Chapter 9

Local Refinements and Grid Adaptation

Non-uniform resolution is needed in many, perhaps most, practical problems. Increasingly finer grids are needed near singularities, near non-smooth boundaries, at boundary layers, around captured shocks, etc., etc. Increasingly coarser grids are needed for exterior problems on unbounded domains. The multi-level FAS approach gives a convenient way to create non-uniform adaptable structures which are very flexible, allowing fast local refinements and local coordinate transformation, and whose equations are still solved with the usual multigrid efficiency. Moreover, the grid adaptation can naturally be governed by quantities supplied by the FAS multigrid processing, and it can naturally be integrated with the FMG algorithm to give increasingly better approximations to the true *differential* solution, at a fast, nearly optimal rate. These techniques, outlined below, are described in more detail in [Bra77a, §7, 8, 9], [Bra77b, §2, 3, 4], [Bra79b, §3, 4]. An application to three-dimensional transonic flows is described in [Bro82].

Another highly flexible discretization using a multigrid solver [Ban81] is based on a finite element approach, which makes the program simpler – especially for complicated structures, but the execution is less efficient. The techniques outlined below are also applicable to finite element formulations as in [Bra79a], [B+78, Lecture 4].

9.1 Non-uniformity organized by uniform grids

Our non-uniform discretization grows from the simple observation that the various grids (levels) used in usual multigrid algorithms need not all extend over the same domain. The domain covered by any grid may be only a proper part of the domains covered by coarser grids. Each grid h can be extended only over those subdomains where the desired meshsize is roughly less than $2h$. In such a structure, the effective meshsize at each

neighborhood will be that of the finest grid covering it: see Fig. 9.1.

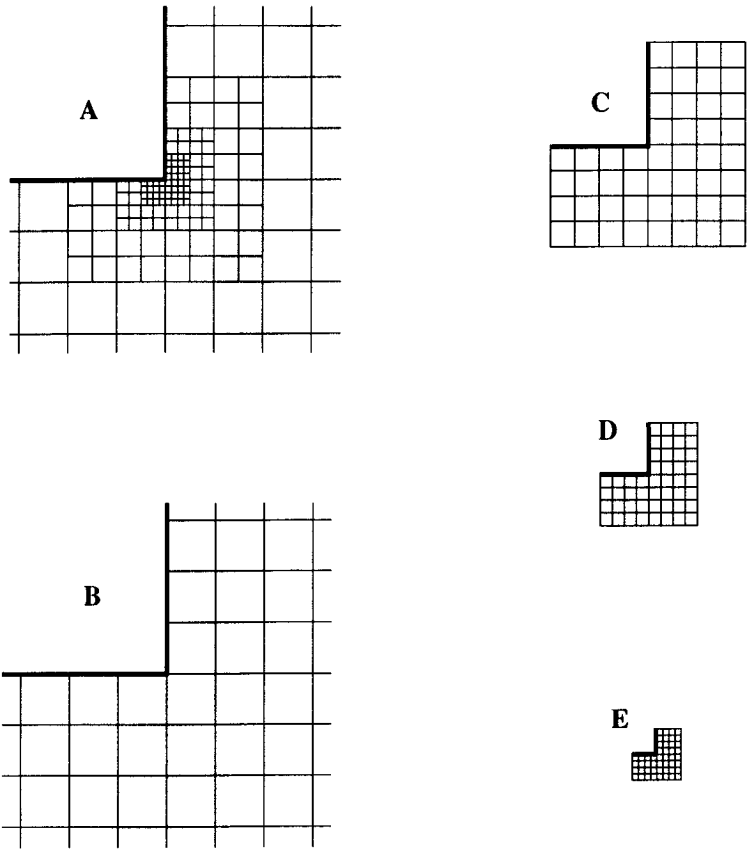


Figure 9.1. ***A piece of non-uniform grid (A) and the uniform levels it is made of (B,C,D,E).***
The heavy line shows a piece of the boundary, with a reentrant corner calling for the local refinements produced by the local patches (C,D,E).

