Computing Projections for the Karmarkar Algorithm

Kathryn Turner
Department of Mathematics
Utah State University
Logan, Utah 84322-3900

Submitted by Richard Tapia

ABSTRACT

Alternatives are considered for computing the projection of a vector onto the nullspace of a matrix, as is required to compute the step direction for the Karmarkar projective algorithm. Among the possibilities considered are forming and factoring the normal matrix, and working with a larger but sparser extended matrix. The extended matrices are symmetric but indefinite. A modification to the Harwell MA27 set of subroutines to make them more efficient for indefinite matrices is presented. Computational results are given to support the conclusion that computing these projections using one of the extended matrices offers significant computational advantages over the other alternatives with which it is compared.

The simplest form of the projective algorithm for linear programming, proposed by Karmarkar (1984), assumes that the standard linear programming problem

$$\text{minimize} \quad \bar{c}^T\bar{x} \tag{1}$$

$$\text{subject to} \quad \bar{A}\bar{x} = b,$$

$$l \leqslant \bar{x} \leqslant u$$

has been transformed so that the feasible region is the intersection of the

0024-3795/91/$3.50

simplex $\{x : x \geqslant 0,\ e^T x = n\}$, where $e = (1, \ldots, 1)^T$, with the vector space $\{x : Ax = 0\}$, so that the problem to be solved is

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} && Ax = 0, \\ & && e^T x = n, \\ & && x \geqslant 0. \end{aligned} \tag{2}$$

A means of transforming the problem (1) into the problem (2) in a manner that preserves sparsity is given in Dennis, Morshedi, and Turner (1987).

If the minimum value of the objective function in (1) is known, we may readily shift the objective function in (2) to have a minimum value of zero. The Karmarkar algorithm for solving the problem (2) with minimum objective value zero assumes we have an initial feasible point $x_0 > 0$, and can be stated as follows:

begin
  set $x = x_0$
  while $c^T x$ is too large
  do
    let $D = \text{diag}(x)$, $B = \begin{bmatrix} AD \\ e^T \end{bmatrix}$
    compute c_p as the projection of Dc onto the nullspace of B
    determine a steplength α and set $\tilde{x} = e - \alpha c_p$
    set $x = \dfrac{n}{e^T D\tilde{x}} D\tilde{x}$
  end do
end

The steplength parameter α is chosen at each step to ensure that iterates remain feasible, and that sufficient reduction in the potential function

$$f(x) = \sum_{i=1}^{n} \log \frac{c^T x}{x_i} \tag{3}$$

to ensure convergence of the algorithm in polynomial time is achieved at each step. In our implementation, the choice of the steplength parameter α at each step is made according to the three-faceted step acceptance criterion given in Dennis, Morshedi, and Turner (1987), which emphasizes reduction of the linear objective function given in (2) while enforcing Karmarkar's

(1984) sufficient reduction of the potential function (3), safeguarded by the Goldstein-Armijo "α-condition." [See Dennis and Schnabel (1983).]

A feasible starting point for the problem (2) may be obtained by solving the phase-one problem (for $x \in \mathbb{R}^{n+1}$)

$$\begin{aligned} &\text{minimize} && x_{n+1} \\ &\text{subject to} && \hat{A}x = 0, \\ & && e^T x = n+1, \\ & && x \geqslant 0, \end{aligned} \tag{4}$$

where $\hat{A} = [A \mid -Ax_0]$. A positive feasible starting point for this problem is

$$\frac{n+1}{1+\sum_{i=1}^{n} \hat{x}_i}(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n, 1)$$

for any strictly positive $\hat{x} \in \mathbb{R}^n$.

