MARS : A METHOD FOR THE ADAPTIVE REMOVAL OF STIFFNESS IN PDES

LAURENT DUCHEMIN[*] AND JENS EGGERS[†]

Abstract. The EIN method was developed recently to remove numerical instability from PDE's, adding and subtracting an operator $\mathcal{D}$ of arbitrary structure, one of which is treated implicitly, the other explicitly. Here we extend this idea by devising an adaptive procedure to find an optimal approximation for $\mathcal{D}$. We propose a measure of the numerical error which detects numerical noise across all wavelengths, and adjust each Fourier component of $\mathcal{D}$ to the smallest value such that numerical instability is suppressed. We show that for a highly nonlinear and non-local PDE, the spectrum of $\mathcal{D}$ adapts automatically and dynamically to the theoretical result for stability. Our method thus has the same stability properties as a fully implicit method, while only requiring an explicit solver. The adaptive implicit part is diagonal in Fourier space, and thus leads to minimal overhead compared to the explicit method.

Key word. Stiff set of PDEs, Hele-Shaw, Birkhoff–Rott integral, surface tension

1. Introduction.

Our ability to model many key physical processes is limited by the numerical stability of the partial differential equations (PDE's) describing them. The reason is that the maximal stable time step of an explicit numerical integration scheme is of the order of the shortest time-scale in the system. In a stable physical system these are typically exponentially damped modes which relax back to equilibrium; the smaller the length scale, the faster the relaxation. This makes it particularly hard to simulate systems at large values of the viscosity or of the surface tension. For instance, surface tension driven flows in the open source fluid dynamics code Gerris [16] require a time step proportional to $\Delta^{3/2}$[4], where Δ is the grid spacing, which this program adapts dynamically in order to ensure a sufficient spatial accuracy [17]. As a result, for small geometries Δ can be very small, resulting in time steps which are prohibitively small. This constraint is more restrictive than the CFL constraint, related to advection. Another example is the numerical computation of solidification/fusion fronts, which uses a non-linear heat equation [22] : the corresponding time step constraint is Δ^2.

If for example relaxation is controlled by a differential operator of order m ($m = 2$ for ordinary diffusion, $m = 3$ for the Hele-Shaw flow to be described below), then the required maximum time step δt scales as $\delta t = C\delta x^m$, where δx is the smallest grid spacing or the size of the smallest sub-division. In a well-resolved numerical simulation, this should be considerably smaller than the smallest relevant physical feature. Rapid exponential decay implies that the amplitude of perturbations on the grid scale is very small, and contributes negligibly to the numerical solution. Thus one arrives at the paradoxical situation that the stability of the numerical scheme is controlled by a part of the solution which contributes negligibly, and which is actually the most stable from a physical perspective. This property is sometimes referred to as the stiffness of the PDE [13], which becomes worse with increasing spatial order m of the operator.

[*]Aix Marseille Université, CNRS, Centrale Marseille, IRPHE UMR 7342, F-13384, Marseille, France (duchemin@irphe.univ-mrs.fr).
[†]School of Mathematics - University of Bristol, University Walk, Bristol BS8 1TW, United Kingdom (Jens.Eggers@bristol.ac.uk).

To deal with this constraint on the time step, which often is so severe that it makes the exploration of important physical parameter regimes impractical, one has to resort to implicit methods. This means that the right hand side of the equation (or at least the stiffest parts of it) has to be evaluated at a *future* time step, making it necessary to solve an implicit equation at each time step [12, 1]. This makes the numerical code both complicated to write and time-consuming to solve. This is true in particular if the operator is non-local (as is the case for example of integral operators, as they appear in boundary integral type codes [18, 11]).

To address this problem, it has long been realized that not the whole of the right hand side of an equation has to be treated implicitly, as long as the "stiffest" part of the operator is dealt with implicitly. This gives rise to the so-called "implicit-explicit methods" [2], which divide up the problem between explicit and implicit parts, such that hopefully the implicit contribution is sufficiently simple to invert. If this is not clear, as is typically the case for an integral operator, the problem can be solved by judiciously slicing off the stiffest part, which can be local [11]). However, this has to be done on a case-by-case basis, and will not always be possible. Recently, we have presented a much more general method to stabilize stiff equations, which makes use of the arbitrariness in which splitting between explicit and implicit parts can take place [8, 7]. We consider a partial differential equation of the form

$$\frac{\partial u}{\partial t} = f(u,t), \tag{1.1}$$

where $u(x,t)$ is a function of space and time or a vector of functions of space and time. In order to stabilize the stiff terms in $f(u,t)$, we add two terms on the right-hand-side of the discretized version of equation (1.1) :

$$\frac{u_i^{n+1} - u_i^n}{\delta t} = f_i(u^n, t^n) - \mathcal{D}_i[u^n] + \mathcal{D}_i[u^{n+1}], \tag{1.2}$$

where n denotes the time variable ($t^n = n\delta t$) and $\mathcal{D}$ is an arbitrary operator. The variable u as well as f are defined on a spatial grid $x_i = i\delta x$, where δx is the grid spacing. Clearly, the added terms are effectively zero apart from the first-order error that comes from the fact that $\mathcal{D}$ is evaluated at different time levels, which motivates the name "Explicit-Implicit-Null" method or "EIN". If $\mathcal{D}$ is the same as the original operator $f(u,t)$, this is a purely implicit method, if $\mathcal{D} = 0$, it is explicit. Similar ideas have been implemented to stabilize the motion of a surface in the diffuse interface and level-set methods [21, 10, 19], and for the solution of PDEs on surfaces [14]. We also show that by a simple step-halving procedure [3], (1.2) can always be turned into a scheme which is second order accurate in time [7].