This structure is very flexible, since local grid refinements (or coarsening) is done in terms of extending (or contracting) uniform grids, which is relatively easy and inexpensive to implement. A scheme named GRIDPACK [MUG84] has been developed for constructing, extending and contracting general uniform grids, together with many service routines for such grids, including efficient sweeping aids, interpolations, displays, treatment of boundaries and boundary data, etc. It is fully described in [BO83]. One of its advantages is the efficient storage: The amount of logical information (pointers) describing a uniform grid is proportional to the number of *strings* of points (contiguous sets of gridpoints on the same gridline), and is therefore usually small compared with the number of points on the grid.

Similarly, the amount of logical operations for sweeping over a grid is only proportional to the number of strings. Changing a grid is inexpensive too. One can easily add finer levels, or extend existing ones, thus effecting any desired refinements.

Moreover, this structure will at the same time provide a very efficient **solution process** to its difference equations, by using its levels also as in a multigrid solver. For this purpose the full approximation scheme must be used, because in parts of the domain not covered by the finer grid h, the coarser grid $H = 2h$ must certainly show the full solution, not just a correction. Indeed, the FAS approach naturally fit here: We use on grid H the equations $L^H u^H = \hat{f}^H$, where $\hat{f}^H$ is given by (8.5b) wherever $L^h u^h$ is well defined (i.e., in the interior of grid h), and $\hat{f}^H = f^H$ otherwise. In other words (cf. (8.8)-(8.9)), the fine-to-coarse correction τ_h^H is simply canceled wherever it is not defined. Applying an FMG algorithm with these structures and equations, we will get a solution that in each subdomain will satisfy its finest-grid equations, while at interior boundaries (of fine levels not covering the entire domain) the solution will automatically be as interpolated from the coarser grid. Note that the coarse-grid solution is influenced by the finer grid even at regions not covered by the latter, since the coarse grid *equations* are modified in the refined region.

In other words, a patch of the next finer level h can be thrown on any part of a given grid $H = 2h$, correcting there the latter's equations to the finer-grid accuracy. Moreover, several such patches may be thrown on the same grid. Some or all of the patches may later be discarded, but we can still retain their τ_h^H corrections in the grid H equations.

An important advantage is that difference equations are in this way defined on uniform grids only. Such difference equations on equidistant points are simple, inexpensive and standard, even for high-order approximations, whereas on general grids their weights would have to be calculated by lengthy calculations separately for each point. Relaxation sweeps are also made on uniform grids only. This simplifies the sweeping and is particularly useful for line relaxation schemes.

9.2 Anisotropic refinements

It is sometimes desired to have a grid which resolves a certain thin layer, such as a boundary layer. Very fine meshsizes are then needed in one direction, namely, across the layer, to resolve its thin width. Even when the required meshsize is extremely small, not many gridpoints are needed, since the layer is comparably thin, provided, of course, that fine meshsizes are used only in that one direction. We need therefore a structure for meshsizes which get finer in one direction only.

In case the thin layer is along coordinate hyperplane $\{x_j = \text{const.}\}$, this is easily done by **semi refinements**: Some levels H are refined by the next level h only in the j coordinate, $h_j = H_j/2$ whereas $h_i = H_i$ for

$i \neq j$. See Fig. 9.2. In fact, different patches may have different refinement directions. Thus, the set of all grids is arranged logically in a *tree*, each grid having a unique next-coarser grid, but possibly *several* next-finer grids, instead of the former linear ordering. All these grids are still uniform, and can still easily be handled by GRIDPACK.

Note that the next-finer grids of a given grid H may have some **overlap**. All that is needed in such cases is to get priority relations, to tell which correction τ_h^H applies at each point of grid H. Such priority relations are simply set by the order in which the corrections are transferred to grid H.

In case the thin layer is not along coordinate lines, the methods of the following sections could be used.

9.3 Local coordinate transformations

Another dimension of flexibility and versatility can be added to the above system by allowing each of the local patches to have its own set of local coordinates.

Near a boundary or an interface, for example, the most effective discretization is made in terms of coordinates in which the boundary (or interface) is a coordinate line. In such coordinates it is much easier to formulate high-order approximations near and on the boundary, and to introduce meshsizes which are much smaller across than along the boundary layer (§9.2); etc. In the interior, local patches of coordinates aligned with characteristic directions (along streamlines, for instance) can greatly reduce the cross-stream numerical viscosity (cf. §2.1), thus yield superior approximations to non-elliptic equations.

