



ILJS-14-043

Convergence Rate Analysis of a Proposed Function Space Algorithm (FSA)

Omolehin¹, J. O., Rauf², K., Mabayoje³, M. A., Arowolo⁴, O. T. and Lukuman⁵, A.

¹Department of Mathematics, Federal University, Lokoja, Kogi State, Nigeria.

²Department of Mathematics, University of Ilorin, Ilorin, Nigeria.

³Department of Computer Science, University of Ilorin, Ilorin, Nigeria.

⁴Department of Mathematics, Lagos State University, Lagos, Nigeria.

⁵Federal College of Fisheries and Marine Technology, Lagos State, Nigeria.

Abstract

In this work, the numerical implementation of Function Space Algorithms (FSA) for the solution of quadratic continuous cost functional was considered. 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.

Keywords: Optimization, FSA, Penalty parameter, Constraints, Convergence rate, Parameter.

1. Introduction

Optimization can be described as a process or methodology of making a design, system, or decision as functional or effective and fully perfect as possible. In particular, it is the mathematical procedures (for finding the maximum of a function) involved in a particular problem as it was used in quadratic functional. See, [6 &7].

Consider the quadratic functional of the form:

$$F(x) = F_0 + \langle a, x \rangle_H + \frac{1}{2} \langle x, Ax \rangle_H,$$

where A is an $n \times n$ symmetric positive definite matrix operator on the Hilbert space H . a is a vector in H and F_0 is a constant term.

Let us also consider what is termed conjugate descent with F . With conjugate descent, it is assumed that a sequence $\{p_i\} = p_0, p_1, \dots, p_k, \dots$ is available with the members of the sequence conjugate with respect to the positive definite linear operator A .

*Corresponding Author: Rauf, K.

E-mail: krauf@unilorin.edu.ng

By conjugate with respect to A , it means that:

$$\langle p_i, Ap_i \rangle_H = \begin{cases} \neq 0, & \text{if } i \neq j \\ = 0, & \text{if } i = j \end{cases}$$

In this case, A is assumed positive definite so $\langle p_i, Ap_i \rangle_H > 0$. The conventional Conjugate Gradient Method (CGM) was originally designed for the minimization of a quadratic objective functional of the form stated above.

2. 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:

$$\text{Stage 2: } p_0 = -g_0 = -(a + Ax_0)$$

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

$$\text{Stage 3: } x_{i+1} = x_i + \alpha_i p_i, \quad \alpha_i = \langle g_i, g_i \rangle_H / \langle p_i, Ap_i \rangle_H$$

$$g_{i+1} = g_i + \alpha_i Ap_i;$$

α is the step length

$$p_{i+1} = -g_{i+1} + \beta_i p_i; \quad \beta_i = \langle g_{i+1}, g_{i+1} \rangle_H / \langle g_i, g_i \rangle_H$$

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 α_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)

$$\text{Minimise } \int_0^\delta \{x^T(t)Qx(t) + u^T(t)Pu(t)\}dt,$$

subject to the dynamic constraints

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

$0 < t < \delta$ (δ 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 $n \times 1$ state vector, $u(t)$ is the $q \times 1$ control vector, C and D are $n \times q$ 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

$$\begin{aligned} \langle A, AZ \rangle_K = J(x, u, \mu) = \int_0^\delta \{x^T(t)Qx(t) + u(t)^TPu(t) \\ + \mu \|\dot{x}(t) - Cx(t) - Du(t)\|^2\} dt \quad (\mu > 0) \end{aligned}$$

by transforming problem (1) into an unconstrained optimal control problem. Where μ is the penalty constant K is given by $K = H_1[0, \delta] \times L_2^q[0, \delta]$, and $H_1[0, \delta]$, denotes sobolev space of the absolutely continuous functions $x(\cdot)$, 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 $[0, \delta]$ into R^q , with norm denoted by $\|\cdot\|_E$ and defined by

$\|u\| = \int_0^\delta \{\|u\|^2\}^{\frac{1}{2}} dt$ and with scalar product conventionally denoted by $\langle \cdot, \cdot \rangle$ and defined by $\langle u_1, u_2 \rangle = \int_0^\delta \langle u_1, u_2 \rangle_E dt$ where $\|\cdot\|_E$ and $\langle \cdot, \cdot \rangle_E$ 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):

$$\begin{aligned} \text{Min } J(x, u, \mu) = & \text{Min} \int_0^T \{x^T(t) P x(t) + u^T(t) Q u(t)\} dt \\ & + \mu \int_0^T ||\dot{x}(t) - Cx(t) - Du(t)||^2 dt \end{aligned}$$

