Mathematical Modelling in the Natural Sciences SS21
Solutions to Exercises on Sheet 5

Exercises und Lecture Notes

Contents

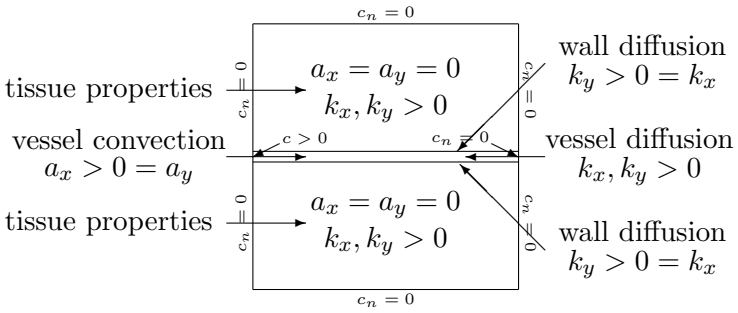
• Exercise 1: Tracer Transport, Forward Problem

◦ Task

Write a Matlab code to simulate the convection and diffusion of contrast agent in a spatial domain $\Omega = (0,1)^2$ containing a single vessel according to

$$\begin{cases} \partial_t c + \partial_x(a_x c) + \partial_y(a_y c) &= \partial_x(k_x \partial_x c) + \partial_y(k_y \partial_y c) \\ &+ a_x \delta(x-0)\delta(y-\frac{1}{2})(c-c_0), & (x,y) \in \Omega, & t > 0 \\ \partial_n c &= 0, & (x,y) \in \partial\Omega, & t > 0 \\ c &= 0, & (x,y) \in \Omega, & t = 0 \end{cases}$$

which is illustrated as follows:



Here, blood flows through the vessel in the x-direction from left to right with convection coefficients $a_x > 0 = a_y$ and diffusion coefficients $k_x, k_y > 0$. In the tissue outside the vessel, there is no convection so $a_x = 0 = a_y$, but there is diffusion with diffusion coefficients $k_x, k_y > 0$. There is also diffusion over the boundary of the vessel in the y-direction with diffusion coefficients $k_y > 0 = k_x$. In general, the diffusion coefficients in the vessel, at the vessel wall and in the tissue outside the vessel are different. The boundary conditions for the concentration c are $c_n = 0$ on all sides. On the left side a constant inflow concentration $c > 0$ holds impulsively at the vessel inflow.

