Applied Numerical Mathematics 13 (1993) 251–270
North-Holland

APNUM 441

Analysis of local uniform grid refinement

J.G. Verwer and R.A. Trompert
CWI, P.O. Box 4079, 1009 AB Amsterdam, Netherlands

Received 22 September 1992
Revised 13 November 1992
Accepted 16 November 1992

Abstract

Verwer, J.G. and R.A. Trompert, Analysis of local uniform grid refinement, Applied Numerical Mathematics 13 (1993) 251–270.

Numerical methods for time-dependent PDEs usually integrate on a fixed grid, a priori chosen for the whole time interval. Similar to a fixed stepsize, a fixed grid may be inefficient when solutions possess large local gradients. While most schemes can easily adapt the stepsize, as in genuine ODE and method-of-lines schemes, the question of how to automatically adapt the grid to rapid spatial transitions is much more involved. The subject of this paper is local uniform grid refinement (LUGR) for finite different methods. The idea of LUGR is to cover the spatial domain with nested, finer-and-finer, locally uniform subgrids. LUGR is applicable both to stationary and time-dependent problems. For time-dependent problems the local subgrids are adapted at discrete values of time to follow moving transitions. The aim of this paper is to discuss, for the class of finite difference methods under consideration, a general error analysis that shows the interplay between local truncation and interpolation errors. This analysis points the way to a theoretically optimal strategy for the local refinement, optimal in the sense that this strategy controls accumulation of interpolation errors and simultaneously strives for the spatial accuracy that would be obtained on the finest grid when used without adaptation. Attention is paid to both the stationary and time-dependent case, while for time-dependent problems the emphasis lies on combining LUGR with Runge–Kutta time stepping.

Keywords. Time-dependent partial differential equations; numerical analysis; adaptive-grid methods; local uniform grid refinement; error analysis.

1. Introduction

Numerical methods for time-dependent PDEs usually integrate on a fixed grid, a priori chosen for the whole time interval. Similar to a fixed temporal stepsize, a fixed grid may be disadvantageous when solutions possess large local gradients. However, while most schemes can easily adapt the stepsize by local error control, as in genuine ODE applications and in method-of-lines schemes, the question of how to automatically adapt the grid to the spatial solution behaviour is much more involved.

Correspondence to: J.G. Verwer, CWI, P.O. Box 4079, 1009 AB Amsterdam, Netherlands.

For time-dependent PDEs we distinguish two main categories of adaptive-grid methods, viz. dynamic and static ones. While dynamic (in time) methods adapt the grid in a continuous-time manner, like classical Lagrangian and moving-grid methods, static (in time) methods adapt the grid only at discrete times such that a time step is carried out on a fixed grid. Static methods bear a close resemblance to similar methods developed for stationary (elliptic boundary-value) problems. These methods again can be divided into two types. We mean the local-pointwise-refinement methods, where truly nonuniform grids arise, and the local-uniform-grid-refinement methods. Obviously, for each type of method one can envisage advantages and drawbacks. In this paper we refrain from discussing these and instead will focus on a particular class of static local-uniform-grid-refinement (LUGR) methods.

The basic idea of local uniform grid refinement [1,3,6] is to cover the spatial domain, say Ω, with nested, finer-and-finer, locally uniform subgrids so as to accurately resolve steep spatial transitions. In addition, for time-dependent problems these local subgrids are adapted in time to follow the moving transitions. In our approach, within a time step the integration then starts at the coarsest (base) grid covering Ω and during this time step the PDE is subsequently reintegrated on these nested subgrids. Loosely speaking, per time step interval an initial boundary value problem is solved on each local subgrid. For each local subgrid required initial values are defined by interpolation from the next coarser subgrid or are taken from a possibly existing subgrid from the previous time step interval. Likewise, boundary values required at internal boundaries are also interpolated from the next coarser subgrid. Having completed the integration on the finest level, the LUGR process is then repeated for the next time step by again starting at the base grid, but by using the most accurate solution available while also grid points already living at a certain level of refinement are used in the step continuation. So to say, to each grid level a numerical integration in time is attached on a discretely moving integration domain.

Both from the practical and theoretical point of view the first question to address is how to select the regions in Ω that ought to be refined. Albeit in essence LUGR is a straightforward adaptive-grid technique, the error analysis and resulting implementation issues readily become complicated. The aim of the present paper is to discuss, for the class of methods and problems under consideration, a general error analysis that shows the interplay between the local truncation and interpolation errors. In theory this analysis points the way to an optimal strategy for the local refinement, optimal in the sense that this strategy controls accumulation of interpolation errors and simultaneously strives for the spatial accuracy that would be obtained on the finest grid when used without any adaptation. Needless to say that this is a natural goal to be achieved when grid adaptivity is employed.

The class of methods and the error analysis were discussed before in [10,11,16]. In these papers only time-dependent problems are considered. However, the same ideas apply to stationary problems and because then the analysis is somewhat simpler to describe, we here first discuss the stationary case and then turn to time-dependent problems. We work with a general class of semi-discrete PDEs obtained through a finite different space discretization. Of course, we have in mind specific problems possessing steep local transitions. However, we restrict ourselves to problems having solutions with a sufficiently large number of derivatives and hence discard those with truly discontinuous solutions (like nonlinear hyperbolic problems with shocks).

2. LUGR: the stationary case

2.1. The semi-discrete problem

Consider a (system of) well-posed, real, abstract Cauchy problem(s)

$$\begin{aligned} &u_t = L(x, t, u), \qquad x \in \Omega, \quad t > 0, \\ &u(x, 0) = u^0(x), \end{aligned} \tag{2.1}$$

where L is a D-space dimensional PDE operator of at most second order, supplemented with appropriate boundary conditions. Assume that the boundary of Ω is locally parallel to coordinate axes and that (the interior of) Ω can be covered with a uniform base grid, here denoted by Ω_1. In addition, let Ω_k, $2 \leqslant k \leqslant r$, be uniform space grids such that Ω_k covers the whole of (the interior of) Ω while Ω_{k+1} is obtained from Ω_k by bisectional cellular refinement. On each Ω_k we introduce a semi-discrete counterpart of (2.1) denoted by

$$\frac{\mathrm{d}}{\mathrm{d}t} u_k(t) = F_k(t, u_k(t)) + \alpha_k(t), \quad t > 0, \qquad u_k(0) \text{ given}, \tag{2.2}$$

where $u_k(t)$ is the (pointwise) restriction of $u(x, t)$ and $\alpha_k(t)$ the spatial truncation error induced by the difference operator F_k. It is assumed that the semi-discrete boundary conditions have been worked into (2.2). Neglecting α_k yields the ODE system

$$\frac{\mathrm{d}}{\mathrm{d}t} U_k(t) = F_k(t, U_k(t)), \quad t > 0, \qquad U_k(0) = u_k(0), \tag{2.3}$$

where $1 \leqslant k \leqslant r$. We assume that r, the number of grid levels, is fixed in time (in [11] r is allowed to be variable). For the stationary case we remove the temporal variable t to obtain the similar stationary equations

$$F_k(u_k) + \alpha_k = 0, \quad k = 1, \ldots, r, \tag{2.2'}$$

$$F_k(U_k) = 0, \qquad k = 1, \ldots, r. \tag{2.3'}$$

We will formulate and analyse the LUGR method for the semi-discrete equations (2.3) and (2.3'), like in the method-of-lines approach. We tacitly assume second-order finite differencing, by imposing at interior points the standard central scheme and, if necessary, on boundaries the one-sided three-point scheme. The error analysis is valid for other finite difference discretizations as well.