The crucial insight is that $\mathcal{D}$ can be chosen for maximum effectiveness, with no regard as to the structure of the original operator f. In particular, we can choose $\mathcal{D}$ to be diagonal in Fourier space, rendering the implicit step almost trivial to perform:

$$\frac{\hat{u}_k^{n+1} - \hat{u}_k^n}{\delta t} = \hat{f}_k(u^n, t^n) + \lambda(k)\hat{u}_k^n - \lambda(k)\hat{u}_k^{n+1}, \tag{1.3}$$

where $\hat{}$ denotes the Fourier transform and the damping spectrum $\lambda(k) \geq 0$ is an arbitrary function. The Fourier transform $\hat{f}_k$ can be calculated effectively from the spatial discretizaton f_i using the fast Fourier transform (FFT). From (1.3), we can calculate $\hat{u}_k^{n+1}$ directly for each k, so we obtain the desired solution from the inverse transform. The scheme (1.3) (as well as any other first order scheme) can be turned into a second order scheme by Richardson extrapolation [3]. Namely, let $u^{1,n+1}$ be the solution for one step δt, $u^{2,n+1}$ the solution for two half steps $\delta t/2$. Then

$$u^{n+1} = 2u^{2,n+1} - u^{1,n+1} + \mathcal{O}(\delta t^3), \tag{1.4}$$

is second order accurate in time, and

$$E = u^{1,n+1} - u^{2,n+1} \tag{1.5}$$

can be used as an estimate for the error.

To analyse (1.3) further, we adopt a "frozen-coefficient" hypothesis, that the solution is essentially constant over the time scale on which numerical instability is developing. Then for small perturbations $\delta \hat{u}_k^n$ about the current solution $\hat{u}_k^n$ we can linearize. At least on the small scale (i.e. in the large k limit), $\hat{f}_k(u^n, t^n)$ is expected to be translationally invariant, making the operator diagonal in Fourier space, so we can write

$$\hat{f}_k(u^n + \delta u, t^n) \sim -e(k)\delta \hat{u}_k^n. \tag{1.6}$$

Here for simplicity we assume that the eigenvalues $e(k)$ are real, as it is typically the case for physical problems, where the dominant process in small scale is dissipative. We have shown in [7] that, as long as $\lambda(k) > e(k)/2$, the system (1.3) is unconditionally stable. This is a generalization of a method first presented, for the case of the diffusion equation in two dimensions, in [5]. If (1.3) is turned into a second order scheme using (1.4), this condition is [7]:

$$\lambda(k) > \lambda_c(k) = 2e(k)/3, \tag{1.7}$$

with $\lambda_c(k)$ the theoretical stability limit. Thus for sufficiently large values of $\lambda(k)$, there is always stability; however, the rounding error increases with λ, and should therefore be kept as small as possible, consistent with the stability constraint. There is a certain similarity here with the preconditioning of matrices, where a matrix is approximated by a simple diagonal matrix [23, 9].

In [7] we have tested the ideas underlying the EIN method, calculating the spectrum $e(k)$ for a variety of operators, including nonlocal operators treated previously in [11]. We approximated $\lambda(k)$ as a power law, derived from the low wavenumber limit of the exact discrete spectrum. As predicted by the above analysis, we find the scheme (1.3) unconditionally stable, and performing with the same accuracy as that proposed in [11]. Obviously, this still requires one to obtain a good estimate for the spectrum.

In the present paper, we aim to remove this analytical step, and to make the calculation of $\lambda(k)$ self-consistent. The idea is to determine $\lambda(k)$ iteratively, by detecting numerical instability. If there is numerical noise, the damping is increased, while $\lambda(k)$ can be reduced if the code is stable. In the simplest version of our procedure, we focus on the high wave number limit, where most of the stiffness is coming from, and approximate $\lambda(k)$ by a power law, determined by one or two parameters, depending on whether the exponent is to be prescribed. While we found this approach to work, it introduces arbitrary assumptions into the procedure, and assumes a separation between a high and low wave number regimes. Instead, here we present the results of a scheme which adjusts each Fourier mode individually, based on noise detected in the same Fourier mode. This models the original operator in much greater detail, and leads to a spectrum $\lambda(k)$ which corresponds exactly to the theoretical stability limit.