Todd and Burrell (1986) proposed a modification of the Karmarkar algorithm to solve problems with unknown optimal objective value. This modification uses estimates of the optimal objective value based on the dual of problem (2),

$$\begin{aligned} &\text{minimize} && nz \\ &\text{subject to} && A^T y + ze \leqslant c. \end{aligned} \tag{5}$$

They noted that for any $y \in \mathbb{R}^m$, (y, z) with

$$z = \min_j (c - A^T y)_j$$

is dual feasible with objective value nz. They proposed computing estimates of the dual variables at each step in such a way that the sequence $\{z_i\}$ is nondecreasing, and using the objective function $(c - ze)^T x$ in the computation of the step direction for the primal problem (2). We shall use the notation $P_A v$ to denote the projection of the vector v onto the nullspace of the matrix A:

$$P_A v = \left[I - A^T (AA^T)^{-1} A \right] v,$$

and

$$P_{e^T} v = \left(I - \frac{ee^T}{n} \right) v.$$

The projective algorithm incorporating Todd and Burrell's dual estimates can be stated as follows:

begin
  set $x = x_0$
  let $D = \text{diag}(x)$
  compute $y = [(AD)(AD)^T]^{-1}(AD)Dc$ and $z = \min_j (c - A^T y)_j$
  while $c^T x - nz$ is too large
  do
    (let $D = \text{diag}(x)$)
    compute $s = P_{AD} Dc$
    compute $p = P_{AD} De$
    if $\min_j (s - zp)_j > 0$
    then
      find a larger z such that $\min_j (s - zp)_j = 0$
      set $y = [(AD)(AD)^T]^{-1}(AD)D(c - ze)$
    end if
    compute the step direction $c_p = P_{e^T}(s - zp)$
    determine a steplength α and set $\tilde{x} = e - \alpha c_p$
    set $x = \dfrac{n}{e^T D\tilde{x}} D\tilde{x}$
  end do
end

It is clear that the computational effort in the Karmarkar algorithm is concentrated in the projection that is required to compute the step direction. As noted by Todd and Burrell (1986), since every iterate in the Karmarkar algorithm is feasible, we have $ADe = 0$ for $D = \text{diag}(x)$, and $P_B = P_{AD} P_{e^T} = P_{e^T} P_{AD}$. Hence, we may focus attention on the computation of $P_{AD} v$. Efficient computation of this projection is crucial in an efficient implementation of the Karmarkar projective algorithm.

We note that one may compute

$$P_{AD} v = \left\{ I - (AD)^T \left[(AD)(AD)^T \right]^{-1} (AD) \right\} v$$

by forming and factorizing the normal matrix $(AD)(AD)^T$ at each step. Among the disadvantages of this approach are

(i) the work involved in forming the normal matrix,
(ii) the tendency for the normal matrix to be ill conditioned, and
(iii) the tendency for the normal matrix to be denser than the matrix AD.

We may certainly avoid disadvantages (i) and (iii), and hope to reduce the severity of ill-conditioning, by working instead with the matrix

$$\begin{pmatrix} I & (AD)^T \\ AD & 0 \end{pmatrix}.$$

We note that we may obtain

$$s = P_{AD}(Dc) = \left\{ I - (AD)^T \left[(AD)(AD)^T \right]^{-1} (AD) \right\} Dc$$

directly by solving

$$\begin{pmatrix} I & (AD)^T \\ AD & 0 \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} Dc \\ 0 \end{pmatrix}.$$

An alternative is to restrict pivoting and solve

$$\begin{pmatrix} I & (AD)^T \\ AD & 0 \end{pmatrix} \begin{pmatrix} ? \\ t \end{pmatrix} = \begin{pmatrix} 0 \\ -ADDc \end{pmatrix},$$

where ? denotes a part of the solution that will not be computed. That is, if the factorization is carried out without pivoting, our problem is to solve the pair of triangular systems

$$\begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} 0 \\ v \end{pmatrix} = \begin{pmatrix} 0 \\ -ADDc \end{pmatrix}$$

and

$$\begin{pmatrix} U_{11} & U_{12} \\ & U_{22} \end{pmatrix} \begin{pmatrix} ? \\ t \end{pmatrix} = \begin{pmatrix} 0 \\ v \end{pmatrix},$$

which requires only the solution of the two small systems $L_{22}v = -ADDc$ and $U_{22}t = v$, since we are not interested in the part of the solution represented by ?. One may then compute s as $Dc - (AD)^T t$. However, carrying out the factorization without pivoting necessitates a factorization of $-(AD)(AD)^T$, which will arise in the lower right-hand corner. Since we have introduced the extended matrix in order to avoid factoring this denser matrix, we do not consider this alternative further. Factorization of the extended matrix

$$\begin{pmatrix} I & (AD)^T \\ AD & 0 \end{pmatrix}$$

in our experiments was carried out without this structural restriction on pivoting, in an attempt to preserve as much sparsity as possible, within the constraint of achieving a stable factorization.

