Understanding the rational approximation of the exponential integrator (REXI)

Martin Schreiber <M.Schreiber@exeter.ac.uk>
Pedro S. Peixoto <pedrosp@ime.usp.br>
et. al.

August 6, 2015

!!!THIS IS A PRELIMINARY, NON PROOF-READ DOCUMENT!!!

This document serves as the basis for implementing the Rational approximation of the EXponential Integrator (REXI). Here, we purely focus on the linear part of the shallow-water equations (SWE) and show the different steps to approximate solving this linear part with an exponential integrator. This paper mainly summarises previous work on REXI.

1 Problem formulation

We use linearised shallow water equations (SWE) with respect to a rest state with mean water depth of H and defined for perturbations of height h (see [1]). The linear operator (L) may be written as

$$L(U) := \begin{pmatrix} 0 & H\delta_x & H\delta_y \\ g\delta_x & 0 & -f \\ g\delta_y & f & 0 \end{pmatrix} U$$

where $U := (h, u, v)^T$. Here, we neglect all non-linear terms and consider f constant (f-plane approximation).

The time evolution of the PDE, with the subscript t denoting the derivative in time, is given by

$$U_t = L(U).$$

It is further worth noting, that this system describes an oscillatory system (2D wave equation), hence the operator L is hyperbolic and has imaginary eigenvalues.

2 Exponential integrator

Linear initial value differential problems are well known to be solvable with exponential integrators for arbitrary time step sizes via

$$U(t) = e^{Lt}U(0).$$

see e.g. [5]. However, this is typically quite expensive to compute and analytic solutions only exist for some simplified system of equations, see e.g. [1] for f-plane shallow-water equations. These exponential integrators can be approximated with rational functions and this paper is on giving insight into this approximation.

3 Underlying idea of rational approximation

Terry et. al. [2] developed a rational approximation of the exponential integrator. First, we like to get more insight into it with a one-dimensional formulation before applying REXI to a rational approximation of a linear operator. Our main target is to find an approximation of an operator with a *complex exponential shape*, in our case e^{ix}, which (in one-dimension) is given as a function $f(x)$. We will end up in an approximation given by the following rational approximation:

$$e^{ix} \approx \sum_{n=-N}^{N} \frac{\beta_n}{ix - \alpha_n}$$

with complex coefficients α_n and β_n. We point out that the coefficients α_n will always have non zero real part, so no singularity occurs with the rational function.

3.1 Step A) Approximation of solution space

First, we assume that we can use Gaussian curves as basis functions for our approximation So first we find an approximation of one of our underlying Gaussian basis function

$$\psi_h(x) := (4\pi)^{-\frac{1}{2}} e^{-x^2/(4h^2)}$$

In this formulation, h can be interpreted as the horizontal "stretching" of the basis function. Note the similarities to the Gaussian distribution, but by dropping certain parts of the vertical scaling as it is required for probability distributions. We can now approximate our function $f(x)$ with a superposition of basis functions $\psi_h(x)$ by

$$f(x) \approx \sum_{m=-M}^{M} b_m \psi_h(x + mh)$$

with M controlling the interval of approximation (~size of “domain of interest”) and h will be related to the accuracy of integration (~resolution in “domain of interest”).

We choose h small enough so that the support of the Fourier transform of f is mainly localised within $[-1/(2h), 1/(2h)]$ (i.e. almost zero outside this interval). M is chosen such that the approximation will be adequate in the interval $|x| < Mh$. M and h are connected in the sense that to obtain a good approximation for a larger interval, M needs to be larger and h smaller.

To compute the coefficients b_m, we rewrite the previous equation in Fourier space with

$$\frac{\hat{f}(\xi)}{\hat{\psi_h}(\xi)} = \sum_{m=\infty}^{\infty} b_m e^{2\pi i m h \xi},$$

where the $\hat{\cdot}$ symbols indicate the Fourier transforms of the respective functions. The b_m are now the Fourier coefficients of the series for the function $\frac{\hat{f}(\xi)}{\hat{\psi_h}(\xi)}$ and be calculated as [1],

$$b_m = h \int\limits_{-\frac{1}{2h}}^{\frac{1}{2h}} e^{-2\pi i m h \xi} \frac{\hat{f}(\xi)}{\hat{\psi_h}(\xi)} d\xi,$$

for $m \in \mathbb{Z}$.