2. Adaptive stabilization.

Our method is based on the formulation (1.3), which together with (1.4) is an unconditionally stable second order scheme, as long as $\lambda(k)$ is sufficiently large. We would like to find an adaptive procedure which refines $\lambda(k)$ at each time step, so as to keep it as small as possible consistent with stability. To achieve this, we have to address two issues: (i) find a measure $\epsilon(k)$ of the noise, or of numerical instability, for each Fourier mode k; (ii) specify the evolution of $\lambda(k)$ for a given noise.

Finding a suitable measure of the error is the crucial question, to be discussed in more detail below. As for (ii), we aim to adjust each Fourier component $\lambda(k)$ individually, although we have also explored representing $\lambda(k)$ by a finite number of parameters. We adopt the simplest possible approach, taking a local relation between $\epsilon(k)$ and $\lambda(k)$. For each Fourier mode, if $\epsilon(k)$ is larger than an upper bound ϵ_u, the corresponding $\lambda(k)$ is increased in a geometric progression. If on the other hand $\epsilon(k) < \epsilon_u$, $\lambda(k)$ is increased at a much smaller progression, in order to avoid a sudden onset of instability.

As to a measure of noise, a first guess might be to take $\epsilon(k)$ as the Fourier transform of the error (1.5). We tested this idea using the interface dynamics discussed in more detail in the next section, and illustrated in Fig. 5. Figure 1 shows the evolution of the Fourier transform $\hat{E}_k$ of this error, as a function of time, without using the EIN method. The time step is chosen to be $\delta t = 3.125 \times 10^{-5}$, the number of points $N = 1024$, and we use a purely explicit scheme (no stabilization), so that the modes with the largest wavenumbers are unstable. Indeed, as explained in the next section, there exists a region in $k-$space which would be stable with an explicit scheme (on the left of the vertical dashed line), and an unstable region (on the right), where we would like to detect numerical instability. As a result, the noise level grows very rapidly for the right-hand side of the spectrum, and for the first two time steps there is little power in the small-k modes. Thus $\hat{E}_k$ could be used to detect correctly the numerical instability for large k.

However, the left part of the spectrum is soon invaded through non-linear mode-coupling, and there grows a considerable component of the error at small k (corresponding to large scales), which would not be damped away if $\lambda(k)$ was increased. The problem is clear: in the proposed scheme, there is no clean distinction between noise resulting from numerical instability, and the broad spectrum of unstable modes which is part of the physical solution. The crucial problem of defining the numerical noise ϵ lies in this distinction.

We have tried various refinements of the definition of ϵ which attempt to make the distinction more clearly. For example, we used a higher order approximation of the truncation error E by taking into account several time steps. However, although the procedure worked for a little longer, it soon failed in the same way. A successful procedure came from the idea of spatial smoothing, taking the truncation error as the starting point. To compute ϵ at the i'th gridpoint, we consider the error E_i, and compare it to a smoothed version $\bar{E}_i$ at the same point. The reasoning is that $\bar{E}_i$ contains the full spectrum coming from the deterministic nonlinear dynamics, so $E_i - \bar{E}_i$ contains the random noise produced by numerical instability. Taking the Fourier transform, we define

$$\epsilon(k) = \hat{E}_k - \hat{\bar{E}}_k. \tag{2.1}$$

There are many possible choices for the smoothed- out error. We chose a polynomial approximation over $2n$ gridpoints, but excluding i itself, otherwise ϵ would be

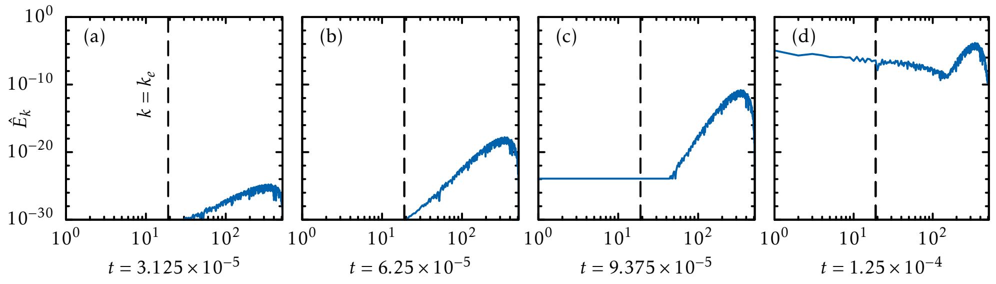

FIG. 1. *The evolution of the Fourier transform $\hat{E}_k$ of the error, for the first four time steps, without damping ($\forall k, \lambda(k) = 0$). Initially the error is uniformly small for a flat interface with a white noise, but eventually develops significant components at smaller k. The vertical dashed line, at $k = k_e$ (see equation (3.11)), separates to the left values of k for which an explicit scheme is stable, and to the right values of k for which our EIN method is needed.*

identically zero. In other words,

$$\bar{E}_i = \mathcal{P}\left(E_{i-n}, \ldots, E_{i-1}, E_{i+1}, \ldots, E_{i+n}\right). \tag{2.2}$$

The measure of the error with this new definition is shown in figure 2. The same parameters as for figure 1 have been used, and the three data curves correspond to $n = 1$ (red), $n = 2$ (blue) and $n = 3$ (green). The results to be reported below are for $n = 2$, i.e. using 4 points for the interpolation. The choice $n = 1$ yielded similar results, but the procedure broke down earlier as the interface became very strongly deformed.

