
      implicit real*8(a-h,o-z)
      dimension A(0:n),B(0:n),C(0:n),x(0:n)
      C(0)=C(0)/B(0)
      x(0)=x(0)/B(0)
      do i=1,n
        t=1/(B(i)-A(i)*C(i-1))
        C(i)=C(i)*t
        x(i)=(x(i)-A(i)*x(i-1))*t
      end do
      do i=n-1,0,-1
        x(i)=x(i)-C(i)*x(i+1)
      end do
      return
      end
```

2.9 Computer codes for Thermodynamic COP Optimization

2.9.1 Optimization of COP for Water–Zeolite-NaX

```
      program zeolitew
      implicit double precision(a-h,o-z)
      parameter (ivar=10)
      parameter (iconst=20)
      parameter (kworka=IVAR*(IVAR+ICONST+25)+11*ICONST+60)
      parameter (kkwork=300)
      parameter (imnn2=iconst+2*ivar+2)
      parameter (iact=3*iconst+15)

      dimension variab(ivar),constr(iconst)
      dimension gradof(ivar)
      dimension vecmul(imnn2),boulow(ivar),bouupp(ivar)
      dimension workar(kworka),iworka(kkwork)
      dimension gradco(iconst,ivar)
      dimension  hessem(ivar,ivar),rhside(ivar)
      logical active(iact),lmerit,lql

      common/cmache/eps100,eps200,eps300
      common /ctrl/Temax,Tcmin,start,fin,num,eregen
      real*8 Temax,Tcmin,eregen
      logical start,fin

      eps100=epsilon(eps100)
      eps200=1.d-7
      eps300=1.d-3

c  NUMBER OF CONSTRAINTS:
      nocons=4

c  NUMBER OF EQUALITY CONSTRAINTS:
      noeqco=0

c  NUMBER OF VARIABLES
      novari=4

c  BOUNDS ON VARIABLES
      boulow(1)=280.0
```

```
      bouupp(1)=500.0
      boulow(2)=280.0
      bouupp(2)=500.0
      boulow(3)=280.0
      bouupp(3)=500.0
      boulow(4)=280.0
      bouupp(4)=500.0

c  USER SPECIFIED ACCURACY
      accura=1.d-8

c  USER SPECIFIED SCALING PARAMETER
      scabou=1.d+3

c  PRINT LEVEL SPECIFIED
      iprint=0

c  MODE OF OPERATION
      modeal=0

c  DESIRED OUTPUT UNIT NUMBER
      ioutst=8

c  IF TRUE, USE L2 MERIT FUNCTION.  OTHERWISE L1.
      lmerit = .true.

c  IF TRUE, USE FULL QP SUBPROBLEM.
      lql=.true.

c  ROW DIMENSION OF DG -  .GE. (MAX (1, NOCONS))
      nommax=iconst

c  ROW DIMENSION OF C - .GE. (MAX (2, NOVARI))
      nonmax=ivar

c  MAX NUMBER OF FUNCTION CALLS DURING LINE SEARCH
      maxfun=8

c  UPPER BOUND ON THE NUMBER OF ITERATIONS
      maxite=40

c  EQUAL TO NOCONS + NOVARI +NOVARI +2
```

```
      nomnn2= nocons + novari + novari + 2
c  OUTPUT PARAMETER REGD. TERMINATION. SHOULD BE ZERO AT START
      infail=0

c  WORKING PARAMETER MIGHT HAVE TO BE ENLARGED IF NEEDED
      lework=nonmax*nommax + 4*nommax + 4*nommax + 19*nonmax+55 +
     1 1.5*(nonmax+1)**2 + 15*nonmax + 4*nommax + 20

c  INTEGER PARAMETER TO DETERMINE SPACE FOR SUBPROBLEM SOLUTION
      leiwor=19+ivar +ivar+iconst