2.2. The method formulation

For $1 \leqslant k \leqslant r$ let S_k be the vector space of grid functions on Ω_k. Let $I_k : S_k \to S_k$ denote the unit matrix. Introduce an interpolation operator $P_{k-1k} : S_{k-1} \to S_k$ from Ω_{k-1} to Ω_k. Let $D_k : S_k \to S_k$ denote a certain diagonal matrix with entries $(D_k)_{ii}$ either unity or zero. Suppose that $D_1 = I_1$. For the sequence of stationary problems (2.3') the multilevel LUGR method is then defined by

$$\begin{aligned} &F_1(U_1) = 0, \\ &D_k F_k(U_k) + (I_k - D_k)(U_k - P_{k-1k} U_{k-1} - b_k) = 0, \quad k = 2, \ldots, r. \end{aligned} \tag{2.4}$$

The second formula is to be interpreted as a difference-interpolation equation to be solved on the grids Ω_k in the order $k = 2, 3, \ldots$. The matrix D_k is supposed to be known prior to the computation at grid Ω_k. These diagonal matrices define the local subgrids upon which the actual finite difference calculations are carried out. Specifically, for nodes in the interior of the local subgrids $(D_k)_{ii} = 1$, while for all other nodes where interpolation is carried out $(D_k)_{ii} = 0$. The actual definition of D_k, and hence the actual selection of the local subgrids, is made by the refinement strategy. The nesting property of the local subgrids is also induced by this strategy and cannot be recovered from the above formulation, as it is hidden in the definition of D_k. In the analysis the matrices D_k play an important role. Note that if we substitute $D_k = I_k$ for all k, we recover the usual single grid finite difference scheme at all specified grids.

For $2 \leqslant k \leqslant r$ an equivalent formulation is

$$(I_k - D_k)(U_k - P_{k-1k}U_{k-1} - b_k) = 0, \tag{2.5a}$$

$$D_k F_k(U_k) = 0, \tag{2.5b}$$

which separates the interpolation part (2.5a). In this part we have included a grid function b_k which serves to collect certain terms possibly emanating from the physical boundary conditions. We include b_k since in (2.2) we have eliminated the semi-discrete unknowns at boundaries. Its presence is purely formal and for the analysis below this term plays no role. The choice of interpolant is in principle still free, although we advocate higher-order interpolation, e.g. fourth-order Lagrangian. Formula (2.5b) represents the finite difference computation over the local subgrid in use, which is carried out after the interpolation (2.5a). Note that (2.5b) is coupled to (2.5a), since the function evaluation in (2.5b) calls for solution components of U_k living at grid interfaces (internal boundaries) through the coupling in the finite difference grid. These grid interface components are defined by (2.5a).

At this stage we should emphasize that (2.4) defines approximations in the spaces S_k whose associated grids Ω_k cover the entire domain Ω. Of course, in practice we only execute (2.5b) at nodes for which the associated entry of D_k equals one and interpolate only at level k nodes where needed in reality, rather than over the whole of the grid Ω_k. For the time being, however, we act as if we truly work in S_k and will futher comment on the (restricted) interpolation in Section 2.4.

2.3. The refinement condition

An essential feature of any LUGR method is that a grid point will never be part of a future local subgrid at a higher refinement level once it gets outside the current local subgrid. This naturally implies that the accuracy at a grid point which is allowed to leave the local subgrid must satisfy a rather stringent test. We now present a general refinement condition defining such a test. In theory this condition guarantees the spatial accuracy that would be obtained on the finest grid when used without any adaptation, up to a grid-independent constant.

We rewrite (2.4) as

$$-D_k F_k(U_k) + (I_k - D_k)(U_k - P_{k-1k}U_{k-1} - b_k) = 0, \quad k = 1, \ldots, r, \tag{2.6}$$

and note that for $k = 1$ the interpolation part is auxiliary since $D_1 = I_1$ (on the base grid Ω_1 there is no interpolation). Substitution of the PDE solution u_k yields

$$-D_k F_k(u_k) + (I_k - D_k)(u_k - P_{k-1k}u_{k-1} - b_k) = \delta_k, \quad k = 1, \ldots, r, \tag{2.7}$$

where the defect δ_k is composed of the spatial truncation error α_k and the interpolation error γ_k as follows,

$$\begin{aligned} &\delta_k = D_k\alpha_k + (I_k - D_k)\gamma_k, \\ &\alpha_k = -F_k(u_k), \qquad \gamma_k = u_k - P_{k-1k}u_{k-1} - b_k. \end{aligned} \tag{2.8}$$

The minus sign in front of $D_kF_k(U_k)$ has been introduced to get $D_k\alpha_k$ in (2.8), rather than $-D_k\alpha_k$. Note that this is allowed in view of (2.5b). If we subtract (2.6) from (2.7), then the global error $\psi_k = u_k - U_k$ in the space S_k is seen to satisfy

$$Z_k\psi_k = (I_k - D_k)P_{k-1k}\psi_{k-1} + \delta_k, \tag{2.9a}$$

$$Z_k = -D_kM_k + I_k - D_k, \tag{2.9b}$$

where M_k is the Jacobian matrix

$$M_k = \int_0^1 [\partial F_k(\theta u_k + (1-\theta)U_k)/\partial U]\, \mathrm{d}\theta, \tag{2.10}$$

which results from applying the mean-value theorem for vector functions.

The refinement condition is now argued as follows. First we rewrite (2.9) to

$$Z_k\psi_k = D_k\alpha_k + (I_k - D_k)\rho_k, \quad \rho_k = \gamma_k + P_{k-1k}\psi_{k-1}, \tag{2.11}$$

and observe that we have two parts in this equation. The first part

$$-D_kM_k\psi_k = D_k\alpha_k \tag{2.12}$$

is defined on the interior of the local subgrid. Here the error is determined by the usual spatial truncation error $D_k\alpha_k$ and the error components on the subgrid interface. These latter components are defined in the second part

$$(I_k - D_k)\psi_k = (I_k - D_k)\rho_k, \tag{2.13}$$

which is defined on the interface and outside the local subgrid. All error components in (2.13) are determined by ρ_k. This error contains the interpolation error γ_k and the prolongation of the previous global error ψ_{k-1}, according to (2.11). In the ideal situation $(I_k - D_k)\rho_k$ is not larger than $D_k\alpha_k$, because then the maximum of the error ψ_k will more or less be determined by the truncation error expression (2.12). This means that the maximum error is nearly equal to the maximum error found if no grid adaptivity was employed. The refinement condition derived below indeed achieves that on grid Ω_r the parasitic error $(I_r - D_r)\rho_r$ will not be larger than $D_r\alpha_r$.

The derivation goes as follows (cf. [10, Section 6]). The error ρ_r is first brought in the form

$$\begin{aligned} &\rho_r = \lambda_r + P_{r-1r}\sum_{k=2}^{r-1}\left\{\prod_{i=r-1}^{k+1} X_i\right\}(Z_k)^{-1}(I_k - D_k)\lambda_k, \\ &X_i = (Z_i)^{-1}(I_i - D_i)P_{i-1i}, \end{aligned} \tag{2.14}$$

where

$$\lambda_k = \gamma_k + P_{k-1k}(Z_{k-1})^{-1}D_{k-1}\alpha_{k-1}. \tag{2.15}$$

Let $\|\cdot\|$ be the maximum norm (in all spaces S_k) and introduce the norm bounds

$$\|(Z_k)^{-1}\| \leqslant C_{\mathrm{FD}}, \qquad \|P_{k-1k}\| \leqslant C_1, \tag{2.16}$$

where both constants are independent of the grids. Note that for linear interpolation $C_I = 1$, while for higher-order interpolants $C_I > 1$. The constant C_{FD} will be very close to the bound that naturally exist for all $(M_k)^{-1}$, since in (2.9b) a part of the rows of M_k remain unchanged and the remaining ones are replaced by unit rows. Inserting the bounds (2.16) in (2.14) yields the inequality

$$\|(Z_r)^{-1}(I_r - D_r)\rho_r\| \leqslant C \max_{2\leqslant k\leqslant r} \|(I_k - D_k)\lambda_k\|, \quad C = C_{FD} = \sum_{k=2}^{r-1} (C_{FD}C_I)^{r-k}. \tag{2.17}$$

Hence, if we succeed in imposing, for two arbitrary constants c and C^*, the condition

$$\|(I_k - D_k)\lambda_k\| \leqslant \frac{c}{C^*}\|(Z_r)^{-1}D_r\alpha_r\|, \quad k = 2,\ldots,r, \tag{2.18}$$

then insertion of (2.17) in (2.11), for $k = r$, implies

$$\|\psi_r\| \leqslant \left(1 + c\frac{C}{C^*}\right)\|(Z_r)^{-1}D_r\alpha_r\| \leqslant \left(1 + c\frac{C}{C^*}\right)C_{FD}\|\alpha_r\|. \tag{2.19}$$

In conclusion, if we manage to satisfy (2.18), then the adaptive-grid global error ψ_r satisfies the same bound as the global error $\psi_r := (-M_r)^{-1}\alpha_r$ that exists on the globally uniform grid Ω_r, up to a grid-independent constant. In other words, asymptotically spoken, (2.18) removes the contribution from interpolation and the use of coarser grids, and thus achieves that there will be no true loss in spatial accuracy by using coarser grids.