Each set of coordinates will generally be used with more than one grid, so that (a) local refinements, isotropic or anisotropic, in the manner described above, can be made within each set of coordinates; and (b) the multigrid processing retains its full efficiency by keeping the meshsize ratio between any grid and its next-coarser one properly bounded.

Since local refinement can be made within. each set of coordinates, the only purpose of the coordinate transformation is to provide the grid with the desired orientation, i.e., to have a given manifold (such as a piece of the boundary) coincide with a grid hyperplane. Since, furthermore, this needs to be done only locally, it can be obtained by a simple and standard transformation. For example, in two dimensional problems, let a curve be given in the general parametric form

$$x = x_0(s), \quad y = y_0(s), \quad (s_1 \leq s \leq s_2) \tag{9.1}$$

where s is the arclength, i.e.

$$x_0'(s)^2 + y_0'(s)^2 = 1. \tag{9.2}$$

To get a coordinate system (r, s) in which this curve coincides with the grid

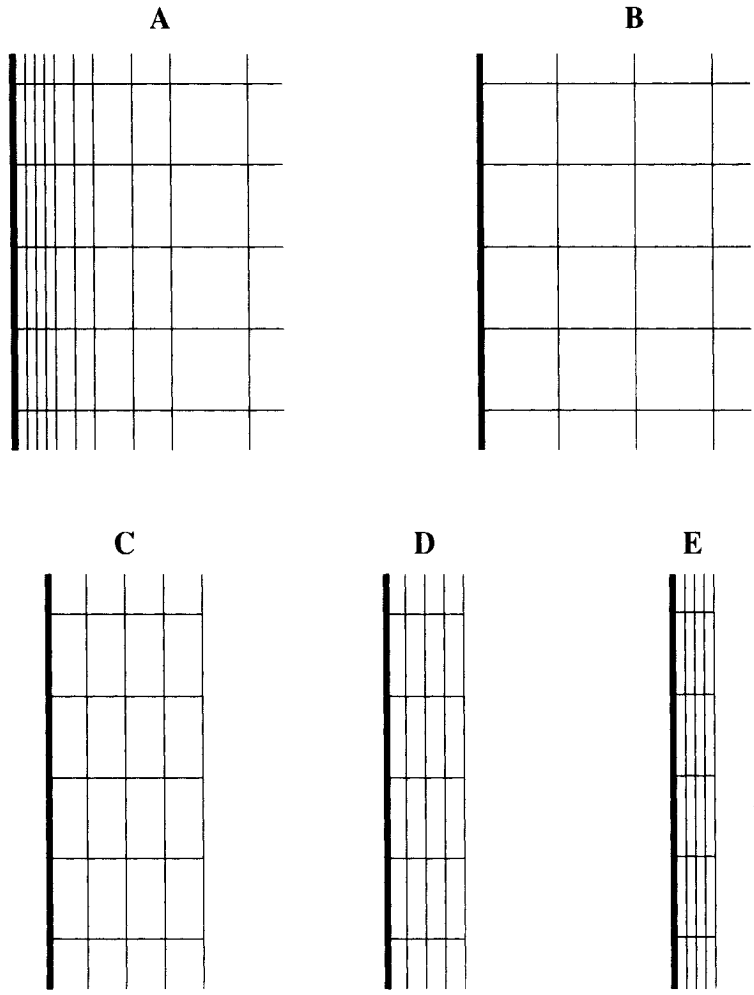


Figure 9.2. ***A piece of non-uniform, boundary-layer type grid (A) and the uniform rectangular subgrids it is made of (B,C,D,E).***

The meshsize in the local patches (C,D,E) is halved horizontally only.

line $\{r = 0\}$, we use the following transformation as standard:

$$x(r,s) = x_0(s) - ry_0'(s), \quad y(r,s) = y_0(s) + rx_0'(s). \tag{9.3}$$

Locally, near $r = 0$, this transformation is isometric (simple rotation).