c  SPACE FOR  LOGICAL VARIABLES OF CONSTRAINT ACTIVITY
      leacti=3*nommax + 15

      open(unit=ioutst,file='nlpql.res')

c  INITIAL VALUES OF VARIABLES
      open(unit=9,file='dat')
      rewind(9)
      read(9,*)(variab(i),i=1,novari),eregen
      close(9)
      Temax=variab(1)
      do 1000 Tcmin=Temax+20,500

      fin=.false.
      num=0
      call nlpql1(nocons,noeqco,nommax,novari,nonmax,nomnn2,variab,
     1  objfun,constr,gradof,gradco,vecmul,boulow,bouupp,hessem,
     2  rhside,accura,scabou,maxfun,maxite,iprint,modeal,ioutst,
     3  infail,workar,lework,iworka,leiwor,active,leacti,lmerit,
     4  lql)
c
c
c   OUTPUT RESULT
c
      num=infail
      fin=.true.
      call nlfunc(nocons,noeqco,nommax,novari,objfun,constr,
     +           variab,active)
      write(*,9020) infail,-objfun*1d-3,(variab(i),i=1,novari)
 9020 format(1x,i4,f8.4,8f8.2)
 1000 continue
```

```
      stop
      end
c
      block data
      logical start,fin
      common /ctrl/Temax,Tcmin,start,fin,num,eregen
      real*8 Temax,Tcmin,eregen
      data fin/.false./,start/.true./
      end
c
      subroutine nlfunc(nocons,noeqco,nommax,novari,objfun,g,
     1     x,active)
      implicit double precision(a-h,o-z)
      dimension g(nommax),x(novari)
      logical active(nommax),fin,start
      common /ctrl/Temax,Tcmin,start,fin,num,eregen

c   EVALUATION OF PROBLEM FUNCTIONS

      afun(a)=(((1.6448d3*a-7.3176d2)*a+1.10854d2)*a+
     +         13.4244d0+4.660517d0)
      bfun(a)=(((-3.4867d3*a+0.564447d4)*a+67.2292d2)*a-
     +         7373.78d0)
      bfuni(a)=(((-3.4867d3/4*a+0.564447d4/3)*a+67.2292d2/2)*a-
     +         7373.78d0)*a
c
      data crefr/4200d0/, cbed/1406.0d0/, Rrefr/461.0d0/
      data cvap/1925.8d0/
c
c     crefr=cvap
      Te=x(1)
      Tc=x(2)
      T1=x(3)
      T3=x(4)

      pcond=psat(Tc)
      pevap=psat(Te)

      omega2=omegaf(pevap,T3)
      omega1=omegaf(pcond,T1)

      T2=bfun(omega1)/(log(pevap)-afun(omega1))
```

```
      T4=bfun(omega2)/(log(pcond)-afun(omega2))
      hfg=hfgw(Tc)
c
      Qih=(cbed+omega2*crefr)*(T4-T3)      ! Isosteric heating
      Qdes=Rrefr*(bfuni(omega1)-bfuni(omega2))
      Rintc=quad(pcond,omega1,omega2)
      Qsd=cbed*(T1-T4)+crefr*(omega1*T1-omega2*T4+rintc)
      Qic=(cbed+crefr*omega1)*(T1-T2)      ! Isosteric cooling
      Qads=Qdes
c
      rinte=quad(pevap,omega1,omega2)
      Qsa=cbed*(T2-T3)+crefr*(omega1*T2-omega2*T3+rinte)
      Qsve=cvap*(rinte-Te*(omega2-omega1))
      Qa=Qads+Qsa-Qsve
      Qd=Qdes+Qsd

