Modified Krylov acceleration for parallel environments ⋆

Caroline Le Calvez

Université de Lille I, LIFL, Bâtiment M3, 59655 Villeneuve d'Ascq Cedex, France

Yousef Saad

University of Minnesota, Deparment of Computer Science and Engineering, 200 Union Street, SE, Minneapolis, MN 55455-0159, USA

Abstract

This paper considers a few variants of Krylov subspace techniques for solving linear systems on parallel computers. The goal of these variants is to avoid global dot-products which hamper parallelism in this class of methods. They are based on replacing the standard Euclidean inner product with a discrete inner product over polynomials. The set of knots for the discrete inner product is obtained by estimating eigenvalues of the coefficient matrix.

1 Introduction

The storage and computational requirements of the Generalized Minimum Residual (GMRES) algorithm [6] in its original form can be prohibitive when a large number of steps are required to achieve convergence. A simple remedy proposed in [6] consists of restarting the algorithm every m steps, using each time a Krylov subspace of dimension m and taking the initial guess to be the latest approximate solution obtained. The convergence behavior of this restarted method, referred to as GMRES(m), is not as well understood as the non-restarted GMRES but it represents a good trade-off and can be quite effective when combined with a good preconditioner.

⋆ Expanded version of a presentation given at the 1997 IMACS World Congress (Berlin, August 1997). This work was partially supported by NSF grant CCR-9618827, DARPA grant number NIST 60NANB2D1272, and by the Minnesota Supercomputer Institute

Another difficulty appears when implementing GMRES in a parallel environment: The modified Gram-Schmidt algorithm of the Arnoldi process requires the computation of inner products which must be performed sequentially and this can be time-consuming. To address this difficulty, we propose a strategy in which the Euclidean inner product is replaced by a discrete inner product over the space of polynomials. In this approach, the Krylov subspace is actually viewed as a space of polynomials and the linear least-squares problem in GMRES translates into a least-squares problem over polynomial spaces. The idea is to replace the standard inner product in this least-squares problem by a discrete inner product over points selected from some approximation to the spectrum.

In the computational procedure, it is desirable to use a basis of orthogonal polynomials. These will satisfy a three-term recurrence when the discrete set of knots is real. Otherwise a Hessenberg recurrence similar to that of the Arnoldi process is used. These polynomials are computed only in the form of the values which they take in the knots used for the discrete inner products. As is done for QMR (see [3]), we then consider this new basis to be orthonormal and perform a quasi-minimization of the resulting least-squares problem which yields a quasi-minimal residual solution. The set of knots used in the discrete inner product, which are approximate eigenvalues of A, are then updated and the process is repeated.

We first cover the case when the spectrum is real and then extend the techniques to cases when the spectrum is complex.

2 Case of a real spectrum

The main cost in GMRES(m), apart from the matrix-vector product and preconditioning operations, is in the Arnoldi process which constructs an orthogonal basis of the Krylov subspace. Since the vectors to be orthogonalized are not known in advance, the Gram-Schmidt process must be performed one step at a time, meaning that each new vector in the Arnoldi sequence must depends on the previous vector of the sequence. A new vector is introduced to the subspace and then it is orthogonalized against all previous basis vectors, using either a Classical (more parallel) or a Modified Gram-Schmidt framework. The main idea proposed here is to replace these inner products of size n, by localized discrete inner products over polynomials.

We begin by recalling the intimate link between Krylov subspaces and polynomial spaces. A Krylov subspace is a subspace of $\mathbb{R}^n$ of the form,

$$\mathrm{K}_m = \mathrm{Span}\{v, Av, \cdots, A^{m-1}v\} \tag{1}$$

where v is an initial vector generally equal to the initial residual r_0. Any member w of K_m is of the form $w = q(A)v$ where q belongs to the space $\mathbb{P}_m$ of polynomials of degree $\leq m-1$. As is well-known the usual Euclidean inner product on K_m gives rise to an inner product between two polynomials q_1, q_2 on $\mathbb{P}_m$ via the mapping,

$$\langle q_1, q_2 \rangle = (q_1(A)v, q_2(A)v)$$

and this has been exploited in a number of earlier studies. Conversely, it is also possible to define an inner product on K_m from an arbitrary inner product on $\mathbb{P}_m$. The defining relation is the equation,

$$(w_1, w_2) = \langle q_1, q_2 \rangle \tag{2}$$

in which $q_i, i = 1, 2$ is the (unique) polynomial which represents w_i by the relation $w_i = q_i(A)v$. One way to define the algorithms presented in this paper is simply this: *run a version of a Krylov subspace algorithm (CG, BiCG, FOM, GMRES, ..) in which the Euclidean inner product is replaced by an inner product of the form (2) where $\langle ., . \rangle$ is some discrete inner product over polynomial spaces.* It is important to note that we have no restriction whatsoever in the choice of the polynomial inner product. The Arnoldi basis thus constructed will no longer be orthogonal with respect to the usual Euclidean inner product but it will be orthogonal relative to another, less natural, inner product.

2.1 Discrete inner products over polynomial spaces

We introduce the following discrete inner product on the space $\mathbb{P}_m$ of polynomials of degree $\leq m-1$,

$$\begin{aligned} \langle \cdot, \cdot \rangle : \mathbb{P}_m \times \mathbb{P}_m &\rightarrow \mathbb{R} \\ \langle p, q \rangle \quad &= \sum_{\theta_k \in \mathrm{S}} \eta_k p(\theta_k) q(\theta_k) \end{aligned} \tag{3}$$

where the η_k's are strictly positive weights and the set of knots S is a set of approximate eigenvalues of A, with $|\mathrm{S}| \geq m$. This does define a nondegenerate inner product if all η_k's are different and when $|\mathrm{S}| \geq m$. The experiments suggest that $|\mathrm{S}| = m$ often works well.

