Introduction to Splitting Methods for PDEs (by A. Ostermann* and M. Caliari)

M. Caliari

a.y. 2016/2017

*University of Innsbruck (Austria)

Chapter 1

FFT Matlab/GNU Octave friendly

1.1 1-dimensional

We consider, for N even,

$$\hat{u}(x) = \sum_{j=-N/2}^{N/2-1} \hat{u}_{j+1+N/2} \frac{\mathrm{e}^{\mathrm{i}j2\pi(x-a)/(b-a)}}{\sqrt{b-a}} = \sum_{j=1}^{N} \hat{u}_j \frac{\mathrm{e}^{\mathrm{i}(j-1-N/2)2\pi(x-a)/(b-a)}}{\sqrt{b-a}} =$$
$$= \sum_{j=1}^{N} \hat{u}_j \phi_j(x)$$

where $\hat{u}_j$ is the approximation by trapezoidal quadrature rule of

$$\int_a^b u(x)\overline{\phi_j(x)}\mathrm{d}x$$

(we assume $u(x)$ periodic in $[a, b]$) that is

$$u_j = \int_a^b u(x)\overline{\phi_j(x)}\mathrm{d}x = \sqrt{b-a}\int_0^1 u(a+y(b-a))\mathrm{e}^{-\mathrm{i}(j-1)2\pi y}\mathrm{e}^{iN\pi y}\mathrm{d}y \approx$$
$$\approx \frac{\sqrt{b-a}}{N} \boxed{\sum_{n=1}^{N} \left(u(x_n)\mathrm{e}^{\mathrm{i}N\pi y_n}\right)\mathrm{e}^{-\mathrm{i}(j-1)2\pi y_n}} = \hat{u}_j$$

where $y_n = (n-1)/N$ and $x_n = a + (b-a)y_n$. We say that the 0 frequency is in the center of spectrum. What is written in the box is the result of `fft`

(`[`$u(x_1)\mathrm{e}^{\mathrm{i}N\pi y_1}, \ldots, u(x_N)\mathrm{e}^{\mathrm{i}N\pi y_N}$`] )`. We consider now, for $1 \le j \le N/2$,

$$\texttt{fft ([}u(x_1), \ldots, u(x_N)\texttt{])}_j = \sum_{n=1}^{N} u(x_n)\mathrm{e}^{-\mathrm{i}(j-1)2\pi y_n} =$$

$$= \sum_{n=1}^{N} \left(u(x_n)\mathrm{e}^{\mathrm{i}N\pi y_n}\right) \mathrm{e}^{-\mathrm{i}(N/2+j-1)2\pi y_n} = \frac{N}{\sqrt{b-a}} \hat{u}_{N/2+j}$$

and, for $N/2 < j \le N$,

$$\texttt{fft ([}u(x_1), \ldots, u(x_N)\texttt{])}_j = \sum_{n=1}^{N} u(x_n)\mathrm{e}^{-\mathrm{i}(j-1)2\pi y_n} =$$

$$= \sum_{n=1}^{N} \left(u(x_n)\mathrm{e}^{\mathrm{i}N\pi y_n}\right) \mathrm{e}^{-\mathrm{i}(j-N/2-1)2\pi y_n} = \frac{N}{\sqrt{b-a}} \hat{u}_{j-N/2}$$

taking into account that $\mathrm{e}^{iN2\pi y_n} = 1$. Therefore `uhat = fftshift (fft (u)) * sqrt (b - a) / N`. Then we have

$$\hat{\hat{v}}_n = \sum_{k=1}^{N} \hat{v}_k \phi_k(x_n) = \sum_{k=1}^{N} \hat{v}_k \frac{\mathrm{e}^{\mathrm{i}(k-1-N/2)2\pi(x_n-a)/(b-a)}}{\sqrt{b-a}} =$$

$$= \frac{N}{\sqrt{b-a}} \boxed{\frac{1}{N}\left(\sum_{k=1}^{N} \hat{v}_k \mathrm{e}^{\mathrm{i}(k-1)2\pi y_n}\right)} \mathrm{e}^{-\mathrm{i}N\pi y_n}$$

