Journal of Asian Scientific Research

journal homepage: http://aessweb.com/journal-detail.php?id=5003

# SOME NUMERICAL RESULTS ON FUNCTION SPACE (FSA) ALGORITHMS

**J. O. Omolehin**[1]
**K. Rauf**[2]
**A. Mabayoje**[3]
**O. T. Arowolo**[4]
**A. Lukuman**[5]

## ABSTRACT

*In this work, we consider the numerical implementation of Function Space Algorithm (FSA) for the solution of quadratic continuous cost functional. It is used to solve Reaction Diffusion Control problems. It considered specifically a parabolic problem characterized by dynamics constraints and the results obtained analyzed. The cumbersome nature of the line search techniques associated with FSA was addressed by time discretization approach. It is shown that the convergence rate of FSA improves as the penalty parameter grows.*

**Key Words:** FSA, Penalty parameter, Constraints, Convergence rate, Time discretization
**2010 Mathematics Subject Classification:** 93B40, 93CXX & G 1.7

## INTRODUCTION

Let us first consider the quadratic functional of the form:

$$F(x) = F_o + < a, x >_H + \frac{1}{2} < x, Ax >_H ,$$

Where $A$ is an $nxn$ symmetric positive definite matrix operator on the Hilbert space $H$. $\alpha$ is a vector in $H$ and $F_0$ is a constant term.

[1] Mathematics and Computer Science Department, IBB University, Lapai, Nigeria

[2] Mathematics Department, University of Ilorin, Ilorin, Nigeria

[3] Computer Science Department, University of Ilorin, Ilorin, Nigeria

[4] Mathematics and Statistics Department, Lagos State Polytechnics, Lagos State

[5] Federal College of Fisheries and Marine Technology, Lagos State

Let us also consider what is termed conjugate descent with $F$. With conjugate descent, it is assumed that a sequence

$$\{p_i\} = p_0, p_1, \dots, p_k, ..$$

is available with the members of the sequence conjugate with respect to the positive definite linear operator $A$.

By conjugate with respect to $A$, we mean that

$$< p_i, Ap_i >_H = \begin{cases} \neq 0, & if\ i \neq j \\ = 0, & if\ i = j \end{cases}$$

In this case, $A$ is assumed positive definite so $< p_i, Ap_i >_H > 0$.

The conventional Conjugate Gradient Method (CGM) was originally designed for the minimization of a quadratic objective functional of the form stated above.

## STAGES INVOLVED IN CONJUGATE GRADIENT METHOD

Stage 1: The first element $x_0 \in H$ of the descent sequence is guessed while the remaining members of the sequence are computed with the aid of the following formulae:

Stage 2: $p_0 = -g_0 = -(a + Ax_0)$

($p_0$ is the descent direction and $g_0$ is the gradient of $F(x)$ when $x = x_0$)

Stage 3: $x_{i+1} = x_i + \alpha_i p_i,\ \alpha_i =< g_i, g_i >_H / < p_i, Ap_i >_H$

$$g_{i+1} = g_i + a_i Ap_i;$$

$\alpha$ is the step length

$$p_{i+1} = -g_{i+1} + \beta_i p_i;\ \ \beta_i =< g_{i+1}, g_{i+1} >_H / < g_i, g_i >_H$$

Stage 4: if $g_i$ for some $i$ terminate the sequence else, set $i = i + 1$ and go to stage 3.

The CGM has a well worked out theory with an elegant convergence profile. It has been proved that the algorithm converges, at most, in n iterations in a well posed problem and the convergence rate is given as:

$$E(x_n) = \left\{ \frac{1 - \frac{m}{M}}{1 + \frac{m}{M}} \right\}^{2n} E(x_0)$$

Where m and M are smallest and spectrums of matrix A respectively.

That is, for an n dimensional problem, the algorithm will converge in at most n iterations.