The first step in the proposed procedure is to compute a basis $\{w_i\}_{i=1,\ldots,m}$ of the Krylov subspace $\mathrm{K}_m = \mathrm{K}_m(A, r_0)$ where r_0 is the initial residual. We denote by W_m the matrix $W_m = [w_1, \ldots, w_m]$. Each basis vector w_j is taken

in the form $w_j = p_{j-1}(A)r_0$ for $1 \leq j \leq m$, where $p_0, \ldots, p_{m-1}$ is a sequence of orthogonal polynomials with respect to the inner product (3). Since the knots and weights are real, it is well-known that these polynomials satisfy a 3-term recurrence of the form

$$\beta_{i+1}p_i(t) = tp_{i-1}(t) - \alpha_i p_{i-1}(t) - \beta_i p_{i-2}(t), \qquad i = 1, \ldots, m. \tag{4}$$

with the convention $p_{-1} \equiv 0$, $\beta_0 \equiv 0$ and $p_0 = 1/\alpha$ such that $\|p_0\| = 1$. This results in a similar relation for the sequence of w_i's.

$$\begin{aligned} w_1 &= p_0(A)r_0 \\ \beta_{i+1}w_{i+1} &= Aw_i - \alpha_i w_i - \beta_i w_{i-1}, \quad i = 1, \ldots, m \end{aligned} \tag{5}$$

with the convention that $w_0 \equiv 0$.

When the matrix is normal, the discrete inner product $\langle p_i, p_j \rangle$ is a good approximation of (w_{i+1}, w_{j+1}), for specific weights η_k. Note that we address here the general case of a complex spectrum since when the spectrum is real and A is normal then A is Hermitian. When A is normal, it admits an orthonormal basis of eigenvectors $u_i, i = 1, \ldots, n$. Expanding r_0 in this basis as $r_0 = \sum_{k=1}^{n} \xi_k u_k$ we obtain

$$p_i(A)r_0 = \sum_{k=1}^{n} p_i(\lambda_k)\xi_k u_k$$

and it follows that

$$(w_i, w_j) = \sum_{k=1}^{n} |\xi_k|^2 \; p_i(\lambda_k)\bar{p}_j(\lambda_k)$$

This inner product is clearly identical with the discrete inner product (3) when S is equal to the spectrum of A and $\eta_k = |\xi_k|^2$.

Assume that S has l distinct elements θ_i, $i = 1, \ldots, l$. Then it is well-known that the roots of the l-th orthogonal polynomial p_l are the knots θ_i, $i = 1, \ldots, l$.

For $i > l$, the orthogonal polynomials are underdetermined. Specifically, any polynomial of the form

$$p_{k+l}(t) = q_k(t)p_l(t) \;\forall\; k \geq 0 \tag{6}$$

where q_k is an arbitrary polynomial of degree k, will satisfy the orthogonality condition.

Note that when a knot θ_i is represented several times in S then clearly we can replace all these points by a single one whose weight is the sum of the weights of all the different instances of θ_i. Since we do not want $\|p_k\|$ to be equal to zero for $k = 0, \ldots, m$, we have to take $|\mathrm{S}| \geq m+1$ with S having at least $m+1$ distinct θ_k. Nevertheless, taking $|\mathrm{S}| = m$ works well in practice, since we do not explicitly use $w_{m+1} = p_m(A)r_0$.

2.2 Computation of the basis W_m

Before computing the basis W_m it is necessary to start by obtaining the coefficients α_i, β_i of the three-term recurrence with the standard Stieljes procedure using the discrete inner product, see [4]. These coefficients can be represented as a tridiagonal matrix $\bar{T}_m$ of size $(m+1) \times m$.

$$\bar{T}_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta 2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \\ & & & & \beta_{m+1} \end{pmatrix}$$

We then compute the basis W_m with the 3-term recurrence (5). The following relation is satisfied

$$AW_m = W_{m+1}\bar{T}_m \tag{7}$$

The vectors w_i, $i = 1, \ldots, m+1$ are not normalized for the 2-norm. Introducing the diagonal matrix $D_m = \mathrm{Diag}(\|w_1\|, \ldots, \|w_m\|)$ we can form the basis $\tilde{W}_m = W_m D_m^{-1}$ whose column-vectors are of norm unity. We next form the new iterate $\tilde{x}$ resulting from a *quasi-minimization* step on the space spanned by $\tilde{W}_m$. This is obtained as follows. Write the approximation in the form

$$\tilde{x} = x + \tilde{W}_m \tilde{y} = x + W_m D_m^{-1} \tilde{y} \tag{8}$$

and express the corresponding residual as

$$\begin{aligned} \tilde{r} &= r - AW_m D_m^{-1}\tilde{y} \\ &= r - W_{m+1}\bar{T}_m D_m^{-1}\tilde{y} \\ &= r - \tilde{W}_{m+1} \underbrace{D_{m+1}\bar{T}_m D_m^{-1}}_{\tilde{S}_m} \tilde{y} \end{aligned} \tag{9}$$

with $r = \alpha\ w_1 = \alpha\|w_1\|\ \tilde{w}_1$, and α such that $p_0(t) = 1/\alpha$. The Galerkin condition $\tilde{r} \perp A\mathrm{K}_k(A, r_0)$ then leads to finding $\tilde{y}$ such that:

$$\begin{aligned} \tilde{y} &= \arg\min_y \|r - A\tilde{W}_m y\| \\ &= \arg\min_y \|\tilde{W}_{m+1}(\alpha\|w_1\|\ e_1 - \tilde{S}_m y)\| \end{aligned} \tag{10}$$

A similar minimization problem arises in the standard GMRES algorithm. However, in GMRES the analogue of the matrix $\tilde{W}_{m+1}$ is orthonormal and this leads to an inexpensive method for minimizing the residual norm over K_m. In the present situation, $\tilde{W}_{m+1}$ is not orthonormal so an exact minimizer of the residual norm may be difficult to obtain. However, we can find a quasi-minimum solution which consists of ignoring the non-orthogonality of the system W_{m+1} as is done in the QMR algorithm [3]. This corresponds to finding $\tilde{y}$ which satisfies:

$$\tilde{y} = \arg\min_y \|\ \alpha\ \|w_1\|\ \ e_1 - \tilde{S}_m y\| \tag{11}$$

It can be easily seen that the solution which would be obtained by a projection process onto the unscaled basis $\tilde{W}_m$, is identical with the one defined above. However, this is only true in exact arithmetic. In the presence of round-off the use of $\tilde{W}_m$ is not as viable numerically.