Since we are interested in approximating $f(x) = e^{ix}$, we can simplify the equation by using the response in frequency space $\hat{f}(\xi) = \delta(\xi - \frac{1}{2\pi})$, where here δ is the Dirac distribution, and

$$b_m = h\, e^{-imh} \hat{\psi_h}(\frac{1}{2\pi})^{-1}.$$

The Fourier transform of the Gaussian function is well known and given by

$$\hat{\psi_h}(\xi) = \int\limits_{-\infty}^{\infty} \frac{1}{\sqrt{4\pi}} e^{-\left(\frac{x}{2h}\right)^2} e^{-2\pi i x \xi} dx = h e^{-(2h\pi\xi)^2}$$

where we used that $\int_{-\infty}^{\infty} e^{-\left(\frac{x}{2h}\right)^2} dx = h\sqrt{4\pi}$. For the case $\xi = \frac{1}{2\pi}$, we get

$$\hat{\psi_h}\left(\frac{1}{2\pi}\right) = h\, e^{-h^2}.$$

Finally, one can obtain the equation

$$b_m = h\, e^{-imh} \frac{1}{h\, e^{-h^2}} = e^{-imh} e^{h^2}$$

to compute the coefficients b_m for $f(x) = e^{ix}$.

[1] see [2], page 11

3.2 Step B) Approximation of basis function

The second step is the approximation of the basis function $\psi_h(x)$ itself with a rational approximation, see [6]. Our basis function is given by

$$\psi_h(x) := (4\pi)^{-\frac{1}{2}} e^{-x^2/(4h^2)}$$

and a close-to-optimal approximation of $\psi_1(x)$ with a sum of rational functions is given by

$$\psi_1(x) \approx Re\left(\sum_{l=-L}^{L} \frac{a_l}{ix + (\mu + i\,l)}\right)$$

with the μ and a_l given in [6], Table 1. We can generalise this approximation to arbitrary chosen h via

$$\psi_h(x) \approx Re\left(\sum_{l=-L}^{L} \frac{a_l}{i\frac{x}{h} + (\mu + i\,l)}\right)$$

3.3 Step C) Approximation of the approximation

We then combine the approximation (B) of the approximation (A), yielding

$$f(x) \approx \sum_{m=-M}^{M} b_m \psi_h(x + mh) = \sum_{m=-M}^{M} b_m Re\left(\sum_{l=-L}^{L} \frac{a_l}{i\frac{x+mh}{h} + (\mu + i\,l)}\right)$$

$$= \sum_{m=-M}^{M} b_m \sum_{l=-L}^{L} Re\left(\frac{ha_l}{ix + h(\mu + i(m+l))}\right).$$

We further like to simplify this equation and we observe, that for $n := m + l$, the denominator is equal. We can hence express parts of the denominator in terms of n by

$$\alpha_n := h(\mu + in).$$

Now, we merge the b_m and a_l coefficients and first have a look at the b_m which is complex values. We observe the following property: Assuming that we want to compute the real value of $f(x)$, only the real value of b_m has to be merged with the sum, since the imaginary component would be dropped afterwards. This allows us to move the $Re(b_m)$ values inside the $\sum_L$:

$$Re(f(x)) := Re\left(\sum_{m=-M}^{M} \sum_{l=-L}^{L} \frac{Re(b_m)\, a_l}{ix + s(\mu + i(m+l))}\right).$$

Now we can collect all nominators with equivalent denominator (if $n = m + l$ and by using δ as the Kronecker delta), yielding

$$\beta_n^{Re} := \sum_{m=-M}^{M} \sum_{l=-L}^{L} Re(b_m) a_l \delta(n,\, m + l)$$

for real values $f(x)$ and

$$\beta_n^{Im} := \sum_{m=-M}^{M} \sum_{l=-L}^{L} Im(b_m) a_l \delta(n,\, m + l)$$

for complex values of $f(x)$. This finally leads us to the REXI approximation

$$e^{ix} \approx \sum_{n=-N}^{N} Re\left(\frac{\beta_n^{Re}}{ix + \alpha_n}\right) + i\, Re\left(\frac{\beta_n^{Im}}{ix + \alpha_n}\right)$$

for the complex-valued function e^{ix}.

4 Apply REXI with a matrix:

Finally, we like to apply REXI to a formulation such as

$$U(t) := e^{tL} U(0).$$

To see the relationship between the approximation of e^{ix} with REXI, we first rewrite the exponential formulation in terms of $f(x) := e^{itx}$

$$f(L) := e^{tL} = \Sigma \Lambda \Sigma^H = \Sigma \begin{pmatrix} ... & & \\ & e^{i\lambda_n t} & \\ & & ... \end{pmatrix} \Sigma^H = \Sigma\, f(\lambda_n)\, \Sigma^H$$

with complex-valued exponentials on the eigenvalues. Hence, the accuracy of the exponential integrator on $f(L)$ only depends on the spectrum of the L and allows to be applied in the same way as e^{ix}, but by replacing x with the matrix L. For error bounds, we like to refer to [2].