Condition (2.18) is the refinement condition referred to above. Note that (2.18) is a result of first bounding terms in (2.14) and then adding norm bounds, resulting in (2.17). Hence (2.18) will tend to be a conservative condition and (2.19) a conservative estimate. The choice of the two parameters c and C^* plays a role too, of course. Their use here is argued as follows. If C would be known, or an upper bound for it, it is natural to put $C^* = C$, so that in (2.19) the extra error bound term due to the interpolation and the use of coarse grids then becomes $cC_{FD}\|\alpha_r\|$. This extra error bound term can then be controlled by the parameter c. Generally C is unknown. In the remained we therefore put $C^* = r - 1$, which is motivated by the observation that, if C would be $\leqslant r - 1$, then the extra error bound term $cC_{FD}\|\alpha_r\|$ applies. We have $C \leqslant r - 1$ in reality if we would interpolate linearly and $C_{FD} \leqslant 1$. In applications we advocate higher-order interpolation, though. As long as the interpolation takes place in low error regions, the influence of a bound $C_I > 1$ is expected to be minor. The stability of the discretization, and thus the size of C_{FD}, is of greater importance.

For actual implementation we use the refinement condition

$$\|(I_k - D_k)\zeta_k\| \leqslant \frac{c}{C^*}\|(Z_r)^{-1}D_r\alpha_r\|, \qquad C^* = r - 1, \quad k = 2,\ldots,r, \tag{2.20}$$

where, componentswise, the grid function ζ_k is defined by

$$(\zeta_k)_i = |(\gamma_k)_i| + \left|\left(P_{k-1k}(Z_{k-1})^{-1}D_{k-1}\alpha_{k-1}\right)_i\right|. \tag{2.21}$$

Condition (2.20) implies (2.18). Replacing λ_k by ζ_k thus provides an extra safety margin.

In theory (2.20) can always be satisfied by adjusting the refinement matrices D_k. An extreme solution would be $D_k = I_k$, implying no local refinement. Note that this also results for $c = 0$.

Obviously the size of the local subgrids is partly determined by the parameter c which is still free. A natural choice is $c \simeq 1$, which is assumed henceforth. The larger c is chosen, the easier it will be to satisfy (2.20), but also the larger the estimated error contribution from interpolation and coarser grids will be that arises in the upperbound

$$\|\psi_r\| \leqslant \left(1 + c\frac{C}{r-1}\right)\|(Z_r)^{-1} D_r \alpha_r\| \leqslant \left(1 + c\frac{C}{r-1}\right) C_{\mathrm{FD}} \|\alpha_r\|. \tag{2.19'}$$

Finally, if the order of the interpolation error is larger than the order of the truncation error and the interpolation takes place in low error regions, then from (2.21) we see that the interpolation error contribution will play a minor role, as is to be expected. This matter, however, is more complicated for time-dependent problems since there the temporal stepsize plays a role too. We will encounter this later in the paper.

2.4. Implementation aspects of the refinement condition

In this section we pay attention to implementation issues for (2.20) (cf. [10, Section 6]). Note that if we are able to efficiently solve the general nonlinear finite difference system (2.3') on a single fine grid, then we can also solve each of the related systems (2.4). Hence typical solving issues are not discussed here (see also [4]), neither do we discuss the data structure (see [12]). For (2.20) the main issues are discussed in the following subsections.

2.4.1. The restricted interpolation and nesting property

Suppose we have computed U_1 on the global grid Ω_1. Then at all points of Ω_2 the numerical version of (2.20) is checked for determining the local subgrid of level 2. Let ω_2 denote this local subgrid. If at the ith point $|(\zeta_2)_i|$ is not small enough, then this ith point is placed into ω_2 ($(D_2)_{ii} = 1$) and otherwise outside ($(D_2)_{ii} = 0$). The numerical condition (2.20) is now satisfied at level 2 so we can compute U_2 from (2.5). We then first execute the interpolation (2.5a), but only at the internal boundary points of ω_2 to omit redundant work. Then we solve the finite difference system (2.5b) at level 2. Because we have omitted the full interpolation (2.5a), we have not computed the whole of U_2 and, so to say, have left the computational space S_2. If a third level is needed, then the next step is to determine ω_3 and we proceed with the numerical condition (2.20) in the same way as at level 2. However, we check it only at the intersection of Ω_3 and ω_2 and thus omit that part of the space domain which at level 2 was found sufficiently accurately resolved. Consequently, ω_3 will then be nested into ω_2. Higher levels are dealt with likewise.

The point we wish to make here is that by not interpolating everywhere and checking (2.20) only at the intersection of Ω_k and ω_{k-1}, we are no longer in S_k and thus deviate from the theory. This deviation, however, is not essential and our claim is that would we stay in S_k and check (2.20) on the whole of Ω_k, local subgrids ω_k would be found equal or very close. To see this we consider (2.20) and (2.21) again for $k = 2$. For $k = 2$ the check has been carried out on Ω_2. Hence, due to the absolute values, outside ω_2 we have

$$|(\gamma_2)_i| \leqslant \frac{c}{r-1}\|(Z_r)^{-1} D_r \alpha_r\| \tag{2.22}$$

(for the analytic expressions their numerical counterparts are substituted). Now consider $k = 3$.

For $k = 3$ the deviation comes from not checking (2.20) outside ω_2. However, there the second term of (2.21) is zero for $k = 3$, just by definition. So there would be no deviation if also $|(\gamma_3)_i|$ satisfies (2.22) outside ω_2. This is a reasonable assumption in view of the fact that the interpolation error on a fine grid may be expected to be smaller than on a coarse grid. We repeat this argument for $k = 4$, etc.

2.4.2. The estimation of the interpolation and truncation error in (2.21)

Because we work at uniform grids this is well feasible. The expression for γ_k is explicitly known once the interpolant is chosen. The truncation error $\alpha_{k-1} = -F_k(u_{k-1})$ is estimated from $-(F_k^*(u_{k-1}) - F_k(u_{k-1}))$, where F_k^* is a second, higher-order finite difference operator. We use a F_k^* of order four. Note that we tacitly assume that substitution of the numerical solution into the theoretical error expressions is allowed in the sense that the numerical estimate is also asymptotically correct with the same order of accuracy. Further, for linear problems the Jacobian matrix Z_{k-1} is always available. In case of nonlinear problems the approximate Jacobian appearing in the iterative Newton process is used. The greater part of the work thus is an additional linear solve to compute the global error expression $(Z_{k-1})^{-1} D_{k-1} \alpha_{k-1}$.

2.4.3. The estimation of the global error expression at level r in (2.20)

Here we exploit the asymptotics of the local truncation error. Let p be the order of consistency of α_k. We then use the asymptotic relation

$$\left\| (Z_r)^{-1} D_r \alpha_r \right\| \simeq 2^{-p(r-k)} \left\| (Z_k)^{-1} D_k \alpha_k \right\|, \quad r \geqslant k+1, \tag{2.23}$$

and reuse this every time k is increased to improve the reliability of the estimation for increasing k.