      Qsvc=cvap*(rintc-Tc*(omega2-omega1))
      Qc=Qsvc+(omega2-omega1)*hfg
      Qe=(omega2-omega1)*(hfg-cvap*(Tc-Te))

c      Qcond=max(Qic,0d0)+max(Qa,0d0)+max(Qc,0d0)
      Qcond=Qic+Qa+Qc
      if(fin)then
         if(Qic.lt.0.or.Qa.lt.0.or.Qc.lt.0)then
           write(*,*)' Wrong sign'
         endif
         if(Qih.lt.0.or.Qd.lt.0)then
           write(*,*) 'negative heat input'
         endif
      endif
      Qin=(Qih+Qd)
      Qird=Qih+Qdes+Qsd
      Qira=Qic+Qads+Qsa
      Qar=min((cbed+crefr*omega1)*(T1-max(T2,T4)),
     +      (cbed+crefr*omega2)*(min(T4,T2)-T3))
c
c     Isosteric regeneration
c
      if(T4.lt.T2)then
         omega5=omegaf(pcond,T2)
         T5=T2
         Qdesr=Rrefr*(bfuni(omega5)-bfuni(omega2))
```

```
         Rintcr=quad(pcond,omega5,omega2)
         Qsdr=cbed*(T5-T4)+crefr*(omega5*T5-omega2*T4+Rintcr)
         Qbr=Qdesr+Qsdr
      else
         Qbr=0
      endif
      Qregen=Qar+Qbr
      Qregen=max(0d0,Qregen)
      cop= (Qcond-eregen*Qregen)/(Qin-eregen*Qregen)
c     cop=Qcond/Qin      ! for basic cycle
c
c      Objective function to minimize, scaled if necessary
c
      f= -cop*1d3              ! - is for minimizer, scaled up
c
c      Equality constraints
c
c      g(1)=afun(omega2)+bfun(omega2)/T3-log(pevap)
c      g(2)=afun(omega1)+bfun(omega1)/T1-log(pcond)
c      g(3)=Qria-Qrid           ! Regenerator balance condition
c
c      Inequality constraints, expressed as G >= 0
c
c     g(1)=(Te-280)             ! Prevent freezeup of evaporator
c     g(2)=(500-T1)             ! Prevent overheating zeolite
      g(1)=(T3-Te)              ! Cycle constraint
      g(2)=(T4-Tc)              ! Desorbed fluid must go out as vapor
      g(3)=(Tc-Tcmin)           ! Reqd. for rejection of waste heat
      g(4)=(Temax-Te)           ! Reqd. for cooling to be possible
c      g(7)=(T3-280)            ! Prevent freezing of zeolite
c
      if(fin.and.num.eq.0)then
         write(*,30)Te,Tc,T1,T2,T3,T4,cop
   30 format(/,' Final cycle temperatures : ',6f6.0,f7.4)
         write(*,32)cop,Qcond,Qin,Qregen
   32 format(' COP = ',f6.4,-3p,'  Qcond = ',f5.0,'  Qin = ',f5.0,
     +       '  Qregen = ',f5.0,0p)
      endif
      objfun= f
      return
      end
```

```
      subroutine nlgrad(nocons,noeqco,nommax,novari,objfun,
     1    constr,gradof,gradco,variab,active,coneps)
      implicit double precision(a-h,o-z)
      dimension constr(nommax),gradof(novari),gradco(nommax,novari),
     1    variab(novari),coneps(nommax)
      logical active(nommax)
c     DIMENSION G(2)
c
c   EVALUATION OF GRADIENTS BY FINITE DIFFERENCES
c
      on=1.d+0
      eps=1.d-7
      do1 i=1,novari
      xeps=eps*dmax1(on,dabs(variab(i)))
      xepsi=on/xeps
      variab(i)=variab(i) + xeps
      call nlfunc(nocons,noeqco,nommax,novari,feps,
     1     coneps,variab,active)
      gradof(i)=(feps - objfun)*xepsi
      do2 j=1,nocons
      if (.not.active(j)) goto 2
      gradco(j,i)=(coneps(j) - constr(j))*xepsi
    2 continue
    1 variab(i)=variab(i) - xeps
      return
      end
c
      double precision function psat(T)
      implicit real*8(a-h,o-z)
      data hfg/2462200d0/, R/461d0/, T0/373.15d0/
      data p0/101325d0/
      psat= exp((1/T0-1/T)*hfg/R)*p0
      return
      end