The CGM algorithm cannot handle quadratic cost functional of the form:

Minimise

$$\int_0^T \{av^2(t) + bu^2(t)\}dt$$

Subject to

$$\dot{v} = cv(t) + du(t)$$

For the reason that operator A was not known explicitly in continuous cost functional, researchers came up with different approximation – based techniques that could estimate $\alpha_i$ that minimizes $F(x_i + \alpha p_i)$. [(1)] In this fashion, there came into being various cumbersome techniques. The most popular among such methods is the conventional function space (CFS) algorithm, [(1)] to minimize the continuous cost functional of the form:

**Problem (1)**

Minimise $\int_o^\delta \{x^T(t)Qx(t) + u^T(t)P\,u(t)\}dt$

Subject to the dynamic constraints

$$\dot{x}(t) = C\,x(t) + Du(t),$$

$0 < t < \delta$ ($\delta$ given); where $x(t)$ denotes the transpose of x(t), $\dot{x}(t)$ stands for the first derivative of x(t) with respect to t. x(t) is the n x 1 state vector, u(t) is the q x 1 control vector, C and D are n x q constant matrices respectively, while Q and R are symmetric, positive definite, constant square matrices of dimensions n and q respectively.

The control operator A is associated with problem (1) satisfying

$$< A, AZ >_K = J(x, u, \mu) = \int_0^\delta \{x^T(t)Qx(t) + u(t)^T Pu(t)$$

$$+\mu || \dot{x}(t) - Cx(t) - Du(t) ||^2\}dt \quad (\mu > 0)$$

by transforming problem (1) into an unconstrained optimal control problem. Where $\mu$ is the penalty constant $K$ is given by $K = H_1[0, \delta] \times L^q_2[0, \delta]$, and $H_1[0, \delta]$, denotes sobolev space of the absolutely continuous functions x(.), square integrable over the closed interval $[0, \delta]$.

$L_2^q[0,\delta]$ stands for the Hilbert space consisting of the equivalence classes of square integrable functions from [0, δ] into Rq, with norm denoted by $||.||_E$ and defined by

$||\ u\ ||=\int_0^\delta\{||u||^2\}^{\frac{1}{2}}\ dt$ and with scalar product conventionally denoted by < .,.> and defined by

$<u_1,\ u_2>=\int_0^\delta <\ u_1, u_2\ >_E\ dt$ where $||.||_E$ and $<.,.>_E$ denote the norm and scalar product in Euclidean q – dimensional space.

## FUNCTION SPACE ALGORITHM

The function space algorithm is constructed to solve the optimal control problem [(1)]:

$$\text{Min J (x, u, μ)} = \text{Min}\int_0^R\{x^T(t)\ Px\ (t)\ +u^T(t)\ Qu(t)\ \}\ dt$$

$$+\mu\int_0^T|\ |(\dot{x}(t)-\ Cx(t)-\ Du(t)\ ||\ ^2dt$$

Where C and D are constant matrices of appropriate dimensions. The steps involve in FSA is as follows:

**Step 1**

choose the initial values $\dot{x}_0(t),\ u_o(t),$

where $0<t<T,\qquad T\ \textit{is known and compute}\ x_o(t)=\int_o^T\dot{x}_0(t)\ dt$

**Step 2**

Initialize the Counter: $i=0,$ and compute $[\nabla_x^J]_{i,}\ \left[\nabla_U^J\right]_i$ using formulae for

$$\nabla J=\begin{bmatrix}\nabla_X^J\\ \nabla_U^J\end{bmatrix}$$

$$\left[\nabla_X^J\right]=2\mu\ (\dot{x}\ (t)-\ f\ (x(t),\ u(t),t)]-\int_T^t(\frac{\partial T}{\partial x})^T-\left(\frac{\partial f}{\partial x}\right)^T\ [2\mu(\dot{x}(s)-\ f(x(s),\ u(s),\ s)]\ \}ds$$

$0\le t\le T$

$$[\nabla_u^J]\ =\ \left(\frac{\partial I}{\partial u}\right)-\ (\frac{\partial F}{\partial v})^T\ [2\mu(\dot{x}\ (t)-\ f(x\ (t),\ u(t),t])],\quad 0<t\ <\ T$$

**Step 3**

Compute the current descent direction.

$$\dot{s}_{x,i}(t) = \begin{cases} -[\nabla]o, & for\ i=o \\ -\left[\nabla_X^J\right]i + \beta_{i-1} S_{X,i-1}(t), & for\ i>0, where\ t \in [o,T] \end{cases}$$

$$s_{x,i}(t) = \int_o^t \dot{s}_{x,i}(t)\, dt$$

$$\dot{s}_{u,i}(t) = \begin{cases} -\left[\nabla_U^J\right]_0, & for\ i=o \\ -\left[\nabla_U^J\right]_i + \beta_{i-1} S_{i-1}(t) & for\ i>0 \end{cases}$$