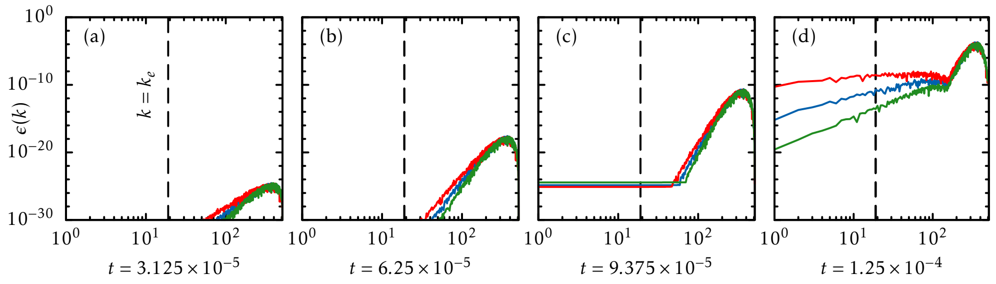

FIG. 2. *The evolution of the error $\epsilon(k)$ defined by equation (2.1), for the first four time steps, without damping ($\forall k, \lambda(k) = 0$). Initially the error is uniformly small for a flat interface with a white noise. The large values observed at small k on $\hat{E}_k$ after a few time steps are significantly reduced with this new definition of the error. The three colors correspond to three averaging methods :* **Red** *: $n = 1$ in equation (2.2), i.e. one point to the left, one to the right;* **Blue** *: $n = 2$;* **Green** *: $n = 3$. The vertical dashed line is the stability boundary $k = k_e$.*

This shows that the procedure does not rely on the selection of a particular length scale, as defined by the ratio of the size of the smoothing region, divided by the grid size. However, for $n = 3$ we ran into numerical difficulties in trying to compute the interpolating polynomial. Instead, we used $n = 2$, *i.e.* two points to the left, two to the right, to define the average error; this procedure worked equally well.

The difference between the naive error measure $\hat{E}_k$ and $\epsilon(k)$ based on smoothing is illustrated in Fig. 3. We ran the same computation as in figures 1 and 2, but using our

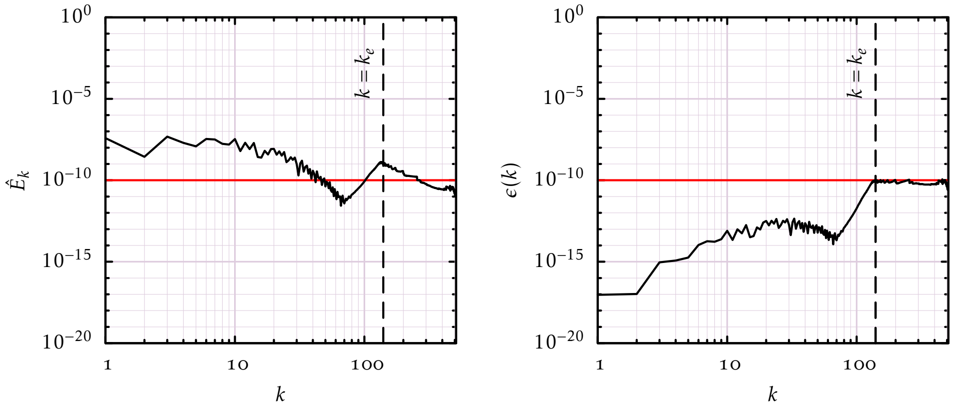

FIG. 3. *The effect of smoothing on the solution at* $t = 0.04$ *(third panel of Fig. 5 below). On the left, we show the spectrum of the error* $\hat{E}_k$*, which is broad in the nonlinear regime. On the right, we show the numerical noise* $\epsilon(k)$ *as defined by (2.1), which is substantial only in a high wave number region where noise is detected. The vertical dashed line is* $k = k_e$*. The horizontal red line is the threshold* ϵ_u *used to adapt* $\lambda(k)$*.*

adaptive procedure. Equation (2.1) is used as a measure of the error (with $n = 2$) to adapt $\lambda(k)$ at each time step, with the threshold $\epsilon_u = 10^{-10}$. The left curve shows the Fourier transform $\hat{E}_k$ as a function of k : clearly, this error alone is ill-suited to detect instability, since it has significant components for $k < k_e$, where the explicit scheme is stable, so there is no instability even for $\lambda(k) = 0$. The right curve shows the error (2.1) used to adapt $\lambda(k)$ as a function of k : the instability is correctly detected at large values of k and this error remains low, *i.e.* does not require any damping, in the region where an explicit scheme is stable.

