Parallel Element-by-Element Spectral Multilevel Techniques

M.B. Davis * G.F. Carey†

Abstract

A parallel element-by-element multilevel strategy is developed and applied to two nonlinear, coupled PDE systems. Spectral (p) finite elements are used to discretize the problems and the multilevel solution strategy uses projections between bases of different degree (level). The projection methods for the p-multilevel schemes are developed and analyzed for Lagrange and hierarchic bases. The approach is implemented in a parallel element-by-element algorithm, which is particularly suitable for the spectral finite element method. Results are presented for two candidate nonlinear elliptic transport problems: the augmented drift-diffusion equations of semiconductor device modeling and the stream function-vorticity equations of incompressible fluid dynamics.

Key words: spectral elements, multilevel method, drift-diffusion equation.

AMS subject classifications: 65N30, 65N35, 65N55.

1 Introduction

Finite element methods in which the refinement is accomplished by increasing the degree p of the polynomial basis can give superior accuracy for similar computational work as compared to the more commonly used h refinement schemes. However, the condition number of the matrix deteriorates with increasing p. This motivates the need for an effective preconditioner, and a multilevel scheme in which the basis degree serves as the grid level is a natural choice. Such schemes may also be easily parallelized.

In the finite element context, the variational statement of the problem on the different grid levels leads naturally to the development of appropriate projection operators [5, 8]. Hierarchic basis functions, which are constructed by adding appropriate functions to the existing lower-degree polynomials, lead to matrices and vectors which are nested. This may be a particularly suitable choice for multilevel methods, since the projections for hierarchic multilevel are performed by merely truncating or appending a part of the vector at little or no computational cost [9].

Element-by-element strategies have proven to be efficient and scalable for parallelization of finite element methods using gradient iterative solvers [2, 3, 6, 7]. The basic idea in the parallel EBE scheme is to avoid assembling the system and instead perform matrix-vector and dot products in parallel at the element level. All matrices and vectors are stored in element format, which means the memory is scalable with the number of elements. Use of high-degree polynomials as the bases makes the element vectors longer and increases the ratio of calculation to communication by increasing the ratio of total degrees-of-freedom to element boundary degrees-of-freedom.

Moreover, in this approach, multilevel operations such as residual calculation, restriction and prolongation can be confined to an element and hence are completely parallel. The only steps that require communication are the smoothing (iteration) phase and coarse grid solves. A further advantage of spectral multilevel methods is that the number of elements in the domain remains constant, and hence the decomposition of the domain is fixed across grid levels. An important issue with parallel multilevel methods defined in this way is the ratio of communication to calculation. Although this ratio may be small for the fine level (high-degree basis), on coarser levels it gets successively larger, and at some level the communication time may dominate the total computational time.

2 Spectral elements and multilevel

An alternative to refining the mesh by making the element size h smaller is to increase the degree p of the polynomial

*University of Texas at Austin
†University of Texas at Austin

ICOSAHOM'95: Proceedings of the Third International Conference on Spectral and High Order Methods.

basis. Use of high-p finite elements can give higher-order accuracy for the same number of grid points. For example, the L^2 finite element error estimate for elliptic PDE's has the form [1]

$$\|e\| = C(p)h^{\mu}, \qquad \mu = min(r, p+1) \tag{1}$$

where r is the regularity of the solution ($u \in H^r$) and p is the degree of the basis. For smooth solutions the convergence rate with respect to h is then $O(h^{p+1})$. If we increase the polynomial degree instead of decreasing the grid spacing we get exponential convergence of the error. This rate is not achieved in the vicinity of a singularity due to the local lack of regularity. In such regions the error is $O(h^r)$ and, therefore, increasing p will not increase the rate beyond r. The optimal refinement strategy should be to decrease the element size h near a singularity, and to increase the degree p in smooth regions. Of course, we can grade the mesh (redistribute the grid) to cluster near a singularity and then increase p uniformly [5].

One disadvantage of the p-type finite element method is that the conditioning of the matrix deteriorates with increasing p. This deterioration is dependent on the type of basis used. One way to counter this is to apply a preconditioner to the system. A p-type multilevel method may be defined by using the degree of the polynomial basis as the grid level. The intergrid transfers can then be naturally defined in terms of expansions in the appropriate bases.

The analysis of a finite element Galerkin multilevel scheme is best carried out in the variational setting. In this way the Galerkin statement can be formulated on each grid level, and the consistency of the projection operators with the finite element discretizations on the associated grid levels is assured. The approach here follows that in [5]. We proceed by considering a representative linear elliptic problem on a domain Ω with a boundary $\partial\Omega$:

$$L(u) = f \quad \text{in} \quad \Omega \tag{2}$$

$$u = g \quad \text{on} \quad \partial\Omega \tag{3}$$

where L denotes the differential operator. Applying the method of weighted residuals and integrating by parts, the variational statement of the problem has the form: Find $u \in H$ with $u = g$ on $\partial\Omega$ such that

$$a(u, v) = f(v) \quad \forall v \in H \tag{4}$$

with $v = 0$ on $\partial\Omega$. Here $a(\cdot,\cdot)$ denotes the bilinear functional, $f(\cdot)$ is a linear functional and H is the appropriate space of admissible functions. Introducing a finite element discretization and a polynomial basis so that $S^p \subset H$, we define the approximate variational problem on grid level p as: Find $u_p \in S^p$ with $u_p = g$ on $\partial\Omega$ such that

$$a(u_p, v_p) = f(v_p) \quad \forall v_p \in S^p \tag{5}$$

with $v_p = 0$ on $\partial\Omega_p$. Introducing the finite element expansion and evaluating the integrals in (5) leads to a linear system of the form

$$\mathbf{A}_p \mathbf{u}_p = \mathbf{b}_p \tag{6}$$

where p once again indicates the grid level. Now consider a multilevel scheme where (6) corresponds to the fine grid system. Application of an iterative smoother to this system yields an approximation u_p^* and associated error $e_p^* = u_p - u_p^*$. Substituting this into (5), the error e_p^* is specified by the residual equation