Where $t \in [0, T]$, and

$$\beta_{i-1} = \frac{\int_o^T \left|\left|\left[\nabla_x^J\right]_i\right|\right|^2 dt + \int_o^T \left|\left|\left[\nabla_u^J\right]_i\right|\right|^2 dt}{\int_o^T \left|\left|\left[\nabla_x^J\right]_{i-1}\right|\right|^2 dt + \int_o^T \left|\left|\left[\nabla_u^J\right]_{i-1}\right|\right|^2 dt}$$

**STEP 4**

Find $P^*_i$

such that $J(x_i + P_i^* s_{x,i},\ u_i + P^*_i s_{u,i},\ \mu,) < J(x_i + p_i s_{X,i}\, u_i + p_i s_{u,i},\ \mu,),\ p \geq o$

**STEP 5**

Test for the stopping criterion of the algorithm by verifying if

$$S(x,\ u,\ \mu) = ||\, Ø\,(z)\ ||^2 + (\nabla^J{}_x)^T\ (\nabla^J{}_x) + (\nabla_u{}^J)^T\ (\nabla_u{}^J) \leq \varepsilon,$$

$$(Ø = \dot{x} - f)$$

Where ε is a chosen predetermined tolerance to indicate that the desired accuracy required for the computational programming method.

**STEP 6**

Set $x_{i+1}(t) = x_i(t) + p *_i s(t)$ $\quad 0 \leq t \leq T$

$u_{i+1}(t) = u_i(t) + p *_i s(t)$ $\quad 0 \leq t \leq T$

**STEP 7**

Set i = i + 1 and goto step 2

To the best knowledge of the authors, no numerical work has been carried out prior to the time this work was concluded.

## NUMERICAL RESULTS ON FSA

In carrying out our numerical investigation using FSA algorithm discussed, we study computationally the convergence rate of various diffusion equation problems which are only dimensionally different from one another. Thus, when we say the dimensionality of the resulting diffusion control problem is $\mathbb{R}^{12}$ and $\mathbb{R}^{11}$ it means that is α, u ∈ $\mathbb{R}^{12}$ and

α, u ∈ $\mathbb{R}^{11}$, respectively.

Thus, in $\mathbb{R}^{12}$ and $\mathbb{R}^{11}$ we have problems (1) and (2) respectively as stated below:

**PROBLEM (1)**

Minimize $\int_o^1 \{\alpha_1^2(t) + \alpha_2^2(t) + \ldots + \alpha_{12}^2(t) + u_1^2(t) + u_2^2(t) + \ldots + u_{12}^2(t)\}dt$

Subject to $\alpha_1(t) = -\pi^2\alpha_1(t) + u_1(t)$

$\alpha_2(t) = -4\pi^2\alpha_2(t) + u_2(t)$

……

$\alpha_{12}(t) = -144\pi^2\alpha_{12}(t) + u_{12}(t)$

The problem is transformed into an unconstrained problem with the introduction of a penalty constant μ and it becomes:

Min J (α, u, μ) = Min $\int_o^1 \{\alpha_1^2(t) + \alpha_2^2(t) + \ldots + \alpha_{12}^2(t) + u_1^2(t) + u_2^2(t) + \ldots + u_{12}^2(t)]\, dt$

$+ \mu\{\int_0^1 \{\left|\left|\alpha_1(t) + \pi^2\alpha_1(t) - u_1(t)\right|\right|^2 + \left|\left|\alpha_2(t) + 4\pi^2\alpha_2(t) - u_2(t)\right|\right|^2 + \left|\left|\alpha_3(t) + 9\pi^2\alpha_3(t) - u_3(t)\right|\right|^2 + \cdots + \left|\left|\alpha_{12}(t) + 144\pi^2\alpha_{12}(t) - u_{12}(t)\right|\right|^2\}\}$dt

Also, problem in $\mathbb{R}^{11}$ i.e. when α, u∈ $\mathbb{R}^{11}$, we obtain the following equivalent problem formulation:

**PROBLEM (2)**

Minimize $\int_o^1 \{\alpha_1^2(t) + \alpha_2^2(t) + \ldots + \alpha_{11}^2(t) + u_1^2(t) + u_2^2(t) + \ldots + u_{11}^2(t)dt$