2.3 Update of the θ_k's and the η_k's

Once the new approximate solution is found, the next step is to update the set S of knots θ_k and the weights η_k in order to restart the process. Initially the θ_k's are taken to be random numbers in the range $[\tilde{\lambda}_{min}, \tilde{\lambda}_{max}]$, where $\tilde{\lambda}_{min}$ and $\tilde{\lambda}_{max}$ are approximations to the smallest and largest eigenvalues of A provided from Gershgörin's theorm. In subsequent steps, the θ_k's are updated from the eigenvalues of the following Generalized Eigenvalue Problem:

$$\tilde{W}_m^T A\tilde{W}_m\ u = \ \lambda\ \tilde{W}_m^T\tilde{W}_m\ u \tag{12}$$

This is obtained from a projection method onto the Krylov subspace [1].

This operation is expensive because it requires forming the matrix $\tilde{W}_m^T\tilde{W}_m$ which requires $m^2/2$ inner products. Once this matrix is available, then the matrix $\tilde{W}_m^T A\tilde{W}_m$ is easily computed from

$$\tilde{W}_m^T A\tilde{W}_m = \tilde{W}_m^T\tilde{W}_{m+1}\tilde{S}_m \tag{13}$$

The problem (12) gives m approximate eigenvalues of A of which m' are distinct. We can update m' values of S or add some of them to S. The size of S should not be too big since it is desirable that the discrete inner product (3) be inexpensive to evaluate. Nevertheless, the tests we made for the different choices of S showed that the convergence rate is not improved by taking a larger S, and that in fact the process can even be slowed.

We need to compute the $m \times (m+1)$ matrix $\tilde{W}_m^T \tilde{W}_{m+1}$ which is equal to the left hand side matrix of (12). Since the square part of size $m \times m$ of $\tilde{W}_m^T \tilde{W}_{m+1}$ is symmetric and has one on the main diagonal, we need only compute its strict upper part, which requires $(m(m+1)/2)$ inner products of size n each. This is about the same number of operations than for computing the inner products in the Arnoldi process of GMRES(m). The main difference is that the vectors to be orthogonalized are now available in advance and therefore all the relevant inner product can be obtained at the same time, exploiting parallelism.

For the computation of η_k, Section 2.1 suggests that in the normal case, a good choice would be $\eta_i = (r_{k+1}, z_i)$, where z_i is an exact eigenvector. Since exact eigenvectors are not available we can use their approximations obtained from the subspace, i.e., the Ritz vectors $W_m u_i$ of (12). This gives $\eta_i = (r_{k+1}, W_m u_i)^2$ assuming that the Ritz vectors $W_m u_i$ are normalized. Then

$$(r_{k+1}, \tilde{W}_m u_i) = (r_k - A\tilde{W}_m \tilde{y}, \tilde{W}_m u_i) \tag{14}$$
$$= (\ \alpha \|w_1\|\ e_1 - \tilde{S}_m \tilde{y}, \tilde{W}_{m+1}^T \tilde{W}_m u_i) \tag{15}$$

Let Q_m be the product of Givens rotation matrices which transform $\alpha\|w_1\|\ e_1$ into $\bar{g}_m = (\gamma_1, \ldots, \gamma_{m+1})^T$ and $\bar{T}_m$ into an $(m+1) \times m$ upper triangular matrix $\bar{R}_m$. Then, $\tilde{y} = R_m^{-1} g_m$ where R_m is the $m \times m$ part of $\bar{R}_m$ and g_m is the vector consisting of the first m components of $\bar{g}_m$. Then we have:

$$(r_{k+1}, \tilde{W}_m u_i) = (Q_m\ (\ \alpha\|w_1\|\ e_1 - \bar{T}_m \tilde{y}), Q_m \tilde{W}_{m+1}^T \tilde{W}_m u_i) \tag{16}$$
$$= (\bar{g}_m - \bar{R}_m \tilde{y}, Q_m \tilde{W}_{m+1}^T \tilde{W}_m u_i) \tag{17}$$
$$= (\gamma_{m+1} e_{m+1}, Q_m \tilde{W}_{m+1}^T \tilde{W}_m u_i) \tag{18}$$

We are just interested in the last components of $Q_m \tilde{W}_{m+1}^T \tilde{W}_m u_i$. The matrix $\tilde{W}_m \tilde{W}_{m+1}^T$ is already computed. The rest of the computation to obtain the term (18) is inexpensive.

In the second part of the tests to be presented in Section 5 the η_k's are taken to be equal to one.

2.4 The Algorithm

The algorithm can be summarized as follow:

Algorithm 1 *Acceleration by Discrete Orthogonal Polynomials*

(1) *Initialize:*
  (a) Obtain the estimates of $\tilde{\lambda}_{min}$ *and* $\tilde{\lambda}_{max}$ *using Gershgörin's theorem.*
  (b) Randomly generate a set S of m points in $[\tilde{\lambda}_{min}, \tilde{\lambda}_{max}]$
  (c) Choose x_0 *as a starting vector, and compute* $r_0 = b - Ax_0$.
(2) *Iterate:*
  (a) Generate the tridiagonal matrix $\bar{T}_m$*, i.e., the* α*'s and* β*'s from the Stieljes procedure.*
  (b) Compute the basis $W_m = [p_0(A)r_0, \ldots, p_{m-1}(A)r_0]$
  (c) Compute $y_m = \arg\min \|\beta e_1 - \bar{T}_m y\|_2$*,* $x_m = x_0 + W_m y_m$*, and* $r_m = r_0 - AW_m y_m$
  (d) If $\|r_m\|$ *is small enough then Stop.*
  (e) Else compute the set $\hat{\mathrm{S}}$ *of eigenvalues of the Generalized Eigenvalue Problem*

$$W_m^T A W_m u = \lambda W_m^T W_m u$$

  (f) Set $x_0 = x_m$*,* $r_0 = r_m$*,* $\mathrm{S} = \hat{\mathrm{S}}$ *and go to 2a.*

2.5 Cost Comparison with GMRES(m)

The costs of the new algorithm with GMRES(m) for the same Krylov subspace dimension m are as follows.