$$a(e_p^*, v_p) = r^*(v_p) \text{ for all } v_p \in S_0^p \tag{7}$$

where

$$r^*(v_p) = f(v_p) - a(u_p^*, v_p). \tag{8}$$

Next, introduce a coarse grid level q such that $S^q \subset S^p$. Since all v_q are in S^q and thus in S^p we can test against the set of bases v_q [5] so the solution of (7) also satisfies the property

$$a(e_p^*, v_q) = r^*(v_q) \text{ for all } v_q \in S_0^q \tag{9}$$

where $r^*(v_q) = f(v_q) - a(u_p^*, v_q)$. This system is obviously underdetermined, so we take the best (Galerkin) approximation $e_q^* \in S^q$ to e_p^*. That is, find $e_q^* \in S_0^q$ such that

$$a(e_q^*, v_q) = r^*(v_q) \text{ for all } v_q \in S_0^q \tag{10}$$

Substituting the finite element expansion in (10) yields the coarse level system for the error correction vector

$$\mathbf{A}_q \mathbf{e}_q^* = \mathbf{r}_q. \tag{11}$$

where $\mathbf{A}_q$ is computed by evaluating the bilinear form on the space S^q and the right side vector defines a natural projection of the residual from S^p to S^q. More specifically, (8) implies

$$r^*(v_q) = f(v_q) - a(u_p^*, v_q). \tag{12}$$

Note that this requires the $a(\cdot,\cdot)$ inner product of u_p^* and v_q.

As an illustration, consider the bilinear form for the Laplacian

$$a(u, v) = \int_{\Omega} \nabla u \cdot \nabla v \, d\Omega \tag{13}$$

Introducing a polynomial expansion for u_p^* and polynomial test function v_q

$$u_p^* = \sum_{j=1}^{N_p} (u_p^*)_j \phi_j^p(\mathbf{x}), \quad v_q = \phi_i^q(\mathbf{x}) \tag{14}$$

where ϕ_j^p and ϕ_i^q denote the respective basis functions for S^p and S^q. Upon substitution this yields

$$a(u_p^*, v_q) = \int_\Omega \nabla(\sum_{j=1}^{N_p}(u_p^*)_j\phi_j^p)\cdot\nabla\phi_i^q\, d\Omega$$

$$= \sum_{j=1}^{N_p}(\int_\Omega \nabla\phi_j^p\cdot\nabla\phi_i^q\, d\Omega)(u_p^*)_j \tag{15}$$

Let this projection operator be denoted $\mathbf{A}^{q,p}$. Then

$$A_{ij}^{q,p} = a(\phi_i^q, \phi_j^p) \tag{16}$$

so that (12) implies

$$r^*(\phi_i^q) = f(\phi_i^q) - \sum_{j=1}^{N_p} A_{ij}^{q,p}(u_p^*)_j \tag{17}$$

or in matrix form

$$\mathbf{r}_q^* = \mathbf{f}_q - \mathbf{A}^{q,p}\mathbf{u}_p^* \tag{18}$$

Now, in the spirit of traditional multigrid methods on finite difference grids, we can characterize the projection as the product of its component parts: residual vector calculation (matrix multiplication) and projection to the coarser space.

To illustrate this idea, let us expand the test function ϕ_i^q in the higher-dimensional basis. Then

$$\phi_i^q = \sum_{k=1}^{N_p} m_{ik}^{q,p}\phi_k^p \tag{19}$$

Substituting (19) into (17)

$$r^*(\phi_i^q) = f(\phi_i^q) - a(u_p^*, \phi_i^q)$$

$$= \sum_{j=1}^{N_p} m_{ij}^{q,p} f(\phi_j^p) - \sum_{j=1}^{N_p} m_{ij}^{q,p} a(u_p^*, \phi_j^p)$$

$$= \sum_{j=1}^{N_p} m_{ij}^{q,p} r^*(\phi_j^p) \tag{20}$$

or in matrix form

$$\mathbf{r}_q^* = \mathbf{M}^{q,p}(\mathbf{b}_p - \mathbf{A}_p\mathbf{u}_p^*) = \mathbf{M}^{q,p}\mathbf{r}_p^* \tag{21}$$

At this point we need to determine the actual values of $m_{ik}^{q,p}$ in order to be able to carry out the projection. The following analysis is for Lagrange bases. These bases have the interpolation property that the value of each basis function is one at the node corresponding to the basis function, and zero at all other nodes, *i.e.* $\phi_i(\mathbf{x}_j) = \delta_{ij}$. It follows that

$$\phi_i^q(\mathbf{x}_j) = \sum_{k=1}^{N_p} m_{ik}^{q,p}\phi_k^p(\mathbf{x}_j) = m_{ij}^{q,p} \tag{22}$$

That is, the components of the projection matrix $\mathbf{M}^{q,p}$ are simply the values of the coarse grid basis at the fine grid nodes.

To complete the multilevel concept in the variational setting, a prolongation operator is needed which will project the error correction in equation (11) to grid level p. A natural choice for the prolongation operator is the transpose of the restriction operator in (22). Then the fine grid correction is computed from the coarse grid result according to

$$\mathbf{e}_p^* = (\mathbf{M}^{q,p})^T\mathbf{e}_q^* \tag{23}$$

As in the standard multigrid method, these error corrections are added to the approximate solution on the finer level to obtain the corrected approximation and smoothed by fine grid iteration.

2.1 Hierarchic bases and multilevel

Hierarchic basis functions are constructed by adding the next degree basis function to an existing basis of lower degree. For example, a quadratic basis may be formed by adding a quadratic polynomial to an existing linear basis. The higher degree basis then explicitly contains the lower degree basis. This implies that the finite element matrix and vector contributions corresponding to the lower degree polynomials are nested in the matrix and vector contributions for the higher degree polynomials. Similarly, coarsening implies simply deleting the appropriate rows and columns of the matrix. These properties are useful in the multilevel context. However, the interpolation property $\phi_i(\mathbf{x}_j) = \delta_{ij}$ for Lagrange polynomials holds only for the $p = 1$ basis.