2.4.4. Accuracy in the estimation

The LUGR method is designed to solve PDEs with steep solutions. Yet the refinement condition (2.20) with the various order relations, like (2.23), are based on asymptotics, which means that they can only be accurate if the solution is sufficiently smooth on the grid in use. At first sight this constitutes a contradiction, because initially error estimations on coarse grids are carried out. However, to our experience the estimation procedure works well in practice and, so far, the results are in very good agreement with the theory. We believe the reason is that even when the estimation is not so accurate, it will still indicate where the error is large and where not. If the error is considered too large at a certain part of the space domain, the finite difference computation is simply redone there. So there is a built-in safety in the algorithm in the sense that it locally refines until the solution becomes smooth relative to the local grid in use. Further, mostly the estimation will be in the same order of magnitude as the true error and then the refinement strategy based on (2.20) and (2.23) should work without any serious difficulty.

2.5. Numerical illustration

To illustrate our local refinement condition (2.20), we have applied the LUGR scheme (2.4) to

$$u_{xx} + u_{yy} = f, \qquad \Omega = (0, 1) \times (0, 1), \tag{2.24}$$

Table 1
Results for problem (2.24)

1	2	3	4	5	6	7
2	1	$1.14_{10^{-1}}$	$1.07_{10^{-1}}$	441		
	2	$2.58_{10^{-2}}$	–	299	1681	$2.58_{10^{-2}}$
3	1	$1.14_{10^{-1}}$	$1.07_{10^{-1}}$	441		
	2	$2.57_{10^{-2}}$	$2.48_{10^{-2}}$	705		
	3	$5.66_{10^{-3}}$	–	625	6561	$6.28_{10^{-3}}$
4	1	$1.14_{10^{-1}}$	$1.07_{10^{-1}}$	441		
	2	$2.58_{10^{-2}}$	$2.48_{10^{-2}}$	1455		
	3	$6.33_{10^{-3}}$	$6.21_{10^{-3}}$	1431		
	4	$1.31_{10^{-3}}$	–	2161	25921	$1.56_{10^{-3}}$
5	1	$1.14_{10^{-1}}$	$1.07_{10^{-1}}$	441		
	2	$2.58_{10^{-2}}$	$2.48_{10^{-2}}$	1681		
	3	$6.29_{10^{-3}}$	$6.23_{10^{-3}}$	2559		
	4	$1.58_{10^{-3}}$	$1.55_{10^{-3}}$	4737		
	5	$3.52_{10^{-4}}$	–	7281	103041	$3.90_{10^{-4}}$

Column 1: r = number of levels.
Column 2: k = the refinement level.
Column 3: Maximum of true global error on subgrid.
Column 4: Maximum of numerical estimation of globar error on subgrid.
Column 5: Number of points of subgrid.
Column 6: Number of points of uniform grid.
Column 7: Maximum of true global error on uniform grid.

with exact solution $u(x, y) = \exp(-A[(x-0.5)^2 + (y-0.75)^2])$ and Dirichlet boundary conditions. Although not really very difficult, this linear elliptic model problem provides a nice example to illustrate the LUGR method and its anticipated error and convergence behaviour. Note that the solution is strongly peaked for A large. In our computation $A = 160$. We have used fourth-order Lagrangian interpolation and recall that the finite difference operators F_k and F_k^* are of accuracy order two and four, respectively.

Table 1 shows results for four experiments using, respectively, two, three, four, and five levels of refinement. The coarsest grid is always 21×21, hence the coarsest and finest meshwidths used are $1/20$ and $1/320$. The free parameter c, left in the refinement condition (2.20), has been put equal to 2. Table 1 shows also results for four additional computations carried out on, respectively, the uniform 41×41, 81×81, 161×161, and 321×321 grid. Figure 1 shows the five-level grid and computed solution on this grid.

From the table we conclude that for the linear elliptic model problem the LUGR scheme works as predicted by the theory. When adding a new level, the true global error decreases by a factor of 4, approximately, reflecting the second-order convergence of the difference approximation. Note that the numerical global error estimation is very accurate, even on the coarse grids. Also note that the local subgrids at levels $k < r$ grow for increasing r, according to the theory. The correspondence with the accuracy found on the uniform grids is striking. Such an excellent correspondence is of course attractive, but it also indicates that the refinement condition indeed may work out too conservative, as already indicated above. This reduces

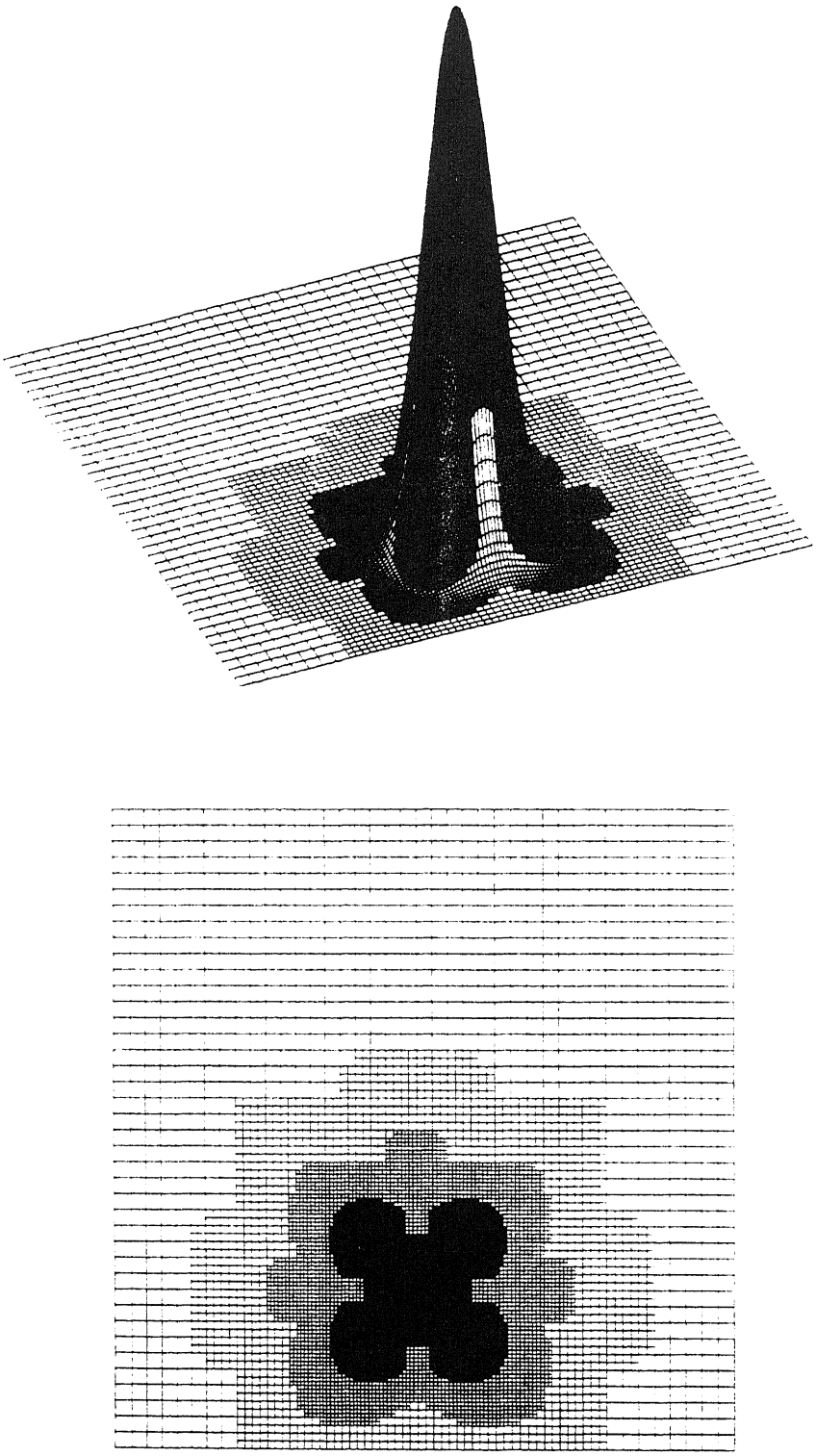

Fig. 1. Problem (2.24): grid and solution for the five-level computation.

efficiency, since it means that the local subgrids have been chosen unnecessarily large, due to a too small parameter c. We have repeated the five-level experiment for $c = 3$, 4, and 5 and found the following maximal global error on the fifth level: $3.65_{10^{-4}}$, $4.19_{10^{-4}}$, and $4.20_{10^{-4}}$ respectively. Note that these are still very close to the maximal global error on the 321×321 grid.