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

Step 1

choose the initial values $\dot{x}_0(t)$, $u_0(t)$,

where $0 < t < T$, T is known and compute, $x_0(t) = \int_0^T \dot{x}_0(t) dt$

Step 2

Initialize the Counter: $i = 0$, and compute $[\nabla_x^J]_i$, $[\nabla_u^J]_i$ using formulae for

$$\nabla J = \begin{bmatrix} \nabla_x^J \\ \nabla_u^J \end{bmatrix}$$

$$\begin{aligned} [\nabla_x^J] &= 2\mu (\dot{x}(t) - f(x(t), u(t), t)) - \int_T^t \left(\frac{\partial T}{\partial x}\right)^T - \left(\frac{\partial f}{\partial x}\right)^T [2\mu(\dot{x}(s) - f(x(s), u(s), s))] ds \\ 0 &\leq t \leq T \end{aligned}$$

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

Step 3

Compute the current descent direction.

$$\dot{s}_{x,i}(t) = \begin{cases} -[\nabla_x^J]_0, & \text{for } i=0 \\ -[\nabla_x^J]_i + \beta_{i-1} s_{x,i-1}(t), & \text{for } i>0, \text{ where } t \in [0, T] \end{cases}$$

$$s_{x,i}(t) = \int_0^t \dot{s}_{x,i}(t) dt$$

$$\dot{s}_{u,i}(t) = \begin{cases} -[\nabla_u^J]_0, & \text{for } i=0 \\ -[\nabla_u^J]_i + \beta_{i-1} s_{u,i-1}(t) & \text{for } i>0 \end{cases}$$

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

$$\beta_{i-1} = \frac{\int_0^T ||[\nabla_x^J]_i||^2 dt + \int_0^T ||[\nabla_u^J]_i||^2 dt}{\int_0^T ||[\nabla_x^J]_{i-1}||^2 dt + \int_0^T ||[\nabla_u^J]_{i-1}||^2 dt}$$

STEP 4

Find P_i^*

such that $J(x_i + P_i^* s_{x,i}, u_i + P_i^* s_{u,i}, \mu_i) < J(x_i + p_i s_{x,i}, u_i + p_i s_{u,i}, \mu_i), p \geq 0$

STEP 5

Test for the stopping criterion of the algorithm by verifying if

$$S(x, u, \mu) = \|\bar{\theta}(z)\|^2 + (\nabla_x^J)^T (\nabla_x^J) + (\nabla_u^J)^T (\nabla_u^J) \leq \varepsilon, (\bar{\theta} = \dot{x} - f)$$

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.

3. Results and Discussion**Numerical Results on FSA**

In carrying out this numerical investigation using FSA algorithm, the convergence rate of various diffusion equation problems which are only dimensionally different from one another were vividly studied. Thus, the dimensionality of the resulting diffusion control problem is $\mathbb{R}^{12}$ and $\mathbb{R}^{11}$ it means that is $\alpha, u \in \mathbb{R}^{12}$ and $\alpha, u \in \mathbb{R}^{11}$, respectively. Thus, in $\mathbb{R}^{12}$ and $\mathbb{R}^{11}$ will have problems (1) and (2) respectively as stated below:

PROBLEM (1)

$$\text{Minimize } \int_0^1 \{\alpha_1^2(t) + \alpha_2^2(t) + \dots + \alpha_{12}^2(t) + u_1^2(t) + u_2^2(t) + \dots + u_{12}^2(t)\} dt$$

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:

$$\begin{aligned} \text{Min } J(\alpha, u, \mu) = & \text{Min} \int_0^1 \{ \alpha_1^2(t) + \alpha_2^2(t) + \dots + \alpha_{12}^2(t) + u_1^2(t) + u_2^2(t) + \dots + \\ & u_{12}^2(t) \} dt \\ & + \mu \{ \int_0^1 \{ | \alpha_1(t) + \pi^2\alpha_1(t) - u_1(t) |^2 + | \alpha_2(t) + 4\pi^2\alpha_2(t) - u_2(t) |^2 + | \alpha_3(t) + \\ & 9\pi^2\alpha_3(t) - u_3(t) |^2 + \dots + | \alpha_{12}(t) + 144\pi^2\alpha_{12}(t) - u_{12}(t) |^2 \} \} dt \end{aligned}$$

Also, problem in $\mathbb{R}^{11}$ i.e. when $\alpha, u \in \mathbb{R}^{11}$, the following equivalent problem formulation was obtained:

PROBLEM (2)

$$\text{Minimize } \int_0^1 \{ \alpha_1^2(t) + \alpha_2^2(t) + \dots + \alpha_{11}^2(t) + u_1^2(t) + u_2^2(t) + \dots + u_{11}^2(t) \} dt$$

