

SIMUMAT SUMMER SCHOOL 2009

NUMERICAL METHODS FOR FORWARD AND BACKWARD PARABOLIC PROBLEMS BASED ON CONFORMAL TRANSFORMATION

María López Fernández
Instituto de Ciencias Matemáticas (CSIC-UAM-UC3M-UCM)

The numerical recovery of holomorphic mappings in the unit disk from a finite set of approximate values. We consider the numerical approximation of a holomorphic mapping f in the unit disk D from a finite set of values, $f(z_1), \dots, f(z_N)$. We will consider the situation when the interpolation nodes z_j $j = 1, \dots, N$ are located on the real segment inside D $[-r, r]$, $0 < r < 1$. More precisely, we will choose the Chebychev nodes. This is a classical ill-posed problem, meaning that small perturbations on the data (the values $f(z_j)$) may lead to big errors in the final result. We will follow the approach in [3], where the effect of these errors is taken into account.

We create the file `recuho10.m` to implement the following experiment for the mapping:

$$f(z) = \frac{1}{1.1 + z}, \quad z \in D,$$

for $D = \{z \in \mathbb{C} : |z| < 1\}$.

1. We set $0 < r < 1$, $N \geq 1$ and compute the Chebychev nodes in the interval $[-r, r]$:

```
r=.3; N=8;  
z = r*cos(pi*[0:N-1]/(N-1));
```

2. We evaluate f at z :

```
sol = 1./(1.1+z).';
```

3. We fix ε and add random perturbations of maximum size ε to the computed values of f :

```
epsilon = 1e-6;  
dsol = epsilon*rand(N,1);  
data = sol + dsol;
```

With these *hand made* data we address the problem of approximating f at any point inside D from the perturbed nodal values $f(z_j) + \delta f(z_j)$.

4. We want to compute the interpolating mapping F_{num} of f in D . This mapping is of the form

$$F_{num}(z) = \sum_{j=1}^N c_j g_j(z), \quad (1)$$

where the interpolation basis $\{g_1(z), \dots, g_N(z)\}$ is given by

$$g_1(z) = \frac{\sqrt{1 - |z_1|^2}}{1 - \bar{z}_1 z} \quad (2)$$

and

$$g_j(z) = \frac{\sqrt{1 - |z_j|^2}}{1 - \bar{z}_j z} \prod_{k=1}^{j-1} \frac{z - z_k}{1 - \bar{z}_k z}, \quad 2 \leq j \leq N. \quad (3)$$

Given an *a priori* estimate M of f in the whole disk, the vector of coefficients $\mathbf{c} = [c_j]_{j=1}^N$ solves the minimization problem

$$\min \|A\mathbf{x} - (\mathbf{f} + \delta\mathbf{f})\|, \quad (4)$$

subject to

$$\|\mathbf{x}\| \leq M, \quad (5)$$

where $A = (a_{k,j})$ is the $N \times N$ matrix defined by:

$$a_{k,j} = g_j(z_k), \quad k, j = 1, \dots, N.$$

The minimization problem is solved by the Least Square Methods (LSM) in combination with a Lagrange multiplier as follows.

- (a) We build the system matrix A :

```
A = sqrt(1-abs(z(1))^2) ./ (1-conj(z(1))*z);
p = 1;
for k = 2:N
p = p .* (z-z(k-1)) ./ (1-conj(z(k-1))*z);
A = [A ; sqrt(1-abs(z(k))^2) ./ (1-conj(z(k))*z) .* p ];
end
A = A.'; %transpose without conjugation of A
```

- (b) For the LSM resolution we use the singular value decomposition (U, S, V) of A . Recall that $AV = US$, where U and V are unitary matrices and S is a diagonal matrix, whose entries are $\sigma_j = +\sqrt{\mu_j}$, for μ_j the (always positive) eigenvalues of A^*A :

```
[U,S,V] = svd(A);
sigma = diag(S); %vector with the singular values of A
```

```

index = find(diag(S)); %remove possible zero singular values
sigma = sigma(index);
omega = U(index,index)'*data(index);
y = omega./sigma;

```

(c) We introduce the constraint:

- If $\|\mathbf{y}\| \leq M$, we take directly $\mathbf{c} = V(\text{index}, \text{index}) * \mathbf{y}$.
- Else, if $\|\mathbf{y}\| > M$, we replace the constraint by $\|x\| = M$ and introduce a positive multiplier λ . In this way, we consider the vector $\mathbf{y}(\lambda) = [y_j(\lambda)]_{j=1}^K$ defined by:

$$y_j(\lambda) = \frac{\sigma_j \omega_j}{\sigma_j^2 + \lambda}, \quad 1 \leq j \leq K, \quad \lambda > 0.$$

We need to find the value λ^* that solves the nonlinear equation

$$N(\lambda) = \|\mathbf{y}(\lambda)\|^2 - M^2 = \sum_{j=1}^K \left| \frac{\sigma_j \omega_j}{\sigma_j^2 + \lambda} \right|^2 - M^2 = 0. \quad (6)$$

Then take $\mathbf{c} = V(\text{index}, \text{index}) * \mathbf{y}(\lambda^*)$.

Program step (c) in MATLAB. Use the routine `newtonsys.m` to solve (6). Recall that you will need to create an auxiliary *.m file to evaluate $N(\lambda)$ and $N'(\lambda)$. You can always take for Newton's method $\lambda_0 = 0$ as the initial iterant. Why?

5. We check the interpolation error at the interpolation nodes:

```
err = norm(A*c-sol,inf);
```

How big should be this error?

6. Once we have computed the coefficients in (1) we can approximate f at any point $\eta \in D$ just by evaluating $F_{num}(\eta)$. You can measure the interpolation error for instance at the points:

```

eta = .6*exp(i*2*pi*[1:4]/8);
sol_eta = 1./(1.1+eta).';
%Evaluation of g_j(eta) for j = 1,...,N.
B = sqrt(1-abs(z(1))^2)./(1-conj(z(1))*eta);
p = 1;
for k = 2:N
p = p .* (eta-z(k-1))./(1-conj(z(k-1))*eta);
B = [B ; sqrt(1-abs(z(k))^2)./(1-conj(z(k))*eta).*p ];
end
B = B. ';
err_eta = norm(B*c-sol_eta,inf)

```

2. Backward problem for the heat equation. Let $T > 0$ be a fixed positive time and consider the backward problem for the homogeneous heat equation:

$$\begin{cases} u_t(x, t) = u_{xx}(x, t), & 0 \leq x \leq 1, t > 0, \\ u(0, t) = u(1, t) = 0, & t \geq 0, \\ u(x, T) = u_T(x) + \delta u_T(x) \in C^0([0, 1]), \end{cases} \quad (7)$$

i.e., we are interested in computing an approximation of u at times $0 \leq t < T$.

This well known ill-posed problem can be reduced to the numerical recovery of a holomorphic mapping f in the unit disk D from values of f on the real interval $[-r, r]$, for some $0 < r < 1$, [4]. More precisely, for every fixed $x \in [0, 1]$, we can define a holomorphic mapping $f_x(s) = u(x, \Psi^{-1}(s))$, $s \in D$, for Ψ any conformal transformation mapping a sector

$$\Sigma_\theta = \{z \in \mathbb{C} : |\arg z| \leq \theta\}, \quad \text{for some } 0 < \theta < \frac{\pi}{2},$$

into D and such that $\Psi(T) = -r$. If we consider again the Chebychev nodes z_j , $j = 1, \dots, N$ on $[-r, r]$, we can integrate forward problem (7) to approximate $u(x, \tau_j)$, at future times $\tau_j = \Psi^{-1}(z_j) > T$, $j = 1, \dots, N$. Then, given $0 < t < T$ and for every $x \in [0, 1]$, we can apply the method implemented in Section 2 to approximate $u(x, t) = u(x, \Psi^{-1}(\Psi(t))) = f_x(\Psi(t))$.