◦ Solution

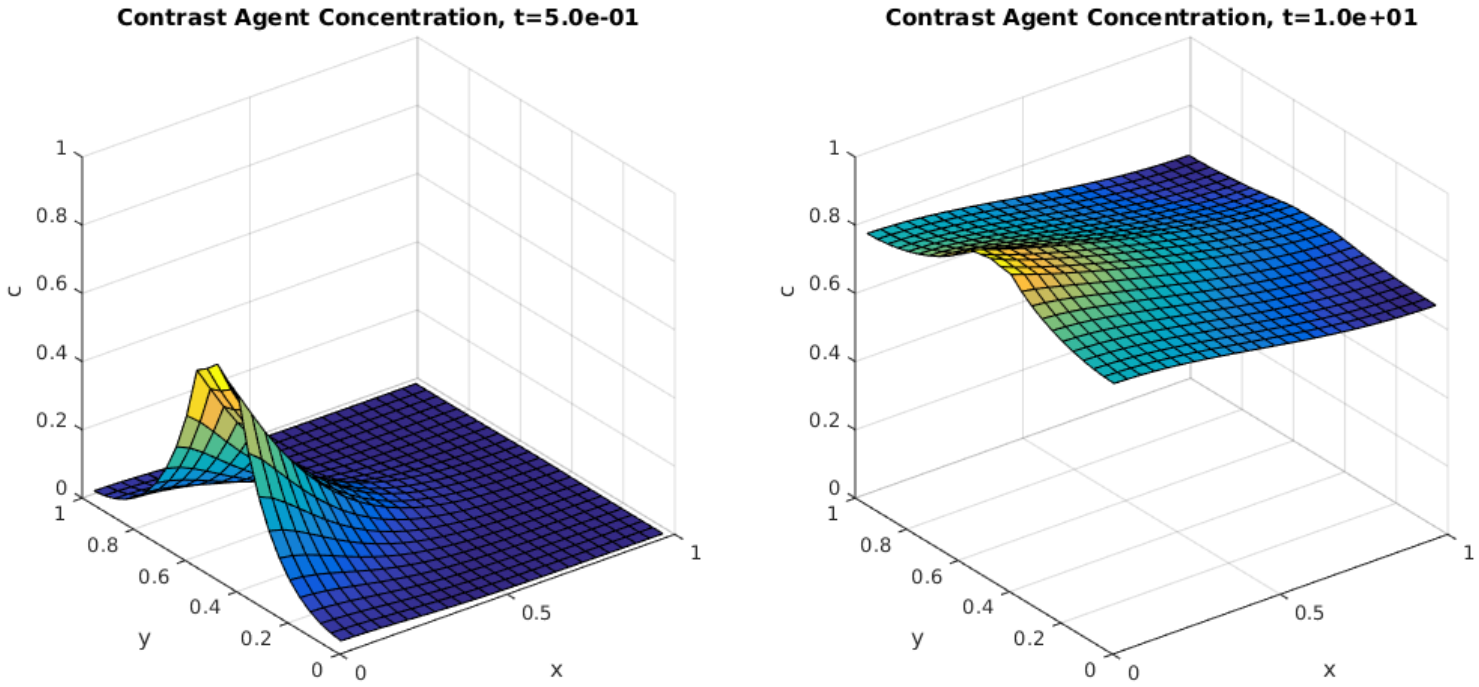
The following Matlab code was used to solve the initial- and boundary-value problem. The vessel is located within the cells with `i=1,...,Nx` and `j=Ny1,...,Ny2`. The vessel wall consists of zero-thickness interfaces between the rows of cells `i=1,...,Nx`, `j=Ny1-1`, `j=Ny1` and `i=1,...,Nx`, `j=Ny2`, `j=Ny2+1`. Contrast agen is injected at `i=1` and `j=Ny1,...,Ny2`.

```
% setup figure
    h1 = figure(1);
    close(h1);
    h1 = figure(1);
    set(h1,'Position',[10 10 500 500]);

% temporal and geometric parameters
    Nt = 1000; Nx = 25; Ny = 25;  % dimensions of space-time grid
    Ny1 = 12; Ny2 = 14;           % positions of vessel

    T  = 10; dt = T/Nt;           % final time and time step

    xmin = 0; xmax = 1;
    hx = (xmax - xmin)/Nx;        % x coordinates of cells
    x  = linspace(xmin,xmax,Nx+1);
    x = x(1:Nx) + hx/2;

    ymin = 0; ymax = 1;
    hy = (ymax - ymin)/Ny;        % y coodinates of cells
    y  = linspace(ymin,ymax,Ny+1);
    y = y(1:Ny) + hy/2;

    xx = kron(x(:),ones(1,Ny));   % coordinates of
    yy = kron(ones(Nx,1),y(:)');  % cell centers

% first order derivatives with Neumann BCs
    dx = spdiags(ones(Nx-1,1),1,Nx-1,Nx) ...
       - spdiags(ones(Nx-1,1),0,Nx-1,Nx);
    dx = dx/hx;
    iy = speye(Ny);
    Dx = kron(iy,dx);
    dy = spdiags(ones(Ny-1,1),1,Ny-1,Ny) ...
       - spdiags(ones(Ny-1,1),0,Ny-1,Ny);
    dy = dy/hy;
```

```
    ix = speye(Nx);
    Dy = kron(dy,ix);

% first order derivative for convection
    dx = spdiags(ones(Nx,1), 0,Nx,Nx) ...
       - spdiags(ones(Nx,1),-1,Nx,Nx);
    dx = dx/hx;
    iy = speye(Ny);
    Cx = kron(iy,dx);

% model parameters
    ax0 = 1;     % x-convection in vessel
    ky0 = 0.5;   % y-diffusion in vessel wall
    kx1 = 0.1;   % x-diffusion in vessel
    ky1 = 0.1;   % y-diffusion in vessel
    kx2 = 0.05;  % x-diffusion in interstitium
    ky2 = 0.05;  % y-diffusion in interstitium
    c0  = 1.0;   % injected concentration

% transport coefficients
    ax = zeros(Nx,Ny);
    ay = zeros(Nx,Ny);
    ax(:,Ny1:Ny2) = ax0;                % convection in the vessel

    kx = zeros(Nx-1,Ny); kx = kx + kx2;  % x-diffusivity in interstitium
    ky = zeros(Nx,Ny-1); ky = ky + ky2;  % y-diffusivity in interstitium

    ky(:,Ny1-1) = ky0;                  % permeability in vessel wall
    ky(:,Ny2) = ky0;                    % permeability in vessel wall

    kx(:,Ny1:Ny2) = kx1;                % x-diffusion in the vessel
    ky(:,Ny1:Ny2-1) = ky1;              % y-diffusion in the vessel

% discrete operators, diffusion L2 and convection L1
    L2 = Dx'*spdiags(kx(:),0,(Nx-1)*Ny,(Nx-1)*Ny)*Dx ...
       + Dy'*spdiags(ky(:),0,Nx*(Ny-1),Nx*(Ny-1))*Dy;
    L1 = spdiags(ax(:),0,Nx*Ny,Nx*Ny)*Cx;
    I  = speye(Nx*Ny);
% impulsive injection
    F = zeros(Nx,Ny);
    F(1,Ny1:Ny2) = c0*ax(1,Ny1:Ny2)/hx;
    F = F(:);

% initial conditions
    c = zeros(Nx,Ny); c = c(:);

    for k=1:Nt
```

```
% time step
% C_t = -L2*C - L1*C + F
        c  = (I + dt*(L1 + L2)) \ (c + dt*F);

% plotting
        surf(xx,yy,reshape(c,Nx,Ny));
        axis([xmin xmax ymin ymax 0 c0])
        xlabel('x')
        ylabel('y')
        zlabel('c')
        title(sprintf('Contrast Agent Concentration, t=%0.1e',k*dt))
        drawnow;

    end
```