The advantages of hierarchic bases become apparent when we extend the previous multilevel analysis to this setting. The change of basis coefficients in (19) for Lagrange bases are simplified for hierarchics because the basis for the space S^q is explicitly contained in the basis for S^p. That is,

$$\phi_i^q = \phi_i^p \quad 1 \le i \le N_q \tag{24}$$

If we follow the same strategy as for the Lagrange basis, then the residual projection in (17) becomes $r^*(\phi_i^q) = r^*(\phi_i^p)$ for $i = 1, 2, \ldots, N_q$. That is, the components of the residual projection to the subspace S^q are precisely the first N_q components of the fine grid residual. Hence, only

the first N_q components of the residual vector need to be computed.

Similarly, the coarse grid matrix $\mathbf{A}_q$ is now the leading $N_q \times N_q$ minor of the fine level matrix $\mathbf{A}_p$. Hence $\mathbf{A}_q$ does not need to be recomputed.

The subspace problem for the error correction in S^q again has the form in (11). That is,

$$\mathbf{A}_q \mathbf{e}_q^* = \mathbf{r}_q^*. \tag{25}$$

In a two level scheme this system is solved for $\mathbf{e}_q^*$. The projection of $\mathbf{e}_q^*$ to the higher level space S^p is trivial because of the explicit inclusion of the basis (recall (24)). Hence the corrected high level approximation is simply obtained by adding the N_q components of $\mathbf{e}_q^*$ to the first N_q components of $\mathbf{u}_p^*$. This new approximation in S^p can then be iteratively smoothed and the cycle repeated.

We can express this alternatively by introducing the projection operator defined by

$$(\mathbf{P}_p^q)_{ij} = \delta_{ij} \qquad i = 1, \ldots, N_q, \quad j = 1, \ldots, N_p. \tag{26}$$

Then $\mathbf{P}_p^q$ extracts the first N_q components of a vector of length N_p. The problem on the coarse grid can then be expressed as

$$\mathbf{A}_q \mathbf{e}_q^* = \mathbf{P}_p^q (\mathbf{b}_p - \mathbf{A}_p \mathbf{u}_p^*) \tag{27}$$

or after projection

$$\mathbf{A}_q \mathbf{e}_q^* = \mathbf{b}_q - \mathbf{P}_p^q \mathbf{A}_p \mathbf{u}_p^* \qquad i = 1, \ldots, N_q. \tag{28}$$

An alternative to the standard error correction method described above takes advantage of the nesting of the matrices and vectors [9]. Instead of computing a correction and adding it to the existing approximation, one can directly compute the corrected solution on the coarse level $\tilde{\mathbf{u}}_q = \mathbf{u}_q^* + \mathbf{e}_q^*$. More specifically, adding $\mathbf{A}_q \mathbf{u}_q^*$ to each side of (28) we have

$$\mathbf{A}_q \tilde{\mathbf{u}}_q = \mathbf{b}_q - \mathbf{P}_p^q \mathbf{A}_p \mathbf{u}_p^* + \mathbf{A}_q \mathbf{u}_q^*. \tag{29}$$

and clearly can solve this system to get $\mathbf{u}_q$. Note also that the matrix $\mathbf{A}_p$ and the vector $\mathbf{u}_p^*$ can be decomposed into blocks in the following fashion

$$\mathbf{A}_p = \begin{bmatrix} \mathbf{A}_q & \mathbf{A}_{qp} \\ \mathbf{A}_{pq} & \mathbf{A}_{pp} \end{bmatrix} \quad \mathbf{u}_p^* = \begin{bmatrix} \mathbf{u}_q^* \\ \mathbf{u}_{pp}^* \end{bmatrix} \tag{30}$$

where $\mathbf{A}_q$ and $\mathbf{u}_q^*$ are the coarse level matrix and solution approximation, respectively. Using this block decomposition and the projection defined in (26), equation (29) simplifies to

$$\mathbf{A}_q \tilde{\mathbf{u}}_q = \mathbf{b}_q - \mathbf{A}_{qp} \mathbf{u}_{pp}^*. \tag{31}$$

This form has two advantages. First, it emphasizes the fact that the full residual need not be computed. Second, no intermediate correction needs to be projected and added to the fine level approximation. The computed coarse level solution from (31) is trivially inserted as the first N_q components of the fine level solution vector.

2.2 Smoothing and correction

For reasons of convenience and parallelization, a simple point Jacobi scheme is the preferred smoother for the multilevel scheme. Any smoother must efficiently damp the high frequency error modes on the respective grids. For the relaxed Jacobi smoother, the relaxation parameter determines which frequencies are damped more quickly than others. If we assume that we wish to eliminate the highest frequency eigenmode corresponding to the leading eigenvalue of the discrete operator we obtain the relaxation factor for optimum multilevel smoothing [8, 14]

$$\omega = \left(\frac{(\mathbf{x}, \mathbf{A}\mathbf{x})}{(\mathbf{x}, \mathbf{D}\mathbf{x})}\right)^{-1} \tag{32}$$

where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = A_{ii}$.

Since this relaxation factor ω is a function of the matrix $\mathbf{A}$, it changes with both the problem and the discretization. Hence the optimum relaxation needs to be repeatedly calculated for each decoupled equation matrix. This value can be conveniently calculated using a power series method.

Calculation of this eigenvalue (or relaxation parameter) in a power series scheme generally requires 30-40 matrix-vector multiplies. If this were done at each nonlinear iteration for each decoupled linear system, the cost would quickly become a significant part of the total computation and communication time. However, if the linear system corresponding to a particular equation doesn't change enough to significantly alter this eigenvalue estimate over several block iterations, then the calculation can be done infrequently, and the cost can be amortized over several nonlinear iterations. In practice, this is found to be the case for both the augmented drift-diffusion and stream function - vorticity equations. Hence the relaxation is only recomputed every ten block iterations, or at the start of a continuation step.