Subject to

$$\alpha_1(t) = -\pi^2\alpha_1(t) + u_1(t)$$

$$\alpha_2(t) = -4\pi^2\alpha_2(t) + u_2(t)$$

……

$$\alpha_{11}(t) = -122\pi^2\alpha_{11}(t) + u_{11}(t)$$

The problem is now transformed into an unconstrained problem with the introduction of a penalty constant μ.

$$\text{Min } J(\alpha, u, \mu) = \text{Min } \int_o^1 \{\alpha_1^2(t) + \alpha_2^2(t) + \ldots + \alpha_{11}^2(t)(\alpha, u)$$

$$+ u_1(t) + u_2^2(t) + \ldots + u_{11}^2(t)]\, dt + \mu\{\int_0^1 \{\left|\left|\alpha_1(t) + \pi^2\alpha_1(t) - u_1(t)\right|\right|^2 +$$

$$\left|\left|\alpha_2(t) + 4\pi^2\alpha_2(t) - u_2(t)\right|\right|^2 + \left|\left|\alpha_3(t) + 9\pi^2\alpha_3(t) - u_3(t)\right|\right|^2 + \cdots +$$

$$\left|\left|\alpha_{11}(t) + 121\pi^2\alpha_{11}(t) - u_{11}(t)\right|\right|^2\}\}dt.$$

The problems in other dimensions can easily be formulated in similar manner. Meanwhile, the convergence rate for various values of penalty constants in $\mathbb{R}^{12}$ is shown by Tables (1.1) – (1.6).

**Table-1. FSA Algorithm in $\mathbb{R}^{12}$**

Table (1.1): μ = 10

| Time (t) | Iteration Num | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ |
|---|---|---|---|---|---|---|---|---|
| **0.2** | 4 | 0.998991 | .992290 | 0.981630 | 0.967777 | 0.951799 | 0.935071 | 0.919281 |
| **0.4** | 9 | 0.997361 | 0.986562 | 0.969175 | 0.94612 | 0.918683 | 0.888516 | 0.857644 |
| **0.6** | 9 | 0.996464 | 0.983578 | 0.962585 | 0.934212 | 0.899475 | 0.859680 | 0.816428 |
| **0.8** | 8 | 0.996073 | 0.982462 | 0.960030 | 0.929157 | 0.890379 | 0.844381 | 0.79200 |

| $u_8$ | $u_9$ | $u_{10}$ | $u_{11}$ | $u_{12}$ | **Objective Function** |
|---|---|---|---|---|---|
| 0.906385 | 0.898730 | 0.898913 | 0.909716 | 0.934522 | 14.064660 |
| 0.828453 | 0.803700 | 0.786510 | 0.780373 | 0.789148 | 11.81770 |
| 0.771602 | 0.727384 | 0.686244 | 0.650937 | 0.624517 | 11.100700 |
| 0.734227 | 0.672205 | 0.607229 | 0.540747 | 0.474358 | 13.85520 |

Table (1.2): $\mu = 20$

| Time (t) | Iteration Num | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ |
|---|---|---|---|---|---|---|---|---|
| **0.2** | 5 | 0.998933 | 0.992232 | 0.981572 | 0.967719 | 0.951742 | 0.935017 | 0.919213 |
| **0.4** | 8 | 0.997315 | 0.986516 | 0.969129 | 0.946074 | 0.918636 | 0.888471 | 0.857598 |
| **0.6** | 9 | 0.996427 | 0.983541 | 0.962549 | 0.934175 | 0.899438 | 0.859644 | 0.816391 |
| **0.8** | 10 | 0.996044 | 0.982433 | 0.960001 | 0.929128 | 0.890350 | 0.844352 | 0.791971 |

| $u_8$ | $u_9$ | $u_{10}$ | $u_{11}$ | $u_{12}$ | **Objective Function** |
|---|---|---|---|---|---|
| 0.906306 | 0.898689 | 0.898843 | 0.909705 | 0.934476 | 14.063360 |
| 0.828406 | 0.803654 | 0.786463 | 0.780326 | 0.789102 | 11.180790 |
| 0.771565 | 0.727347 | 0.686207 | 0.650901 | 0.624480 | 11.099970 |
| 0.734198 | 0.672176 | 0.607200 | 0.540718 | 0.474329 | 13.854660 |

