Approximation Theory

Simulating porous flow with chebfun

Almut Eisenträger

almut.eisentraeger@sjc.ox.ac.uk

28th October 2009

1 Introduction

The aim of the *chebfun* project is a system in MATLAB in which numerical computations with functions can be done as easily, conveniently and accurately as it is already possible for numbers. Therefore, functions are approximated by Chebyshev polynomials of "sufficiently large" degree, represented by their values at suitably rescaled Chebyshev points (Trefethen, 2007). The extension *chebops* then provides linear operators on chebfuns, e. g. differential operators, and commands to solve linear systems thereof, by means of spectral collocation methods (Driscoll et al., 2008). Here, we want to test this system on a simple one-dimensional model problem, motivated by poroelastic models of CSF flow through the brain (Sobey and Wirth, 2006).

2 Model Problem

2.1 Derivation of the ODE system

We consider the one-dimensional steady-state model problem shown in Figure 1, where a fluid is pumped through an elastic porous solid material. Within the saturated medium, the stress in x-direction

$$\sigma = \hat{E}\frac{\mathrm{d}u}{\mathrm{d}x} - \alpha p$$

is made up of an elastic part induced by the strain $\varepsilon = \frac{\mathrm{d}u}{\mathrm{d}x}$, where we write

$$\hat{E} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$$

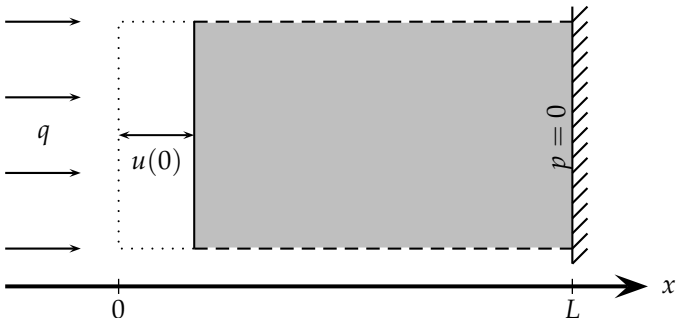

Figure 1: Model example

for convenience of notation with the material parameters Young's modulus E and Poisson's ratio ν of the saturated medium, and a part induced by the pressure p, where the Biot-Willis parameter $\alpha > 0$ describes the relation between the change in fluid content and the strain (Biot, 1941). Since we have no outer forces, the equilibrium of force

$$\frac{\mathrm{d}\sigma}{\mathrm{d}x} = -F_x = 0$$

leads to

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[\hat{E}\frac{\mathrm{d}u}{\mathrm{d}x} - \alpha p\right] = 0,$$

hence,

$$\hat{E}\frac{\mathrm{d}^2u}{\mathrm{d}x^2} - \alpha\frac{\mathrm{d}p}{\mathrm{d}x} = 0. \tag{1}$$

Furthermore, in the steady state, the flux

$$q = -\frac{k}{\mu}\frac{\mathrm{d}p}{\mathrm{d}x},$$

where μ is the viscosity of the fluid and the permeability of the solid matrix

$$k = k_0 f\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)$$

depends on the strain through a suitable function f, must satisfy the fluid continuity (Biot, 1941)

$$\frac{\mathrm{d}q}{\mathrm{d}x} = 0.$$

Hence, we receive

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[\frac{k_0}{\mu}f\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)\frac{\mathrm{d}p}{\mathrm{d}x}\right] = 0. \tag{2}$$

Equations (1) and (2) form a system of two ordinary second order differential equations, so we need four boundary conditions. At the right end, the solid matrix is restricted by a rigid wall, so the displacement there must be zero, i. e.

$$u|_L = 0. \tag{3}$$

In addition, we set the pressure there to be zero, giving us the second boundary condition

$$p|_L = 0. \tag{4}$$

On the left end, fluid is pumped into the matrix with constant flux

$$-\left(\frac{k_0}{\mu}f\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)\frac{\mathrm{d}p}{\mathrm{d}x}\right)\Big|_0 = q_0,$$

yielding

$$\left(f\left(\frac{\mathrm{d}u}{\mathrm{d}x}\right)\frac{\mathrm{d}p}{\mathrm{d}x}\right)\Big|_0 = -\hat{q} \tag{5}$$

with

$$\hat{q} = \frac{q_0\mu}{k_0} > 0.$$

The stress at the left end must be balanced by the outer pressure, so

$$p|_0 = -\sigma|_0,$$

hence, we get our fourth and last boundary condition as

$$\left(\hat{E}\frac{\mathrm{d}u}{\mathrm{d}x} + (1-\alpha)p\right)\Big|_0 = 0. \tag{6}$$

