Programming methodologies

Metodología 1: Xtreme Programming

Hay dos personas, y cada uno tiene su función:

→ Driver (escribe código)
→ Reviewer (solo comenta sobre el programa, critica constructivamente)

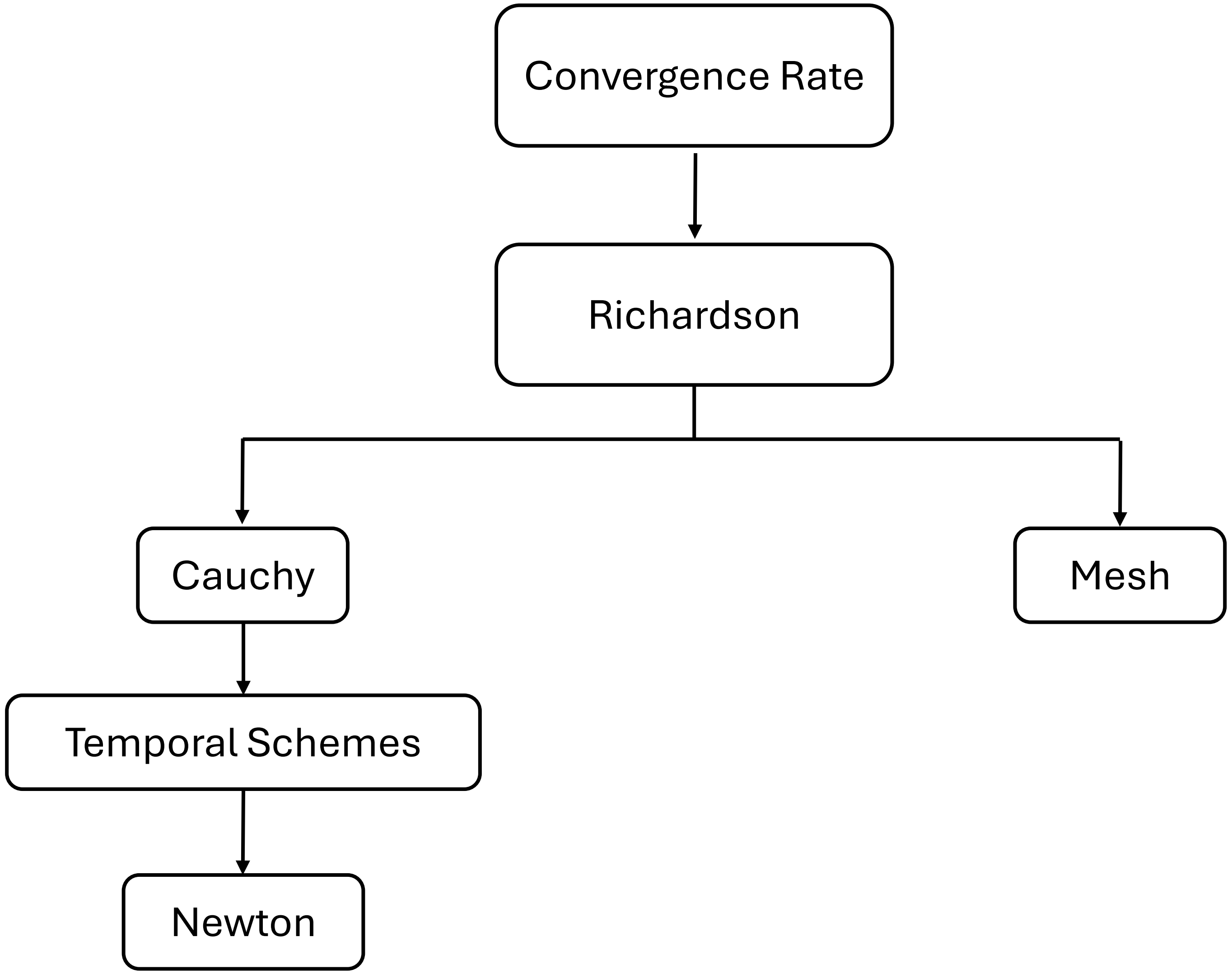
Metodología 2: Test Driven Development (TDD)

Escribir código a través de pruebas muy pequeñas.

Metodología 3: Top-Down

Escoger un problema, e ir derivando. Es decir, identificando cosas, dividirlas, y esas cosas divididas volver a dividirlas. De modo que primero resolvamos lo más básico, y nos vaya dando las soluciones más complejas.