Earlier and later states of the evolution are shown graphically as follows.



- **Exercise 2: Tracer Transport, Parameter Estimation**
  - **Task**

Let $c^\star$ be the solution to the previous problem using known values $a_x^\star, k_x^\star, k_y^\star$ in the vessel, $k_y^\dagger$ at the vessel wall and $k_x^\ddagger, k_y^\ddagger$ in the tissue outside the vessel. Use a perturbation of $c^\star$ as data and write a Matlab code to estimate the known convection and diffusion parameters. For this, it is recommended to use the Matlab function `lsqnonlin`.

- **Solution**

The following Matlab code was used to solve the inverse problem.

```
function S04Exp2

% exact parameters
    ax0 = 1;     % x-convection in vessel
    ky0 = 0.5;   % y-diffusion in vessel wall
    kx1 = 0.1;   % x-diffusion in vessel
    ky1 = 0.1;   % y-diffusion in vessel
    kx2 = 0.05;  % x-diffusion in interstitium
    ky2 = 0.05;  % y-diffusion in interstitium
    pexact = [ax0;ky0;kx1;ky1;kx2;ky2];
    cexact = capprox(pexact);
    cdata  = cexact.*(1 + 0.1*rand(size(cexact)));

    p0 = pexact.*(1 + 0.1*rand(size(pexact)));
    p  = lsqnonlin(@(p) cdata - capprox(p),p0)

end

function cv = capprox(p)

% extract parameters
    ax0 = p(1);   % x-convection in vessel
    ky0 = p(2);   % y-diffusion in vessel wall
    kx1 = p(3);   % x-diffusion in vessel
    ky1 = p(4);   % y-diffusion in vessel
    kx2 = p(5);   % x-diffusion in interstitium
    ky2 = p(6);   % y-diffusion in interstitium
    c0  = 1.0;    % injected concentration

% temporal and geometric parameters
    Nt = 1000; Nx = 25; Ny = 25;  % dimensions of space-time grid
    Ny1 = 12; Ny2 = 14;           % positions of vessel

    T  = 10; dt = T/Nt;           % final time and time step

    xmin = 0; xmax = 1;
    hx = (xmax - xmin)/Nx;        % x coordinates of cells
    x  = linspace(xmin,xmax,Nx+1);
    x = x(1:Nx) + hx/2;

    ymin = 0; ymax = 1;
    hy = (ymax - ymin)/Ny;        % y coodinates of cells
    y  = linspace(ymin,ymax,Ny+1);
    y = y(1:Ny) + hy/2;
```

```
% first order derivatives with Neumann BCs
    dx = spdiags(ones(Nx-1,1),1,Nx-1,Nx) ...
       - spdiags(ones(Nx-1,1),0,Nx-1,Nx);
    dx = dx/hx;
    iy = speye(Ny);
    Dx = kron(iy,dx);
    dy = spdiags(ones(Ny-1,1),1,Ny-1,Ny) ...
       - spdiags(ones(Ny-1,1),0,Ny-1,Ny);
    dy = dy/hy;
    ix = speye(Nx);
    Dy = kron(dy,ix);

% first order derivative for convection
    dx = spdiags(ones(Nx,1), 0,Nx,Nx) ...
       - spdiags(ones(Nx,1),-1,Nx,Nx);
    dx = dx/hx;
    iy = speye(Ny);
    Cx = kron(iy,dx);

% transport coefficients
    ax = zeros(Nx,Ny);
    ay = zeros(Nx,Ny);
    ax(:,Ny1:Ny2) = ax0;                % convection in the vessel

    kx = zeros(Nx-1,Ny); kx = kx + kx2; % x-diffusivity in interstitium
    ky = zeros(Nx,Ny-1); ky = ky + ky2; % y-diffusivity in interstitium

    ky(:,Ny1-1) = ky0;                  % permeability in vessel wall
    ky(:,Ny2) = ky0;                    % permeability in vessel wall

    kx(:,Ny1:Ny2) = kx1;                % x-diffusion in the vessel
    ky(:,Ny1:Ny2-1) = ky1;              % y-diffusion in the vessel

% discrete operators, diffusion L2 and convection L1
    L2 = Dx'*spdiags(kx(:),0,(Nx-1)*Ny,(Nx-1)*Ny)*Dx ...
       + Dy'*spdiags(ky(:),0,Nx*(Ny-1),Nx*(Ny-1))*Dy;
    L1 = spdiags(ax(:),0,Nx*Ny,Nx*Ny)*Cx;
    I  = speye(Nx*Ny);
% impulsive injection
    F = zeros(Nx,Ny);
    F(1,Ny1:Ny2) = c0*ax(1,Ny1:Ny2)/hx;
    F = F(:);

% initial conditions
    c = zeros(Nx,Ny); c = c(:);

% save state
    cv = c;
```

```
    for k=1:Nt

% time step
% C_t = -L2*C - L1*C
        c  = (I + dt*(L1 + L2)) \ (c + dt*F);
        cv = [cv,c];

    end
end
```