3. Forward problem with the numerical inversion of the Laplace transform. Integrating forward (7) can be also a difficult and expensive task due to the stiffness of the problem. The difficulty increases if we need to approximate the solution u at very different scaled future times $t > T$ and if the initial data is *bad*, for instance, highly oscillating. This can be the case, for instance, in the procedure explained in Section 2 for the backward problem (7). Recall that the times τ_1, \dots, τ_N for the forward integration of (7) are given by the conformal transformation $\tau_j = \Psi^{-1}(z_j)$, for z_j , $j = 1, \dots, N$, the Chebychev nodes on $[-r, r]$, $0 < r < 1$. Standard time stepping methods may perform very slowly for the integration forward.

Let us consider for instance the initial value problem

$$\begin{cases} u_t(x, t) = u_{xx}(x, t), & 0 \leq x \leq 1, t > 0, \\ u(0, t) = u(1, t) = 0, & t \geq 0, \\ u(x, 0) = \sin(25\pi x^2). \end{cases} \quad (8)$$

An alternative approach to the use of time stepping methods is the application of the Laplace transform in time to (7) and the numerical inversion method developed in [1, 2]. To this end, we create the file `fhe.m`.

1. We consider the spatial discretization of (7) by the finite difference method. Then, setting $M \geq 1$, the number of spatial grid nodes, $\Delta x = 1/(M + 1)$ and $x_j = j\Delta x$,

$j = 1, \dots, M$, we consider a vector of approximations $\mathbf{U}(t) = [u_j(t)]_{j=1}^M$, where $u_j(t) \approx u(x_j, t)$, $j = 1, \dots, M$.

```
M = 500;
dx = 1/(M+1);
x = [dx:dx:1-dx]';
e = ones(M,1);
L = spdiags([e -2*e e], -1:1, M,M);
L = 1/dx/dx*L; %1D Laplacian with homogeneous Dirichlet b.c.
u_ini = sin(25*pi*x.^2);
```

The semidiscrete version of (7) results

$$\mathbf{U}_t(t) = L\mathbf{U}(t), \quad \mathbf{U}(0) = [\sin(25\pi x_j^2)]_{j=1}^M. \quad (9)$$

2. We apply the Laplace transform to both members in (9). Recall that the Laplace transform of a mapping $f : (0, +\infty) \rightarrow X$, taking values in a Banach space X , with exponential growth $\|f(t)\| \leq Ct^{\mu-1}e^{\sigma t}$, $t > 0$, is given by

$$\mathcal{L}[f](z) = F(z) = \int_0^{+\infty} e^{-zt} f(t) dt, \quad \operatorname{Re} z > \sigma. \quad (10)$$

In our case $X = \mathbb{R}^M$ and, setting $\mathcal{U}(z) = \mathcal{L}[\mathbf{U}](z)$, the transformed problem of (9) becomes

$$z\mathcal{U}(z) - \mathbf{U}(0) = L\mathcal{U}(z). \quad (11)$$

Thus

$$\mathcal{U}(z) = (zI - L)^{-1}\mathbf{U}(0). \quad (12)$$

This means that to evaluate the Laplace transform of the solution to (9) at a given $z \in \mathbb{C}$ we only have to solve a linear system. Since L is a sparse matrix, these linear systems can be very efficiently solved.

In order to recover the solution of (9) from few values of $\mathcal{U}(z)$, we use the inversion method for the Laplace transform developed in [1, 2]. This is possible due to the symmetry of L and the location of its spectrum, which lies on the negative semi-axis $\operatorname{Re} z < 0$. The inversion method is nothing but a quadrature to approximate the inversion formula

$$\mathbf{U}(t) = \frac{1}{2\pi i} \int_{\Gamma} e^{tz} \mathcal{U}(z) dz, \quad t > 0, \quad (13)$$

where Γ is a contour from $-i\infty$ to $+i\infty$, which can be taken beginning and ending in the left half plane. Choosing Γ as the left branch of a hyperbola and considering

a suitable parameterization in terms of the complex sine $\sin z$, it is possible to obtain an approximation of the form

$$\mathbf{U} \approx \mathbf{U}_N(t) = \sum_{k=-N}^N w_k e^{tz_k} \mathcal{U}(z_k), \quad (14)$$

for certain quadrature weights and nodes $w_k, z_k, k = -K, \dots, K$, so that the error is uniformly bounded by

$$\|\mathbf{U}(t) - \mathbf{U}_N(t)\| \sim O(e^{-cN}), \quad \text{for } t \in [t_0, \Lambda t_0], \Lambda \geq 1.$$

Notice that this provides a continuous output approximation of the original mapping $\mathbf{U}(t)$ and that the same evaluations of the transform \mathcal{U} can be used to approximate \mathbf{U} at different t .