There are two main choices for a multilevel strategy applied to a linear system: a V (or W) cycle, or a full multigrid cycle. The full multigrid (FMV) cycle uses nested iteration to improve the initial guess on the fine grid, hastening convergence. The strategy for solution of the nonlinear problem uses block iteration and successive approximations. Hence, at each nonlinear (or block) iteration, there

exists a good initial guess on the fine grid. For this reason only V-cycles are used as a multigrid cycling scheme.

The Jacobi smoother can generate oscillations in the cross-wind direction for convection dominated problems. The magnitude of these oscillations is proportional to the magnitude of the residual. In order to minimize these oscillations, an initial coarse grid correction (no pre-smooths) is performed at the first V-cycle. This initial correction further improves the initial guess from the previous block iterate, and convergence is improved [8].

Multigrid cycling schemes such as the Full Approximation Scheme can be used on the full nonlinear problem. Two alternative approaches are used here for the nonlinear problem. First, the multilevel solver is used only as the linear system solver for the fine grid problem, which is run to convergence using successive approximations and continuation. The second approach is a nested iteration scheme: The coarsest grid problem is run to convergence on the full nonlinear problem, including continuation in the boundary voltage or Reynolds number. The solution is then projected to the next finest grid and the problem on this grid level is then run to convergence at the final voltage or Reynolds number. This strategy is repeated until the highest grid level is reached.

3 Parallelization

Finite element methods divide a given problem domain into a union of elements for discrete solution. Hence schemes in which blocks of elements are operated on by a processor and the processor decomposition follows element boundaries provide a natural way in which to parallelize finite element methods [2, 3, 6, 7]. Adjacent elements share nodes on the element interface, so the information associated with these nodes may be stored on different processors. This information is updated during matrix-vector product or inner product operations. This means that messages must be passed between processors in order to update these values. The ratio of communication to computation is important because it can limit efficiency. The use of high-p elements, which have more internal degrees of freedom, results in a higher computation to communication ratio.

For a message passing paradigm, the time to send a message is given by

$$t_m = \alpha + \beta L_m \tag{33}$$

where α is the startup time or latency, β is the time per byte for message transfer, and L_m is the length of the message in bytes. For transfers in which a large amount of data is to be transferred, the key is to send as few messages as possible so that the startup time is minimized. Otherwise the startup time may dominate the communication time. The optimum situation would be to send one long message so that the latency is essentially hidden.

The previous argument motivates the need for message bundling using sendlists. A data structure is developed in which each processor has a pointer array which contains the element and node numbers that are shared with another processor. The order in which this information is to be placed into a message is also stored. Thus, when a vector is to be updated, a message vector is filled in order and sent to the appropriate processor. In turn, a message is received from that processor. A pointer array indicates which element and local node corresponds to which position in the array, in the same way as for the message which was sent. In this fashion all of the communication between adjacent processors can be accomplished using one message each way, and message latency is minimized. There is, however, some overhead in the packing and unpacking phases.

In the present work we can use an element-type data structure and recast all matrix-vector or projection operations at the element level. This means that instead of addressing a vector by its global node number, it is addressed by its element and local node number. In addition, each element has a pointer array which stores its neighbor elements and which nodes are shared with this neighbor. A specific processor will store information only for elements local to that processor. Elements are therefore addressed by the number local to that processor rather than a global element number. The pointer array for neighbor information includes the local element number and processor number for neighboring elements. This format facilitates parallel coding.

The formation of the matrix and RHS vector for finite element methods is usually accomplished by forming the local element matrices and vectors and summing them to get the global matrix and RHS as implied in the multilevel formulation of the previous sections. However, in the present parallel algorithm we no longer form the global matrix and RHS, but leave them in element form. The matrix and RHS calculation phase is therefore completely parallel. If the matrix is to be preconditioned using a global Jacobi preconditioner (diagonal scaling), then the diagonal elements of the matrices may be assembled to find the scaling factor. This accumulation phase will involve communication across processor boundaries.

Iteration by point iterative methods (Jacobi, SOR, etc.) as a smoother or gradient methods (CG, BCG, etc.) for the coarse grid solve involves repeated matrix-vector multiplications or dot products. Calculation of either one re-

quires that the information on shared nodes be updated. For instance, to compute a matrix-vector product such as

$$\mathbf{A}_p \mathbf{u}_p^* = \mathbf{v}_p \tag{34}$$

in the residual calculation (21), we write $\mathbf{u}_p^e = \mathbf{B}_e \mathbf{u}_p^*$ where $\mathbf{B}_e$ is the Boolean (adjacency or connectivity) matrix for element e and relates local to global variables. Then

$$\mathbf{A}_p = \sum_{e=1}^{E} \mathbf{B}_e^T \mathbf{A}_p^e \mathbf{B}_e \tag{35}$$

and

$$\begin{aligned} \mathbf{A}_p \mathbf{u}_p^* &= (\sum_{e=1}^{E} \mathbf{B}_e^T \mathbf{A}_p^e \mathbf{B}_e) \mathbf{u}_p^* = \sum_{e=1}^{E} \mathbf{B}_e^T \mathbf{A}_p^e \mathbf{u}_p^e \\ &= \sum_{e=1}^{E} \mathbf{B}_e^T \mathbf{v}_e = \mathbf{v} \end{aligned} \tag{36}$$

Hence the calculation requires element matrix-vector products that can be carried out independently in parallel.

Note that the solution vector is stored in summed form, but still in element format. The element accumulation in (36) requires communication if the element boundary corresponds to a processor boundary. High-p elements, which have more internal degrees of freedom, will result in a higher ratio of computation to communication.

Hence we see that when structured in this way, multilevel methods are a natural extension of the parallel EBE solution of finite element problems using gradient or other iterative methods. Obviously the smoothing phase proceeds as before with matrix-vector products updated across element boundaries. The issues of the residual calculation, restriction, and prolongation also need to be addressed.