The main advantage of this transformation is that it is fully characterized by the single-variable functions $x_0(s)$, $y_0(s)$. These functions, together with $x_0'(s)$, $y_0'(s)$ and $q(s) = x_0''/y_0' = -y_0''/x_0'$ can be stored as one-dimensional arrays, in terms of which efficient interpolation routines

from (x, y) grids to (r, s) grids, and vice versa, can be programmed once for all. The difference equations in (r, s) coordinates are also simple to write in terms of these stored arrays, since, by (9.2)–(9.3),

$$\frac{\partial}{\partial x} = -y_0' \frac{\partial}{\partial r} + \frac{x_0'}{1 + rq} \frac{\partial}{\partial s}, \quad \frac{\partial}{\partial y} = x_0' \frac{\partial}{\partial r} + \frac{y_0'}{1 + rq} \frac{\partial}{\partial s}. \tag{9.4}$$

A different kind of multi-level procedure using a combination of cartesian grids and grids curved along boundaries is described in [ST82, §11]. The main difference is that all levels, from coarsest to finest, are used there both for the cartesian and for the curved grids, and at each level the relaxation includes interpolations between the two types of grids, while the present approach is to regard the curved grids as a finer level which correct the finest cartesian grid near the boundary. The present approach is perhaps more economic and flexible, but it requires a (crude) approximation to the boundary conditions to be given on the cartesian grids, too.

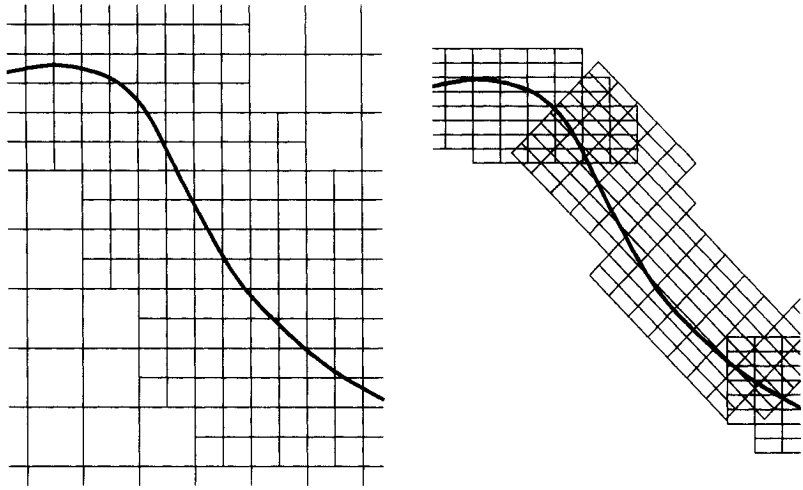
9.4 Sets of rotated cartesian grids

Another variant of this procedure is required in case the location of the thin layer (interface, shock, etc.) is not fully defined. For this purpose, each level will be a set of rotated cartesian grids, possibly overlapping. The finer the level, the finer (richer) is also the set of rotations. See Fig. 9.4. The self-adaptive criteria (see §9.5) can be employed to decide where to refine the set of rotations (together with refining the meshsize in one direction). Hence the scheme can capture discontinuities (thin layers), without defining them as such. The stronger the discontinuity, the better its resolution.

Since only rotated cartesian grids are needed in this scheme, the finite difference equations are as simple as ever. Hence this method is sometimes preferable even in cases where the location of the thin layer is known.

9.5 Self-adaptive techniques

The flexible organization and solution process, described above, facilitate the implementation of variable meshsize $h(x)$ and the employment of high and variable approximation order $p(x)$. How, then, are meshsizes and approximation orders to be chosen? Should boundary layers, for example, be resolved by the grid? What is their proper resolution? Should high-order approximations be used at such layers? How does one detect such layers automatically? In this section we survey a general multigrid framework for automatic selection of $h(x)$, $p(x)$ and other discretization parameters in a (nearly) optimal way. This system automatically resolves or avoids from resolving thin layers, depending on the goal of the computations, which can be stated through a simple function. (For more details see [Bra77a, §8], [Bra77b, §3]).



(A) (B)

Figure 9.3. ***Grid orientation around an interior thin layer.*** *The two coarsest levels (A) have the usual orientation* 0*. The next level (B) has four possible orientations:* $-\frac{\pi}{2}$, $-\frac{\pi}{4}$, 0 *and* $\frac{\pi}{4}$ *(only the last two are applied here). The next level (not shown) admits eight orientations,* $\frac{k\pi}{8}$, $-4 \leq k < 4$*; etc. The descendant refinements of a grid will always either have the same or one of the two nearest orientations (e.g., each descendant of the* $\frac{\pi}{4}$*-oriented grid at level B will either be* $\frac{\pi}{4}$*-,* $\frac{\pi}{8}$*-, or* $\frac{3\pi}{8}$*-oriented).*

As our directive for sensible discretization we consider the problem of minimizing a certain error estimator E subject to a given amount of solution work W (or minimizing W for a given E. Actually, the practical control quantity will often be neither E nor W, but their rate of exchange $\lambda = -dE/dW$, as shown below). This optimization problem should of course be taken quite loosely, since full optimization would require too much control work and would thus defeat its own purpose.