      double precision function hfgw(T)
      implicit real*8(a-h,o-z)
      hfgw=2462200d0
      return
      end

      double precision function omegaf(p,T)
```

```
      implicit real*8(a-h,o-z)
      afun(a)=(((1.6448d3*a-7.3176d2)*a+1.10854d2)*a+
     +         13.4244d0+4.660517d0)
      bfun(a)=(((-3.4867d3*a+0.564447d4)*a+67.2292d2)*a-
     +         7373.78d0)
      dafun(x)=(1.6448d3*3*x-7.3176d2*2)*x+1.10854d2
      dbfun(x)=(-3.4867d3*3*x+0.564447d4*2)*x+67.2292d2
      rp=log(p)
      x=(rp-4.660517d0-13.4244d0+7373.78d0/T)/
     +  (1.10854d2+67.2292d2/T)
      do 10 i=1,20
        dx=(afun(x)+bfun(x)/T-rp)/(dafun(x)+dbfun(x)/T)
        x=x-dx
        if(abs(dx).lt.1d-6)goto 20
   10 continue
   20 omegaf=x
      return
      end

      double precision function quad(p,ome1,ome2)
      implicit real*8(a-h,o-z)
      parameter (Nint=8)
      afun(a)=(((1.6448d3*a-7.3176d2)*a+1.10854d2)*a+
     +         13.4244d0+4.660517d0)
      bfun(a)=(((-3.4867d3*a+0.564447d4)*a+67.2292d2)*a-
     +         7373.78d0)
      funi(x)=bfun(x)/(rlnp-afun(x))
c
      dx=(ome2-ome1)/Nint
      rlnp=log(p)
      quad=funi(ome1)+funi(ome2)
      do 10 i=1,Nint-1,2
   10    quad=quad+4*funi(ome1+i*dx)
      do 20 i=2,Nint-2,2
   20    quad=quad+2*funi(ome1+i*dx)
      quad=quad*dx/3
      return
      end
```

2.9.2 Include file 'fun1' for Methanol–Carbon-Carbon

```
afun(a)=((52.3793d0*a-11.66841d0)*a+6.53035d0)*a+
     +        20.3305d0+4.660517d0
bfun(a)=((40.5379d3*a-2.605877d4)*a+63.1516d2)*a-
     +        6003.58d0
bfuni(a)=(((40.5379d3/4*a-2.605877d4/3)*a+63.1516d2/2)*a-
     +        6003.58d0)*a
dafun(a)=(52.3793d0*3*a-11.66841d0*2)*a+6.53035d0
dbfun(a)=(40.5379d3*3*a-2.605877d4*2)*a+63.1516d2
```

2.9.3 Optimization of COP for Methanol–Carbon-Carbon

```
      program methcarb
c
c     Compute optimum COP of adsorption heat pump with
c     methanol as refrigerant and Carbon-Carbon as adsorbent
c
      implicit double precision(a-h,o-z)
      parameter (ivar=10)
      parameter (iconst=20)
      parameter (kworka=IVAR*(IVAR+ICONST+25)+11*ICONST+60)
      parameter (kkwork=300)
      parameter (imnn2=2*ivar+iconst+2)
      parameter (iact=3*iconst+15)

      dimension variab(ivar),constr(iconst)
      dimension gradof(ivar)
      dimension vecmul(imnn2),boulow(ivar),bouupp(ivar)
      dimension workar(kworka),iworka(kkwork)
      dimension gradco(iconst,ivar)
      dimension  hessem(ivar,ivar),rhside(ivar)
      logical active(iact),lmerit,lql