For example, consider the residual calculation $\mathbf{r}_p = \mathbf{b}_p - \mathbf{A}_p \mathbf{u}_p^*$. In the EBE structure we obtain

$$\sum_{e=1}^{E} \mathbf{B}_e^T \mathbf{r}_p^e = \sum_{e=1}^{E} \mathbf{B}_e^T \mathbf{b}_p^e - \sum_{e=1}^{E} \mathbf{B}_e^T \mathbf{A}_p^e \mathbf{B}_e \mathbf{u}_p^* \tag{37}$$

but $\mathbf{B}_e \mathbf{u}_p^* = \mathbf{u}_p^e$ so (37) implies

$$\sum_{e=1}^{E} \mathbf{B}_e^T \mathbf{r}_p^e = \sum_{e=1}^{E} \mathbf{B}_e^T (\mathbf{b}_p^e - \mathbf{A}_p^e \mathbf{u}_p^e) \tag{38}$$

and we can use directly the element residuals

$$\mathbf{r}_p^e = \mathbf{b}_p^e - \mathbf{A}_p^e \mathbf{u}_p^e \tag{39}$$

Note also that because the element bases are defined locally we can introduce a local change of basis at the element level and corresponding to the global matrix $\mathbf{M}^{q,p}$ in (21) we have the element projection matrix $\mathbf{M}_e^{q,p}$. Then the element residual projection follows in a manner analogous to (21) as

$$\mathbf{r}_q^e = \mathbf{M}_e^{q,p} \mathbf{r}_p^e \tag{40}$$

Thus residual calculation and restriction take place on the element level, without communication, and are completely parallel operations. The prolongation to finer grid operates on the error vector, which is the solution on the coarser grid. This vector is stored in summed format, and hence no updating is necessary. Therefore, prolongation can also take place on an element and is once again completely parallel.

To summarize, the basic steps of the parallel algorithm for a two-level scheme are:

1. Processor partition. An element-by-element partitioning of the domain is made (contiguous element blocks are desirable). Sendlists for interprocessor communication are constructed.

2. High-level smoothing iteration.

   (a) For each processor subdomain in parallel compute element matrix and vector contributions at every level and store elementwise $\{\mathbf{A}_p^e\}, \{\mathbf{b}_p^e\}, \{\mathbf{A}_q^e\}, \{\mathbf{b}_q^e\}$.

   (b) For $k = 1, 2, \ldots, K$ iterations carry out relaxed Jacobi iteration (or a similar scheme). This involves local element matrix-vector products with element solution vector iterate $\{\mathbf{u}_p^e\}$ and communication between adjacent processors for element nodes on an interprocessor boundary.

3. Residual computation and projection. For each element in parallel, compute element residuals (level p) and locally project to level q to get residuals $\mathbf{r}_q^e$. For the hierarchic basis this reduces to simply computing the first N_q^e components of the residual for each element e.

4. Coarse grid solution. The coarse grid system is solved in parallel using an element-by-element generalized conjugate gradient solver.

5. High-level update. The coarse level correction for each element is projected elementwise to the higher level using (23) and these p-level element corrections are added to the current p-level element iterate.

6. Return to Step 2(b) and repeat the cycle until the fine grid iterate satisfies a specified stopping test.

Remark 1: In the above procedure all matrices and vectors are generated and stored elementwise. This permits a more straightforward parallel implementation and simplifies coding.

Remark 2: Additional level projections and smoothing iterations can be included in the usual way.

4 Applications

The above method is now formulated for two nonlinear, coupled transport problems. The first test case is the augmented drift-diffusion equations, which model the transport of electrons and holes (carriers) in semiconductor devices. The steady state, scaled form of the equations is [4, 11, 15, 16]

$$\begin{aligned} \lambda^2 \Delta\psi &= n - p - C \\ \nabla \cdot (\mu_n \nabla n - \mu_n n \nabla\psi) &= R \\ \nabla \cdot (\mu_p \nabla p + \mu_p p \nabla\psi) &= R \end{aligned} \tag{41}$$

where ψ is the electrostatic potential, n and p are carrier concentrations, μ_n and μ_p are mobilities, R is the recombination-generation rate, C is the doping, and λ is the scaled Debye length. An augmented mobility model is used for the carrier mobilities [4].

Equations (41) are decoupled iteratively and successive approximations used to solve the nonlinear problem [10]. A linearized Newton step is used on the potential equation to facilitate convergence[8, 10]. The equations are then discretized using a spectral finite element method. The approximate variational statement of the decoupled problem is: For iterate $k = 1, 2, 3, \ldots$ find ψ_h^{k+1}, n_h^{k+1}, and $p_h^{k+1} \in H^h(\Omega) \subset H^1(\Omega)$ satisfying the respective essential boundary conditions and such that

$$\begin{aligned} &\int_\Omega (\lambda^2 \nabla\psi_h^{k+1} \cdot \nabla w_h + (n_h^k + p_h^k) w_h)\, dx \\ &= \int_\Omega (p_h^k - n_h^k + C) w_h \, dx \quad \forall w_h \in H^h(\Omega) \\ &\int_\Omega \mu_n (-\nabla n_h^{k+1} \cdot \nabla u_h + n_h^{k+1} \nabla\psi_h^{k+1} \cdot \nabla u_h)\, dx \\ &= \int_\Omega R u_h \, dx \quad \forall u_h \in H^h(\Omega) \end{aligned}$$

and

$$\begin{aligned} &\int_\Omega \mu_p (-\nabla p_h^{k+1} \cdot \nabla v_h - p_h^{k+1} \nabla\psi_h^{k+1} \cdot \nabla v_h)\, dx \\ &= \int_\Omega R v_h \, dx \quad \forall v_h \in H^h(\Omega) \end{aligned}$$

for test functions w_h, u_h, v_h vanishing on those parts of the boundary where respective essential data is given and where zero flux conditions are taken elsewhere. Further details are given in [8].

Upon integration, three linear systems are obtained, which are solved successively with a multilevel method using available solution iterates of the other field variables [8]. Due to discontinuities in the doping C for these problems, the issue of gridding is critical for high-p elements. A transition region must be used between different doping values, and at least one element must be in this region. Otherwise, Gibbs-type oscillations are set up in the interior of the element containing the discontinuity, resulting in divergence.