We use the inversion method for the Laplace transform to approximate the solution to (9) at a given sequence of times. We will use the auxiliary routine `paracontours.m`, which computes the quadrature weights and nodes according to [2] and measure CPU time:

```
T = 1/8192;
tvec = [T,1.2*T,1.5*T,2*T,3*T,10*T];

tic
tini = tvec(1);
ratio = 10;
tfinal = ratio*tini;
K = 30;
uvec = [];
index = find(tvec <= tfinal);
tt = tvec(index)

while (~isempty(index) & index(end) < length(tvec))
[weights,nodes] = paracontours(tini,K,ratio);
I = speye(M); LTM = [];
for j=1:K+1
transf = (nodes(j)*I-A)\u_ini;
LTM = [LTM transf];
end
for t = tt
u = zeros(M,1);
for j=1:K+1
u = u + weights(j)*exp(nodes(j)*t)*LTM(:,j);
end
```

```

u = real(u);
uvec =[uvec u];
end
tini = tvec(index(end)+1);
tfinal = ratio*tini;
index = find(tini <= tvec & tvec <=tfinal);
tt = tvec(index)
end

%For the last vector of times
[weights,nodes] = paracontours(tini,K,ratio);
I = speye(M); LTM = [];
for j=1:K+1
transf = (nodes(j)*I-A)\u_ini;
LTM = [LTM transf];
end
for t = tt
u = zeros(M,1);
for j=1:K+1
u = u + weights(j)*exp(nodes(j)*t)*LTM(:,j);
end
u = real(u);
uvec =[uvec u];
end
cpult = toc

```

3. VISUALIZING:

```

for ll= 1:length(tvec)
figure(1)
plot(x,uvec(:,ll));
axis([0 1 -1 1])
drawnow
pause(.5)
end

```

4. Compare CPU times and error with some of the MATLAB solvers like `ode45` and `ode15s`. Recall that to this end you need to create an `*.m` file to evaluate the right hand side of (9), for instance `fode_he.m`.

Try with different error tolerances, for instance:

```

opts = odeset('AbsTol',1e-10,'RelTol',1e-10);
tic

```

```

[tv45,uvec45] = ode45 (@(t,u) fode_he(t,u,L),[0,tvec],u_ini,opts);
cpu45 = toc
err45 = max(max(abs(uvec-uvec45.')))

```

Try also with different values of M , the number of spatial grid nodes, K , the number of quadrature nodes, and the ratio $\Lambda = t_{final}/t_{ini}$.

The time integration by using the numerical inversion of the Laplace transform can be equally used in higher dimensional problems for many of the spatial discretizations commonly used, including the finite element method. It is also useful for the time integration of problems with fractional time derivatives and for nonhomogeneous problems, provided that they admit a Laplace transform which is computable and fit the *sectorial* setting.

References

- [1] M. LÓPEZ-FERNÁNDEZ, C. PALENCIA, *On the numerical inversion of the Laplace transform of certain holomorphic mappings*, Appl. Numer. Math., 51 (2004), pp. 289–303.
- [2] M. LÓPEZ-FERNÁNDEZ, C. PALENCIA, A. SCHÄDLE, *A spectral order method for inverting sectorial Laplace transforms*, SIAM J. Numer. Anal., 44 (2006), pp. 1332–1350.
- [3] J.M. MARBAN, C. PALENCIA, *On the numerical recovery of a holomorphic mapping from a finite set of approximate values*, Numer. Math., 91 (2002), pp. 57–75.
- [4] J.M. MARBAN, C. PALENCIA, *A new numerical method for backward parabolic problems in the maximum-norm setting*, SIAM Numer. Math., 40 (2002), pp. 1405–1420.