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

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

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

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

$$\begin{aligned} \text{Min } J(\alpha, u, \mu) = & \text{Min} \int_0^1 \{ \alpha_1^2(t) + \alpha_2^2(t) + \dots + \alpha_{11}^2(t) + u_1^2(t) + u_2^2(t) + \dots + u_{11}^2(t) \} dt \\ & + \mu \{ \int_0^1 \{ | \alpha_1(t) + \pi^2\alpha_1(t) - u_1(t) |^2 + | \alpha_2(t) + 4\pi^2\alpha_2(t) - u_2(t) |^2 + | \alpha_3(t) + 9\pi^2\alpha_3(t) - u_3(t) |^2 + \dots + | \alpha_{11}(t) + \\ & 121\pi^2\alpha_{11}(t) - u_{11}(t) |^2 \} \} dt. \end{aligned}$$

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): $\mu = 10$

Time (t)	Iteration Num	u_1	u_2	u_3	u_4	u_5		
u_6	u_7							
0.2	4	0.99899 1	.99229 0	0.9816 30	0.9677 77	0.95179 9	0.93507 1	0.9192 81
0.4	9	0.99736 1	0.9865 62	0.9691 75	0.9461 2	0.91868 3	0.88851 6	0.8576 44
0.6	9	0.99646 4	0.9835 78	0.9625 85	0.9342 12	0.89947 5	0.85968 0	0.8164 28
0.8	8	0.99607 3	0.9824 62	0.9600 30	0.9291 57	0.89037 9	0.84438 1	0.7920 0
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	0.99893 3	0.9922 32	0.9815 72	0.9677 19	0.95174 2	0.93501 7	0.9192 13
0.4	8	0.99731 5	0.9865 16	0.9691 29	0.9460 74	0.91863 6	0.88847 1	0.8575 98
0.6	9	0.99642 7	0.9835 41	0.9625 49	0.9341 75	0.89943 8	0.85964 4	0.8163 91
0.8	10	0.99604 4	0.9824 33	0.9600 01	0.9291 28	0.89035 0	0.84435 2	0.7919 71
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	0.9989 14	0.9922 12	0.98155 2	0.96769 9	0.95172 2	0.9349 90	0.9192 00
0.4	9	0.9972 99	0.9865 00	0.96911 3	0.94605 8	0.91862 0	0.8884 55	0.8575 82
0.6	10	0.9964 15	0.9835 29	0.96253 6	0.93416 3	0.89942 6	0.8596 32	0.8163 78
0.8	10	0.9960 34	0.9824 23	0.95999 1	0.92911 9	0.89035 0	0.8443 42	0.7919 61
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							
0.6	9	0.997287	0.986487	0.969100	0.946045	0.918608	0.888442	0.857570
0.8	9	0.996405	0.983519	0.962526	0.934153	0.899416	0.859622	0.816369
		0.996002	0.982415	0.959985	0.929111	0.890333	0.844334	0.791953
u ₈ Function		u ₉	u ₁₀	u ₁₁	u ₁₂	Objective		
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	0.9988 95	0.9921 92	0.98153 1	0.9676 76	0.9516 96	0.93496 9	0.91917 3
0.4	10							
0.6	9	0.9972 84	0.9864 84	0.96909 7	0.9460 42	0.9186 05	0.88843 9	0.85756 6
0.8	10	0.9964 03	0.9835 17	0.96252 4	0.9341 51	0.8994 13	0.85962 0	0.81636 6
		0.9902 4	0.9824 14	0.95998 1	0.9291 09	0.8903 31	0.84433 2	0.79195 1
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			

From obtained results, it is easily shown that the convergence rate of proposed 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 μ grows. Part of future research in this work is to determine the optimal value for μ .

4. Conclusion

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, see [2-5 & 8].

The advantage of this method is that, it can handle continuous quadratic functional which cannot be tackled 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*, **3**, 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, *Analele stiintifice ale Universitata "Al. I. Cuza"* **1**, 127-132.
- Omolehin, J. O., Abdullahi, I. and Rauf, K. (2013): New Computational Results on some Methods with Fortran in Optimization, *International Journal of Mathematics and Computer Science*, **8**(2), 37-53.
- Omolehin, J. O., Abdullahi, I., Rauf, K. and Dickmu, P. L. (2014): Fibonacci Scheme On Gradient Method For Some Control Problems, *International Journal of Mathematics and Computer Science*, **9**(1), 1-10.
- Russel, D. L. (1970): *Optimization theory*, W.A. Benjamin, Inc. New York.

APPENDIX

```

C      OPEN 12, 'JULU', AT='AP'
C      THIS PROGRAM 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), DOT SX(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(1)*SX(1))+(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)*SU(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,SNOM1,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

```