A second extended system option that we considered is solution of the linear system

$$\begin{pmatrix} D^{-2} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} r \\ t \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix},$$

and then setting $s = D^{-1}r$. This has the added advantage that only n elements of the matrix change from one iteration to the next, so that the cost of adjusting the numerical values of the matrix at each iteration is reduced.

The two extended matrices are $(m+n)\times(m+n)$, but are usually sparser than the normal matrix, and thus it was hoped that the time spent in factorization and in subsequent solution of the linear system could be significantly reduced. We found this to generally be the case in our numerical experiments.

Our implementation employs the MA27 subroutines from Harwell [Duff and Reid (1982)] for factorization of the symmetric matrices and solution of the linear systems. This collection of subroutines divides the work into three phases: (1) an analysis phase, in which a tentative pivot sequence for the factorization, based only on the sparsity pattern of the matrix, is identified; (2) a numerical factorization phase, in which the actual factorization is carried out; and (3) the solution phase, in which the solution to the linear system is determined from the factorization obtained in step (2). For either the phase-1 problem to obtain an initial feasible point or the phase-2 problem to find the optimal solution, with or without Todd and Burrell's dual estimates, the Karmarkar projective algorithm requires that the analysis phase be executed only once, since the sparsity pattern of the matrix does not

change from one iteration to the next. Using the extended matrix with the identity in the upper left, for example, we require a single numerical factorization and two solves at each iteration:

$$\begin{pmatrix} I & (AD)^T \\ AD & 0 \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} Dc \\ 0 \end{pmatrix},$$

and

$$\begin{pmatrix} I & (AD)^T \\ AD & 0 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} De \\ 0 \end{pmatrix}.$$

The step direction for the transformed variable $\tilde{x}$ is given by

$$c_p = P_{e^T}(s - zp),$$

and when updates to the dual vector y are required, they are readily available from the work already done:

$$y = t - zq.$$

Numerical experiments reported in this paper are on problems with known optimal values, and dual estimates were not used.

In the course of numerical factorization, the tentative pivot sequence that was determined during the analysis step is modified as necessary to maintain stability of the factorization. During numerical factorization, 2×2 pivots are sometimes employed, although they are not identified in the analysis phase. Extensive alteration of the tentative pivot sequence during numerical factorization usually results in a significant loss of sparsity, and hence slower execution. In our numerical experiments, we found that the saving in time to update the extended matrix with D^{-2} in the upper left (as compared with the extended matrix with the identity in the upper left) was more than offset by significant loss of sparsity during numerical factorization.

In the currently released version of the MA27 subroutines, the analysis phase selects a tentative pivot sequence with the implicit assumption that all diagonal entries in the matrix are nonzero, and hence every variable is admissible from the outset as a singleton pivot. This is certainly not the case for our extended matrices. Hence, the tentative pivot sequence determined under this erroneous assumption during the analysis phase would be almost certain to require modification during numerical factorization. We therefore

modified the analysis phase of MA27 to deal with indefinite matrices having some zeros on the diagonal. The modified version of MA27 used in the numerical experiments reported in this paper used the minimum-degree pivot selection criterion, modified to take the possibility of zeros on the diagonal of the matrix into account.

MA27 is based on a multifrontal scheme, described fully by Duff and Reid (1983). Briefly, the matrix A is regarded as having the form

$$A = \sum_l B^{(l)},$$

where each matrix $B^{(l)}$ is zero except in a small number of rows and columns. At each step, a dense matrix of workspace, called the frontal matrix and consisting of the merged collection of those $B^{(l)}$ having nonzeros in the pivot row or column, is employed. Elimination is carried out in this full matrix. The minimum-degree criterion for selection of a pivot sequence consists of identifying each potential pivot with the row for which it is the diagonal element, initially assigning as the degree of each variable the number of nonzeros in its row, and choosing pivots in turn to minimize the degree of the selected pivot in the current reduced problem. At each step, the degrees of the variables appearing in the frontal matrix change because of the eliminations that are performed and the fill-in that occurs. During numerical factorization, stability of the factorization is maintained by requiring that singleton pivots satisfy