3. Example: Hele–Shaw flow.

3.1. Equations of motion. We consider an interface in a vertical Hele-Shaw cell, separating two viscous fluids with the same dynamic viscosity, with the heavier fluid on top [11]. As heavy fluid falls, small perturbations on the interface grow exponentially: this is known as the Rayleigh-Taylor instability [6]. However, surface tension assures regularity on small scales. For simplicity, we assume the flow to be periodic in the horizontal direction. We briefly recall the dynamics of the interface here; for more details, see [7].

The interface is discretized using marker points labeled with α, which are advected according to :

$$\frac{\partial \mathbf{X}(\alpha)}{\partial t} = U\mathbf{n} + T\mathbf{s}. \tag{3.1}$$

Here $\mathbf{X}(\alpha) = (x, y)$ is the position vector, $\mathbf{n} = (-y_\alpha/s_\alpha, x_\alpha/s_\alpha)$ and $\mathbf{s} = (x_\alpha/s_\alpha, y_\alpha/s_\alpha)$ are the normal and tangential unit vectors, respectively, and $s_\alpha = (x_\alpha^2 + y_\alpha^2)^{1/2}$. Hence $U = (u, v) \cdot \mathbf{n}$ and $T = (u, v) \cdot \mathbf{s}$ are the normal and tangential velocities, respectively. The tangential velocity does not affect the motion, but is chosen so as to maintain a reasonably uniform distribution of points [11, 7]. If $z(\alpha, t) = x + iy$ is the complex position of the interface, which is assumed periodic with period 1 ($z(\alpha + 2\pi) = z(\alpha) + 1$), the complex velocity becomes :

$$u(\alpha) - iv(\alpha) = \frac{1}{2i} PV \int_0^{2\pi} \gamma(\alpha', t) \cot\left[\pi(z(\alpha, t) - z(\alpha', t))\right] d\alpha', \tag{3.2}$$

where γ is the vortex sheet strength. For two fluids of equal viscosities [15],

$$\gamma = S\kappa_\alpha - Ry_\alpha, \tag{3.3}$$

where κ is the mean curvature of the interface :

$$\kappa(\alpha) = \frac{x_\alpha y_{\alpha\alpha} - y_\alpha x_{\alpha\alpha}}{s_\alpha^3}, \qquad \text{recalling that} \qquad s_\alpha = (x_\alpha^2 + y_\alpha^2)^{1/2}. \tag{3.4}$$

Here S is the non-dimensional surface tension coefficient and R is the non-dimensional gravity force. To compute the complex Lagrangian velocity of the interface (3.2), we use the spectrally accurate alternate point discretizaton [20] :

$$u_j - iv_j \simeq -\frac{2\pi i}{N} \sum_{\substack{l=0 \\ j+l\ odd}}^{N-1} \gamma_l \cot\left[\pi(z_j - z_l)\right]. \tag{3.5}$$

Derivatives κ_α and y_α are computed at each time step using second-order centered finite differences, and α is defined by $\alpha(j) = 2\pi j/N$, where $j \in [0, N]$ and N is the number of points describing the periodic surface. Note that the numerical effort of evaluating (3.5) requires $\mathcal{O}(N^2)$ operations, and thus will be the limiting factor of our algorithm.

3.2. Stabilization. Combining (1.3) with (3.1), the numerical scheme becomes :

$$\hat{x}_k^{n+1} = \hat{x}_k^n + \frac{\hat{u}_k^n}{\delta t^{-1} + \lambda(k)}, \quad \hat{y}_k^{n+1} = \hat{y}_k^n + \frac{\hat{v}_k^n}{\delta t^{-1} + \lambda(k)}, \tag{3.6}$$

where $\hat{u}_k^n$ and $\hat{v}_k^n$ are calculated from the Fourier transform of (3.5). The new grid points x_i^{n+1}, y_i^{n+1} are obtained from the inverse Fourier transform. Using the Richardson scheme (1.4), from two half steps $\delta t/2$ of the first-order method (3.6), one obtains

$$x_i^{n+1} = 2x_i^{2,n+1} - x_i^{1,n+1}, \quad y_i^{n+1} = 2y_i^{2,n+1} - y_i^{1,n+1}, \tag{3.7}$$

which is second-order accurate.

In [7], we performed a linear analysis (1.6) of the discrete modes of (3.5) about a flat interface. We found that

$$e(k) = \frac{SN^3}{L^3}\left(1 - \cos\frac{2\pi k}{N}\right)\sin\frac{2\pi k}{N} \equiv \tilde{e}(x) = (1 - \cos x)\sin x, \quad x = \frac{2\pi k}{N}, \tag{3.8}$$

where L is the length of the interface, and N the number of gridpoints. The normalized form of the spectrum is shown in Fig. 4. For long wavelengths, *i.e.* small wavenumbers, this can be approximated by the power law

$$e(k) \approx \frac{S}{2L^3}(2\pi k)^3 \equiv \tilde{e}(x) = \frac{x^3}{2}, \tag{3.9}$$

which is shown as the dashed line in the same figure.