The exact solution to the inverse problem obtained by using noise-free data is given by

```
p =
    1.0000
    0.5000
    0.1000
    0.1000
    0.0500
    0.0500
```

With data given by a 10% perturbation of the exact solution, the computed solution to the inverse problem is given by the following output.

```
p =
    1.0651
    0.4818
    0.0761
    0.0978
    0.0475
    0.0500
```

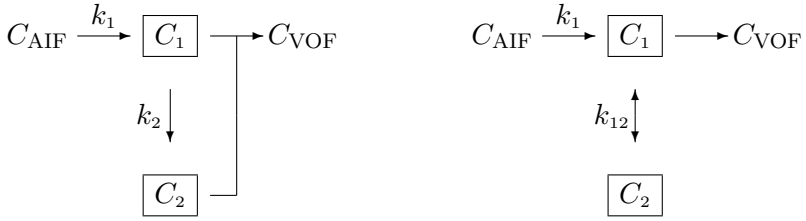
• Exercise 3: Tracer Transport, Convolution Kernel

◦ Task

Verify that the convolution kernel $K(t)$ for each of the two problems on page 161 of the lecture notes is given in section 2 of this article.

◦ Solution

The two examples are illustrated graphically as follows,



where the example on the left exhibits pure convection among the compartments and the example on the right exhibits convection and diffusion among the compartments. (Both examples exhibit diffusion in the sense that within each compartment the fluid is well mixed.) The systems of ODEs for these models are given respectively by

$$(1)\quad \left\{ \begin{array}{rcl} V_1C_1' + k_1(C_1 - C_{\rm AIF}) &=& 0 \\ V_2C_2' + k_2(C_2 - C_1) &=& 0 \end{array} \right. \qquad (2)\quad \left\{ \begin{array}{rcl} V_1C_1' + k_1(C_1 - C_{\rm AIF}) &=& k_{12}(C_2 - C_1) \\ V_2C_2' &=& k_{12}(C_1 - C_2) \end{array} \right.$$

It will be shown that the kernel $K(t)$ corresponding to both models has the bi-exponential form $K(t) = \alpha_1 e^{-\lambda_1 t} + \alpha_2 e^{-\lambda_2 t}$.

With

$$\mu_1 = k_1/V_1, \quad \mu_2 = k_2/V_2$$

and

$$\boldsymbol{C}(t) = \left[\begin{array}{c} C_1(t) \\ C_2(t) \end{array} \right], \quad A = \left[\begin{array}{cc} -\mu_1 & 0 \\ +\mu_2 & -\mu_2 \end{array} \right], \quad \boldsymbol{b}(t) = \left[\begin{array}{c} \mu_1 C_{\rm AIF}(t) \\ 0 \end{array} \right]$$

system (1) can be written as

$$\boldsymbol{C}'(t) = A\boldsymbol{C} + \boldsymbol{b}(t)$$

with solution

$$\boldsymbol{C}(t) = e^{At}\boldsymbol{C}(0) + \int_0^t e^{A(t-s)}\boldsymbol{b}(s)ds.$$

For the determination of $K(t)$ take $\boldsymbol{C}(0) = \boldsymbol{0}$ and $C_{\rm AIF}(t) = \delta(t)$. The eigenspace decomposition of A is

$$AS = S\Lambda, \quad S = \left[\begin{array}{cc} \mu_2 - \mu_1 & 0 \\ \mu_2 & 1 \end{array} \right], \quad \Lambda = \left[\begin{array}{cc} -\mu_1 & 0 \\ 0 & -\mu_2 \end{array} \right], \quad S^{-1} = \frac{1}{\mu_2 - \mu_1} \left[\begin{array}{cc} 1 & 0 \\ -\mu_2 & 1 \end{array} \right]$$