The systems of linear equations, arising in (2.5b) and (2.21), are solved in sufficiently high accuracy using a BI-CGSTAB iteration method with ILU preconditioning (a code taken from NETLIB, see [4] for details, authors, and references). For this iterative approach the LUGR technique offers an advantage due to the nesting of the higher-level subgrids, which implies decreasing condition numbers of the associated linear systems when compared with those found for the corresponding globally uniform grids. Hence, in general a more rapid convergence of the linear solver is experienced on the higher-level subgrids. The convergence is further accelerated by using as initial guess the interpolated solution from the previous level, like in straightforward nested iteration. A drawback is that each level requires a new ILU decomposition. However, preconditioning is still very effective.

Our iteration strategy has not been thoroughly optimized, but works so far very satisfactorily (see also [4,13]). For example, in the five-level computation of Table 1 we counted, respectively, 11, 17, 19, 22, and 19 iterations to compute the numerical solution on the five subgrids and 10, 16, 19, and 22 iterations in the four global error estimations. The total effort for these nine iterations is to be compared with the iteration costs for the uniform 321×321 grid, plus the costs of the ILU decomposition. The number of iterations for the uniform 21×21, 41×41, 81×81, 161×161, and 321×321 grids are, respectively, 11, 21, 41, 61, and 82.

3. LUGR: the time-dependent case

We now turn our attention to time-dependent problems (2.3) and show how the stationary LUGR scheme (2.6) can be modified by attaching to each grid level a numerical integration in time on a discretely moving local subgrid. The approach here is that of a step-by-step one, by which we mean that the local refinement is carried out over all levels, time step after time step. Hence the time variable itself does not play a role in the local refinement.

3.1. The implicit and explicit Euler LUGR schemes

Let $\tau = t_n - t_{n-1}$ denote the stepsize and $R_{rk} : S_r \to S_k$ the natural pointwise restriction operator from Ω_r to Ω_k. The implicit Euler scheme then reads, $k = 1, \ldots, r$,

$$D_k^n\left[\left(U_k^n - R_{rk}U_r^{n-1}\right)/\tau - F_k(t_n, U_k^n)\right] + (I_k - D_k^n)[U_k^n - P_{k-1k}U_{k-1}^n - b_k^n] = 0, \quad (3.1)$$

where U_k^n is the numerical solution at time $t = t_n$ and grid Ω_k. This integration/interpolation scheme can be interpreted as to be obtained from the stationary scheme (2.6) by first replacing $-F_k(U_k)$ by the time-dependent expression $\mathrm{d}U_k(t)/\mathrm{d}t - F_k(t, U_k(t))$ and then invoking the first-order backward difference

$$\frac{\mathrm{d}}{\mathrm{d}t}U_k(t_n) \simeq \left(U_k^n - R_{rk}U_r^{n-1}\right)/\tau. \quad (3.2)$$

Taking $R_{rk}U_r^{n-1}$ instead of U_k^{n-1} means that for time stepping the finest grid solution from the past time level is used. Similar to (2.6), scheme (3.1) is applied for $k = 1, 2, \ldots,$ but now for each time step $t_{n-1} \to t_n$. The matrices D_k^n are adjusted per time step which means that the local subgrids, or integration domains, move in time in a discrete manner. Note that $D_1^n = I_1$, so that in (3.1) the interpolation part is to be ignored for $k = 1$. Also note that for $\tau \to \infty$ the stationary equation (2.6) is recovered.

In [10] we use the equivalent formula

$$U_k^n = D_k^n\big[R_{rk}U_r^{n-1} + \tau F_k(t_n, U_k^n)\big] + (I_k - D_k^n)\big[P_{k-1k}U_{k-1}^n + b_k^n\big], \quad k = 1, \ldots, r, \tag{3.3}$$

which is obtained after multiplying (3.1) by τ and reordering terms. The explicit Euler LUGR scheme is now also immediately found to be

$$U_k^n = D_k^n\big[R_{rk}U_r^{n-1} + \tau F_k\big(t_{n-1}, R_{rk}U_r^{n-1}\big)\big] + (I_k - D_k^n)\big[P_{k-1k}U_{k-1}^n + b_k^n\big], \quad k = 1, \ldots, r. \tag{3.4}$$

In fact, the stationary LUGR approach can be coupled to any existing time stepping technique of interest to PDEs. To illustrate this, the next section is devoted to the class of Runge–Kutta LUGR methods [11] for which (3.3) and (3.4) are special cases. The discussion will again focus on the refinement condition.

Note that in our method description we follow the method of lines approach of first converting PDEs to continuous in time ODE systems through spatial discretization. Starting from the alternative Rothe method, also called the method of discretization in time [9], Deuflhard and his coworkers developed interesting finite element type adaptive multilevel methods (see [5,7] and the references therein).

3.2. *The Runge–Kutta LUGR schemes*

For ease of presentation we use the direct product notation where augmented variables and operators will be underlined. For ODE systems (2.3) the general s-stage RK formula thus reads

$$\underline{U}_k^n = e \otimes U_k^{n-1} + \tau(A \otimes I)\underline{F}_k(t_{n-1}, \underline{U}_k^n), \tag{3.5}$$

where we have included the output stage in the formula for the intermediate stages. Hence, A is here the Butcher matrix (a_{ij}) of order $s + 1$, rather than of order s, whose $(s + 1)$st column contains zeros and whose remaining entries in the $(s + 1)$st row are just the weights of the final output stage. Scheme (3.5) lives in the augmented space $\underline{S}_k = (S_k)^{s+1}$. The augmented vector $\underline{U}_k^n$ is composed of the intermediate stage vectors

$$U_k^{(i)} = U_k^{n-1} + \tau \sum_{j=1}^{s} a_{ij}F_k\big(t_{n-1} + c_j\tau, U_k^{(j)}\big), \quad 1 \leqslant i \leqslant s, \tag{3.6}$$

and the output vector

$$U_k^n = U_k^{n-1} + \tau \sum_{j=1}^{s} a_{s+1j}F_k\big(t_{n-1} + c_j\tau, U_k^{(j)}\big). \tag{3.7}$$

For $1 \leqslant i \leqslant s$ the ith component vector of $\underline{F}_k$ is $F_k(t_{n-1} + c_i\tau, U_k^{(i)})$, while the $(s+1)$st component vector of $\underline{F}_k$ is the zero vector. We assume the usual condition $c_i = a_{i1} + \cdots + a_{is}$.

Analogous to the implicit Euler scheme (3.3), we now introduce the general RK-LUGR scheme

$$\begin{aligned} \underline{U}_k^n = \underline{D}_k^n\big[\underline{R}_{rk}\big(e \otimes U_r^{n-1}\big) + \tau(A \otimes I_k)\underline{F}_k(t_{n-1}, \underline{U}_k^n)\big] \\ + (\underline{I}_k - \underline{D}_k^n)\big[\underline{P}_{k-1k}\underline{U}_{k-1}^n + \underline{b}_k^n\big], \end{aligned} \tag{3.8}$$

where $1 \leqslant k \leqslant r$ and

$$\begin{aligned} &\underline{I}_k = \operatorname{diag}(I_k), \qquad \underline{R}_{rk} = \operatorname{diag}(R_{rk}), \qquad \underline{P}_{k-1k} = \operatorname{diag}(P_{k-1k}), \\ &\underline{D}_k^n = \begin{cases} \operatorname{diag}(I_k, D_k^n, \ldots, D_k^n), & \text{if } a_{1j} = 0,\ 1 \leqslant j \leqslant s, \\ \operatorname{diag}(D_k^n, D_k^n, \ldots, D_k^n), & \text{otherwise.} \end{cases} \end{aligned} \tag{3.9}$$

An important observation is that the multistage nature of the RK method is carried over. This means that, completely similar to the Euler case, grid interface components at intermediate stages are defined by the scheme itself. Note that (3.8) contains the class of all explicit and implicit one-step RK methods. The definition of $\underline{D}_k^n$ avoids interpolation at the first stage if $a_{1j} = 0$, $1 \leqslant j \leqslant s$, which holds for all explicit methods, but e.g. also for the implicit trapezoidal rule. Hence in these cases the first (trivial) stage value is assigned to the restriction of the finest grid approximation $R_{rk}U_r^{n-1}$. The Euler methods (3.3) and (3.4) are obtained by elaborating (3.8), respectively, for the Butcher matrices

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad A = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}. \tag{3.10}$$

Finally we note that levelwise (3.8) takes the same temporal stepsize. For explicit RK methods this naturally leads to a too small stepsize on the coarser grids since for these methods the stepsize is readily constrained by stability dictated by the finest grid. For implicit methods this stability constraint normally does not exist. Of course, although the coarse grid computations are the cheapest ones, using the same stepsize leads to extra overhead. Taking a larger stepsize on coarser grids, as in [12], may be advantageous. It should be backed, however, by a solid error analysis.