Table (1.3): $\mu = 30$

| Time(t) | Iteration Num | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ |
|---|---|---|---|---|---|---|---|---|
| **0.2** | 4 | 0.998914 | 0.992212 | 0.981552 | 0.967699 | 0.951722 | 0.934990 | **0.919200** |
| **0.4** | 9 | 0.997299 | 0.986500 | 0.969113 | 0.946058 | 0.918620 | 0.888455 | **0.857582** |
| **0.6** | 10 | 0.996415 | 0.983529 | 0.962536 | 0.934163 | 0.899426 | 0.859632 | **0.816378** |
| **0.8** | 10 | 0.996034 | 0.982423 | 0.959991 | 0.929119 | 0.890350 | 0.844342 | **0.791961** |

| $u_8$ | $u_9$ | $u_{10}$ | $u_{11}$ | $u_{12}$ | **Objective Function** |
|---|---|---|---|---|---|
| **0.906321** | 0.89671 | 0.89880 | 0.909671 | 0.934411 | 14.062860 |
| **0.828391** | 0.803639 | 0.786448 | 0.780311 | 0.789086 | 11.180460 |
| **0.771553** | 0.727336 | 0.686194 | 0.650888 | 0.62448 | 11.099730 |
| **0.734188** | 0.672166 | 0.607190 | 0.540707 | 0.474318 | 13.854470 |

Table (1.4): $\mu = 40$

| Time(t) | Iteration Num | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ |
|---|---|---|---|---|---|---|---|---|
| **0.2** | 10 | 0.998904 | 0.992201 | 0.981540 | 0.967686 | 0.951706 | 0.934978 | **0.919183** |
| **0.4** | 10 | 0.997291 | 0.986492 | 0.969105 | 0.946050 | 0.918612 | 0.888447 | **0.857574** |
| **0.6** | 10 | 0.996409 | 0.983523 | 0.962530 | 0.934157 | 0.899419 | 0.859626 | **0.816372** |
| **0.8** | 9 | 0.996029 | 0.982418 | 0.959986 | 0.929114 | 0.890335 | 0.844337 | **0.791956** |

| $u_8$ | $u_9$ | $u_{10}$ | $u_{11}$ | $u_{12}$ | **Objective Function** |
|---|---|---|---|---|---|
| **0.906309** | 0.898689 | 0.898797 | 0.909663 | 0.934455 | 14.067680 |
| **0.828393** | 0.803631 | 0.786440 | 0.780303 | 0.789079 | 11.180300 |
| **0.771547** | 0.727330 | 0.686188 | 0.650882 | 0.624461 | 11.099610 |
| **0.734183** | 0.672161 | 0.607185 | 0.540703 | 0.474313 | 13.85438 |

Table (1.5): $\mu = 50$

| Time(t) | Iteration Num | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ |
|---|---|---|---|---|---|---|---|---|
| **0.2** | 10 | 0.998890 | 0.992196 | 0.981535 | 0.96768 | 0.951700 | 0.934972 | 0.919177 |
| **0.4** | 9 | 0.997287 | 0.986487 | 0.969100 | 0.946045 | 0.918608 | 0.888442 | 0.857570 |
| **0.6** | 9 | 0.996405 | 0.983519 | 0.962526 | 0.934153 | 0.899416 | 0.859622 | 0.816369 |
| **0.8** | 9 | 0.996002 | 0.982415 | 0.959985 | 0.929111 | 0.890333 | 0.844334 | 0.791953 |

| $u_8$ | $u_9$ | $u_{10}$ | $u_{11}$ | $u_{12}$ | Objective Function |
|---|---|---|---|---|---|
| 0.906303 | 0.898641 | 0.898791 | 0.909658 | 0.934450 | 14.062570 |
| 0.828379 | 0.803626 | 0.786435 | 0.780298 | 0.789074 | 11.180200 |
| 0.771543 | 0.727326 | 0.686184 | 0.650878 | 0.624455 | 11.099540 |
| 0.734180 | 0.672158 | 0.607183 | 0.540700 | 0.474311 | 13.854330 |