      common/cmache/eps100,eps200,eps300
      common /ctrl/Temax,Tcmin,start,fin,num,etareg
      real*8 Temax,Tcmin,etareg
      logical start,fin

      eps100=epsilon(eps100)
      eps200=1.d-7
      eps300=1.d-3
```

```
c  NUMBER OF CONSTRAINTS:
      nocons=4

c  NUMBER OF EQUALITY CONSTRAINTS:
      noeqco=0

c  NUMBER OF VARIABLES
      novari=4

c  BOUNDS ON VARIABLES
      boulow(1)=280.0
      bouupp(1)=500.0
      boulow(2)=280.0
      bouupp(2)=500.0
      boulow(3)=280.0
      bouupp(3)=500.0
      boulow(4)=280.0
      bouupp(4)=500.0

c  USER SPECIFIED ACCURACY
      accura=1.d-8

c  USER SPECIFIED SCALING PARAMETER
      scabou=1.d+3

c  PRINT LEVEL SPECIFIED
      iprint=0

c  MODE OF OPERATION
      modeal=0

c  DESIRED OUTPUT UNIT NUMBER
      ioutst=8

c  IF TRUE, USE L2 MERIT FUNCTION.  OTHERWISE L1.
      lmerit = .true.

c  IF TRUE, USE FULL QP SUBPROBLEM.
      lql=.true.

c  ROW DIMENSION OF DG -  .GE. (MAX (1, NOCONS))
      nommax=iconst
```

```
c  ROW DIMENSION OF C - .GE. (MAX (2, NOVARI))
      nonmax=ivar

c  MAX NUMBER OF FUNCTION CALLS DURING LINE SEARCH
      maxfun=8

c  UPPER BOUND ON THE NUMBER OF ITERATIONS
      maxite=40

c  EQUAL TO NOCONS + NOVARI +NOVARI +2
      nomnn2= nocons + novari + novari + 2
c  OUTPUT PARAMETER REGD. TERMINATION. SHOULD BE ZERO AT START
      infail=0

c  WORKING PARAMETER MIGHT HAVE TO BE ENLARGED IF NEEDED
      lework=nonmax*nommax + 4*nommax + 4*nommax +
     1 19*nonmax+55 +1.5*(nonmax+1)**2 + 15*nonmax
     2 + 4*nommax + 20
c  INTEGER PARAMETER TO DETERMINE SPACE FOR SUBPROBLEM SOLUTION
      leiwor=19+ivar +ivar+iconst

c  SPACE FOR  LOGICAL VARIABLES OF CONSTRAINT ACTIVITY
      leacti=3*nommax + 15

      open(unit=ioutst,file='nlpql.res')

c  INITIAL VALUES OF VARIABLES
      open(unit=9,file='dat')
      rewind(9)
      read(9,*)(variab(i),i=1,novari),etareg
      close(9)
      Temax=variab(1)
      do 1000 Tcmin=Temax+20,500

      fin=.false.
       num=0
c
c     Call nonlinear constrained optimization routine
c
      call nlpql1(nocons,noeqco,nommax,novari,nonmax,nomnn2,
     1  variab,objfun,constr,gradof,gradco,vecmul,boulow,bouupp,
```

```
     2  hessem,rhside,accura,scabou,maxfun,maxite,iprint,modeal,
     3  ioutst,infail,workar,lework,iworka,leiwor,active,leacti,
     4  lmerit,lql)
c
c    OUTPUT RESULT
c
      num=infail
      fin=.true.
      call nlfunc(nocons,noeqco,nommax,novari,objfun,constr,
     +           variab,active)
      write(*,9020) infail,-objfun*1d-3,(variab(i),i=1,novari)
 9020 format(1x,i4,f8.4,8f8.2)
 1000 continue
      stop
      end
c
      block data
      logical start,fin
      common /ctrl/Temax,Tcmin,start,fin,num,etareg
      real*8 Temax,Tcmin,etareg
      data fin/.false./,start/.true./
      end
c
      subroutine nlfunc(nocons,noeqco,nommax,novari,objfun,g,
     1     x,active)
      implicit double precision(a-h,o-z)
      dimension g(nommax),x(novari)
      logical active(nommax),fin,start
      common /ctrl/Temax,Tcmin,start,fin,num,etareg