3.3. The local space error for the RK-LUGR scheme

For simplicity of presentation we assume now that (2.3) is of the constant coefficient linear type

$$\frac{\mathrm{d}}{\mathrm{d}t}U_k(t) = M_kU_k(t) + f_k(t), \quad t > 0, \qquad U_k(0) = u_k(0). \tag{3.11}$$

Let $\underline{u}_r^n$ be the augmented vector composed of the true PDE solution vectors $u_r^{(i)} = u_r(t_{n-1} + c_i\tau)$, $1 \leqslant i \leqslant s+1$, where $u_r^{(s+1)} = u_r^n$. Introduce the global error $\underline{e}_r^n = \underline{u}_r^n - \underline{U}_r^n$ committed by (3.8) in the augmented space $\underline{S}_r$. An elementary calculation [11] shows that $\underline{e}_r^n$ is defined by a general error scheme

$$\underline{e}_r^n = \underline{G}_r^n\big(e \otimes e_r^{n-1}\big) + \underline{\psi}_{r\mathrm{t}}^n + \underline{\psi}_{r\mathrm{s}}^n, \quad n = 1, 2, \ldots, \tag{3.12}$$

where the amplification operator $\underline{G}_r^n$ and the local error $\underline{\psi}_{rt}^n + \underline{\psi}_{rs}^n$ are defined by recursions connecting the grid levels. The part $\underline{\psi}_{rt}^n$ contains all temporal error contributions from the RK method, while $\underline{\psi}_{rs}^n$ is the local error part composed of all existing spatial truncation and interpolation errors. In the remainder we will call $\underline{\psi}_{rs}^n$ the local space error (at level r). It is emphasized that this local space error is the counterpart of the stationary (global space) error ψ_r featuring in Section 2.3.

We now concentrate on $\underline{\psi}_{rs}^n$. In connection with local refinement this error is the most interesting one since it is $\underline{\psi}_{rs}^n$ that we wish to dictate the local refinement. We note in passing that both the stability and the temporal accuracy will not differ significantly from what we experience in the usual single-grid application, as long as interpolation takes place in low error regions. Hence a good refinement strategy which guarantees this is most desirable. In [11] it is shown that $\underline{\psi}_{rs}^n$ can be written in the form

$$\underline{\psi}_{rs}^n = (\underline{Z}_r^n)^{-1}\left[\tau \underline{D}_r^n \underline{\sigma}_r^n + (\underline{I}_r - \underline{D}_r^n)\underline{\rho}_r^n\right], \tag{3.13}$$

where $\underline{Z}_r^n = \underline{I}_r - \tau \underline{D}_r^n (A \otimes M_r)$ can be recognized as an RK operator, $\tau \underline{D}_r^n \underline{\sigma}_r^n$ as the contribution to the local space error inside the rth-level integration domain, and $(\underline{I}_r - \underline{D}_r^n)\underline{\rho}_r^n$ as the contribution to the local space error outside the rth-level integration domain (note the correspondence with (2.11)). Specifically, $\underline{\sigma}_r^n$ is the space truncation error expression

$$\underline{\sigma}_r^n = (A \otimes I_r)\left[\alpha_r^{(1)\mathrm{T}}, \ldots, \alpha_r^{(s)\mathrm{T}}, \alpha_r^{(s+1)\mathrm{T}}\right]^{\mathrm{T}}, \tag{3.14}$$

i.e. the ith component $\sigma_r^{(i)}$ of $\underline{\sigma}_r^n$ is given by the sum

$$\sigma_r^{(i)} = a_{i1}\alpha_1^{(1)} + \cdots + a_{is}\alpha_r^{(s)}, \tag{3.15}$$

where $\alpha_r^{(j)}$ is the space truncation error in S_r at time $t = t_{n-1} + c_j\tau$ (see (2.2)).

The error $\underline{\rho}_r^n$ depends on all previous levels and, similar to $\underline{\rho}_r$ in (2.14), is given by

$$\underline{\rho}_r^n = \underline{\lambda}_r^n + \underline{P}_{r-1r} \sum_{k=2}^{r-1} \left\{ \prod_{i=r-1}^{k+1} (\underline{X}_i^n) \right\} (\underline{Z}_k^n)^{-1} (\underline{I}_k - \underline{D}_k^n)\underline{\lambda}_k^n,$$

$$\underline{X}_i^n = (\underline{Z}_i^n)^{-1}(\underline{I}_i - \underline{D}_i^n)\underline{P}_{i-1i}, \tag{3.16}$$

where

$$\underline{\lambda}_k^n = \underline{\gamma}_k^n + \underline{P}_{k-1k}(\underline{Z}_{k-1}^n)^{-1}\tau \underline{D}_{k-1}^n \underline{\sigma}_{k-1}^n,$$

$$\underline{\gamma}_k^n = \underline{u}_k^n - \underline{P}_{k-1k}\underline{u}_{k-1}^n - \underline{b}_k^n. \tag{3.17}$$

Like in the stationary case, the local error expression (3.13) plays a crucial role as it uncovers the local space error $\tau(\underline{Z}_r^n)^{-1}\underline{D}_r^n\underline{\sigma}_r^n$ of the RK method defined in $\underline{S}_r$ on the rth-level integration domain. The idea is again to control by a sensible choice of all refinement matrices $\underline{D}_k^n$ the parasitic error $\underline{\rho}_r^n$ in such a way that the adaptive-grid local error (3.13) satisfies a similar bound as the fixed-grid local space error $\tau(\underline{Z}_r^n)^{-1}\underline{\sigma}_r^n$.

3.4. The refinement condition for the RK-LUGR scheme

The derivation of the refinement condition goes completely similar as in the stationary case. It reads

$$\left\|(\underline{I}_k - \underline{D}_k^n)\underline{\lambda}_k^n\right\| \leqslant \frac{c}{C^*}\left\|\tau(\underline{Z}_r^n)^{-1}\underline{D}_r^n\underline{\sigma}_r^n\right\|, \quad k = 2, \ldots, r, \tag{3.18}$$

where $\|\cdot\|$ is again the maximum norm and both c and C^* are arbitrary constants. Condition (3.18) implies the local space error bound

$$\|\underline{\psi}_{rs}^n\| \leqslant \left(1 + c\frac{C}{C^*}\right)\|\tau(\underline{Z}_r^n)^{-1}\underline{D}_r^n\underline{\sigma}_r^n\| \leqslant \left(1 + c\frac{C}{C^*}\right)\tau C_{RK}\|\underline{\sigma}_r^n\|, \tag{3.19}$$

where

$$\|(\underline{Z}_k^n)^{-1}\| \leqslant C_{RK}, \qquad C = C_{RK} + \sum_{k=2}^{r-1}(C_{RK}C_I)^{r-k}, \tag{3.20}$$

and C_{RK} is a constant, independent of k, that still ought to be specified.

For the (one-stage) Euler methods (3.3) and (3.4) the augmented formulation is of course redundant and it is more transparant to present the counterparts of (3.18)–(3.20) directly for (3.3) and (3.4).