2.2 Models for the permeabilty

Following Sobey and Wirth (2006), we consider different possible functions f to model the relationship between strain and permeability

$$f_{\mathrm{a}}(v) = 1, \tag{7a}$$

$$f_b(v) = \frac{1}{1 - Mv} \tag{7b}$$
$$f_c(v) = e^{Mv} \tag{7c}$$

where the parameter is given as $M \approx 4$. We will see, that we need the first and second derivative of f, so consider

$$\begin{aligned} f_a(v+w) &= 1 = f_a(v) \\ \Rightarrow\ f_a'(v) &= 0 \end{aligned} \tag{8a}$$

$$\begin{aligned} f_b(v+w) &= \frac{1}{1 - M(v+w)} = \frac{1}{(1-Mv)} \cdot \frac{1}{1 - \frac{Mw}{(1-Mv)}} \\ &= \frac{1}{(1-Mv)} \left(1 + \frac{Mw}{(1-Mv)} + \mathcal{O}\left(\|w\|^2\right)\right) \\ &= f_b(v) + \frac{Mw}{(1-Mv)^2} + \mathcal{O}\left(\|w\|^2\right) \\ \Rightarrow\ f_b'(v) &= \frac{M}{(1-Mv)^2} \end{aligned} \tag{8b}$$

$$\begin{aligned} f_c(v+w) &= e^{M(v+w)} = e^{Mv}e^{Mw} = e^{Mv}\left(1 + Mw + \mathcal{O}\left(\|w\|^2\right)\right) \\ &= f_c(v) + e^{Mv}Mw + \mathcal{O}\left(\|w\|^2\right) \\ \Rightarrow\ f_c'(v) &= Me^{Mv} \end{aligned} \tag{8c}$$

and, hence,

$$\begin{aligned} f_a'(v+w) &= 0 = f_a'(v) \\ \Rightarrow\ f_a''(v) &= 0 \end{aligned} \tag{9a}$$

$$\begin{aligned} f_b'(v+w) &= \frac{M}{(1 - M(v+w))^2} = \frac{M}{(1-Mv)^2} \cdot \frac{1}{\left(1 - \frac{Mw}{1-Mv}\right)^2} \\ &= \frac{M}{(1-Mv)^2} \cdot \left(1 + \frac{Mw}{1-Mv} + \mathcal{O}\left(\|w\|^2\right)\right)^2 \\ &= f_b'(v) + \frac{M}{(1-Mv)^2} \cdot \frac{2Mw}{1-Mv} + \mathcal{O}\left(\|w\|^2\right) \\ \Rightarrow\ f_b''(v) &= \frac{2M^2}{(1-Mv)^3} \end{aligned} \tag{9b}$$

$$\begin{aligned} f_c'(v+w) &= Me^{M(v+w)} = Me^{Mv}e^{Mw} = Me^{Mv}\left(1 + Mw + \mathcal{O}\left(\|w\|^2\right)\right) \\ &= f_c'(v) + M^2e^{Mv}w + \mathcal{O}\left(\|w\|^2\right) \end{aligned}$$

$$\Rightarrow \; f_c''(v) = M^2 e^{Mv} \tag{9c}$$

3 Existence of Solutions

3.1 Case a

In the linear case $f = f_a = 1$, the solution can easily be obtained analytically, as the system of ODEs simplifies to

$$\hat{E}u_a'' - \alpha p_a' = 0, \tag{10}$$
$$p_a'' = 0, \tag{11}$$

with boundary conditions

$$u_a\big|_L = 0, \tag{12}$$
$$p_a\big|_L = 0, \tag{13}$$
$$p_a'\big|_0 = -\hat{q}, \tag{14}$$
$$\left(\hat{E}u_a' + (1-\alpha)p_a\right)\big|_0 = 0. \tag{15}$$

From (11) follows that p_a is a linear polynomial, with its coefficients defined by (13) and (14), so that

$$p_a(x) = -\hat{q}(x - L). \tag{16}$$

Thus, (10) yields

$$u_a''(x) = -\frac{\alpha\hat{q}}{\hat{E}},$$

giving that u_a is a quadratic polynomial with two more coefficients to be determined via the other two boundary conditions (12) and (15). Finally, we receive

$$u_a(x) = \frac{\hat{q}L^2}{\hat{E}}\left(-\frac{\alpha}{2}\left(\frac{x}{L}\right)^2 - (1-\alpha)\frac{x}{L} + \left(1 - \frac{\alpha}{2}\right)\right). \tag{17}$$