The second application is the stream function-vorticity equations for incompressible Navier-Stokes flow in two dimensions. The steady state form of the equations is [8, 12, 13]

$$\begin{aligned} -\nu \boldsymbol{\Delta}\zeta + \mathbf{u} \cdot \nabla\zeta &= f \\ -\boldsymbol{\Delta}\psi &= \zeta \end{aligned} \tag{42}$$

where ψ is the stream function, ζ is the vorticity, $\mathbf{u}$ is the velocity, and f is the divergence of the body force.

Following the procedure outlined above, the equations are decoupled and discretized to obtain the variational statement: For iterate $k = 1, 2, 3, \ldots$ find $\zeta_h^{k+1} \in H^h(\Omega) \subset H^1(\Omega)$ satisfying the essential boundary conditions and such that

$$\begin{aligned} &\nu \int_\Omega \nabla\zeta_h^{k+1} \cdot \nabla w_h \, dx + \int_\Omega ((\psi_h^k)_y (\zeta_h^{k+1})_x - \\ &(\psi_h^k)_x (\zeta_h^{k+1})_y) w_h \, dx = \int_\Omega f w_h \, dx \quad \forall w_h \in H^h(\Omega) \end{aligned}$$

where vorticity boundary data is computed from the available stream function iterate. Then find $\psi_h^{k+1} \in H^h(\Omega)$ satisfying the essential boundary conditions and such that

$$\begin{aligned} \int_\Omega \nabla\psi_h^{k+1} \cdot \nabla v_h \, dx &= \int_\Omega \zeta_h^{k+1} v_h \, dx \\ &\forall v_h \in H^h(\Omega) \end{aligned}$$

Again, the linear systems arising from substitution of the appropriate basis and integration are solved with a multilevel scheme, and available solution iterates are used.

5 Results

The first test case is the augmented drift-diffusion equations. The example chosen is an $n^+ - n - n^+$ diode with doping of 5×10^{17} and 2×10^{15} in the n^+ and n regions

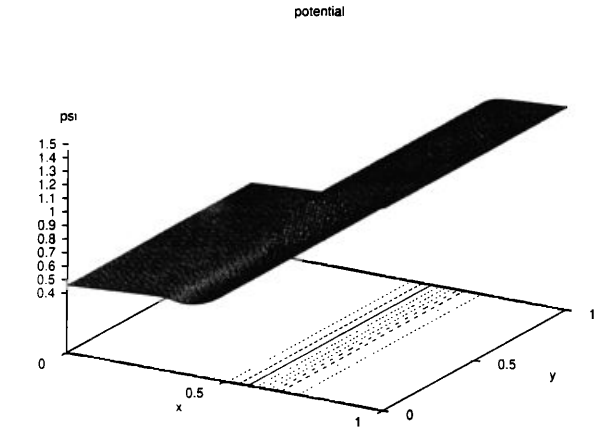

Figure 1: Electrostatic potential, $1V$ bias

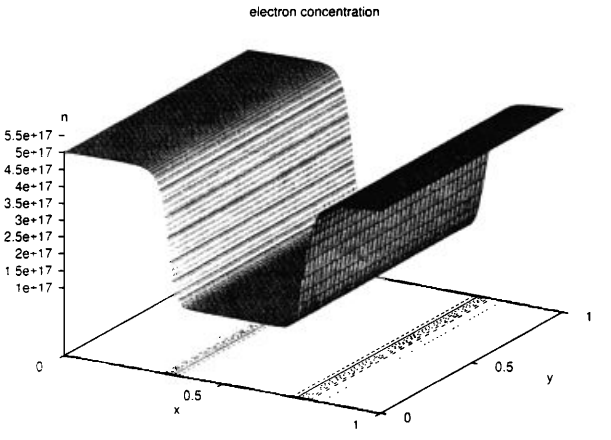

Figure 2: Electron concentration, $1V$ bias

respectively, device length of $.3\mu m$, active length of $.1\mu m$, and an applied bias of $1V$. Surface plots of the electrostatic potential and electron concentration are shown in Figures 1 and 2. This solution was computed using a uniform 12×12 grid of 144 bicubic elements, and a multilevel solver which used bilinear elements as the coarsest level.

A further study examined the performance of the nonlinear successive approximation scheme for different choices of element degree. The convergence history of the nonlinear iterations is shown in Figure 3 for biquadratic and biquintic elements. The graph shows the L_2 norm of the residual of the electron transport linear system at each block iteration. Experience has shown that for the drift-diffusion problem, a good initial iterate is very important for convergence of the decoupled iterations. Hence, both continuation in the applied bias and nested iteration are used. The spikes in Figure 3 correspond to either a new continuation step or the beginning of the solution on a finer grid in a nested iteration step. Note that for both cases, the nonlinear convergence is not smooth early in the history, but becomes smoother at later nested iteration steps and also later in the specific continuation or nested iteration cycle.