$$|a_{kk}| > u \max_{j \neq k} |a_{kj}|,$$

where u is a user-specified parameter in the range [0, 0.5]. If this condition is not satisfied and $u > 0$, MA27 attempts to use a 2×2 pivot that satisfies the stability condition

$$\left\| \begin{pmatrix} a_{kk} & a_{k,k+1} \\ a_{k+1,k} & a_{k+1,k+1} \end{pmatrix}^{-1} \right\|_\infty \max_{j \neq k,k+1} \left[\max\left(|a_{kj}|, |a_{k+1,j}| \right) \right] \leqslant u^{-1}.$$

For matrices with some zero diagonal elements, we extended the minimum-degree criterion to take into account the possible later use of 2×2 pivots. First, we identified zeros on the diagonal of the matrix with defective variables, that is, variables which are not eligible to serve as singleton pivots. The degree assigned to each nondefective variable was the number of nonzeros in its row, just as was done in the library version of MA27. For each defective variable, an attempt was made to find a defective partner such that the use of the pair as a 2×2 pivot could be considered within the minimum-

degree framework. We considered only the use of 2×2 pivots of the form

$$\begin{pmatrix} 0 & x \\ x & 0 \end{pmatrix},$$

pairing defective variable i with defective variable j for some nonzero matrix element $a_{ij} = a_{ji}$. The degree assigned to such a 2×2 pivot was the larger of the number of nonzeros in row i or row j, plus 1. Among the candidate partners for a defective variable, we chose the one that gave the smallest degree for the resulting 2×2 pivot. We did not consider the use of 2×2 pivots of the form

$$\begin{pmatrix} 0 & x \\ x & y \end{pmatrix},$$

where y is nonzero, since the amount of fill-in resulting from the use of such a pivot would be at least as much as would occur by using the nondefective variable as a singleton pivot. In the event that no partner could be found for a defective variable, we initially assigned its degree as n, the size of the matrix, so that the variable would not be chosen as a pivot until after fill-in had occurred on the diagonal, at which time the variable would no longer be defective and would have its degree recomputed as in the unmodified code.

Having thus assigned an initial degree to each variable, we proceeded with determination of a tentative pivot sequence, selecting pivot candidates based on the minimum-degree criterion. Modification of the strategy was necessary only in dealing with the defective variables, which was carried out as follows: When a defective variable is a pivot candidate,

```
if its partner is still defective
then
    select this variable as the current pivot
    force its partner to be placed next in the pivot sequence
else
    determine a new partner
    if the degree of the new 2×2 pivot = the current minimum degree
    then
        select this variable as the current pivot
        force its (new) partner to be placed next in the pivot sequence
    else
        reject the pivot candidate
    end if
end if
```

This modification was made to deal with general matrices having some zeros on the diagonal. Note that for the extended matrices of interest here, the defective variables are associated with the diagonal entries of the zero block, and none of them could have a partner. This is because no two defective variables can be paired, there being no case in which the coefficient of a defective variable in the row of another defective variable is nonzero. Thus the effect of the modified analysis phase on these matrices was simply to defer selection of variables that began as defective until fill-in had occurred in the corresponding diagonal position.