c    EVALUATION OF OBJECTIVE FUNCTIONS
c
      include 'fun1'
c
      data crefr/2540d0/, cbed/920.0d0/, Rrefr/259.8d0/
      data cvap/1340.0d0/,cliq/2540d0/
c
c     crefr=cvap
      Te=x(1)
      Tc=x(2)
      T1=x(3)
      T3=x(4)
```

```
      pcond=psat(Tc)
      pevap=psat(Te)

      omega2=omegaf(pevap,T3)
      omega1=omegaf(pcond,T1)

      T2=bfun(omega1)/(log(pevap)-afun(omega1))
      T4=bfun(omega2)/(log(pcond)-afun(omega2))
      hfg=hfgw(Tc)
c
      Qih=(cbed+omega2*crefr)*(T4-T3)       ! Isosteric heating
      Qdes=Rrefr*(bfuni(omega1)-bfuni(omega2))
      Rintc=quad(pcond,omega1,omega2)
      Qsd=cbed*(T1-T4)+crefr*(omega1*T1-omega2*T4+rintc)
      Qic=(cbed+crefr*omega1)*(T1-T2)       ! Isosteric cooling
      Qads=Qdes
c
      rinte=quad(pevap,omega1,omega2)
      Qsa=cbed*(T2-T3)+crefr*(omega1*T2-omega2*T3+rinte)
      Qsve=cvap*(rinte-Te*(omega2-omega1))
      Qa=Qads+Qsa-Qsve
      Qd=Qdes+Qsd

      Qsvc=cvap*(rintc-Tc*(omega2-omega1))
      Qc=Qsvc+(omega2-omega1)*hfg
      Qe=(omega2-omega1)*(hfg-cliq*(Tc-Te))

c      Qcond=max(Qic,0d0)+max(Qa,0d0)+max(Qc,0d0)
      Qcond=Qic+Qa+Qc
      if(fin)then
         if(Qic.lt.0.or.Qa.lt.0.or.Qc.lt.0)then
           write(*,*)' Wrong sign'
         endif
         if(Qih.lt.0.or.Qd.lt.0)then
           write(*,*) 'negative heat input'
         endif
      endif
      Qin=(Qih+Qd)
      Qird=Qih+Qdes+Qsd
      Qira=Qic+Qads+Qsa
      Qar=min((cbed+crefr*omega1)*(T1-max(T2,T4)),
```

```
     +      (cbed+crefr*omega2)*(min(T4,T2)-T3))
c
c     Isosteric regeneration
c
      if(T4.lt.T2)then
         omega5=omegaf(pcond,T2)
         T5=T2
         Qdesr=Rrefr*(bfuni(omega5)-bfuni(omega2))
         Rintcr=quad(pcond,omega5,omega2)
         Qsdr=cbed*(T5-T4)+crefr*(omega5*T5-omega2*T4+Rintcr)
         Qbr=Qdesr+Qsdr
      else
         Qbr=0
      endif
      Qregen=Qar+Qbr
      Qregen=max(0d0,Qregen)*etareg
      cop= (Qcond-Qregen)/(Qin-Qregen)
c     cop=Qcond/Qin         ! simple cycle w/o regeneration
c
c     Objective function to minimize, scaled if necessary
c
      f= -cop*1d3            ! - for minimizer, scaled up
c     Equality constraints  ! for cocurrent flow
c
c     g(1)=afun(omega2)+bfun(omega2)/T3-log(pevap)
c     g(2)=afun(omega1)+bfun(omega1)/T1-log(pcond)
c     g(3)=Qria-Qrid      ! Regenerator balance condition
c
c     Inequality constraints, expressed as G >= 0
c
c     g(1)=(Te-280)            ! Prevent freezeup of evaporator
c     g(2)=(500-T1)            ! Prevent damage to zeolite
      g(1)=(T3-Te)             ! Cycle constraint
      g(2)=(T4-Tc)             ! Desorbed fluid must go out as vapor
      g(3)=(Tc-Tcmin)          ! Reqd. for rejection of waste heat
      g(4)=(Temax-Te)          ! Reqd. for cooling to be possible
c     g(7)=(T3-280)            ! Prevent freezing of zeolite
c
      if(fin.and.num.eq.0)then
         write(*,30)Te,Tc,T1,T2,T3,T4,cop
   30 format(/,' Final cycle temperatures : ',6f6.0,f7.4)
         write(*,32)cop,Qcond,Qin,Qregen
```