What written in the box is the result of `ifft (vhat)`. We observe that $\mathrm{e}^{-\mathrm{i}N\pi y_n} = (-1)^{n+1}$. We consider now

$$\sum_{k=1}^{N} \texttt{ifftshift ([}\hat{v}_1, \ldots, \hat{v}_N\texttt{])}_k \mathrm{e}^{\mathrm{i}(k-1)2\pi y_n} = \sum_{k=1}^{N/2} \hat{v}_{N/2+k} \mathrm{e}^{\mathrm{i}(k-1)2\pi y_n} + \sum_{k=N/2+1}^{N} \hat{v}_{k-N/2} \mathrm{e}^{\mathrm{i}(k-1)2\pi y_n} =$$

$$= \sum_{k=N/2+1}^{N} \hat{v}_k \mathrm{e}^{\mathrm{i}(k-N/2-1)2\pi y_n} + \sum_{k=1}^{N/2} \hat{v}_k \mathrm{e}^{\mathrm{i}(N/2+k-1)2\pi y_n} =$$

$$= (-1)^{n+1} \sum_{k=1}^{N/2} \hat{v}_k \mathrm{e}^{\mathrm{i}(k-1)2\pi y_n} + (-1)^{n+1} \sum_{k=N/2+1}^{N} \hat{v}_k \mathrm{e}^{\mathrm{i}(k-1)2\pi y_n} =$$

$$= \left(\sum_{k=1}^{N} \hat{v}_k \mathrm{e}^{\mathrm{i}(k-1)2\pi y_n}\right) \mathrm{e}^{-\mathrm{i}N\pi y_n}$$

Therefore, `vhathat=ifft (ifftshift (vhat)) * N / sqrt (b - a)`. It is not difficult to prove that

$$\hat{u}(x_n) = u(x_n)$$

that is, $\hat{u}(x)$ is an approximation of $u(x)$ which interpolates $u(x)$ at x_n, $n = 1, 2, \ldots, N$.

Chapter 2

Linear systems

2.1 Unsymmetric systems

2.2 Symmetric systems

Chapter 3

Finite Differences

3.1 1-dimensional

On `x = linspace (a, b, n)'`, with `h = (b - a) / (n - 1)` we can construct the discretization matrices of first derivative and second derivative

```
Dx = toeplitz (sparse (1, 2, -1 / (2 * h), 1, n), ...
               sparse (1, 2, 1 / (2 * h), 1, n));
Dxx = toeplitz (sparse ([1, 1], [1, 2], [-2, 1] / h ^ 2, 1, n));
```

3.1.1 Boundary conditions

Dirichlet b.c.

Periodic b.c.

Neumann b.c.

3.2 2-dimensional

Let's check how to construct a grid in Matlab/GNU Octave.

```
x = linspace (a, b, n)';
hx = (b - a) / (n - 1);
Dx = toeplitz (sparse (1, 2, -1 / (2 * hx), 1, n), ...
                sparse (1, 2, 1 / (2 * hx), 1, n));
Dxx = toeplitz (sparse ([1, 1], [1, 2], [-2, 1] / hx ^ 2, 1, n));
y = linspace (c, d, m)';
hy = (d - c) / (m - 1);
Dy = toeplitz (sparse (1, 2, -1 / (2 * hy), 1, m), ...
```

```
                 sparse (1, 2, 1 / (2 * hy), 1, m));
Dyy = toeplitz (sparse ([1, 1], [1, 2], [-2, 1] / hy ^ 2, 1, m));
[X, Y] = ndgrid (x, y);
```

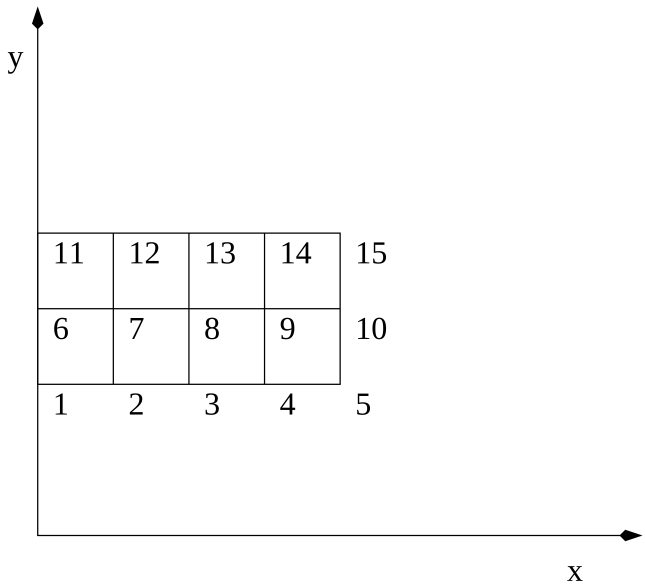

Figure 3.1: 2-dimensional grid.