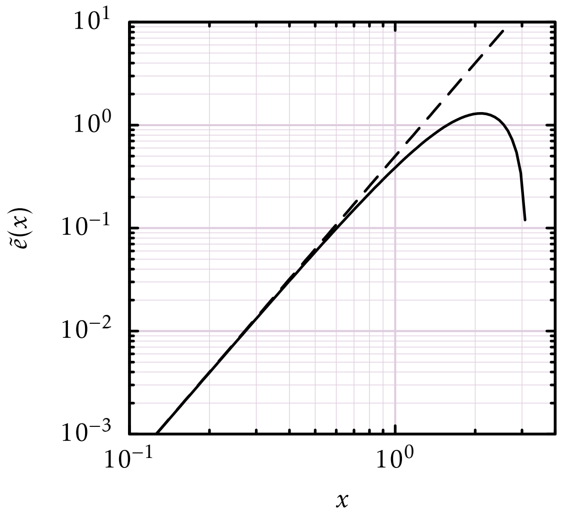

FIG. 4. *A double logarithmic plot of the dispersion relation $\tilde{e}(x)$, as defined in* (3.8). *The long-wavelength approximation* (3.9) *is shown as the dashed line.*

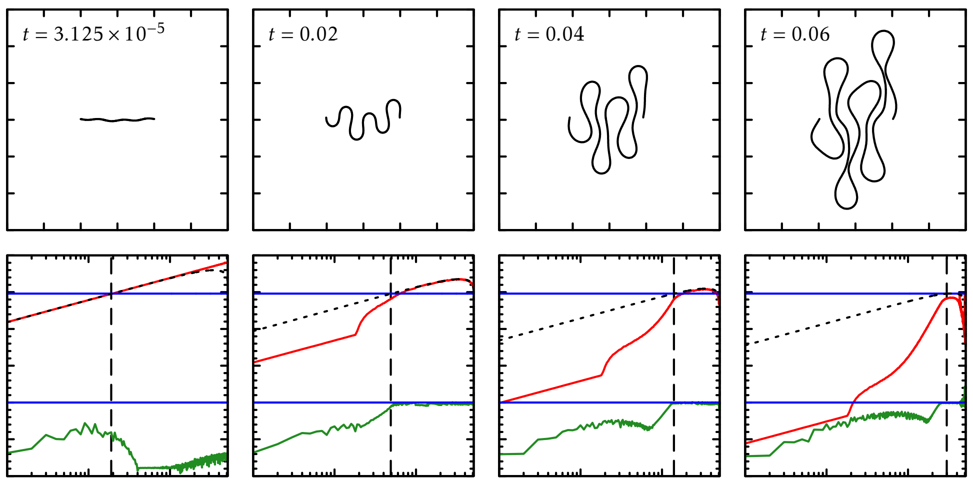

FIG. 5. *A simulation of the Hele-Shaw problem, with the interface shown on the top row. On the lower row, the corresponding spectrum of $\epsilon(k)$ defined by equation* (3.12) *(green), as well as $\lambda(k)$ (red). The dotted line is the stability limit $\lambda_c(k) = 2e(k)/3$, with $e(k)$ given by* (3.8)*, the top horizontal blue line the explicit stability boundary $2/\delta t$. The vertical dashed line is $k = k_e$. The ranges in the interface plots are $[-1 : 2]$ in x, $[-1.5 : 1.5]$ in y, and in the log-log plots $[1 : 512]$ in x, $[10^{-20} : 10^{10}]$ in y.*

In [7] we found that (3.7) was stable as long as $\lambda(k)$ was chosen according to the stability criterium (1.7). For simplicity, we chose the asymptotic form (3.9), which is always larger than the true spectrum. In other words,

$$\lambda(k) \geq \frac{S}{3}\left(\frac{2\pi k}{L}\right)^3 \qquad (3.10)$$

is a sufficient condition for stability. In addition, for $e(k)\delta t < 2$ the purely explicit scheme ($\lambda \equiv 0$) is stable; this is the stability boundary for an explicit Euler scheme. Thus for a given time step δt, the condition $e(k)\delta t = 2$ defines a stability boundary k_e for the wave number, below which an explicit step is stable:

$$k_e \approx \frac{L}{2\pi}\left(\frac{4}{S\delta t}\right)^{1/3}. \tag{3.11}$$

Since this boundary typically lies in the limit of small wavenumbers, we can use the asymptotic expression (3.9).

Using the procedure described in Sec. 2, for each Fourier mode if ϵ is larger than an upper bound $\epsilon_u = 10^{-10}$, λ is multiplied by 1.2. If on the other hand ϵ is smaller than ϵ_u, λ is decreased by a factor of 1/1.02. Figure 5 shows our adaptive scheme at work, as the interface (shown on the top row) deforms, and the length L of the interface increases. The error $\epsilon(k)$ is defined by :

$$\epsilon(k) = MAX(\hat{E}^x_k - \hat{\hat{E}}^x_k, \hat{E}^y_k - \hat{\hat{E}}^y_k). \tag{3.12}$$