```
   32 format(' COP = ',f6.4,-3p,'  Qcond = ',f5.0,
     +       '  Qin = ',f5.0,'  Qregen = ',f5.0,0p)
        endif
        objfun= f
        return
        end
c
        subroutine nlgrad(nocons,noeqco,nommax,novari,objfun,
       1    constr,gradof,gradco,variab,active,coneps)
        implicit double precision(a-h,o-z)
        dimension constr(nommax),gradof(novari),
       1    gradco(nommax,novari),variab(novari),
       2    coneps(nommax)
        logical active(nommax)
c       DIMENSION G(2)
c
c   EVALUATION OF GRADIENTS BY FINITE-DIFFERENCES
c
        on=1.d+0
        eps=1.d-7
        do1 i=1,novari
        xeps=eps*dmax1(on,dabs(variab(i)))
        xepsi=on/xeps
        variab(i)=variab(i) + xeps
        call nlfunc(nocons,noeqco,nommax,novari,feps,
       1     coneps,variab,active)
        gradof(i)=(feps - objfun)*xepsi
        do2 j=1,nocons
        if (.not.active(j)) goto 2
        gradco(j,i)=(coneps(j) - constr(j))*xepsi
    2 continue
    1 variab(i)=variab(i) - xeps
        return
        end
c
        double precision function psat(T)
c
c       Vapor pressure of methanol
c
        implicit real*8(a-h,o-z)
        data hfg/1174109d0/, R/259.8d0/, T0/337.80d0/
        data p0/101325d0/
```

```
      psat= exp((1/T0-1/T)*hfg/R)*p0
      return
      end
c
      double precision function hfgw(T)
c
c     Latent heat of methanol
c
      implicit real*8(a-h,o-z)
      hfgw=1174109d0
      return
      end
c
      double precision function omegaf(p,T)
      implicit real*8(a-h,o-z)
c
      include 'fun1'
c
      rp=log(p)
      x=(rp-4.660517d0-20.3305d0+6003.58d0/T)/
     1  (6.53035d0+63.1516d2/T)
      do 10 i=1,20
        dx=(afun(x)+bfun(x)/T-rp)/(dafun(x)+dbfun(x)/T)
        x=x-dx
        if(abs(dx).lt.1d-6)goto 20
   10 continue
   20 omegaf=x
      return
      end
c
      double precision function quad(p,ome1,ome2)
      implicit real*8(a-h,o-z)
      parameter (Nint=8)
c
      include 'fun1'
c
      funi(x)=bfun(x)/(rlnp-afun(x))
c
      dx=(ome2-ome1)/Nint
      rlnp=log(p)
      quad=funi(ome1)+funi(ome2)
      do 10 i=1,Nint-1,2
```

```
10     quad=quad+4*funi(ome1+i*dx)
   do 20 i=2,Nint-2,2
20     quad=quad+2*funi(ome1+i*dx)
   quad=quad*dx/3
   return
   end
```