For instance, for `x = linspace (0, 4, 5)'` and `y = linspace (2, 4, 3)'` it corresponds to the grid in Figure 3.1. Point i has coordinates `X(i)`,`Y(i)`. Matrices of first and second partial derivatives are obtained by

```
DDx = kron (speye (m), Dx);
DDy = kron (Dy, speye (n));
DDxx = kron (speye (m), Dxx);
DDyy = kron (Dyy, speye (n));
```

Given a function `U = f (X, Y)`, it is possible to approximate its partial derivative with respect to x (but pay attention to the boundaries!) by

```
reshape (DDx * U(:), n, m)
```

Chapter 4

Matrix functions

4.1 Matrix exponential

The matrix exponential for $A \in \mathbb{C}^{n\times n}$ is defined as

$$\exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!}$$

In order to numerically evaluate it, we may think to two simple strategies: Taylor's truncated series and eigenvalue decomposition.

4.1.1 Taylor's truncated series

It is a strong temptation to approximate

$$\exp(A) \approx \sum_{k=0}^{m} \frac{A^k}{k!} = T_m(A)$$

Unfortunately, it does not work, even in simple scalar cases (try to compute the relative error $\mathrm{e}^{100}(\mathrm{e}^{-100} - T_m(-100))$ for increasing values of m). The problem is that Taylor's series, although everywhere convergent, is fast enough (that is it converges to machine precision before truncation errors show up) only in a neighborhood of 0. To this aim, the *scaling and squaring* technique may help. That is, taken $s = 2^j$ such that $\|A\| < 1$, $E = \exp(A/s)$ is approximated by $T_m(A/s)$ and later $\exp(A)$ is recovered by

$$E = E^2 \ j \text{ times}$$

Padé approximation

We can change the way $\exp(A/s)$ is approximated. For instance, a very common approximation is the rational *Padé* one. That is

$$e^z \approx \frac{a_p z^p + a_{p-1}z^{p-1} + \ldots + a_1 z + a_0}{b_q z^q + b_{q-1}z^{q-1} + \ldots + b_1 z + 1} = r_{p,q}(z)$$

such that $T_{p+q}(z) = r_{p,q}(z) + \mathcal{O}(z^{p+q+1})$ for $z \to 0$. Let's try for $p = q = 1$: we have

$$e^z \approx \frac{\frac{1}{2}z + 1}{-\frac{1}{2}z + 1} = r_{1,1}(z)$$

By the way, this corresponds to the trapezoidal rule $y_1 = \left(\frac{1}{2}k + 1\right) / \left(-\frac{1}{2}k + 1\right) = r_{1,1}(k)$ for the solution of $y'(t) = y(t)$, $y(0) = 1$ at $t = k$. The extension to the matrix case is trivial

$$\exp(A/s) \approx \left(-\frac{1}{2}A/s + I\right)^{-1} \left(\frac{1}{2}A/s + I\right)$$

Then, the scaling and squaring technique is used. In practice, the degree $p = q$ of common Padé approximations is around 10. Padé approximations for the exponential share the following property

$$r_{p,p}(-z) = \frac{1}{r_{p,p}(z)}$$

4.1.2 Eigenvalue decomposition

If A is diagonalizable, $AV = V\Lambda$, then

$$\exp(A) = V\exp(\Lambda)V^{-1}$$

where $\exp(\Lambda)$ is easily seen to be $\operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Unfortunately, even in this case it is possible to do very wrong, try with

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1+\varepsilon \end{bmatrix}$$

and compare in Matlab

```
A=[1,1;0,1+eps];,expm(A),[V,Lambda]=eig(A);,V*diag(exp(diag(Lambda)))/V
```

4.2 Matrix exponential-related functions

We would like to approximate the following functions

$$\varphi_\ell(A) = \sum_{k=0}^{\infty} \frac{A^k}{(k+\ell)!}$$

We have $\varphi_0(z) = \mathrm{e}^z$,

$$\varphi_1(z) = \frac{\mathrm{e}^z - 1}{z}$$

and, in general,

$$\begin{aligned} \varphi_\ell(z) &= z\varphi_{\ell+1}(z) + \frac{1}{\ell!}, && \ell \geq 0 \\ \varphi_\ell(z) &= \frac{1}{(\ell-1)!}\int_0^1 \mathrm{e}^{(1-\theta)z}\theta^{\ell-1}\mathrm{d}\theta, && \ell \geq 1 \end{aligned}$$

Of course it is possible a Taylor or Padé approximation in a neighborhood of 0. The scaling and squaring technique is more involved, however. For instance

$$\varphi_1(z) = \frac{1}{2}(\mathrm{e}^{z/2} + 1)\varphi_1\left(\frac{z}{2}\right)$$

The function ϕ_1 is very important. For instance, the solution of a linear, constant coefficients system of ODEs

$$\begin{cases} y'(t) = Ay(t) + b \\ y(t_0) = y_0 \end{cases}$$

is

$$y(t) = y_0 + (t - t_0)\varphi_1((t - t_0)A)b$$

If one is interested only in $\varphi_\ell(A)w$, $w \in \mathbb{C}^n$, then the following theorem can be used (see [1]).