Metodología Top-Down para Error con extrapolación de Richardson y convergencia de esquemas temporales

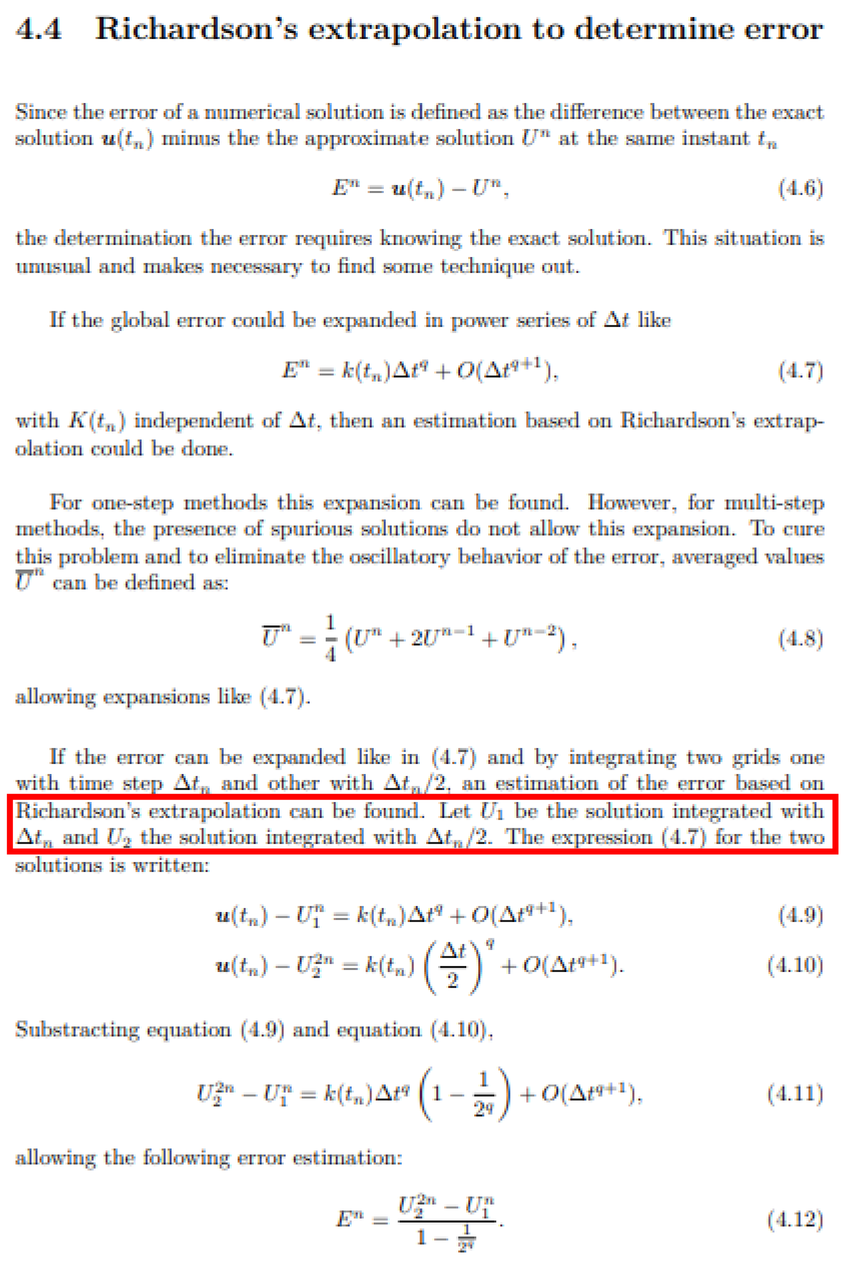


Global error with Richardson's extrapolation

Bibilography:

[1] J.A. Hernández, J.Escoto (2017). *How to learn applied mathematics through modern Fortran*

[2] J.A. Hernández, M.A. Zamecnik (2019). *Interpolación polinómica de alto orden. Métodos espectrales. Aplicación a problemas de contorno y de condiciones iniciales*



4.4 Richardson's extrapolation to determine error

Since the error of a numerical solution is defined as the difference between the exact solution $\mathbf{u}(t_n)$ minus the the approximate solution U^n at the same instant t_n

$$E^n = \mathbf{u}(t_n) - U^n, \tag{4.6}$$

the determination the error requires knowing the exact solution. This situation is unusual and makes necessary to find some technique out.

If the global error could be expanded in power series of Δt like

$$E^n = k(t_n)\Delta t^q + O(\Delta t^{q+1}), \tag{4.7}$$

with $K(t_n)$ independent of Δt, then an estimation based on Richardson's extrapolation could be done.

For one-step methods this expansion can be found. However, for multi-step methods, the presence of spurious solutions do not allow this expansion. To cure this problem and to eliminate the oscillatory behavior of the error, averaged values $\overline{U}^n$ can be defined as:

$$\overline{U}^n = \frac{1}{4}\left(U^n + 2U^{n-1} + U^{n-2}\right), \tag{4.8}$$

allowing expansions like (4.7).