5 Filtering

Since we are approximating the exponential function with a series, one of the most important properties can be violated: The evaluation of e^{ix} is bounded by unity (think of it as a series of real-valued *cos* and imaginary-valued *sin*).

However, the interpolated values can exceed this unity due to interpolation properties (think of a Lagrangian interpolation of high order, leading to large oscillations with the possibility of exceeding the local min/max of the interpolated function in the area of support). This can lead to long-term effects such as amplifying unphysical solutions. Therefore a filtering may be required to assure that the function in the interpolation range is always bounded by unity.

6 REXI, our little dog

In the following, we use $L := \tau L'$ and assume an a-priori fixed time step size, making a REXI approximation more efficient. Then, the REXI approximation is given by

$$exp(\tau L') \approx \sum_{k=-K}^{K} \beta_k (L - \alpha_k)^{-1} \tag{1}$$

The coefficients α_k (corresponding to $s(\mu + i\,n)$ in step C for the one-dimensional formulation) can be precomputed or computed during program start. μ is based on a one-dimensional approximation, see the paper, and the α_k can be interpreted as shifts of the rational approximations. The coefficients β_k (corresponding to c_n in step C) are describing the scaling of the basis function and are also constant and independent of the solution itself. Note, that for debugging purpose, their *imaginary values have to cancel out.*

Note an important property (see Sec. 3.3 in [2]). There's an anti-symmetry in the α_i coefficients, which avoids computing half of the inverses:

$$\overline{(L - \alpha)^{-1} U(0)} = (L - \overline{\alpha})^{-1} U(0)$$

7 Computing inverse of $(L - \alpha)^{-1}$

For computing the inverse, arbitrary solvers can be used. However we like to note, that α is a complex number. Hence, requiring solvers with support for solving in complex space. As an example, we consider a specialization on the shallow-water equations given above with

$$L(U(t)) := \begin{pmatrix} & H\delta_x & H\delta_y \\ g\delta_x & & -f \\ g\delta_y & f & \end{pmatrix} U(t)$$

$$U_t(t) := L(U(t))$$

and we set $g := 1$ and the average height $H := 1$.

7.1 Handling τ in REXI

We recall the formulation of the solution as an exponential integrator

$$U(t) := e^{tL} U(0)$$

which formally allows us to join the integration in time givey by t with the L operator in case of such a formulation. There are basically two different ways to handle this scaling:

The first one is rescaling all parameters by τ:

$$g' := \tau g, \quad f' := \tau f, \quad h_0' := \tau h_0.$$

The second way is to reformulate the REXI approximation scheme given by

$$(\tau L - \alpha)^{-1}.U(\tau) = U(0)$$

and by factoring τ out, yielding

$$(L - \frac{\alpha}{\tau})^{-1}.U(\tau)\tau^{-1} = U(0)$$

So instead of solving for $U(\tau)$, we are solving for $U^\tau(\tau) := U(\tau)\tau^{-1}$ as well as $\alpha^\tau := \frac{\alpha}{\tau}$.

To summarize, we have to solve the system of equations given by

$$(L - \alpha^\tau)^{-1}.U^\tau(1) = U(0) \tag{2}$$

with $U(0)$ the initial conditions, $\alpha^\tau := \frac{\alpha}{\tau}$ and $U(\tau) := U^\tau(1)\tau$. For sake of simplicity, we stick to the formulation without the prime notation.

One final scaling has to be done: the exponential is computing $e^{\tau L}$, hence the real-valued τ has to be included in the operator L. There are basically two different ways: The first one is rescaling all parameters by τ:

$$g^\tau := \tau g, \quad f^\tau := \tau f, \quad h_0^\tau := \tau h_0$$

The second way is to factor the τ parameter out:

$$(\tau L - \alpha)^{-1}.U(\tau) = U(0)$$

$$\left(L - \frac{\alpha}{\tau}\right)^{-1}.U(\tau)\tau^{-1} = U(0)$$

So instead of solving for $U(\tau)$, we are solving for $U^\tau(\tau) := U(\tau)\tau$ as well as $\alpha^\tau := \frac{\alpha}{\tau}$and have to divide the computed solution by τ in the end.