Accelerated Orthogonal Polynomial Method	GMRES(m)
1- Generation of $\bar{T}_m$ and W_m	1- Generation of V_m
2- Computation of $\tilde{y}_m$	with the Arnoldi process
3- Update of S	2- Computation of $\tilde{y}_m$

Step 2 requires about the same number of operations in both cases. Steps 1 and 3 of the Accelerated Orthogonal Polynomial Method require:

(1) $2m$ inner products of size m for the computation of $\bar{T}_m$.
(2) for the computation of W_m
  (a) m matrix vector produtcs with a sparse matrix of size n
  (b) multiplication of $\bar{T}_m$ by 2 diagonal matrices to form $\tilde{\mathrm{S}}_m$
  (c) $2m - 1$ saxpy on vectors of size n
  (d) $m + 1$ divisions of vectors of size n by a scalar

(3) for the generalized eigenvalues problem
   (a) a matrix - matrix product of size $mn \times n(m+1)$ which requires $m(m+1)/2$ inner products of size n for the computation of $W_m^T W_{m+1}$
   (b) matrix - matrix product of size $m(m+1) \times (m+1)m$ for the computation of $W_m^T A W_m$ via (13)
   (c) solve the eigenvalue problem (12)

Since it is assumed that $m \ll n$, steps 1, 2-b, 3-b and 3-c are inexpensive relative to the rest of the computation. In a parallel environment they can be redundantly computed on each node of the computer.

For GMRES(m), the amount of work to compute the basis V_m is as follows.

(1) $m(m+3)/2$ inner products of size n
(2) m matrix vector products, with a sparse matrix of size n
(3) $m(m+1)/2$ saxpy on vectors of size n
(4) $m+1$ divisions of vectors of size n by a scalar

Steps 2 and 4 of GMRES(m) require exactly the same number of operations as steps 2-a and 2-d of the ADOP algorithm.

The only steps to compare are 1 and 3 for GMRES(m), with 2-c and 3-a of ADOP. The total amount of work for the method is less than for GMRES(m). This is due to the fact that a 3-term recurrence is used for generating the Krylov basis W_m, because we assumed that the spectrum is real and that all approximate eigenvalues obtained subsequently are assumed to be real.

If the spectrum is complex and the polynomials p_j do not obey a three term recurrence then these polynomials can be computed as in the Arnoldi process, namely via a long recurrence of the form

$$h_{j+1,j} p_j(t) = t p_{j-1}(t) - \sum_{i=1}^{j} h_{ij} p_{i-1}(t) \tag{19}$$

The cost of this procedure then becomes comparable with that of GMRES(m). One notable difference however, is that while the computation of the inner products in GMRES(m) must be computed in sequence, the new procedure only requires a matrix-matrix product with far more inherent parallelism.

3 Case of a complex spectrum

When the spectrum is complex, then in general there is no three-term recurrence for the orthogonal polynomials associated with the approximate spec-

trum S and the weights η_i. There are two possible alternatives. The first, which was just mentioned is to use the full-term recurrence (19). The second is to use indefinite (non-Hermitian) inner products in order to recover the three-term recurrence. Each of these approaches has advantages and disadvantages.

3.1 Hermitian inner products on S

The discrete symmetric inner product (3) is replaced by the Hermitian discrete inner product:

$$\langle p, q \rangle = \sum_{\theta_k \in \mathrm{S}} \eta_k p(\theta_k) \overline{q(\theta_k)} \tag{20}$$

Here $\mathbb{P}_m$ is the set of all polynomials of degree strictly less than m with coefficients in $\mathbb{C}$. The coefficients η_k are real and positive weights.

The 3-term recurrence of the orthonormal polynomials (4) is now replaced by the recurrence (19). The scalars $(h_{i,j})_{1 \leq i \leq j+1}$ are computed in order for the polynomial p_j to be orthonormal to all previous ones with respect to the inner product (20). The Krylov basis K_m can be computed similarly, with the recurrence

$$h_{j+1,j} w_{j+1} = A w_j - \sum_{i=1}^{j} h_{i,j} w_i, \qquad j = 1, \ldots, m \tag{21}$$

with the convention that $w_0 \equiv 0$. Apart from this difference and the fact that the matrix T_m is replaced by a Hessenberg matrix, the algorithm is identical with Algorithm 1.

3.2 Indefinite Inner Products

An alternative to the previous approach is to define inner products in the form

$$\langle p, q \rangle = \sum_{\theta_k \in S} \eta_k p(\theta_k) q(\theta_k) \tag{22}$$

This inner product is again on the set $\mathbb{P}_m$ of all complex polynomials of degree strictly less than m and the coefficients η_k are real and positive weights. The main difference with the previous case is that the above bilinear form does not define a proper inner product. Indeed, the scalar $\langle p, p \rangle$ is not necessarily a positive number and therefore this inner product does not yield a proper

norm. We will make the assumption that the points of S come in conjugate pairs, i.e., if $\lambda \in$ S then also $\bar{\lambda} \in$ S. This means that the set S is symmetric with respect to the real axis which is clearly true if the elements of S are eigenvalues of a real matrix. In this situation $\langle p, q \rangle$ is real for real polynomials p and q and the computation can be kept in real arithmetic provided A is real and the initial set S satisfies the symmetry requirement discussed above.

The Stieljes procedure should be somewhat modifed to take into account the fact that $\langle p_i, p_i \rangle$ can be negative. In the following algorithm, it is assumed that $p_{-1} \equiv 0$ and $\beta_1 \equiv 0$.

Algorithm 2 *Stieljes procedure for indefinite inner products*

1 *Init: Define* $p_0(t) = 1/\alpha$, *where* $\alpha = \sqrt{| < 1, 1 > |}$.
2 *For* $j = 1, \ldots, m$ *Do*
3 $\quad q = tp_{j-1} - \beta_j p_{j-2}$
4 $\quad \alpha_j =< q, p_{j-1} >$
5 $\quad q := q - \alpha_j p_{j-1}$
6 $\quad \delta_{j+1} = \sqrt{| < q, q > |}$. *If* $\delta_{j+1} == 0$ *abort.*
7 $\quad p_j = q/\delta_{j+1}$;
8 $\quad \beta_{j+1} = \delta_{j+1} \frac{<p_j, p_j>}{<p_{j-1}, p_{j-1}>}$
9 *EndDo*

Note that the sequence of polynomials obtained is such that $\langle p_j, p_j \rangle = \pm 1$, so that the scalar β_{j+1} defined in line 8 is equal to $\pm\delta_{j+1}$ and requires just a sign adjustment to δ_{j+1}. We obtain a recurrence which is quite similar to that obtained in the real spectrum case. In the particular case when S is a real set, then $\langle q, q \rangle$ is always non-negative and therefore $\beta_{j+1} = \delta_{j+1}$, so we recover the case seen in Section 2.