and so the solution for the kernel function is given by

$$\boldsymbol{C}(t) = \frac{\mu_1}{\mu_2 - \mu_1} \int_0^t \left[\begin{array}{cc} \mu_2 - \mu_1 & 0 \\ \mu_2 & 1 \end{array} \right] \left[\begin{array}{cc} e^{-\mu_1(t-s)} & 0 \\ 0 & e^{-\mu_2(t-s)} \end{array} \right] \left[\begin{array}{cc} 1 & 0 \\ -\mu_2 & 1 \end{array} \right] \left[\begin{array}{c} \delta(s) \\ 0 \end{array} \right] ds$$

$$= \frac{\mu_1}{\mu_2 - \mu_1} \int_0^t \left[\begin{array}{cc} (\mu_2 - \mu_1)e^{-\mu_1(t-s)} & 0 \\ \mu_2 e^{-\mu_1(t-s)} & e^{-\mu_2(t-s)} \end{array} \right] \left[\begin{array}{c} 1 \\ -\mu_2 \end{array} \right] \delta(s)ds$$

$$= \frac{\mu_1}{\mu_2 - \mu_1} \int_0^t \left[\begin{array}{c} (\mu_2 - \mu_1)e^{-\mu_1(t-s)} \\ \mu_2(e^{-\mu_1(t-s)} - e^{-\mu_2(t-s)}) \end{array} \right] \delta(s)ds = \left[\begin{array}{c} \mu_1 e^{-\mu_1 t} \\ (e^{-\mu_1 t} - e^{-\mu_2 t})\mu_1\mu_2/(\mu_2 - \mu_1) \end{array} \right]$$

Defining $h_1(t) = \mu_1 e^{-\mu_1 t}$ and $h_2(t) = \mu_2 e^{-\mu_2 t}$ leads to

$$[h_1 * h_2](t) = \int_0^t \mu_1 e^{-\mu_1(t-s)} \mu_2 e^{-\mu_2 s} ds = \mu_1\mu_2 e^{-\mu_1 t} \left. \frac{e^{(\mu_1 - \mu_2)s}}{\mu_1 - \mu_2} \right|_0^t = \frac{\mu_1\mu_2}{\mu_1 - \mu_2}[e^{-\mu_2 t} - e^{-\mu_1 t}]$$

or

$$\boldsymbol{C}(t) = \left[\begin{array}{c} h_1(t) \\ [h_1 * h_2](t) \end{array} \right]$$

The kernel function is given by the average concentration for the whole unit,

$$K(t) = \frac{V_1C_1(t) + V_2C_2(t)}{V_1 + V_2} = \frac{V_1h_1(t) + V_2[h_1 * h_2](t)}{V_1 + V_2}$$
$$= \frac{\mu_1 V_1}{V_1 + V_2} e^{-\mu_1 t} - \frac{\mu_1\mu_2 V_2}{(\mu_2 - \mu_1)(V_1 + V_2)}(e^{-\mu_2 t} - e^{-\mu_1 t})$$

a bi-exponential form.

With

$$\mu_1 = k_1/V_1, \quad \alpha = k_{12}/V_1, \quad \beta = k_{12}/V_2$$

and

$$\boldsymbol{C}(t) = \begin{bmatrix} C_1(t) \\ C_2(t) \end{bmatrix}, \quad A = \begin{bmatrix} -\mu_1 - \alpha & \alpha \\ +\beta & -\beta \end{bmatrix}, \quad \boldsymbol{b}(t) = \begin{bmatrix} \mu_1 C_{\text{AIF}}(t) \\ 0 \end{bmatrix}$$

system (2) can be written as

$$\boldsymbol{C}'(t) = A\boldsymbol{C} + \boldsymbol{b}(t)$$

with solution

$$\boldsymbol{C}(t) = e^{At}\boldsymbol{C}(0) + \int_0^t e^{A(t-s)}\boldsymbol{b}(s)ds.$$

For the determination of $K(t)$ take $\boldsymbol{C}(0) = \boldsymbol{0}$ and $C_{\text{AIF}}(t) = \delta(t)$. With

$$\lambda_1 = -\tfrac{1}{2}\left[\alpha + \beta + \mu_1 + \sqrt{(\alpha + \beta + \mu_1)^2 - 4\beta\mu_1}\right]$$
$$\lambda_2 = -\tfrac{1}{2}\left[\alpha + \beta + \mu_1 - \sqrt{(\alpha + \beta + \mu_1)^2 - 4\beta\mu_1}\right]$$

the eigenspace decomposition of A is

$$AS = S\Lambda, \quad S = \begin{bmatrix} \lambda_1 + \beta & \lambda_2 + \beta \\ \beta & \beta \end{bmatrix}, \quad \Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \quad S^{-1} = \frac{1}{\beta(\lambda_1 - \lambda_2)} \begin{bmatrix} +\beta & -\lambda_2 - \beta \\ -\beta & +\lambda_1 + \beta \end{bmatrix}$$