7.2 Solving as an eliptic problem

[TODO (Pedro): Derive dimensional formulation]

We like to mention again, that we can use arbitrary solvers and in this work, we focus on a reformulation into an eliptic problem. Following the idea in [4], instead of solving this relatively large system of equations we can split the problem into an elliptic one for the height which then allows to use an explicit formulation for the velocities. We use the abbreviation$\vec{v} := (u, v)$ in the following paragraph and the formulation with a unit time step (see Eq. 2). Using the formulation in [2], the height can be computed with the elliptic equation given by

$$(\nabla^2 - (\alpha^2 + f^2))h(1) = \frac{\alpha^2 + f^2}{\alpha}(h(0) + H\nabla \cdot (A\,\vec{v}(0)) \tag{3}$$

with

$$A := \frac{1}{\alpha^2 + f^2} \left(\begin{array}{cc} \alpha & -f \\ f & \alpha \end{array} \right).$$

7.3 f-plane

Assuming an f-plane approximation (f is constant), we can rearrange this equation by using the abbreviations $\kappa := \alpha^2 + f^2$ in the following way:

$$(\nabla^2 - \kappa)\, h(1) = \frac{\kappa}{\alpha}\, (h(0) + \nabla \cdot (A\,\vec{v}(0))$$

$$(\nabla^2 - \kappa)\, h(1) = \frac{\kappa}{\alpha} h(0) + \frac{1}{\alpha}\nabla \cdot \left(\begin{array}{cc} \alpha & -f \\ f & \alpha \end{array} \right) \vec{v}(0)$$

$$(\nabla^2 - \kappa)h(1) = \frac{\kappa}{\alpha} h(0) - \frac{f}{\alpha}\nabla \times v(0) + \nabla \cdot \vec{v}(0) \tag{4}$$

Here, the α and κ denote the terms with imaginary numbers and this formulation should also simplify programming.

We continue with an interpretation of this formulation: on the right hand side we see an update-like scheme $h(0)$ in the first scheme, then a vorticity-like formulation $\times$, and an advective part ∇. To simplify the notation for solving the system, we rewrite it as

$$(\nabla^2 - \kappa)\, h(1) = D \tag{5}$$

with real-and-imaginary-valued D and $\nabla^2 - \kappa$ as well as a real-and-imaginary-valued $h(1)$ for which we want to solve. Then, the solution is given e.g. in spectral space directly via

$$h(1) := D(\nabla^2 - \kappa)^{-1}$$

Once computed the height, the velocities can be directly computed via

$$\vec{v}(1) = -A.\vec{v}(0) + A.\nabla h(1) = -A.(\vec{v}(0) - \nabla h(1)) \tag{6}$$

giving us our final solution

$$U(\tau) := \tau(h(1),\, u(1),\, v(1))^T$$

with the scaling with τ as discussed in Sec. 7.1 and we like to mention, that also α has to be scaled appropriately before using it for REXI.

7.4 Interpretation of τ

We like to close this section with a brief discussion of τ by having a look on the REXI reformulation

$$(L - \frac{\alpha}{\tau})^{-1}.U(\tau)\tau = U(0)$$

We see, that for an increasing τ, hence an integration in time over a larger time period, the poles given by α are getting closer. This can possibly lead to a loss in accuracy for the data sampled by the outer poles α_{-N} and α_N. Therefore, the number N of poles is expected to scale linearly with the size of the coarse time step:

$$|N| \sim \tau$$

[TODO (Terry): There's probably a tighter relationship somewhere hidden in the paper]

8 Bringing everything together

Using the spectral methods (e.g. in SWEET), we can directly solve the height Eq. (3) and then solver for the velocity in Eq. (6). Then, the problem is reduced to computing the REXI as given in Eq. (1). We like to note again, that the α_n and β_n are independent of the system L to solve, and the number of coefficients only depends on the accuracy and the resolution.

9 Notes on HPC

- The terms in REXI to solve are all independent. Hence, for latency avoiding, the communication can be interleaved with computations.
- The iterative solvers are memory bound. Instead of computing $c := a * b$ for the stencil operations, we could compute $\vec{c} := a\vec{b}$ with a one coefficient in the stencil. This allows vectorization over c and b on accelerator cards with strided memory access.
- It is unknown which method is more efficient to solve the system of equations:
  - iterative solvers have low memory access,
  - inverting the system and storing it as a sparse matrix allows fast direct solving but can yield more memory access operations.
- Splitting the solver into real and complex number would store them consecutively in memory. This has a potential to avoid non-strided memory access and using the same SIMD operations (Just a rough idea, TODO: check if this is really the case).

10 Acknowledgements

Thanks to Pedro & Terry for the feedback & discussions!

References

[1] Formulations of the shallow-water equations, M. Schreiber

[2] High-order time-parallel approximation of evolution operators, T. Haut et. al.

[3] An asymptotic parallel-in-time method for highly oscillatory PDEs, T. Haut et. al.

[4] An invariant theory of the linearized shallow water equations with rotation and its application to a sphere and a plane, N. Paldor et. al.

[5] Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later, Cleve Moler and Charles Van Loan, SIAM review

[6] Near optimal rational approximations of large data sets, Damle, A., Beylkin, G., Haut, T. S. & Monzon