Table (1.6): $\mu = 60$

| Time(t) | Iteration Num | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ |
|---|---|---|---|---|---|---|---|---|
| **0.2** | 10 | 0.998895 | 0.992192 | 0.981531 | 0.967676 | 0.951696 | 0.934969 | 0.919173 |
| **0.4** | 10 | 0.997284 | 0.986484 | 0.969097 | 0.946042 | 0.918605 | 0.888439 | 0.857566 |
| **0.6** | 9 | 0.996403 | 0.983517 | 0.962524 | 0.934151 | 0.899413 | 0.859620 | 0.816366 |
| **0.8** | 10 | 0.99024 | 0.982414 | 0.959981 | 0.929109 | 0.890331 | 0.844332 | 0.791951 |

| $u_8$ | $u_9$ | $u_{10}$ | $u_{11}$ | $u_{12}$ | Objective Function |
|---|---|---|---|---|---|
| 0.906299 | 0.888637 | 0.898787 | 0.909654 | 0.934446 | 14.062480 |
| 0.828376 | 0.803623 | 0.786432 | 0.780295 | 0.789070 | 11.180130 |
| 0.771541 | 0.727324 | 0.686182 | 0.650876 | 0.624455 | 11.099480 |
| 0.734178 | 0.672176 | 0.607180 | 0.540698 | 0.474309 | 13.854290 |

## ANALYSIS OF RESULTS

If you look at the results of the tables, it is easily shown that the convergence rate of our method improves as the penalty parameter grows without bound. It is observed that the values of the calculated objective function keep on decreasing as the penalty parameter $\mu$ grows. Part of future research in this work is to determine the optimal value for $\mu$.

## CONCLUSION AND RECOMMENDATION FOR FUTURE WORK

It is clear that the conventional function space algorithms for solving minimization of penalized cost functional for optimal control problem characterized by linear system integral quadratic cost due to Di pillo et al [(1)] though falling within the framework of conjugate gradient method algorithm, is difficult to apply computationally. For further work [(2-6)].

The advantage of this method is that, it can handle continuous quadratic functional which cannot be tacked by conventional conjugate method algorithm. The area of future research is to find a way of circumventing the line search techniques associated with FSA.

## REFERENCES

**Di Pillo, G and Grippo, L. (1972)**: A Computing Algorithm for the Epsilon Method to Identification and Optimal Control Problems, Ricerche di Automatica, Vol.**3**, pp.54-77.

**Ibiejugba, M.A. (1985)**: Computational methods in optimization, Ph.D. Thesis, University of Leeds, Leeds, U.K..

**Omolehin, J. O. (1986)**: Experiment with extended conjugate gradient method algorithm, Unpublished M. Sc. Thesis, University of Ilorin, Ilorin. , Nigeria.

**Omolehin, J. O. (1991)**: On the control of reaction diffusion equation, Unpublished Ph.D Thesis, University of Ilorin, Ilorin, Nigeria.

**Omolehin, J. O. (2005)**: Eigen value perturbation for gradient method, Anaele Stiintifice Ale Universitata "Al,I.Cuza,Tomul L1,s.I.Matematical,f. Vol. 1, pp. 127-132. **MR2187363 (2006h:65083)** 65K10.