At each step of the Stieljes procedure we have a recurrence of the form

$$\delta_{j+1} p_j(t) = t p_{j-1}(t) - \alpha_j p_{j-1}(t) - \beta_j p_{j-2}(t) \tag{23}$$

It is clear that if $w_j = p_{j-1}(A) w_1$ then the relation (7) is still valid, but with a tridiagonal matrix of the form

$$\bar{T}_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \delta_2 & \alpha_2 & \beta_3 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \alpha_{m-1} & \beta_m \\ & & & & \delta_m & \alpha_m \\ & & & & & \delta_{m+1} \end{pmatrix}$$

in which $\beta_i = \pm\delta_i$ and $\delta_i > 0$ for $i = 2, \ldots, m$. In a sense this technique is comparable with the BiCG and QMR algorithms which, similarly, are based on an indefinite inner product. At each step there is the potential for a breakdown in Line 6 of the Stieljes procedure. If this happens it is probably best to restart the algorithm – though look-ahead strategies similar to the ones developed for the nonsymmetric Lanczos algorithm and QMR are also possible [3].

4 Preconditioning

It is also possible to precondition the ADOP method and the three preconditioning options, left, right and split preconditioning are all available.

4.1 Left Preconditioning

When considering left preconditioning, the linear system $M^{-1}Ax = M^{-1}b$ is solved by building the Krylov subspace $\mathrm{K}_m(M^{-1}A, r_0)$. The approximate eigenvalues considered are those of $M^{-1}A$. Usually M^{-1} is not available and only its action on a vector can be performed. Approximating the eigenvalues with the Gershgörin's theorem is no longer possible, and one iteration of an Arnoldi-like method can be used. The steps 1-a and 1-b of the Algorithm 1 is replaced by one iteration of the |S|-restarted Arnoldi method.

Next the tridiagonal matrix T_m is constructed, and the vectors w_i satisfying the 3-term recurrence

$$\beta_{i+1}w_{i+1} = M^{-1}Aw_i - \alpha_i w_i - \beta_i w_{i-1}, \quad i = 1, \ldots, m \tag{24}$$

are build. This yields the relation $M^{-1}AW_m = W_{m+1}\bar{T}_m$. The new approximate solution takes the form $\tilde{x} = \tilde{x}_0 + W_m\tilde{y}$ and $\tilde{r} = \tilde{r}_0 + M^{-1}AW_m\tilde{y}$.

Finally, the Generalized Eigenvalue Problem 2-e is replaced by

$$\begin{aligned} W_m^T M^{-1}AW_m \; u &= W_m^T W_{m+1}\bar{T}_m \; u \\ &= \lambda \; W_m^T W_m \; u \end{aligned} \tag{25}$$

More generally, the difference between the left preconditioned and the non preconditioned ADOP method lies in the initialization of the approximate eigenvalues, and the application of $M^{-1}A$ instead of A each time A occurs in the non preconditioned algorithm.

4.2 Right Preconditioning

When using the right preconditioning, the Krylov subspace $\mathrm{K}_m(AM^{-1}, r_0)$ is build to find x such that $Mx = u$ and $AM^{-1}u = b$. The set S is initialized with |S| approximate eigenvalues of AM^{-1} and the matrix W_m satisfies $AM^{-1}W_m = W_{m+1}\bar{T}_m$. The vector u is never computed and the solution is $\tilde{x} = \tilde{x}_0 + M^{-1}W_m\tilde{y}$ and $\tilde{r} = \tilde{r}_0 + AM^{-1}W_m\tilde{y}$.

The Generalized Eigenvalue Problem 2-e is replaced by

$$\begin{aligned} W_m^T AM^{-1}W_m \; u &= W_m^T W_{m+1}\bar{T}_m \; u \\ &= \lambda \; W_m^T W_m \; u \end{aligned} \tag{26}$$

which yields no further changements in the computation of the new approximate eigenvalues.

Recall that GMRES(m) gives rise to a flexible version ([5]), which enables changing the precondioning matrix at each new building vector of the basis. This is not possible with ADOP since it is based on the eigenvalues of the matrix AM^{-1}, which has to be the same at each step.

4.3 Split Preconditioning

The split preconditioning takes into account the two previous forms by solving $L^{-1}AU^{-1}u = U^{-1}b$ with $Ux = u$.

5 Experiments

All tests have been performed on the CRAY T3E of I.D.R.I.S. [1] with matrices taken from the Harwell-Boeing, and the SPARSKIT collection available at the University of Minnesota. The size of the matrices studied are between 886 and 21200, and some are very sparse. The maximum number of processors used is 8. The program was originally coded in F90, but for efficiency reasons, we translated most of the code into F77. However, we are still using the F90 compiler. The Message Passing Library used is MPI.

The first part of the tests focus on the behavior of ADOP and compares the use of the Hermitian and indefinite discrete inner product with the GMRES

[1] Institut du Développement et des Ressources en Informatique Scientifique - CNRS, Bâtiment 506 - B.P. 167, 91403 ORSAY CEDEX, FRANCE

method by testing small and medium size matrices. From the point of view of speed, it appears that the indefinite inner product is superior. The second part compares the ADOP method with GMRES method through the use of the P_SPARSLIB library (see [7]) and larger matrices.

For all the cases taken into account, the residual of the ADOP and GMRES methods are never computed as $\tilde{r} = b - A\tilde{x}$, but updated as

$$\tilde{r} = \gamma_{m+1}\Omega_{m+1}Q_m e_{m+1} \tag{27}$$

with $\Omega_{m+1} = W_{m+1}$ for the ADOP method and $\Omega_{m+1} = V_{m+1}$ for the GMRES method, in order to avoid the matrix-vector products with the matrix A, which are time-consuming operations.

In the next sections, the acronym ADOP stands for the ADOP method with the indefinite discrete inner product and the weights taken to one, ADOPW for the ADOP method with the same indefinite inner product and the weights taken as explained in the section 2.3, and ADOPH means the ADOP method with the Hermitian inner product.