and so the solution for the kernel function is given by

$$\boldsymbol{C}(t) = \frac{\mu_1}{\beta(\lambda_1 - \lambda_2)} \int_0^t \begin{bmatrix} \lambda_1 + \beta & \lambda_2 + \beta \\ -\beta & \beta \end{bmatrix} \begin{bmatrix} e^{\lambda_1(t-s)} & 0 \\ 0 & e^{\lambda_2(t-s)} \end{bmatrix} \begin{bmatrix} +\beta & -\lambda_2 - \beta \\ -\beta & +\lambda_1 + \beta \end{bmatrix} \begin{bmatrix} \delta(s) \\ 0 \end{bmatrix} ds$$

$$= \frac{\mu_1}{\lambda_1 - \lambda_2} \int_0^t \begin{bmatrix} (\lambda_1 + \beta)e^{\lambda_1(t-s)} & (\lambda_2 + \beta)e^{\lambda_2(t-s)} \\ -\beta e^{\lambda_1(t-s)} & +\beta e^{\lambda_2(t-s)} \end{bmatrix} \begin{bmatrix} +1 \\ -1 \end{bmatrix} \delta(s)ds$$

$$= \frac{\mu_1}{\lambda_1 - \lambda_2} \begin{bmatrix} (\lambda_1 + \beta)e^{\lambda_1 t} - (\lambda_2 + \beta)e^{\lambda_2 t} \\ -\beta e^{\lambda_1 t} - \beta e^{\lambda_2 t} \end{bmatrix}$$

The kernel function is given by the average concentration for the whole unit,

$$K(t) = \frac{V_1 C_1(t) + V_2 C_2(t)}{V_1 + V_2}$$
$$= \frac{\mu_1}{(\lambda_1 - \lambda_2)(V_1 + V_2)} \left\{ \left[V_1(\lambda_1 + \beta)e^{\lambda_1 t} - V_2(\lambda_2 + \beta)e^{\lambda_2 t} \right] - \beta \left[V_1 e^{\lambda_1 t} + V_2 e^{\lambda_2 t} \right] \right\}$$

a bi-exponential form. These results are reported explicitly in the article.

• Exercise 4: Ill-Posedness of Deconvolution

◦ Task

As discussed on page 162 of the lecture notes, construct a simple example $(N_\epsilon, C_{\text{AIF}}, E_\epsilon)$ satisfying $N_\epsilon = C_{\text{AIF}} * E_\epsilon$ and $N_\epsilon = \mathcal{O}(\epsilon)$ while $E_\epsilon = \mathcal{O}(\epsilon^{-n})$ for some arbitrary $n \in \mathbb{N}$.

◦ **Solution**

Let $n = 1$ and take

$$C_{\mathrm{AIF}}(t) = \tfrac{1}{2}(1 + \mathrm{sign}(t)) \quad \text{and} \quad E_\epsilon(t) = \frac{1}{\epsilon} \sin\left(\frac{t}{\epsilon^2}\right) = \mathcal{O}(e^{-1})$$

so that

$$N_\epsilon(t) = \int_0^t C_{\mathrm{AIF}}(s) E_\epsilon(t-s) ds = \int_0^t E_\epsilon(t-s) ds = \epsilon \left[1 - \cos\left(\frac{t}{\epsilon^2}\right)\right] = \mathcal{O}(\epsilon).$$

Then let $n = 2$ and take

$$C_{\mathrm{AIF}}(t) = t\mathrm{sign}(t) \quad \text{and} \quad E_\epsilon(t) = \frac{1}{\epsilon^2} \sin\left(\frac{t}{\epsilon^4}\right) = \mathcal{O}(e^{-2})$$

so that

$$N_\epsilon(t) = \int_0^t C_{\mathrm{AIF}}(s) E_\epsilon(t-s) ds = \int_0^t s E_\epsilon(t-s) ds = \epsilon^2 \left[t - \epsilon^4 \sin\left(\frac{t}{\epsilon^4}\right)\right] = \mathcal{O}(\epsilon^2).$$

Then let $n = 3$ and take

$$C_{\mathrm{AIF}}(t) = t^2\mathrm{sign}(t) \quad \text{and} \quad E_\epsilon(t) = \frac{1}{\epsilon^3} \sin\left(\frac{t}{\epsilon^6}\right) = \mathcal{O}(e^{-3})$$

so that

$$N_\epsilon(t) = \int_0^t C_{\mathrm{AIF}}(s) E_\epsilon(t-s) ds = \int_0^t s^2 E_\epsilon(t-s) ds = \epsilon^3 \left[t^2 - 2\epsilon^{12} + 2\epsilon^{12} \cos\left(\frac{t}{\epsilon^6}\right)\right] = \mathcal{O}(\epsilon^3).$$

Clearly a higher order of ϵ is obtained by taking C_{AIF} ever smoother at $t = 0$.