In equation (3.12), the error defined by equation (2.1) is computed, for each mode, for both x and y spectra, and the global error is defined as the maximum of the two. We chose this definition for the error, because rapid oscillations can occur in both x and y variables. We have initialized $\lambda(k)$ to the asymptotic power-law form (3.10) of the stability boundary, also used in [7]. After the first time step (first panel of Fig. 5), $\lambda(k)$ still has its initial value, while the error $\epsilon(k)$ is very small as expected. As seen in the second panel, $\lambda(k)$ has converged onto the theoretical stability limit $\lambda_c(k) = 2e(k)/3$ (cf. (1.7)), with $e(k)$ given by (3.8). The damping spectrum has thus dropped from the initial condition (3.9), which overpredicts the stability boundary. However, convergence only occurs for $k > k_e$, since below $k = k_e$ no numerical instability occurs. As a result, for $k < k_e$ the stabilizing spectrum $\lambda(k)$ is reduced at every time step, and has already fallen by orders of magnitude below the stability limit of the EIN scheme. Correspondingly, by adjusting $\lambda(k)$ the error $\epsilon(k)$ is kept close to the threshold $\epsilon_u = 10^{-10}$ for $k > k_e$. Below k_e, $\epsilon(k)$ remains very small, since there is no instability to trigger it.

In addition, owing to the increase in L, the explicit stability boundary k_e (3.11) moves in time to the right (vertical dashed line), and the stability limit (3.10) (dotted line) moves down. As seen in the third and fourth panel of Fig. 5), $\lambda(k)$ continues to adapt to the stability limit as it drops further, but only in the regime $k > k_e$, which becomes progressively smaller. The results described above are not changed significantly as ϵ_u is varied over several orders of magnitude up or down from 10^{-10}, but of course the value must be significantly over the rounding error, and below the expected truncation error.

In our earlier EIN scheme [7], we used $\lambda(k)$ based on the the simplified stability boundary (3.10) to stabilize the Hele-Shaw interface motion shown in (5). However, this overpredicts the necessary damping for large k. For $k < k_e$, on the other hand, no damping is necessary, and our adaptive scheme reflects that by decreasing $\lambda(k)$ more and more. As a result, the damping in the adaptive scheme is significantly smaller than in our previous EIN scheme. In Fig. 6 we show a comparison of the numerical results to those of the earlier scheme, and find very good agreement. The major advance is of course that $\lambda(k)$ no longer needs to be prescribed, but is found

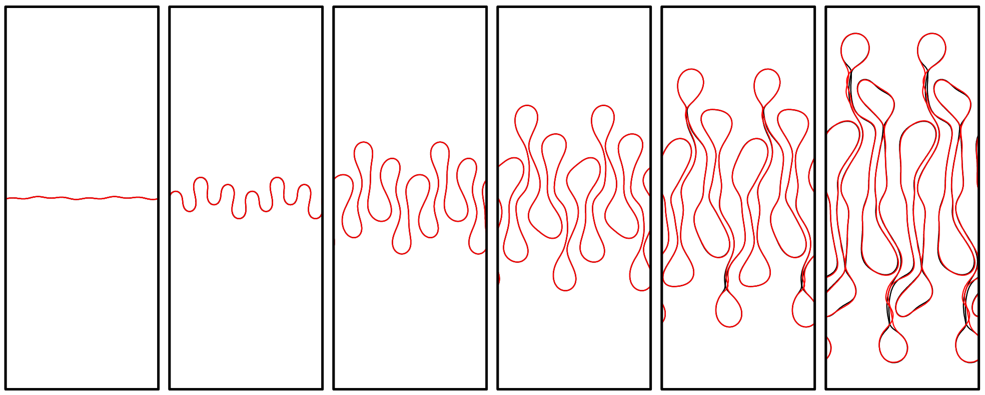

Fig. 6. *A comparison of the interface as obtained from our current adaptive scheme (red curves) and our earlier EIN scheme (black curves) [7], which used the theoretical stability boundary* (3.10).

self-consistently as part of the algorithm which ensures stability. Only in the last panel is there a significant discrepancy between the two results. This occurs in places where two sides of the interface have come in close proximity, comparable to the spacing between grid points. But this means our evaluation of the velocity integral is no longer sufficiently accurate to be reliable.

4. Outlook and conclusions.

We have thus demonstrated the feasibility of our method using a difficult model problem, in that the operator is both very stiff and non-local. However, there are many ways in which to extend and improve the present approach. Firstly, we estimated the damping spectrum by analyzing the current solution in Fourier space, which is particularly easy for the periodic domain considered by us. However, this may be circumvented by periodically continuing a solution defined over a finite domain only. In addition, one could formulate the entire method in real space, as done in some cases described in [7].

A second, more important issue is our assumption of the spectrum $e(k)$ in (1.6) being real. This assumption is well founded, since the ultimate physical damping process is dissipative, leading to real eigenvalues. However, as demonstrated by the example of an inertial vortex sheet considered in [11], even problems lacking dissipation can display significant stiffness. This case leads to a system of PDEs, with pairs of complex eigenvalues $e(k)$ on the right-hand-size of (1.6), corresponding to traveling waves. In that case the damping spectrum $\lambda(k)$ would also have to be complex to ensure stability, a case we have not yet considered.