Our technique for obtaining an initial feasible point by solving a phase-1 problem introduces a dense column into the constraint matrix for the phase-1 problem. Since a dense column results in a dense normal matrix, fairness to the normal-matrix approach required that we split off the dense column in phase 1 and use the correction provided by the Sherman-Morrison-Woodbury formula. The normal matrix that was factored thus had the same number of nonzeros in phase 1 as in phase 2. We did not, however, make any attempt to treat other dense columns separately. The most dramatic difference in sparsity occurs in problems for which dense columns appear in the original problem formulation. We also experimented with splitting off the dense column in phase 1 for the extended matrices, but it made very little difference, since a dense column in the extended matrix is handled nicely by the minimum-degree pivot selection strategy. We concluded that the additional work associated with coding special treatment of dense columns for the extended matrices was not worthwhile. The numerical results reported here were obtained with no special treatment of dense columns in the extended matrices. Table 1 gives the problem statistics for twelve test problems in the Netlib test set which we obtained from David Gay. The number of nonzeros in the matrices is given for the phase-2 problem. To obtain the number of nonzeros in the phase-1 problem in the matrix A, add to the given number of phase-2 nonzeros the number of constraints, m, and in the extended matrices, add $m+1$ to the number of phase-2 nonzeros. Note in Table 1 that the extended matrices have fewer nonzeros than the normal matrix in all but the smallest test problem. The last column in Table 1 gives the number of correct digits obtained in the final objective value. These vary among problems because an absolute, rather than a relative, stopping criterion was used. The problems were solved to the same accuracy using each strategy for computing step directions.

Table 2 shows total execution times in seconds on an IBM 3084. The normal-matrix approach failed on four of the problems, probably due to ill-conditioning of the normal matrix. Our experimental code did not provide condition estimates, but did attempt to control the size of the residuals by iterative refinement. These four failures occurred when the iterative refine-

TABLE 1
PROBLEM STATISTICS

Problem	m	n	Non-zeros, phase 2			Iterations (ph. 1, ph. 2)	Obj. digits
			A	Normal	Extend.		
Afiro	28	53	162	138	215	7,9	4
ADLittle	57	140	601	1045	741	6,18–19	6–7
Share 2b	97	164	965	1240	1129	10,11	4
Share 1b	118	255	1537	5709	1792	9,22	7
Beaconfd	174	297	3772	5227	4069	16,21	7
Israel	175	318	2932	14998	3250	9,25–27	7
Brand Y	194	305	2561	4289	2866	16,22	5
E 226	224	474	3341	7366	3815	19,17	3
Bandm	306	474	3086	10725	3560	18,24	4
Fffff800	525	1030	7635	29013	8665	21,26	6–7
Shell	654	1646	5314	73111	6960	11,35	9
25 fv 47	821	1878	12870	52996	14748	14,37	5

ment was not successful. The normal-matrix solution was fastest on two of the smaller problems, Afiro and Share 2b, by 0.1 second or less. The extended matrix with D^{-2} in the upper left was fastest only for the problem ADLittle, and its margin of victory was slim. Use of the extended matrix with the identity in the upper left gave faster execution in 9 of the 12 problems, with the most dramatic differences occurring in the larger problems.

TABLE 2
EXECUTION TIMES IN SECONDS, IBM 3084

Problem	Normal matrix	D^{-2} extended		I extended	
		Unmodified	Modified	Unmodified	Modified
Afiro	0.24	0.30	0.31	0.29	0.30
ADLittle	1.66	1.35	1.44	1.40	1.45
Share 2b	1.68	1.80	1.86	1.78	1.80
Share 1b	11.73	4.66	4.22	4.50	4.16
Beaconfd	24.19	25.01	26.52	16.79	18.31
Israel	44.03	10.24	10.31	9.96	10.01
Brand Y	*	19.36	18.50	16.46	14.39
E 226	21.19	42.25	47.31	15.42	15.68
Bandm	41.87	31.51	31.28	17.54	17.46
Fffff800	*	121.15	100.22	80.90	67.95
Shell	*	53.66	54.87	46.19	49.07
25 fv 47	*	701.41	837.72	214.21	184.24

TABLE 3
AVERAGE NONZEROS IN FACTORS, PHASE 2

Problem	Normal matrix	D^{-2} extended		I extended	
		Unmodified	Modified	Unmodified	Modified
Afiro	166	356	354	344	344
ADLittle	1163	1197	1197	1193	1193
Share 2b	1271	2208	2243	2177	2206
Share 1b	5711	3856	3250	3715	3189
Beaconfd	6350	10461	10666	8573	8457
Israel	14998	5902	5902	5721	5721
Brand Y	6358*	9241	8729	7987	7527
E 226	8298	18521	19084	9370	9219
Bandm	14266	14830	13963	9645	9051
Fffff800	41722*	43718	37904	29998	25796
Shell	76649*	14545	14002	12910	12367
25 fv 47	69224*	138818	129021	63358	55777

We focus on the issue of sparsity in Table 3. Some difference in execution time is also attributable to the work involved in forming the normal matrix. We used the Harwell subroutine MC35, without modification, for this purpose. Savings might have been realized in the time required to form the normal matrix in the second and subsequent iterations by saving and using knowledge of the locations of the nonzero entries that would arise, but we did not attempt to do so.