The common local space error that exists on the single rth-level grid satisfies an entirely similar bound as the adaptive-grid bound (3.19) (just take $c = 0$). Hence, if C_{RK} can be taken as a grid-independent constant, we may conclude that imposing (3.18) guarantees that in the asymptotic sense the spatial accuracy will not decrease by using coarser grids. This result is very similar to that drawn earlier for the stationary case. However, for time-dependent problems an additional comment must be made, which has to do with the dependency of the local subgrids on the temporal stepsize τ. From (3.18) we see that if τ decreases, more and more entries of $\underline{D}_k^n$ must be taken zero to satisfy this condition. Hence the smaller τ, the larger the local subgrids must be in order to keep control of the interpolation errors. In fact, if $\tau \to 0$, then the interpolation error $\underline{\gamma}_k^n$ can no longer be controlled so that necessarily $\underline{D}_k^n \to \underline{I}_k$. This is what we should expect to happen for a static-regridding method due to the fact that interpolation errors can accumulate for evolving time. Our refinement condition prevents this, even though we interpolate at each time step.

The norm $\|(\underline{Z}_k^n)^{-1}\|$ naturally depends on τ, on all coefficients of the RK method, on D_k^n and on the finite difference operator M_k. Assuming a sensible choice of RK method and a ratio between stepsize τ and finest mesh width dictated by the usual time step stability, in actual application $\|(\underline{Z}_k^n)^{-1}\|$ indeed can be bounded by a grid-independent constant C_{RK} since otherwise a contradiction exists with the supposed applicability of the RK method. Further, C_{RK} will be very close to the constant that exist for application on the entire grids Ω_k. Note that C_{RK} is supposed to be independent of k. This assumption is allowed since we take the same stepsize τ at all levels. Because $(\underline{Z}_k^n)^{-1}$ is an augmented operator, the intermediate stages do play a role in the determination of C_{RK}. It is not possible to get rid of these intermediate stages for the simple fact that interpolation must take place there, which shows up in the local error analysis (and also in the stability analysis which is not discussed here, see [11]). Finally, we observe that when implementing the refinement condition (3.18), working in the augmented space may be too costly so that a simplification of (3.18) towards the output stage, for example, becomes a necessity. Normally such a simplification will not be essential in the sense that the local grid refinement would no longer comply to our claim that the use of coarse grids does not truly diminish the overall spatial accuracy of the global finest grid (see [11] where a three-stage DIRK method has been implemented).

4. Practical experience

So far we only have practical experience with time-dependent problems in two space dimensions. In [12] the explicit Runge–Kutta–Chebyshev method is applied in an LUGR manner to a nonlinear parabolic model problem from combustion theory. The grid selection, however, is still heuristic and does not underlie the current error analysis. On the other hand, this heuristic strategy is expected to work satisfactorily in many practical cases. In [10] the implicit Euler method (3.3) is discussed in detail and applied to a linear parabolic model problem. In [11] a specific three-stage DIRK method belonging to class (3.8) is discussed in detail and applied to a linear parabolic model problem. There we also discuss the use of a variable number of grid levels in time, which in many practical cases can lead to an additional reduction of workload. For a combination of LUGR with the second-order linear multistep BDF method we refer to [13–15]. In these three papers the application is centered around a specific groundwater flow system dealing with transport of brine in porous media. The system is constituted by a nonlinear elliptic equation for the pressure and two nonlinear convection–diffusion equations for the salt mass fraction and temperature, respectively. Due to the elliptic equation, at the semi-discrete level we here obtain a stiff DAE system. It has turned out that for DAE systems the current local refinement analysis may not be adequate and needs to be adapted for proper use. Recall that in this paper we assume either a genuine ODE system (2.3), or a genuine stationary system (2.3′). While in [13] the grid selection is still heuristic, and similar to that in [12], it is shown in [14,15] how such an adaptation towards DAE systems can be accomplished, thereby concentrating on the brine transport problem.

5. Concluding remarks

5.1. Survey

Adaptive-grid methods are meant for problems possessing rapid local transitions in their solution. By their very nature, these problems remain difficult to solve accurately and cheaply, also when using adaptive grids. We believe a promising approach to adaptive grids is based on LUGR. This approach of computing on nested, finer-and-finer, local uniform subgrids, overlaying one another, is widely applicable in any number of space dimensions and provides much flexibility in the selection of discretization schemes and solvers. The technique is equally well applicable to stationary and evolutionary problems. Note that a finite difference cartesian grid structure, which for certain applications can be too restrictive for geometrical reasons, is not necessary. Finite element or finite volume schemes on structured grids can be developed too.

The main theoretical and practical question for any LUGR method should be how to govern the local refinement. The refinement analysis presented in this paper underlies the assumption that the spatial error of the multilevel scheme should be "equal" to the spatial error of the finest grid level used without any adaptation. This is an optimal situation for local grid refinement methods, but of course has a price in terms of overhead. Obviously, an heuristic strategy, as used in [12,13] where no additional linear solves are needed, has less overhead. On the other hand, a strategy that takes the true spatial discretization and interpolation errors into account is generally expected to be more reliable and therefore in the long run may also

expected to be more efficient. Finally we note that we have not yet thoroughly investigated the validity of the estimation of the interpolation and truncation errors (cf. Section 2.4.2) in connection with substituting the numerical solution values into the estimators.

5.2. Efficiency

In general it remains a difficult question to tell which strategy and/or static adaptive technique is to be preferred, local pointwise or local uniform, since this may depend on a variety of interfering factors. We mention datastructure overhead, interpolation and estimation overhead, accuracy on uniform versus accuracy on nonuniform grids, implicit solution costs, and no doubt the computer architecture plays a role. However, whether a local uniform or pointwise approach is followed—assuming a reliable refinement strategy—for computing solutions with greatly varying gradients, grid adaptivity may be very cost-effective in terms of CPU time. To illustrate this for the class of LUGR methods here discussed, we carry out a simple comparison based on the following scenario.

Suppose that the volume of a subgrid is decreased by a factor $g^D < 1$ any time a new grid level is introduced. Let N be the number of grid points of Ω_1. The total number of grid points processed in one full LUGR computation then equals, approximately.

$$N\left[1 + (2g)^D + \cdots + (2g)^{D(r-1)}\right]. \tag{5.1}$$

Note that we may consider this both for a stationary problem and per time step for a time-dependent one, assuming that the local subgrid integrations span the same time interval (cf. Section 3). The number of points of Ω_r is approximately equal to $2^{D(r-1)}N$. Dividing the two yields a first measure of the expected gain factor in CPU time. This factor is to be corrected for all overhead costs not encountered on a single uniform grid. If the final factor is substantially larger than one, it pays to use local refinement. Table 2 lists expected CPU gain factors for the volume-decrease factor $g = 2/3$ and a CPU overhead factor of $3/2$, which means that in the table the entries have been divided by $3/2$.

This simple comparison of course assumes that the accuracies of the LUGR computation and the standard rth-level computation are equal. In connection with the spatial accuracy this assumption is allowed in view of our refinement strategy. Further, one can show that the temporal local error is the same on all grid levels, thanks to using the same stepsize and to always injecting finest grid values into coinciding coarse grid points [10,11]. Hence this assumption is not unrealistic. Also note that the CPU overhead factor is taken relative to the uniform rth-level computation, so that $3/2$ is substantial since coarse grid computations are

Table 2
Expected CPU gain factors

	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$
$D=1$:	0.6	0.7	0.8	1.1	1.5
$D=2$:	0.9	1.8	3.7	7.9	17.4
$D=3$:	1.6	4.7	15.3	50.6	169.7

much cheaper and thus involve relatively little overhead. Yet we believe this factor may be realistic.

We conclude that for the current scenario there is no gain for one-dimensional problems, unless a very large number of levels is used. Although for smaller volume-reduce factors g and a smaller CPU overhead factor larger gains are found, we think for one-dimensional problems the nested LUGR technique is probably not so worthwile if reduction of CPU time is the main objective. On the other hand, in two dimensions, and particularly so in three dimensions, a very substantial gain can be realized.