Finally, a problem we still need to address is how to choose an initial condition for the damping spectrum $\lambda(k)$. In the present work we choose a power-law spectrum which can be inferred from a simple analysis of the continuum version of the equations of motion, which then adapts to an optimal spectrum. It would be ideal if no input whatsoever was necessary, choosing for example $\lambda(k) = 0$ initially. At present, this is not possible, as the quality of the numerical solution deteriorates before $\lambda(k)$ can adapt. We suspect that in order for such a scheme to be successful, one needs to implement a variable time step, such that initial steps during which $\lambda(k)$ is found are very small.

In conclusion, following our previous study on this subject, we propose a new method to remove the stiffness of PDEs containing non-linear stiff terms, *i.e.* high spatial derivatives embedded into non-linear terms. This method allows for the self-consistent estimation of a stabilizing term on the right-hand-side of the PDE, that ensures absolute stability for the numerical scheme. Analyzing the spectrum of the solution at each time step, we adapt automatically the stabilizing term such that each unstable Fourier mode is damped optimally. The computational cost of this method is essentially the same as that of the explicit method.

REFERENCES

[1] W. F. Ames, *Numerical methods for partial differential equations*, Academic press, 2014.

[2] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton, *Implicit-explicit methods for time-dependent partial differential equations*, SIAM J. Numer. Amal., 32 (1995), p. 797.

[3] B. P. Ayati and T. F. Dupont, *Convergence of a step-doubling galerkin method for parabolic problems*, Math. Comput., 74 (2004), pp. 1053–1065.

[4] J. Brackbill, D. B. Kothe, and C. Zemach, *A continuum method for modeling surface tension*, Journal of computational physics, 100 (1992), pp. 335–354.

[5] J. Douglas Jr. and T. F. Dupont, *Alternating-direction galerkin methods on rectangles*, in Numerical Solution of Partial Differential Equations II, B. Hubbard, ed., Academic Press, 1971, pp. 133–214.

[6] P. G. Drazin and W. H. Reid, *Hydrodynamic stability*, Cambridge University Press, Cambridge, 1981.

[7] L. Duchemin and J. Eggers, *The explicit-implicit-null method: removing the numerical instability of pdes*, Journal of Computational Physics, 263 (2014), pp. 37–52.

[8] J. Eggers, J. R. Lister, and H. A. Stone, *Coalescence of liquid drops*, J. Fluid Mech., 401 (1999), pp. 293–310.

[9] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*, Numerical Mathematics and Scientific Computation, 2014.

[10] K. Glasner, *A diffuse interface approach to hele-shaw flow*, Nonlinearity, 16 (2003), pp. 49–66.

[11] T. Hou, J. Lowengrub, and M. Shelley, *Removing the stiffness from interfacial flows with surface tension*, J. Comp. Physics, 114 (1994), pp. 312–338.

[12] A. Iserles, *A first course in the numerical analysis of differential equations*, no. 44, Cambridge university press, 2009.

[13] A.-K. Kassam and L. Trefethen, *Fourth-order time-stepping for stiff pdes*, SIAM J. Sci. Comput., 26 (2005), pp. 1214–1233.

[14] C. B. Macdonald and S. J. Ruuth, *The implicit closest point method for the numerical solution of partial differential equations on surfaces*, SIAM J. Sci. Comput., 31 (2009), pp. 4330–4350.

[15] A. J. Majda and A. L. Bertozzi, *Vorticity and Incompressible Flow*, Cambridge University Press, Cambridge, 2002.

[16] S. Popinet, *An accurate adaptive solver for surface-tension-driven interfacial flows*, J. Comp. Phys., 228 (2009), pp. 5838–5866.

[17] S. Popinet, *Numerical models of surface tension*, Annual Review of Fluid Mechanics, 50 (2018), pp. 49–75, https://doi.org/10.1146/annurev-fluid-122316-045034.

[18] C. Pozrikidis, *Boundary Integral and singularity methods for linearized flow*, Cambridge University Press, Cambridge, 1992.

[19] D. Salac and W. Lu, *A local semi-implicit level-set method for interface motion*, J. Sci. Comput., 35 (2008), pp. 330–349.

[20] M. Shelley, *A study of singularity formation in vortex sheet motion by a spectrally accurate vortex method*, J. Fluid Mech., 244 (1992), p. 493.

[21] P. Smereka, *Semi-implicit level set methods for curvature and surface diffusion motion*, J. Sci. Comput., 19 (2002), pp. 439–456.

[22] M. Ulvrov, S. Labrosse, N. Coltice, P. Raback, and P. Tackley, *Numerical modelling of convection interacting with a melting and solidification front: Application to the thermal evolution of the basal magma ocean*, Physics of the Earth and Planetary Interiors, 206207 (2012), pp. 51 – 66.

[23] A. Wathen and D. Silvester, *Fast iterative solution of stabilised stokes systems. part i:*

Using simple diagonal preconditioners, SIAM Journal on Numerical Analysis, 30 (1993), pp. 630–649.