**Russel, David, L. (1970)**: Optimization theory, W.A. Benjamin, Inc. New York .

## APPENDIX

```
C       OPEN  12, ''JULU'',ATI=''AP''
                        C THIS PROGRAME IS USED TO MINIMIZE AN OBJECTIVE FUNCTIONAL THROUGH THE METHOD OF FUNCTION SPACE ALGORITHM  DIMENSION X(12),U(12),PAE(12),LAMDA(12),GX(12),GU(12)
                                DIMENSION UPGX(12), UPGU(12),SX(12),SU(12),DOTSX(12)
        DIMENSION UPDOTSX(12), PSU(12),UPSX(12),DOTX(12),UPDOTX(12)
                        DIMENSION UPX(12), UPU(12),BITA (12)

    TIME=-0.2

    ITERA=0
                DO 3 I=1, 12
                        X(I)=0.0
                                U(I)=0.0
                                        PAE(I)=0.0
                                                LAMDA(I)=0.0
                                                        G X(I)=0.0

    GU(I)=0.0
        UPGX(I)=0.0
                UPGU(I)=0.0
                        BITA (I)=0.0
                                UPLAMDA (I)=0.0

     SX(I)=0.0
```

```
      SU(I)=0.0
                  DOT SX(I)=0.0
                        UPDOTSX(I)=0.0
                                    PSU(I)=0.0

      UPX(I)=0.0

      UPU(I)=0.0
            UPSX(I)=0.0
                  DOTX(I)=0.0
                        UPDOTX(I)=0.0
                              CONTINUE

     AM=0.0
                   70     AM=AM+10.0
                              WRITE (12, 301)AM
                                          301
FORMAT(70X,''AM='',F15.6//////)
            IF(AM.GT.(10.0))GOTO 60

     TIME=0.
            TIME=TIME+0.2
                        IF(TIME.GT.1)GOTO 70
                              DO 4 I=1, 2
                                    DOTX(I)=1.0
                                          X(1)=TIME

      BITA(I)=0.0
            U(I)=1.0

      CONTINUE

      ITERA=0

      BITA(I)=0.0
            ITERA=ITERA+1
                        SUMU=0.0

     SUMX=0.0
                     C  THE CONSTRUCTION OF THE GRADIENT FOLLOWS
```

```
                                                    DO 5 I=1, 2
                                                        PAE(I)=(I**2)*(3.142**2)

        DI=2*AM*(PAE(I)*X(I)- U(I))
              D2=2*X(I)*TIME-2*X(I)
         D3=PAE(I)*2*AM*(PAE(I)*X(I)-U(I)*(TIME)-1)

         GX(I)=D1-(D2+D3)
              GU(I)=2*U(I)-2*AM*(PAE(I)*X(I)-U(I))

         CONTINUE
         DO 13 I=1,2
        DOTSX(I)=-GX(I)+BITA(I)*SX(I)
        SX(I)=DOTSX(I)*TIME

         SU(I)=-GU(I)+BITA(I)*SU(I)

         CONTINUE
        RNI=(X(I)*SX(I))+(X(2)*SX(2))+(X(3)*SX(3))+(X(4)*SX(4))+(X(5)*SX(5))+(X(6)*SX
(6))
        RN2=(X(7)*SX(7))+(X(8)*SX(8))+(X(9)*SX(9))+(X(10)*SX(10))+(X(11)*SX(11))+(X(
12)*SX(12))
        RN3=(U(1)*SU(1))+(U(2)*SU(2))+(U(3)*SU(3))+(U(4)*SU(4))+(U(5)*SU(5))+(U(6)*S
U(6))
        RN4=(U(7)*SU(7))+(U(8)*SU(8))+(U(9)*SU(9))+(U(10)*SU(10))+(U(11)*SU(11))+(U(
12)*SU(12))          RN5=(SX(1)+(PAE(1)*SX(1))-SU(1))*(DOTX(1)+(PAE(1)*X(1))-
U(1))                                RN6=(SX(2)+(PAE(2)*SX(2))-
SU(2))*(DOTX(2)+(PAE(2)*X(2))-U(2))
        RN7=(SX(3)+(PAE(3)*SX(3))-SU(3))*(DOTX(3)+(PAE(3)*X(3))-U(3))
                    RN8=(SX(4)+(PAE(4)*SX(4))-SU(4))*(DOTX(4)+(PAE(4)*X(4))-
U(4))
          RN9=(SX(5)+(PAE(5)*SX(5))-SU(5))*(DOTX(5)+(PAE(5)*X(5))-U(5))