Theorem 1. *Let* $A \in \mathbb{C}^{n\times n}$, $W = [w_1, \ldots, w_p] \in \mathbb{C}^{n,p}$, $\tau \in \mathbb{C}$ *and*

$$\tilde{A} = \begin{bmatrix} A & W \\ 0 & J \end{bmatrix} \in \mathbb{C}^{(n+p)\times(n+p)}, \quad J = \begin{bmatrix} 0 & I_{p-1} \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{p\times p}$$

then for $X = \exp(\tau\tilde{A})$ *we have*

$$X(1:n, n+j) = \sum_{k=1}^{j} \tau^k \varphi_k(\tau A)w_{j-k+1}, \quad j = 1, 2, \ldots, p$$

Before giving the proof, let us consider a simple example: we want to compute $\varphi_1(\tau A)w$. We consider

$$\tilde{A} = \begin{bmatrix} A & w \\ 0 & 0 \end{bmatrix}$$

and compute $X = \exp(\tau\tilde{A})$, extract the first n rows, last column and divide by τ.

```
n=4;
tau=rand;
A=rand(n);
w=rand(n,1);
Atilde=[A,w;zeros(1,n),0];
X=expm(tau*Atilde);
X(1:n,n+1)/tau
(tau*A)\((expm(tau*A)-eye(n))*w)
```

If interested into $\exp(\tau A)v + \tau\varphi_1(\tau A)w$, we can do

```
n=4;
tau=rand;
A=rand(n);
v=rand(n,1);
w=rand(n,1);
eta=2^(-ceil(log2(norm(w,1))));
Atilde=[A,eta*w;zeros(1,n),0];
X=expm(tau*Atilde)*[v;1/eta];
X(1:n)
expm(tau*A)*v+tau*((tau*A)\((expm(tau*A)-eye(n))*w))
```

The use of the parameter η is for numerical stability (see [1]).

The possibility to compute $\varphi_\ell(A)w$ without computing $\varphi_\ell(A)$ is similar to the possibility to compute the solution of $Ax = w$ without computing A^{-1}.

Proof. We start computing

$$\tilde{A}^2 = \begin{bmatrix} A & W \\ 0 & J \end{bmatrix} \begin{bmatrix} A & W \\ 0 & J \end{bmatrix} = \begin{bmatrix} A^2 & AW + WJ \\ 0 & J^2 \end{bmatrix}$$

and we easily get

$$\tilde{A}^k = \begin{bmatrix} A^k & M_k \\ 0 & J^k \end{bmatrix}$$

with $M_0 = 0$, $M_1 = W$, $M_k = A^{k-1}W + M_{k-1}J$. Then $WJ(:,j) = w_{j-1}$ and $JJ(:,j) = J(:,j-1)$ for $1 \le j \le p$ where we define $w_0 = J(:,0) = 0$. Thus

$$\begin{aligned} M_k(:,j) &= A^{k-1}w_j + (A^{k-2}W + M_{k-2}J)J(:,j) = \\ &= A^{k-1}w_j + A^{k-2}w_{j-1} + M_{k-2}J(:,j-1) = \\ &= \sum_{i=1}^{\min\{k,j\}} A^{k-i}w_{j-i+1} \end{aligned}$$

Moreover

$$X(1:n, n+1:n+p) = \sum_{k=0}^{\infty} \frac{\tau^k M_k}{k!} = \sum_{k=1}^{\infty} \frac{\tau^k M_k}{k!}$$

and now we can compute

$$\begin{aligned} X(1:n, n+j) = \sum_{k=1}^{\infty} \frac{\tau^k M_k(:,j)}{k!} &= \sum_{k=1}^{\infty} \frac{1}{k!} \left(\sum_{i=1}^{\min\{j,k\}} \tau^i (\tau A)^{k-i} w_{j-i+1} \right) = \\ &= \sum_{i=1}^{j} \tau^i \left(\sum_{k=i}^{\infty} \frac{(\tau A)^{k-i}}{k!} \right) w_{j-i+1} = \\ &= \sum_{i=1}^{j} \tau^i \left(\sum_{k=0}^{\infty} \frac{(\tau A)^{k}}{(k+i)!} \right) w_{j-i+1} = \sum_{i=1}^{j} \tau^i \varphi_i(\tau A) w_{j-i+1} \end{aligned}$$

$\square$

Bibliography

[1] A. H. Al-Mohy and N. J. Higham. Computing the action of the matrix exponential with an application to exponential integrators. *SIAM J. Sci. Comput.*, 33(2):488–511, 2011.