• Exercise 5: Regularized Deconvolution

◦ **Task**

For $t \in [0, T]$, $T = 4$, let

$$\begin{aligned} C_{\mathrm{AIF}}(t) &= 20te^{-3t} \\ C_{\mathrm{T}}(t) &= 5e^{-3t}[-5 + 4e^t + e^{2t} - 6t] \\ K(t) &= e^{-t} + e^{-2t} \end{aligned}$$

For $N \in \mathbb{N}$, let $t_i = i\Delta t$, $i = 0, \ldots, N$, $\Delta t = T/N$, let $\boldsymbol{C}_{\mathrm{AIF}} = \{C_{\mathrm{AIF}}(t_i)\}_{i=1}^N$, $\boldsymbol{C}_{\mathrm{T}} = \{C_{\mathrm{T}}(t_i)\}_{i=1}^N$ and $\boldsymbol{K} = \{K(t_i)\}_{i=1}^N$. Then let $\tilde{\boldsymbol{C}}_{\mathrm{AIF}}$ and $\tilde{\boldsymbol{C}}_{\mathrm{T}}$ be perturbations of $\boldsymbol{C}_{\mathrm{AIF}}$ and $\boldsymbol{C}_{\mathrm{T}}$, respectively. For the estimation of $\boldsymbol{K}$, compare the method of truncated singular value decomposition of pages 163 – 165 in the lecture notes with the method of contrained exponentials of pages 167 – 168 in the lecture notes. The Matlab functions `svd` and `lsqlin` are useful for the respective tasks.