In the following tables, m indicates the size of the Krylov subspaces. For each test, the Iter/time data shows the number of iterations, before the algorithm stops and the time of the method (ADOP or GMRES) to stop. This is the time spent within the method itself, including matrix vector products, and the preconditioning computations, but excluding the preprocessing. The second line of each performance data shows the final residual norm reduction achieved. The algorithm is stopped if a residual norm reduction of 10^{-6} is not achieved after a maximum of 400 iterations is reached. The third line represents the time spent exclusively in the accelerator. i.e., in the GMRES or ADOP method, till the convergence or the number of maximum of iterations is reached.

5.1 Hermitian versus Indefinite Inner Product

In this section, we wish to compare the general behavior of the ADOP, ADOPW, ADOPH and GMRES methods.

The matrix is split column-wise among the processors, in order that each partition has *ncol* consecutive columns of the initial matrix A and approximatively the same number *nnz* of non zero elements. When preconditioning is employed, a (left) block Jacobi preconditioner is used in which each block corresponds to the local matrix on each processor. The solution needed in the Block Jacobi Preconditioning operation is computed with an ILU factorization of these blocks. The solution is not known in advance, and the right hand-side is the

Table 1
Matrix ORSIRR1, NbPE = 4, Precon = no

m	Perf.	GMRES(m)	ADOP(m)	ADOPW(m)	ADOPH(m)
5	Iter/time	400 / 3.162	400 / 3.121	400 / 3.091	400 / 4.906
	reduction	0.674E+00	0.226E-01	0.460E+00	0.658E-02
	Accel. time	1.013	0.954	0.956	1.495
7	Iter/time	400 / 4.551	400 / 4.318	400 / 4.318	400 / 7.012
	reduction	0.505E+00	0.233E-05	0.160E+00	0.125E-03
	Accel. time	1.570	1.319	1.315	2.243
9	Iter/time	400 / 6.058	344 / 4.742	400 / 5.590	336 / 7.906
	reduction	0.405E+00	0.888E-06	0.219E-01	0.672E-06
	Accel. time	2.234	1.496	1.782	2.632
10	Iter/time	400 / 6.886	312 / 4.843	400 / 6.245	358 / 9.531
	reduction	0.218E-01	0.144E-06	0.156E-01	0.927E-06
	Accel. time	2.611	1.539	2.003	3.297
15	Iter/time	276 / 7.786	205 / 5.106	400 / 9.962	194 / 8.387
	reduction	0.880E-06	0.438E-06	0.666E-03	0.754E-06
	Accel. time	3.385	1.828	3.561	3.419
20	Iter/time	364 / 14.998	116 / 4.083	324 / 11.662	168 / 10.823
	reduction	0.100E-05	0.781E-06	0.949E-06	0.580E-07
	Accel. time	7.189	1.644	4.775	4.978
30	Iter/time	95 / 6.775	191 / 10.996	343 / 19.956	88 / 10.166
	reduction	0.998E-06	0.663E-06	0.623E-06	0.686E-06
	Accel. time	3.687	5.012	9.085	5.570
40	Iter/time	56 / 6.009	161 / 14.353	400 / 35.181	102 / 18.339
	reduction	0.999E-06	0.400E-06	0.564E-05	0.615E-06
	Accel. time	3.636	7.490	18.253	11.324
50	Iter/time	33 / 4.851	236 / 31.181	379 / 49.18	110 / 29.393
	reduction	0.999E-06	0.968E-06	0.525E-06	0.280E-07
	Accel. time	3.080	18.552	28.980	19.791

unit vector. The initial vector is zero and the different restarts are 11 values in the range [5−50]. The process is stopped whenever $\text{norm}(r)/\text{norm}(b) < 1.0\text{e}{-6}$ or the maximum number 400 of iterations is reached.

Tests were done on the matrices SHERMAN4, SHERMAN5, ORSIRR1, ORSIRR2, ORSREG1. Their sizes are between 886 and 3312. The size of the set S of the approximate eigenvalues of A is equal to m. Tests performed on sequential machines and not reported here seemed to indicate that that though convergence tended to be slower with Hermitian inner products, overall the method tended to be more realiable.

The results shown in Table 1 are with the matrix ORSIRR1 from the Harwell-Boeing collection. It has 6858 non zero elements, and a size of 1030. We can make the following remarks

(1) One iteration of ADOPW and ADOP take roughly the same time. This

is due to the fact that the ADOPW methods requires very few additional computations compared to ADOP, and these are negligible.

(2) From a convergence and time point of view ADOPW is not competitive compared to GMRES and ADOP.

(3) One iteration of ADOPH is more expensive than one iteration of ADOP: the recurrence for building the vectors of the basis of K_m is longer for ADOPH than for ADOP. The code ADOPH requires complex arithmetic. Although most of the complex arrays have been implemented by two real arrays the solution of the generalized eigenvalue problem is performed by LAPACK routine in complex arithmetic.

(4) The number of iterations required for ADOPH to converge decreases as m increases. Nevertheless, ADOPH remains uncompetitive from the point of view of time compared with ADOP.

(5) ADOP converges faster than GMRES for small size of restarts, that are between 5 and 30. When m is too large, the matrix W_m becomes ill-conditionned and the method fails to converge. This is also true for ADOPW. In contrast ADOPH is more reliable and still converges for larger values of restart.

(6) One iteration of ADOP is always faster than one iteration of GMRES. The time gained by ADOP compared with GMRES for one iteration grows up to 33% as m increases to $30 - 40$ and then decreases.

(7) The best overall time of computation is reached for m equal to 10 for ADOP, and m equal to 50 for GMRES.

(8) The best overall time is reached by ADOP for all the tested considered.

When considering the same matrix preconditioned with a left block Jacobi, the previous remarks are still valid except that the best overall time for GMRES is reached for $m = 30$, whereas the best overall time for ADOP still remains with $m = 10$. Furthermore, ADOP behaves better for greater restarts, especially for $m = 50$.

We now turn to the test results with the second matrix shown in the table 2. The matrix is SHERMAN4 another matrix from the Harwell-Boeing collection. It has 3786 non zero elements, and a size of 1104. The first five observations made above for the unpreconditioned matrix ORSIRR1 are still valid. Otherwise:

(1) One iteration of ADOP is always faster than one iteration of GMRES. However, now the time gained by ADOP compared with GMRES for one iteration grows up to 30% as m increases to only 20 and then decreases.