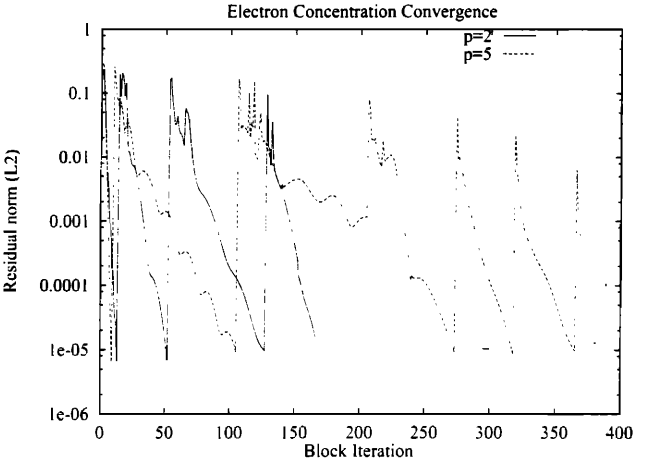

Figure 3: Electron concentration convergence with block iterations

The convergence of the multilevel method for the Lagrange and hierarchic bases is next considered. A comparison is given for both the potential and electron transport equations in Table 1. Here the tabulated values correspond to the fraction of the residual remaining at the end of the V-cycle. Note that the residual reduction factor is very good for the Lagrange basis for $p < 6$ in the potential equation, but deteriorates at $p = 6$. The reduction factor for the transport equation is not as good, as should be expected, and deteriorates as the basis degree is increased. For a Lagrange basis and the multilevel scheme used, this makes sense. The coarse level linear system is solved exactly, so the higher the basis degree, the further away it is from an exact solution since the intermediate levels are not solved exactly. The hierarchic basis reduction factors exhibit the opposite trend with basis degree. This makes sense since the problem at intermediate and coarse grids doesn't begin with an initial guess of zero. Since the solution is nested, intermediate grids have a non-zero initial guess, which is improved by the smoothing steps on intermediate grid levels.

The second example is the stream function-vorticity equations applied to the driven cavity problem. The ve-

	Lagrange		Hierarchic	
p	ψ	n	ψ	n
2	.07	.28	.53	.58
3	.07	.30	.49	.55
4	.17	.31	.52	.42
5	.17	.52	.51	.37
6	.69	.79	.49	.28

Table 1: Multilevel residual reduction factor for the augmented drift-diffusion problem

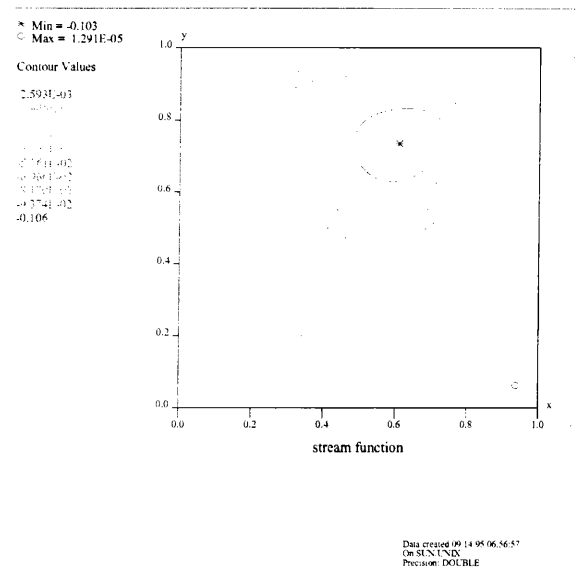

Figure 4: Stream function, $Re = 100$

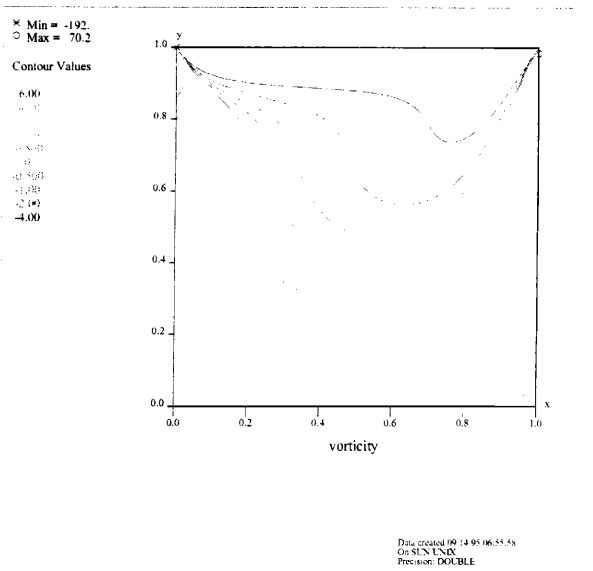

Figure 5: Vorticity, $Re = 100$

locity of the top of the cavity is normalized to one and the viscosity is chosen so that the Reynolds number of the flow is 100. Contour plots for the stream function and vorticity are shown in Figures 4 and 5. The solution was computed using a uniform 16×16 grid of 1024 quadratic elements, and a multilevel solver which used bilinear elements as the coarse grid.

The convergence history for the stream function linear system is shown in Figure 6. Notice that the convergence is smooth and that it asymptotes to a specific rate. The rate is determined by the largest block relaxation which gives convergence. This relaxation is lower for the higher degree elements, and hence convergence is slower. Experience has shown that at this Reynolds number continuation is unnecessary and more computationally expensive, as is nested iteration.

The residual reduction factor for the stream function

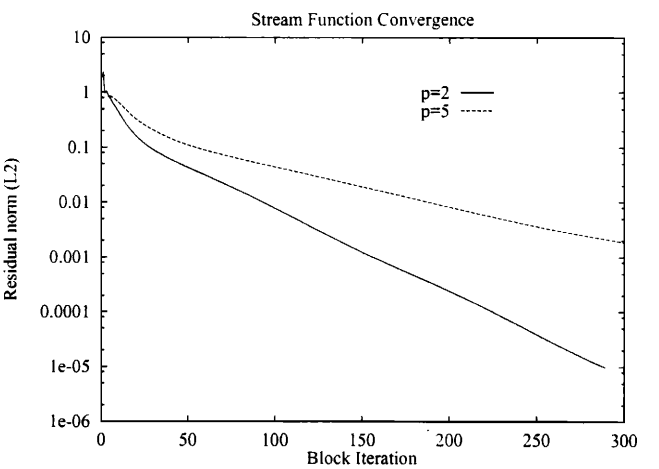

Figure 6: Stream function convergence with block iterations

	Lagrange		Hierarchic	
p	ψ	ζ	ψ	ζ
2	.20	.26	.23	.19
3	.17	.29	.19	.17
4	.38	.34	.19	.17
5	.49	.53	.10	.10
6	.70	.70	.07	.07

Table 2: Multilevel residual reduction factor for the stream function-vorticity problem at $Re = 100$

equation and the vorticity transport equation are shown in Table 2. Once again, the Lagrange basis exhibits a steady deterioration of the reduction factor with degree p for both equations. In this case the values are more similar since the problem is not as convection-dominated. The hierarchic basis performs well on this problem, with a gradual reduction of the factor with basis degree.