The error estimator E has generally the form

$$E = \int_{\Omega} G(x)\tau^h(x)dx, \tag{9.5}$$

where $\tau^h(x)$ is the local truncation error (cf. (8.11)) at x. $G(x) \geq 0$ is the error-weighting function. It should in principle be imposed by the user, thus defining his goal in solving the problem. In practice $G(x)$ serves as a convenient control. It is only the relative orders of magnitude of $G(x)$ at different points x that really matter, and therefore it can be chosen by some simple rules. For example, if it is desired to compute l-order derivatives of the solution up to the boundary then $G(x) \approx d_x^{m-1-l}$, where d is the

distance of x from the boundary, and m is the order of the differential equation.

The work functional W is roughly given by

$$W = \int_\Omega \frac{w(p(x))}{h(x)^d} dx, \tag{9.6}$$

where d is the dimension and h^{-d} is therefore the number of gridpoints per unit volume. (Replace h^d by $h_1 \cdots h_d$ in case of anisotropic grids.) $w = w(p)$ is the solution work per gridpoint. In multigrid processing, this work depends mainly on the approximation order (consistency order) $p(x)$. If the high-order techniques of §10 are used then usually $w(p) \approx w_0 p$, although sometimes $w(p) = O(p^3)$ for unusually high p (see §10.1).

Treating $h(x)$ as a continuous variable, the Euler equations of minimizing (9.5) for fixed (9.6) can be written as

$$G\frac{\partial \tau}{\partial h} - \lambda d w(p) h^{-d-1} = 0, \tag{9.7}$$

where λ is a constant (the Lagrange multiplier), representing the marginal rate of exchanging optimal accuracy for work: $\lambda = -dE/dW$.

In principle, once λ is specified, equation (9.7) determines, for each $x \in \Omega$, the local optimal values of $h(x)$, provided the truncation $\tau^h(x)$ as a function of h is fully known. In some problems the main behavior of $\tau^h(x)$ near singularities or in singular layers is known in advance by some asymptotic analysis so that approximate formulae for h(x) can a-priori be derived from (9.7). (Near source-type singularity (9.7) should be modified for that purpose, since $\tau^h(x)$ has an essential and singular sign reversal at the source [BB87].) More generally, however, equation (9.7) is coupled with, and should therefore be solved together with, the given differential equations. Except that (9.7) is solved to a cruder approximation. This is done in the following way:

In the FAS solution process we readily obtain the quantity τ_h^H. By (8.12) and (9.5), the quantity $-\Delta E(x) = G(x)\tau_h^H(x)$ can serve as an estimate for the decrease in E per unit volume owing to the refinement from H to h in the vicinity of x. By (9.6), this refinement requires the additional work (per unit volume) $\Delta W = w(p)h^{-d}(1 - 2^{-d})$. The local rate of exchanging accuracy for work is $Q = -\Delta E/\Delta W$. If Q is much larger than the control parameter λ, we say that the transition from H to h was highly profitable, and it pays to make a further such step, from h to $h/2$. So we will next establish grid $h/2$ in the neighborhood of x, as in any other neighborhood where there are points with $Q \gg \lambda$.

The computer work invested in the test is negligible compared with the solution work itself, since Q is calculated by just a couple of operations per point on the coarser grid H once per cycle. A similar test can be used to decide on changing the local approximation order $p(x)$ with τ_H^h being replaced by the p_1-to-p_0 defect correction (10.1) and correspondingly

$\Delta W = (w(p_1) - w(p_0))h^{-d}$. Or, if we treat p as a constant over the domain, but we like to optimize that constant, we can measure ΔE globally; i.e., measure directly the change in some quantity of interest (e.g., some functional of the solution we are most interested in), due to the transition from p_1 to p_0. Correspondingly, global ΔW will be used. Whether locally or globally, the order will be increased beyond p_1 if $-\Delta E/\Delta W \gg \lambda$. Other discretization parameters, such as the computational boundaries (when the physical domain is unbounded), or refinements in grid orientations (see §9.4), can be decided by similar tests all based on comparing some $-\Delta E/\Delta W$ to the exchange-rate parameter λ. How to control λ and coordinate it with the solution algorithm is discussed in the next section.