(2) The best overall time of computation is reached for m equal to 15 for ADOP, and m equal to 50 for GMRES.

The best overall time is still achieved by ADOP.

Table 2
Matrix SHERMAN4, NbPE = 4, Precon = no.

m	Perf.	GMRES(m)	ADOP(m)	ADOPW(m)	ADOPH(m)
5	Iter/Time	143 / 1.087	104 / 0.779	355 / 2.714	136 / 1.667
	Reduction	0.663E-06	0.941E-06	0.941E-06	0.565E-06
	Accel. Time	0.361	0.250	0.876	0.527
8	Iter/Time	100 / 1.280	56 / 0.682	159 / 1.953	83 / 1.718
	Reduction	0.949E-06	0.996E-06	0.907E-06	0.979E-06
	Accel. Time	0.471	0.216	0.632	0.573
9	Iter/Time	88 / 1.298	58 / 0.799	133 / 1.854	56 / 1.339
	Reduction	0.980E-06	0.866E-06	0.933E-06	0.132E-06
	Accel. Time	0.492	0.257	0.604	0.455
10	Iter/Time	70 / 1.163	52 / 0.803	82 / 1.280	49 / 1.326
	Reduction	0.963E-06	0.828E-06	0.868E-06	0.777E-06
	Accel. Time	0.455	0.262	0.422	0.466
15	Iter/Time	54 / 1.476	25 / 0.609	44 / 1.087	30 / 1.313
	Reduction	0.993E-06	0.978E-06	0.877E-06	0.321E-06
	Accel. Time	0.649	0.222	0.406	0.551
20	Iter/Time	28 / 1.134	19 / 0.65	29 / 1.023	22 / 1.423
	Reduction	0.970E-06	0.667E-06	0.860E-06	0.540E-06
	Accel. Time	0.551	0.265	0.426	0.676
30	Iter/Time	14 / 0.985	19 / 1.148	46 / 2.750	14 / 1.644
	Reduction	0.976E-06	0.705E-06	0.677E-06	0.717E-06
	Accel. Time	0.546	0.556	1.330	0.926
40	Iter/Time	7 / 0.956	20 / 1.805	41 / 3.702	12 / 2.280
	Reduction	0.985E-06	0.770E-06	0.874E-06	0.109E-06
	Accel. Time	0.445	0.985	2.013	1.456
50	Iter/Time	5 / 0.736	16 / 2.162	22 / 3.088	6 / 1.619
	Reduction	0.962E-06	0.248E-06	0.908E-06	0.107E-06
	Accel. Time	0.478	1.349	1.968	1.101

When considering the preconditioned case, it appears that ADOP behaves better than GMRES in the number of iterations for only very small sizes of restarts (under 9) and that for the restart sizes equal to 15 and 30, ADOPW is superior to ADOP.

ADOP is not competitive for restart sizes between 20 and 50, where the number of iterations of GMRES considerably decreases with m. For $m = 50$, the number of iterations for GMRES is 2. This is also the case for ADOP, ADOPW, ADOPH. The difference is that ADOP must finish the second iteration, i.e., to build the 50 vectors of the basis even when the first few would be sufficient to achieve convergence. The 2 iterations of GMRES are indeed 1 complete iteration plus 2 new vectors of the basis. So in overall time, GMRES converge faster, because it has fewer matrix-vectors and preconditionings to perform.

This case of fast convergence of the ADOP method for large restarts, is un-

common. The tests showed that for large m, when GMRES converge rapidly in 1 or 2 iterations, ADOP is slow. One solution to this case could be to start with one iteration of GMRES. Since it would also enable us to approximate the first eigenvalues, the algorithm would be fast and reliable: When GMRES converges very fast that is in 1 or 2 iterations, so will the general method. When ADOP converges faster, the global method will converge at the same speed. This requires practically not much additional computations in the preconditioned case, since a method for approximating the eigenvalues has to used anyway. This can be done within the first iteration of GMRES.

In general the tests showed that: (1) ADOPH is not attractive from the point of view of the execution time; and (2) ADOPW is not competitive from the point of view of the number of iterations.

5.2 ADOP versus GMRES

In this section, we compare exclusively the ADOP method with the indefinite inner product and the weights taken all equal to one, with GMRES, on larger matrices. For this, we chose the PSPARSLIB library, which already implements GMRES with right preconditioning and flexible GMRES. Before beeing mapped to the processors, the matrix is partitioned with the Recursive Spectral Bisection algorithm. When using a preconditioning, we again employ block Jacobi. The diagonal blocks are those of the reordered matrix and are again approximated through an ILU factorization. The solution is taken to be equal to the vector of all ones. The initial vector is zero and the different restarts are 11 values in the range $[5 - 50]$. The process is stopped whenever $\text{norm}(r)/\text{norm}(b) < 1.0\text{e} - 6$ or the maximum number 400 of iterations is reached.

The matrices tested are EX11, EX19 and EX35 from the FIDAP collection, RAEFSKY3 and INACCURA from the Simon collection, The sizes of these matrices are between 12005 and 21000.

The matrices considered first are EX11 and EX35 shown in the tables 3 and 4 without preconditioning. EX11 has 1096948 non zero elements and its size is 16614. EX35 has 228208 non zero elements and its size is 19716. EX35 is highly sparse. The following remarks can be made:

(1) The ADOP method converges faster than GMRES for all the cases tested.
(2) One iteration of ADOP is faster than one iteration of GMRES. This is not always true for m equal to 5.
(3) For EX35, GMRES never converges, while ADOP converges for m between 25 and 50. The reduction in the number of iterations decrease as m increases for ADOP. The best execution time is achieved with ADOP

Table 3
Matrix EX11, NbPE = 4, Precon = no.

m	Perf.	GMRES(m)	ADOP(m)
5	Iter/Time	400 / 43.747	400 / 43.954
	Reduction	0.112E-03	0.851E-05
	Accel. Time	4.773	5.035
10	Iter/Time	400 / 90.196	400 / 87.792
	Reduction	0.106E-04	0.664E-05
	Accel. Time	12.550	10.165
15	Iter/Time	400 / 139.688	400 / 131.902
	Reduction	0.744E-05	0.251E-05
	Accel. Time	23.315	15.655
20	Iter/Time	400 / 192.702	201 / 90.275
	Reduction	0.607E-05	0.509E-06
	Accel. Time	37.208	12.386
25	Iter/Time	400 / 245.705	172 / 96.992
	Reduction	0.469E-05	0.974E-06
	Accel. Time	53.786	13.732
30	Iter/Time	400 / 303.537	153 / 105.229
	Reduction	0.331E-05	0.425E-06
	Accel. Time	73.282	16.401
40	Iter/Time	400 / 427.784	114 / 105.873
	Reduction	0.133E-05	0.160E-07
	Accel. Time	120.791	17.652
50	Iter/Time	283 / 397.936	84 / 100.635
	Reduction	0.100E-05	0.939E-06
	Accel. Time	126.791	19.352

for m equal to 25. Because of the high degree of sparsity of the matrix, the time spent in the method istelf accounts for a large part of the global time. The gain due to ADOP is then quite high.