Figure 7 shows the speedup on the Intel iPSC/860 hypercube for the stream function-vorticity problem. The speedups are presented for a grid of 1024 quadratic elements and a grid of 64 quintic elements, with a Lagrange basis used in both instances. The processor decomposition is performed by ordering the elements in the square domain naturally and distributing them to the processors in order. *i.e.* the first $\frac{N_e}{N_p}$ elements go to the first processor and so on, with N_e the number of elements and N_p the number of processors. The speedup for less than 16 processors is very good, with a parallel efficiency of .83 for 8 processors. The deterioration of performance above this level is due to the smaller problem sizes on each processor, meaning the communication-computation ratio is larger. The speedups are similar since the $p = 5$ case has fewer grid points (smaller problem size). For the same number of elements as for the $p = 2$ case, the speedup will obviously be better.

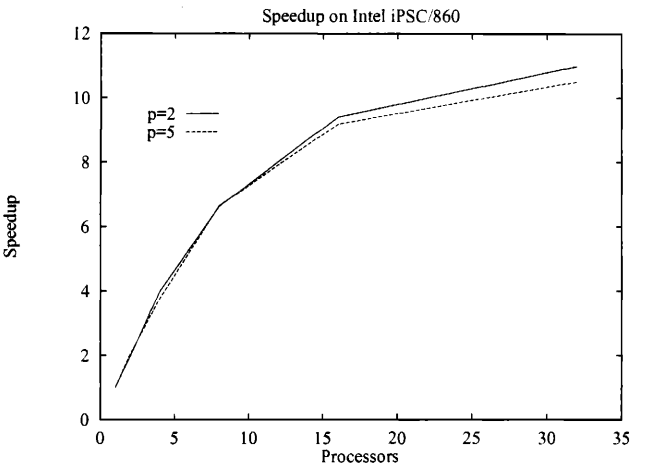

Figure 7: Speedup, stream function-vorticity, $Re = 100$, Lagrange basis

6 Conclusions

Spectral elements have proven to be a practical discretization technique for both the augmented drift-diffusion and stream function-vorticity problems. However, care must be taken in the augmented drift-diffusion problem with the placement of the elements. At least one element must lie in a transition region between different doping values, or the solution diverges due to Gibbs-type oscillations.

Spectral multilevel is an effective preconditioner for the high-p systems for both applications and both Lagrange and hierarchic bases. Adaptive calculation of the relaxation parameter ω allows use of simple point Jacobi relaxation as a multilevel smoother, even for high-p elements. Nested iteration has proven to be an efficient solution method for the augmented drift-diffusion problem, reducing the number of fine grid nonlinear iterations significantly. Extension of the multilevel scheme to include additional smoothers, especially gradient solvers, may further improve efficiency.

In an element-by-element scheme, multilevel methods are easily parallelizable, with very good speedups if the problem size per processor is moderate. Use of very high-degree elements, problems with multiple degrees-of-freedom per node, or three-dimensional problems may extend this parallel performance down to decompositions which have only one element per processor.

7 Acknowledgments

This research has been supported in part by the Texas Advanced Technology Program, ARPA and the National Science Foundation.

References

[1] I. Babuška, B.A. Szabo, and I.N. Katz. The p-version of the finite element method. *SIAM Jour. Num. Anal.*, 18(3):515–544, 1981.

[2] E. Barragy and G.F. Carey. A parallel element-by-element solution scheme. *Int. Jour. Num. Meth. Eng.*, 26:2367–2382, 1988.

[3] E. Barragy and G.F. Carey. Parallel-vector computations with high-p element-by-element methods. *Int. Jour. Comp. Math.*, 44:329–339, 1992.

[4] P.A. Blakey, X.L. Wang, C.M. Maziar, and P.A. Sandborn. A new technique for including velocity overshoot phenomena in conventional drift-diffusion simulators. In *Computational Electronics: Semiconductor Transport and Device Simulation*, pages 51–54, 1991.

[5] G.F. Carey. *Computational Grids: Generation, Refinement and Solution Strategies.* Wiley, 1996. In preparation.

[6] G.F. Carey, E. Barragy, R. McLay, and M. Sharma. Element-by-element vector and parallel computations. *Comm. App. Num. Meth.*, 4:299–307, 1988.

[7] G.F. Carey and B. Jiang. Element-by-element linear and nonlinear solution schemes. *Comm. App. Num. Meth.*, 2:145–153, 1986.

[8] M.B. Davis. *Parallel Multilevel Algorithms Applied to Iteratively Decoupled Transport Problems.* PhD thesis, University of Texas at Austin, 1996. In preparation.

[9] S. Foresti, G. Brussino, S. Hassanzadeh, and V. Sonnad. Multilevel solution method for the p-version of finite elements. *Comp. Phys. Comm.*, 53:349–355, 1989.

[10] H.K. Gummel. A self-consistent iterative scheme for one-dimensional steady state transistor calculations. *IEEE Trans. Elec. Dev.*, ED11:455–465, 1964.

[11] M. Lundstrom. *Modular Series on Solid State Devices, Volume X: Fundamentals of Carrier Transport.* Addison-Wesley, 1990.

[12] R.L. Panton. *Incompressible Flow.* Wiley, 1984.

[13] P. Roache. Finite difference methods for the steady-state Navier-Stokes equations. In *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*, volume 1, pages 139–145. Springer-Verlag, 1973.

[14] E.M. Ronquist and A.T. Patera. Spectral element multigrid. I. formulation and numerical results. *J. Sci. Comp.*, 2(4):389–406, 1987.

[15] S. Selberherr. *Analysis and Simulation of Semiconductor Devices.* Springer-Verlag, Wien, NY, 1984.

[16] C.M. Snowden. *Semiconductor Device Modelling.* Peter Peregrinus Ltd., London, 1988.