For stationary problems—and time-dependent problems when an implicit integration technique is used—an efficiency comparison should definitely take into account the costs of solving systems of nonlinear and linear algebraic equations, since it is here that readily most of the CPU time is spent. The choice of solution method then may play a decisive role. When good iterative solvers are used, like in [4,13] the LUGR technique offers an advantage due to the decreasing volume of the higher-level grids, which implies (relatively) decreasing condition numbers of the associated linear systems. Hence, in general, a more rapid convergence of the linear solver is experienced on the higher-level subgrids when compared with the convergence on the corresponding globally uniform grid (see also the numerical example of Section 2.5).

5.3. *Adjusting the temporal stepsize levelwise*

In this paper local subgrid integrations span the same time step interval (cf. Section 3). For time stepping this is attractive because it means that at each grid level the same local temporal errors arise [10,11], which simplifies stepsize control. Although more complicated, it is conceivable to include also the time variable in the local refinement and to work with so-called local timeslabs, determined by the selected temporal stepsize on Ω_1. In this approach this coarse-grid stepsize is halved every time a new grid level is introduced, in accordance with the spatial refinement. We have heuristically implemented it in [12].

To illustrate what additional gain in CPU time it might yield, we repeat the comparison based on the scenario of Section 5.2. The total number of grid points processed in the full LUGR computation over one timeslab then approximately equals

$$N\left[1 + 2(2g)^D + \cdots + 2^{r-1}(2g)^{D(r-1)}\right] \tag{5.2}$$

and the number of points of Ω_r processed over this timeslab is given by $\simeq 2^{r-1}2^{D(r-1)}N$. For $g = 2/3$, a CPU overhead factor of $3/2$, and considering one local timeslab, Table 3 is the counterpart of Table 2.

Table 3
Expected CPU gain factors

	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$
$D=1$:	0.7	0.9	1.5	2.1	3.2
$D=2$:	1.2	2.5	5.5	12.3	27.6
$D=3$:	2.2	7.2	24.3	81.8	276.0

We see that the one-dimensional case is still questionable for application. In the two- and three-dimensional case the expected decrease in CPU time compared to the situation of Section 5.2 is certainly of some interest, but not really dramatic. The present comparison of course also assumes that the accuracies of the LUGR computation based on local time steps and of the standard rth-level computation are equal. Amongst others this implies that the accuracy on the coarser grids may not suffer from the larger stepsizes in time. In situations where stability is dictated by the finest grid, as usually occurs with explicit time stepping, this threat will readily be absent. On the other hand, when approaching a steady state with an implicit method, there is no reason to take a smaller stepsize on a finer level. A more thorough examination seems justified.

5.4. *Treating the time variable globally and global error estimation*

To include the time variable in the local refinement, it is even conceivable to treat time in a global manner, just like the spatial variables. This means that given a time-dependent problem, we create stationary type systems (2.2′) on a sequence of finer-and-finer space–time grids. The solution step on the base space–time grid then involves that the numerical solution is considered over the entire (finite) time interval. After this, the local refinement is carried out, simultaneously in space and time, followed by the second solution step on the now locally refined, global space–time grid, etc. Note that the refinement analysis of Section 2 is general enough to encompass this global approach. For the temporal discretization one can in principle consider the use of common integration formulas of linear multistep or RK type, in which case any solution step can be carried out in the traditional step-by-step way, or one can choose alternative boundary-value type discretizations as discussed in [2,8].

There exists a clear drawback, though. The complete solution computed over the space–time domain must be kept in storage for error estimation and local refinement purposes. This of course may become an obstacle when the PDE already has two or three space dimensions. Also note that when the boundary-value approach of [2,8] is followed, the dimension in the discretization scheme itself is increased by one, which may require special solvers [8]. However, there also exists a distinct advantage from the theoretical point of view, viz. this way the local refinement is governed by estimates of the true global error (recall that in the stationary case ψ_r is a global error, whereas in the time-dependent case ψ_{rs} is local). Taking the global error into account is of interest for the simple fact that we then mimic the local error propagation of the time integration in the error estimation procedure. For example, if the computation is highly stable, as often happens for dissipative problems like parabolic PDEs or stiff ODEs, the local errors are rapidly damped so that past local errors readily play no role in the global accuracy. On the other hand, in other cases, notably hyperbolic PDEs, the computation is often marginally stable and local errors really add up. Also for DAE systems, e.g. originating from elliptic–parabolic problems, the error propagation may considerably differ per component which in turn may interfere with the local error estimation [15].

Hence it may be of interest to mimic local error propagation. The global approach provides this possibility and in view of our positive experiences with LUGR for the spatial variables, it seems worthwile to combine and extend the ideas to global error estimation for time-dependent problems. This will make codes more reliable and therefore, in the long run, also more efficient. Finally, in connection with PDEs a most interesting point is that this also opens up

possibilities for balancing spatial and temporal errors through locally adjusting the temporal and spatial mesh widths. Balancing of spatial and temporal errors is of interest in itself for attempting to increase efficiency of PDE calculations.

References

[1] S. Adjerid and J.E. Flaherty, A local refinement finite element method for two-dimensional parabolic systems, *SIAM J. Sci. Statist. Comput.* 9 (1988) 792–811.

[2] A.O.H. Axelsson and J.G. Verwer, Boundary value techniques for initial value problems in ordinary differential equations, *Math. Comp.* 45 (1985) 153–171.

[3] M.J. Berger and J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, *J. Comput. Phys.* 53 (1984) 484–512.

[4] J.G. Blom, J.G. Verwer and R.A. Trompert, A comparison between direct and iterative methods to solve the linear systems arising from a time-dependent 2D groundwater flow model, CWI Report NM-R9205, Amsterdam (1992).

[5] F.A. Bornemann, An adaptive multilevel approach to parabolic equations I: general theory and 1D-implementation, *IMPACT Comput. Sci. Engrg.* 2 (1990) 279–317.

[6] W.D. Gropp, Local uniform mesh-refinement with moving grids, *SIAM J. Sci. Statist. Comput.* 8 (1987) 292–304.

[7] J. Lang and A. Walter, A finite element method adaptive in space and time for nonlinear reaction-diffusion-systems, Preprint SC 92-5, Konrad-Zuse-Zentrum, Berlin (1992).

[8] J.M.L. Maubach, Iterative methods for nonlinear partial differential equations, Ph.D. Thesis, Catholic University of Nijmegen, Netherlands (1991).

[9] K. Rektorys, The Method of Discretization in Time, Series Mathematics and Its Applications 4 (Reidel, Dordrecht, Netherlands, 1992).

[10] R.A. Trompert and J.G. Verwer, Analysis of the implicit Euler local uniform grid refinement method, *SIAM J. Sci. Comput.* 14 (1993) 259–278.

[11] R.A. Trompert and J.G. Verwer, Runge–Kutta methods and local uniform grid refinement, *Math. Comp.* 60 (1993) 591–616.

[12] R.A. Trompert and J.G. Verwer, A static-regridding method for two-dimensional parabolic partial differential equations, *Appl. Numer. Math.* 8 (1991) 65–90.

[13] R.A. Trompert, J.G. Verwer and J.G. Blom, Computing brine-transport in porous media with an adaptive-grid method, *Internat. J. Numer. Methods in Fluids.* 16 (1993) 43–63.

[14] R.A. Trompert, Local uniform mesh refinement and brine transport in porous media, Presented at the Conference on Numerical Methods for Fluid Dynamics, Reading, UK (1992).

[15] R.A. Trompert, Local uniform grid refinement and systems of coupled partial differential equations, *Appl. Numer. Math.* 12 (1993) 331–356.

[16] J.G. Verwer and R.A. Trompert, An adaptive-grid finite-difference method for time-dependent partial differential equations, in: D.F. Griffiths and G.A. Watson, eds., *Proceedings 14th Biennial Conference on Numerical Analysis, Dundee, Scotland, 1992*, Pitman Research Notes in Mathematics Series 260 (Pitman, London, 1992) 267–284.