(4) For EX11, GMRES reduces the norm as m increases but only converges for m equal to 50. ADOP reduces the norm as m increases but converges for m between 20 and 50. The best time computed is for m equal to 20.

What is true for the unpreconditioned case, is not true for the right preconditioned Block Jacobi case. Table 5 shows the behavior of EX11 when it is preconditioned. In this case, the number of iterations to reach converge is better for ADOP only for $m = 5,\ 7$ and 20. The overall best time is reached by GMRES for $m = 10$. The best time for ADOP is reached for $m = 20$, but it is no longer ompetitive with the best GMRES time.

In general all the experiments performed show the following points. First, without preconditioning, ADOP converges much faster than GMRES for all the tests performed, and the gain can be high. Second, when Block-Jacobi preconditioning is used, the behavior of ADOP either remains the same, or changes in an unpredictable manner. Furthermore, its associated residual has an erratic

Table 4
Matrix EX35, NbPE = 4, Precon = no

m	Perf.	GMRES(m)	ADOP(m)
5	Time/Iter	400 / 23.711	400 / 23.636
	Reduction	0.222E-03	0.551E-04
	Accel. Time	7.534	7.516
10	Time/Iter	400 / 51.218	400 / 47.068
	Reduction	0.724E-04	0.255E-04
	Accel. Time	19.115	15.057
20	Time/Iter	400 / 119.687	400 / 99.792
	Reduction	0.254E-04	0.154E-05
	Accel. Time	55.768	35.859
25	Time/Iter	400 / 160.722	386 / 121.621
	Reduction	0.194E-04	0.100E-05
	Accel. Time	80.897	44.598
30	Time/Iter	400 / 206.388	341 / 133.234
	Reduction	0.140E-04	341 / 0.993E-06
	Accel. Time	110.632	51.624
40	Time/Iter	400 / 311.198	289 / 155.591
	Reduction	0.618E-05	0.994E-06
	Accel. Time	183.622	63.434
50	Time/Iter	400 / 434.195	199 / 142.275
	Reduction	0.171E-05	0.913E-06
	Accel. Time	274.790	62.999

behavior. A better adapted preconditioner should be tested; possibly a polynomial one with the polynomial taken to be the Chebyshev or least-squares polynomial, which could take advantage of the approximate eigenvalues computed.

6 Conclusion

The method proposed is attractive, for the following reasons. First, for matrices not requiring any preconditioning, the gain obtained in time by ADOP compared with GMRES is not negligible. This is particularly true for highly sparse matrices. Second, the method is amenable to efficient combinations with other techniques. We can for example use GMRES for the first iterations and then switch to ADOP to take advantage of its speed in later steps. We can also use it as a preconditioner in flexible GMRES ([5]).

Better adapted preconditioners should be sought when ADOP is used as the numerical experiments seem to indicate that ADOP is much more sensitive to poor preconditioners than GMRES. In particular, preconditioners that produce indefinite matrices may be ineffective when used with ADOP, though this remains to be studied more carefully. Preconditioners which could take

Table 5
Matrix EX11, NbPE = 4, Precon = Block Jacobi

m	Perf.	GMRES(m)	ADOP(m)
5	Iter/Time	134 / 19.554	122 / 17.821
	Reduction	0.982E-06	0.462E-06
	Accel. Time	2.071	1.909
7	Iter/Time	80 / 16.438	75 / 15.182
	Reduction	0.890E-06	0.985E-06
	Accel. Time	1.896	1.562
9	Iter/Time	55 / 14.415	63 / 16.474
	Reduction	0.100E-05	0.974E-06
	Accel. Time	1.811	1.811
10	Iter/Time	30 / 8.667	52 / 15.075
	Reduction	0.991E-06	0.821E-06
	Accel. Time	1.133	1.627
15	Iter/Time	29 / 12.925	38 / 16.517
	Reduction	0.996E-06	0.617E-06
	Accel. Time	2.009	1.811
25	Iter/Time	22 / 17.666	30 / 22.099
	Reduction	0.999E-06	0.855E-06
	Accel. Time	3.578	2.812
30	Iter/Time	18 / 17.030	46 / 41.038
	Reduction	0.999E-06	0.352E-06
	Accel. Time	3.822	5.619
40	Iter/Time	9 / 11.966	33 / 39.596
	Reduction	0.961E-06	0.240E-06
	Accel. Time	3.118	5.739
50	Iter/Time	6 / 10.012	62 / 94.958
	Reduction	0.982E-06	0.159E-05
	Accel. Time	2.936	15.507

advantage of the eigenvalues approximated in the method can be effective.

References

[1] Y. Saad. *Numerical Methods for Large Eigenvalue Problem.* Halstead Press, New York, 1992.

[2] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS publishing, New York, 1996.

[3] R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik*, 60:315–339, 1991.

[4] W. Gautschi. On generating orthogonal polynomials. *SIAM Journal on Scientific and Statistical Computing*, 3:289–317, 1982.

[5] Y. Saad A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific and Statistical Computing*, 14:461-469, 1993.

[6] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856-869, 1986.

[7] Y. Saad and A. Malevsky PSPARSLIB: A portable library of distributed memory sparse iterative solvers. In V. E. Malyshkin et al., editor, *Proceedings of Parallel Computing Technologies (PaCT-95), 3-rd international conference, St. Petersburg, Sept. 1995*, 1995.