We feel that the largest contributing factor to the differences in execution times between the two different extended matrices is the superior preservation of sparsity exhibited when the diagonal elements of the factored matrix are not changed. Table 3 shows a comparison of the average number of nonzeros in the factors, taken over all iterations in phase 2, for each problem, using the following five options: factor the normal matrix, or factor the extended matrix with D^{-2} or I in the upper left block, using the unmodified analysis phase of MA27 or the modified analysis phase. The modified analysis phase was not used in factoring the normal matrix, since this matrix does not have zeros on its diagonal. When the extended matrix with D^{-2} in the upper left was used, the modified as compared to the unmodified MA27 analysis phase resulted in sparser factors for seven of the problems and the same average number of nonzeros in the factors in three of the problems. Using the extended matrix with I in the upper left, the modified analysis phase produced sparser factors for eight problems and equally sparse factors for three problems, compared to the unmodified code. For all of these problems, sparser factors were obtained with the identity in the upper left of the

extended matrix than with D^{-2} in that position. The tentative pivot sequences determined during the analysis phase are, of course, identical for the two extended matrices. The differences in the average number of nonzeros in the factors over all iterations arise because of changes to the pivot sequence that are required to maintain stability in the numerical factorizations.

These numerical experiments indicate that computing projections using an extended matrix, rather than forming and factoring the normal matrix, offers significant advantages. The matrices are better conditioned, and the larger but sparser matrices result in faster overall execution times when both approaches succeed. Comparing the two extended matrices, the objective of preserving sparsity is much better met if the diagonal elements of the extended matrix are not changed. Overall execution times, as shown in Table 2, indicate a significant computational advantage for factoring the extended matrix with the identity in the upper left block. The minor modification to the analysis phase of MA27 did not produce a significant improvement in overall execution times for these problems. While some gains were made in the sparsity of the factors using the strategy outlined here, further improvements in this area are possible. For further discussion of alternative ways of choosing pivot sequences, see Duff, Gould, Reid, Scott, and Turner (1989).

Helpful discussions with Nick Gould, Iain Duff, and John Reid are gratefully acknowledged, as is the support of AERE Harwell.

REFERENCES

Dennis, J. E., Jr., Morshedi, A. M., and Turner, Kathryn. 1987. A variable-metric variant of the Karmarkar algorithm for linear programming, *Math. Programming* 1:1–20.

Dennis, J. E., Jr., and Schnabel, Robert B., 1983. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, N.J.

Duff, I. S. and Reid, J. K. 1982. MA27—A Set of Fortran Subroutines for Solving Sparse Symmetric Sets of Linear Equations, Report R-10533, Computer Science and Systems Division, AERE Harwell, Oxford OX11 ORA, England.

Duff, I. S. and Reid, J. K. 1983. The multifrontal solution of indefinite sparse symmetric linear equations, *AMS Trans. Math. Software* 9(3):302–325. (Sept.)

Duff, I. S., Gould, N. I. M., Reid, J. K., Scott, J. A., and Turner, K. 1989. The Factorization of Sparse Symmetric Indefinite Matrices, CSS Report 236, Computer Science and Systems Division, AERE Harwell, Oxford OX11 ORA, England; *IMA. J. Numer. Anal.*, to appear.

Gay, David M. 1985. Electronic mail distribution of linear programming test problems, *Committee on Algorithms Newsletter, Math. Programming Soc.* 13, Dec.

Karmarkar, Narendra. 1984. A new polynomial-time algorithm for linear programming, *Combinatorica* 4:373–395.

Todd, Michael J. and Burrell, Bruce P. 1986. An extension of Karmarkar's algorithm for linear programming using dual variables, *Algorithmica* 1:409–424.

Received 16 April 1990; final manuscript accepted 22 October 1990