If the error can be expanded like in (4.7) and by integrating two grids one with time step Δt_n and other with $\Delta t_n/2$, an estimation of the error based on Richardson's extrapolation can be found. Let U_1 be the solution integrated with Δt_n and U_2 the solution integrated with $\Delta t_n/2$. The expression (4.7) for the two solutions is written:

$$\mathbf{u}(t_n) - U_1^n = k(t_n)\Delta t^q + O(\Delta t^{q+1}), \tag{4.9}$$

$$\mathbf{u}(t_n) - U_2^{2n} = k(t_n)\left(\frac{\Delta t}{2}\right)^q + O(\Delta t^{q+1}). \tag{4.10}$$

Substracting equation (4.9) and equation (4.10),

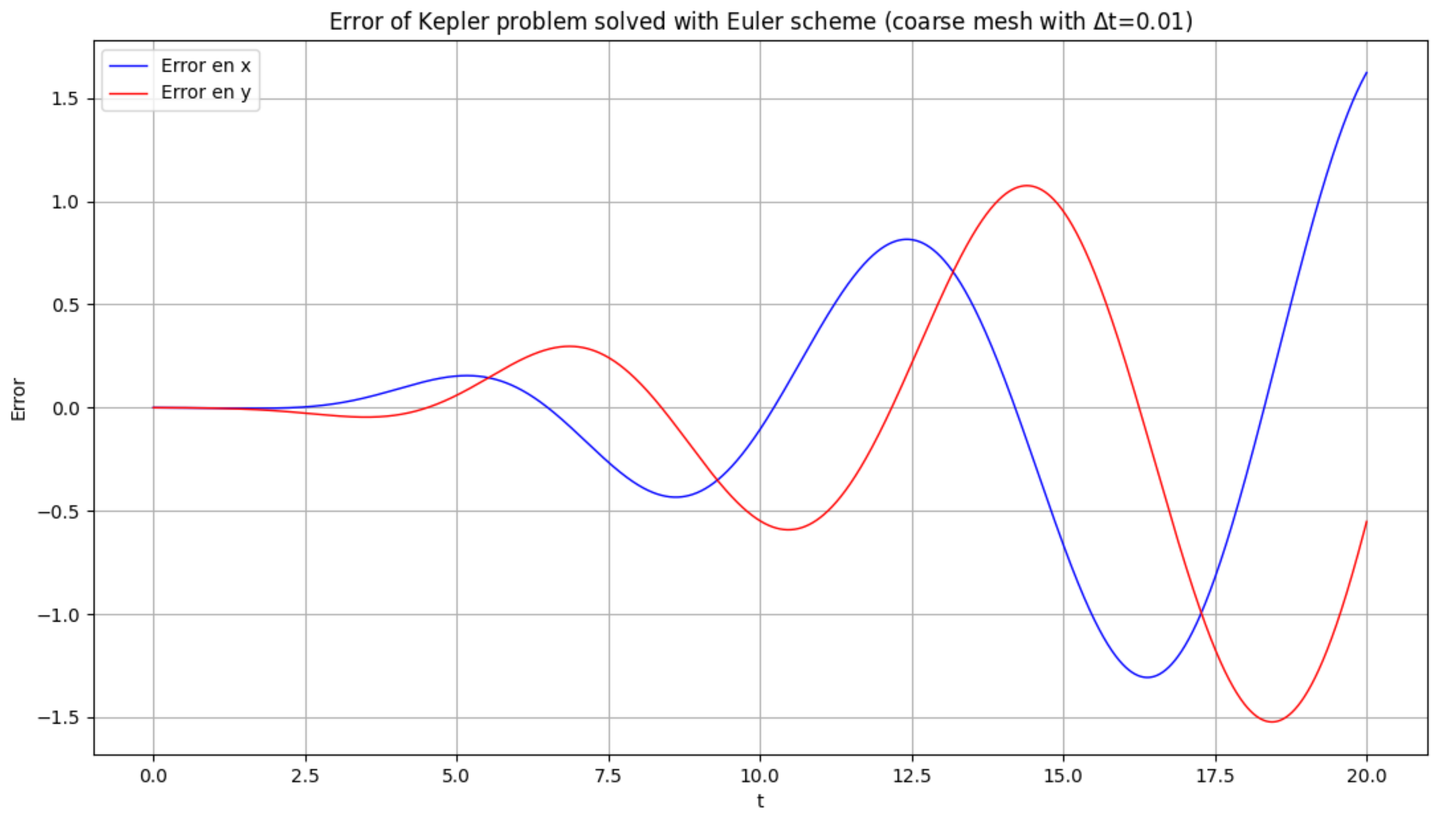
$$U_2^{2n} - U_1^n = k(t_n)\Delta t^q\left(1 - \frac{1}{2^q}\right) + O(\Delta t^{q+1}), \tag{4.11}$$

allowing the following error estimation:

$$E^n = \frac{U_2^{2n} - U_1^n}{1 - \frac{1}{2^q}}. \tag{4.12}$$

Bibliography:

[1] J.A. Hernández, J. Escoto (2017). *How to learn applied mathematics through modern Fortran*

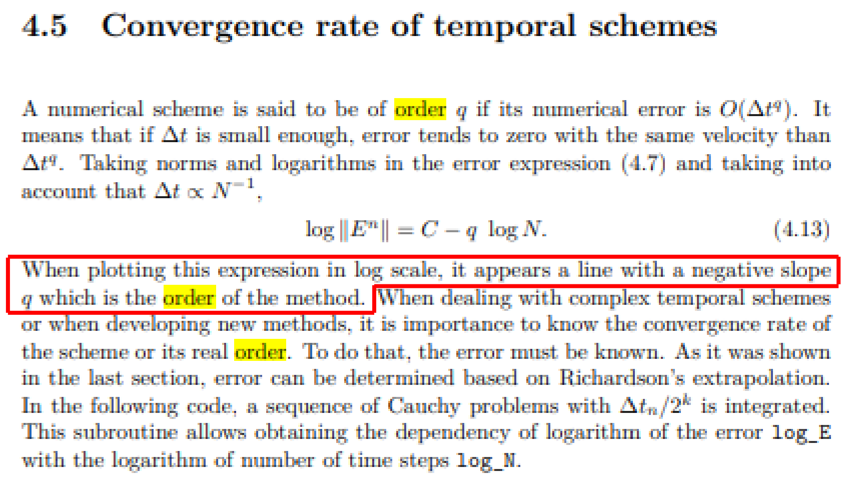


Convergence rates of temporal schemes

Bibilography:

[1] J.A. Hernández, J.Escoto (2017). *How to learn applied mathematics through modern Fortran*

[2] J.A. Hernández, M.A. Zamecnik (2019). *Interpolación polinómica de alto orden. Métodos espectrales. Aplicación a problemas de contorno y de condiciones iniciales*

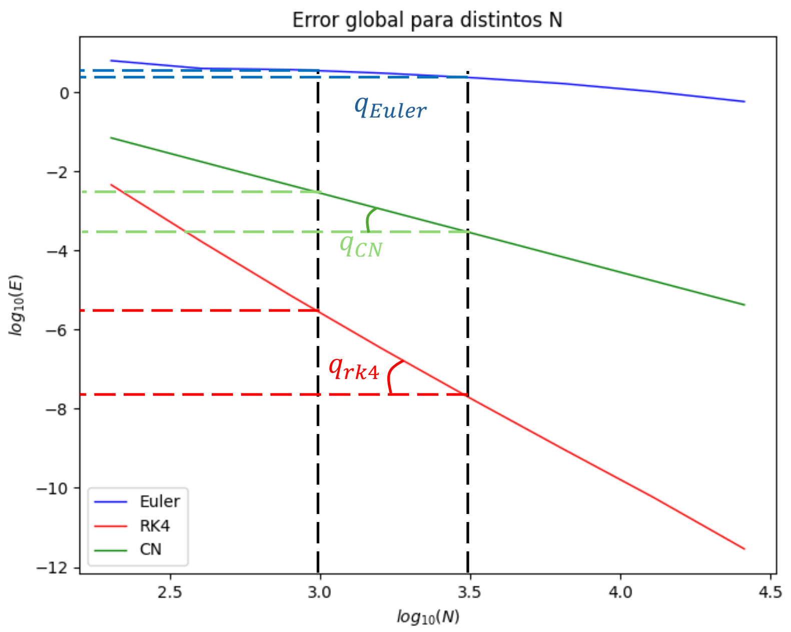


4.5 Convergence rate of temporal schemes

A numerical scheme is said to be of order q if its numerical error is $O(\Delta t^q)$. It means that if Δt is small enough, error tends to zero with the same velocity than Δt^q. Taking norms and logarithms in the error expression (4.7) and taking into account that $\Delta t \propto N^{-1}$,

$$\log \|E^n\| = C - q \ \log N. \tag{4.13}$$

When plotting this expression in log scale, it appears a line with a negative slope q which is the order of the method. When dealing with complex temporal schemes or when developing new methods, it is importance to know the convergence rate of the scheme or its real order. To do that, the error must be known. As it was shown in the last section, error can be determined based on Richardson's extrapolation. In the following code, a sequence of Cauchy problems with $\Delta t_n/2^k$ is integrated. This subroutine allows obtaining the dependency of logarithm of the error `log_E` with the logarithm of number of time steps `log_N`.



$$q = \frac{\log_{10}(E_2) - \log_{10}(E_1)}{\log_{10}(N_2) - \log_{10}(N_1)}$$

Bibliography:

[1] J.A. Hernández, J. Escoto (2017). *How to learn applied mathematics through modern Fortran*