         RN10=(SX(6)+(PAE(6)*SX(6))-SU(6))*(DOTX(6)+(PAE(6)*X(6))-U(6))

         RN11=(SX(7)+(PAE(7)*SX(7))-SU(7))*(DOTX(7)+(PAE(7)*X(7))-U(7))

         RN12=(SX(8)+(PAE(8)*SX(8))-SU(8))*(DOTX(8)+(PAE(8)*X(8))-U(8))
```

```
RN13=(SX(9)+(PAE(9)*SX(9))-SU(9))*(DOTX(9)+(PAE(9)*X(9))-U(9))

RN14=(SX(10)+(PAE(10)*SX(10))-SU(10))*(DOTX(10)+(PAE(10)*X(10))-U(10))

RN15=(SX(11)+(PAE(11)*SX(11))-SU(11))*(DOTX(11)+(PAE(11)*X(11))-U(11))

RN16=(SX(12)+(PAE(12)*SX(12))-SU(12))*(DOTX(12)+(PAE(12)*X(12))-U(12))

RD1=(SX(1)**2)+(SX(2)**2)+ (SX(3)**2)+ (SX(4)**2)+ (SX(5)**2)+ (SX(6)**2)

RD2=(SX(7)**2)+(SX(8)**2)+ (SX(9)**2)+ (SX(10)**2)+ (SX(11)**2)+
(SX(12)**2)
RD3=(SU(1)**2)+(SU(2)**2)+ (SU(3)**2)+ (SU(4)**2)+ (SU(5)**2)+ (SU(6)**2)

RD4=(SU(7)**2)+(SU(8)**2)+ (SU(9)**2)+ (SU(10)**2)+ (SU(11)**2)+
(SU(12)**2)                    RD5=(SX(1) +(PAE(1)*SX(1))-(SU (1)**2))
                                        RD6=(SX(2) +(PAE(2)*SX(2))-(SU
(2)**2))                                RD7=(SX(3)
+(PAE(3)*SX(3))-(SU (3)**2))
        RD8=(SX(4) +(PAE(4)*SX(4))-(SU (4)**2))
                RD9=(SX(5) +(PAE(5)*SX(5))-(SU (5)**2))
                        RD10=(SX(6) +(PAE(6)*SX(6))-(SU (6)**2))
                                RD11=(SX(7)  +(PAE(7)*SX(7))-
(SU (7)**2))                            RD12=(SX(8)
+(PAE(8)*SX(8))-(SU (8)**2))
        RD13=(SX(9) +(PAE(9)*SX(9))-(SU (9)**2))
                RD14=(SX(10) +(PAE(10)*SX(10))-(SU (10)**2))
                        RD15=(SX(11) +(PAE(11)*SX(11))-(SU (11)**2))
                                RD16=(SX(12) +(PAE(12)*SX(12))-(SU (12)**2))

 RNT=(2*(RD1+RD2+ RD3+ RD4)+2*AM*( RN5+ RN6+ RN7+ RN8+ RN9+
RN10+ RN11+ RN12+ RN13+ RN14+ RN15+ RN16))          RMT=(2*(RD1+RD2+
RD3+ RD4)+2*AM*( RD5+ RD6+ RD7+ RD8+ RD9+ RD10+ RD11+ RD12+ RD13+ RD14+
RD15+ RD16))  ROW=RNT/RMT
                        IF (ITERA.GT.50)GOTO 19
                                C  CS  MEANS  CONSTRAINT
SATISFACTION
C       CSAT MEANS THE TOTAL CONSTRAINT SATISFACTION
```

```
      CSAT=0.0
                  DO 6 I=1, 12
                        CS(I)=(I**I)*(3.142**2)*(X(I)-U(I))
                              CSAT=CSAT+(CS(I)**2)

6     CONTINUE
C     WE DECIDED TO LEAVE ALL THE WRITE STATEMENTS OUT SINCE ANY VARIABLE NEEDED CAN EASILY BE CALCULATED      DO 9 I=1, 12

      BITA=((UPGX(I)**2)+(UPGU(I)**2))/((GX(I)**2)+(GU(I)**2))
            UPX(I)=X(I)+ROW*SX(I)
                        UPU(I)=U(I)+ROW*SU(I)

9     CONTINUE
            SNURM1=0.0
                  SNURM2=0.0
                        DO 7 I=1, 12
                              SNURM1=SNURM1+(GX(I)**2)
                                    SNURM2=SNURM2+(GU(I)**2)

7     CONTINUE
            SNOM1=SQRT (SNURM1)
                  SNOM2=SQRT (SNURM2)
                              DO  10  I=1, 12
                                    X(I)=UPX(I)
                                          U(I)=UPU(I)

      SUMU=SUMU+(U(I)*U(I))
            STOTAL=SUMU+SUMX
                  OB=STOTAL

10    CONTINUE
            WRITE(12,112)OB,SNOMI,SNOM2

112   FORMAT(//40X,''OB='',F15.6/40X,''SNOM1='',F15.6/10X,''SNOM2='',F15.6)
                  IF(SNOM1.LE.(0.01).AND. SNOM2.LE.(0.01))GOTO 19 GOTO 20

60    STOP
            END
```