◦ **Solution**

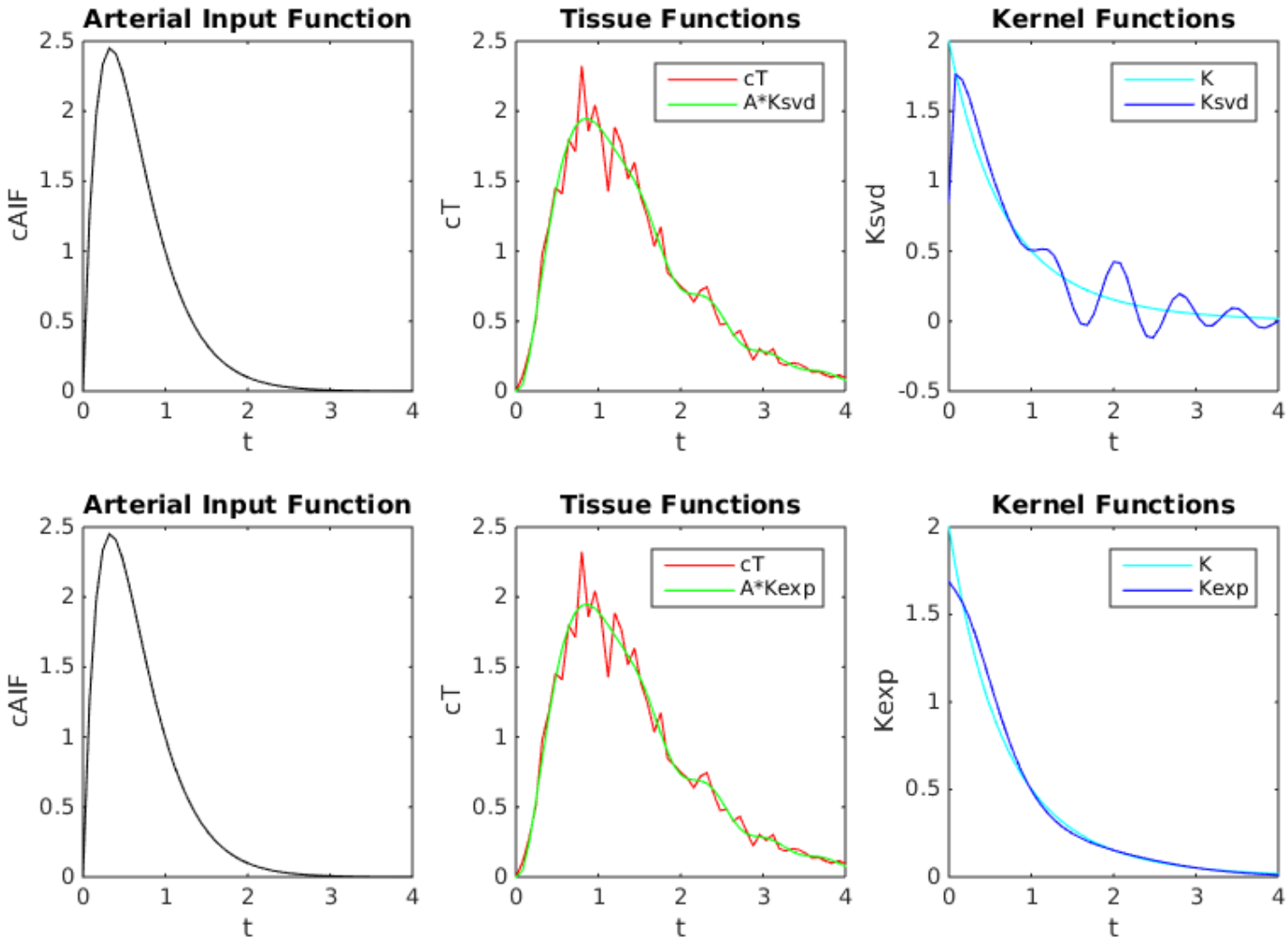
The following Matlab code was used to solve the deconvolution problem using the SVD and EXP approaches.

```
% setup figure
    h1 = figure(1);
    close(h1);
    h1 = figure(1);
    set(h1,'Position',[10 10 900 600]);

% temporal parameters
    N = 51;
    T = 4;
    t = linspace(0,T,N)';

% exact data and solution
    cAIF = 20*t.*exp(-3*t);
    cT   = 5*exp(-3*t).*(-5 + 4*exp(t) + exp(2*t) - 6*t);
    K    = exp(-t) + exp(-2*t);
% perturbed data
    ns   = 0.1;
    cTns = cT.*(1 + ns*randn(N,1));

% matrix approximation to convolution
    dtr = t(2)-t(1);
    A = cAIF*dtr;
    for j=2:N-1
        dtl = t(j)-t(j-1);
        dtc = t(j+1)-t(j-1);
        Aj = [zeros(j-1,1);cAIF(1)*dtl;cAIF(2:N-j+1)*dtc];
        A  = [A,Aj];
    end
    dtl = t(N)-t(N-1);
    A = [A,[zeros(N-1,1);cAIF(1)]*dtl];
    A = A/2;

% truncated svd solution
    [U,S,V] = svd(A);    % A = U*S*V'
    Sd = diag(S);
    Sm = 0.1*max(Sd);
% Sm=min(Sd);
    S1 = diag((Sd > Sm)./(Sd + (Sd <= Sm)));
    Ksvd = V*S1*U'*cTns;

% Ksvd(N) = 2*Ksvd(N-1)-Ksvd(N-2);
% Ksvd = (A'*A + 1.0e-5*eye(N))\(A'*cTns);

% plot solution
    subplot(2,3,1)
    plot(t,cAIF,'k')
    xlabel('t')
    ylabel('cAIF')
```

```
    title('Arterial Input Function')
    subplot(2,3,2)
    plot(t,cTns,'r',t,A*Ksvd,'g')
    legend('cT','A*Ksvd')
    xlabel('t')
    ylabel('cT')
    title('Tissue Functions')
    subplot(2,3,3)
    plot(t,K,'c',t,Ksvd,'b')
    legend('K','Ksvd')
    xlabel('t')
    ylabel('Ksvd')
    title('Kernel Functions')

% number of exp functions
    M  = 20;
% time scales in the exponential basis
    la = (1:M)/T;

% constraint matrix D where D*k <= 0 insures that K is decreasing
    D = -diag(la);
    for m=1:(M-1)
        Dm = eye(M);
        for j=1:(M-m)
            i = m+j;
            Dm(j,j) = 1/(la(i)-la(j));
            Dm(j,j+1) = -Dm(j,j);
        end
        D = Dm' \ D;
    end

% add a row to A so that -K(0) <= 0
    D = [D;-ones(1,M)];

% the constraint is D*k <= b
    b = zeros(M+1,1);

% matrix multiplying coefficients k
    AE = A*exp(-kron(t,la));

% box constraints in lsqlin for good measure
    mx = 1.0e5*ones(M,1);

% options for lsqlin below
    opts = optimset('Display','off','MaxIter',100000);

% solve
    k = zeros(M,1);
```

```
    k = lsqlin(AE,cTns,D,b,[],[],-mx,mx,k,opts);

% solution
    Kexp = exp(-kron(t,la))*k;

% plot solution
    subplot(2,3,4)
    plot(t,cAIF,'k')
    xlabel('t')
    ylabel('cAIF')
    title('Arterial Input Function')
    subplot(2,3,5)
    plot(t,cTns,'r',t,A*Ksvd,'g')
    legend('cT','A*Kexp')
    xlabel('t')
    ylabel('cT')
    title('Tissue Functions')
    subplot(2,3,6)
    plot(t,K,'c',t,Kexp,'b')
    legend('K','Kexp')
    xlabel('t')
    ylabel('Kexp')
    title('Kernel Functions')
```

The results obtained with 10% noise added to the data C_{T} are shown graphically as follows.



Here a 10% threshold is used for the truncated SVD approach, and 20 exponential functions are used in the EXP approach.