9.6 Exchange rate algorithms. λ-FMG

Near a severe singularity many levels of increasingly finer grids on increasingly narrower subdomains may be needed, and formed by the above criteria. If the usual FMG algorithm were applied to these levels, too much work would be spent, since too many passes on coarser grids would be made. Only when all grids cover the same domain is it true that the coarse-grid work is small compared to the next-finer grid work, since the latter deals with about 2^d as many points. This is no longer so when local refinements are used: Finer grids may include *less* points than some coarser grids. The amount of work in a usual FMG algorithm would therefore be much greater than proportional to the total number of gridpoints; (9.6) would not hold.

A better procedure is to decrease the accuracy-to-work exchange rate λ in a gradual sequence $\lambda_1 > \lambda_2 > \cdots$, and to use the solution obtained for λ_{j-1} as the first approximation to the solution on the grids formed for λ_j. In the absence of singularities, and for uniformly p-order approximations, this process with the ratio $\lambda_j/\lambda_{j-1} = 2^{-p-d}$ would yield the regular FMG algorithm, so generally it is called λ-FMG. It was tested [BB87] for the Poisson equation $\Delta u = f$, with severe singularity in f (e.g., $f = r^{-3.5}$, where r is the distance to a certain boundary point), or with 2π reentrant corner. The ratio $\lambda_j/\lambda_{j-1} = 1/16$ was used, for each λ_j the collection of local refinements was determined by (9.7), using the roughly known behavior of τ^h as function of the distance from the singularity. The 5-point Laplacian was employed, with red-black Gauss-Seidel relaxation. In the reentrant corner case, it was essential to use the local relaxation technique (§5.7). Results were invariably excellent: Algebraic errors $\|\tilde{u}^\lambda - u^\lambda\|$ smaller than the truncation errors $\|u^\lambda - u\|$ were obtained by one V(1,1) cycle per λ. More importantly, *the differential error* $E = \|\tilde{u}^\lambda - u\|$ *as function of the total work* W (measured as the total number of points traversed in all the relaxation sweeps) *behaves the same as in regular cases*: In regular problems with second-order approximations on two-dimensional uniform grids $E = O(h^2) = O(W^{-1})$, and the above experiments indeed clearly yield $E = O(W^{-1})$, for several orders of increase in W. This confirms

the validity not only of the λ-FMG algorithm, but also of the adaptation criterion (9.6). Had we used only uniform grids, the singularity would severely cripple $E(W)$; e.g., in the reentrant corner case it would yield $E = O(h) = O(W^{-\frac{1}{2}})$.

Switching criteria based on the exchange rate λ could also be used *in the multigrid cycles themselves.* Typically, the algorithm would continue relaxing on a given grid wherever $\hat{Q}(x) := -G(x)\Delta r(x)/\Delta W(x) > a_1\lambda$, where $-\Delta r(x)$ is the local decrease in the residual $|r(x)|$ per gridpoint per sweep and $\Delta W(x)$ is the corresponding work. Wherever uniformly over a substantial subdomain, $\hat{Q}(x) \lesssim a_1\lambda$ but $\overline{Q}(x) := -G(x)|r(x)|/\Delta W(x) > a_2\lambda$, a switch would be made to the next coarser grid. Wherever $\overline{Q}(x) < a_2\lambda$, a switch should in principle be made to the next finer level; if the current level is already locally the finest one, the processing for the current λ should terminate. Note that although for theoretical optimality different switches should thus in principle be required at different subdomains, for practical simplicity such a fragmentation of the algorithm should most often be avoided, except for the case of continuing relaxation locally (as mentioned in §5.7).

Ultimately, such exchange-rate criteria unify all the multigrid switching and adaptation criteria, integrating them into one algorithm, in which λ is gradually decreased. The process can be continued indefinitely, with increasingly finer levels created, globally and locally, in an almost optimal way. It can be terminated when either E or W or λ reach preassigned limits. The above techniques of anisotropic grids, local transformations and rotations, and adaptation of approximation orders can all be integrated into such an exchange-rate algorithm.