3.2 Case c

For the exponential permeability function of case c, an analytic solution of the form

$$\frac{\mathrm{d}u}{\mathrm{d}x} = \frac{1}{M}\ln\left|C_1 - M\frac{x}{L}\right| + C_2,$$

hence

$$u(x) = -\left(\frac{L}{M^2} + \frac{x}{M}\right) \ln\left|C_1 - M\frac{x}{L}\right| - \frac{C_1 L}{M^2} + \frac{x}{M} - C_2 x + C_4,$$

and

$$p(x) = \frac{\hat{E}}{\alpha M} \ln\left|C_1 - M\frac{x}{L}\right| + C_3$$

can be found in Eisenträger (2009) or Wirth (2005). Following the analysis to derive the integration constants from the boundary conditions, we obtain the following criterion for solvability[1]

$$\hat{q} \leq \frac{\hat{E}}{LM}(1-\alpha)^{\frac{1-\alpha}{\alpha}}.$$

3.3 Case b

Using the same steps as for case c in Eisenträger (2009), we obtain a solution of the form

$$u(x) = -\frac{C_2}{C_1 M^2} e^{-C_1 M x} + \frac{x}{M} + C_4$$
$$p(x) = \frac{\hat{E} C_2}{\alpha M} e^{-C_1 M x} + C_3,$$

where the integration constants can be derived from the boundary conditions for all parameter values.

4 Numerical Approach

Chebops can only solve linear systems of ordinary differential equations, but (2) and the boundary condition (5) are nonlinear for $f = f_b$ or $f = f_c$. So, we use Newton's Method to overcome this restriction.

Define the spaces

$$\mathcal{V} := \left(C^2[0, L]\right)^2,$$
$$\mathcal{W} := \left(C[0, L]\right)^2 \times \mathbb{R}^4$$

[1] Inequality (16) on page 21 of Eisenträger (2009) should be $\frac{a_1}{a_5}\bar{Q}e^{\bar{M}a_6\bar{p}_s} \leq \frac{\alpha}{\bar{M}}(1-\alpha)^{\frac{1-\alpha}{\alpha}}$.

and the nonlinear operator $A : \mathcal{V} \mapsto \mathcal{W}$, such that

$$A(V) := \begin{pmatrix} \hat{E}V_1'' - \alpha V_2' \\ f'(V_1')V_1''V_2' + f(V_1')V_2'' \\ V_1|_L \\ V_2|_L \\ (f(V_1')V_2')|_0 + \hat{q} \\ \left(\hat{E}V_1' + (1-\alpha)\, V_2\right)\big|_0 \end{pmatrix}. \tag{18}$$

Hence, we are looking for an approximate root $V^\star = \begin{pmatrix} u \\ p \end{pmatrix} \in \mathcal{V}$ of A.

For Newton's method, we need $A'(V)$, such that

$$A(V+W) = A(V) + A'(V)W + \mathcal{O}\left(\|W\|^2\right)$$

for all $V, W \in \mathcal{V}$. So, consider the nonlinear components of $A(V+W)$

$$\begin{aligned}
A_2(V+W) &= f'(V_1'+W_1')(V_1''+W_1'')(V_2'+W_2') + f(V_1'+W_1')(V_2''+W_2'') \\
&= f'(V_1')V_1''V_2' + \left[f''(V_1')W_1'\right] V_1''V_2' + f'(V_1')W_1''V_2' + f'(V_1')V_1''W_2' \\
&\quad + f(V_1')V_2'' + \left[f'(V_1')W_1'\right] V_2'' + f(V_1')W_2'' + \mathcal{O}\left(\|W\|^2\right) \\
&= A_2(V) + \left[f''(V_1')V_1''V_2' + f'(V_1')V_2''\right] W_1' + f'(V_1')V_2'W_1'' \\
&\quad + f'(V_1')V_1''W_2' + f(V_1')W_2'' + \mathcal{O}\left(\|W\|^2\right) \\
A_5(V+W) &= \left(f(V_1'+W_1')\left(V_2'+W_2'\right)\right)\big|_0 + \hat{q} \\
&= \left(f(V_1')V_2'\right)\big|_0 + \hat{q} + \left(\left[f'(V_1')W_1'\right] V_2'\right)\big|_0 + \left(f(V_1')W_2'\right)\big|_0 + \mathcal{O}\left(\|W\|^2\right) \\
&= A_5(V) + \left(f'(V_1')V_2'W_1'\right)\big|_0 + \left(f(V_1')W_2'\right)\big|_0 + \mathcal{O}\left(\|W\|^2\right)
\end{aligned}$$

Putting all this together with the linear components, we obtain

$$\begin{aligned}
A'(V) &= \\
&= \begin{pmatrix} \hat{E}\frac{\mathrm{d}^2}{\mathrm{d}x^2} & -\alpha\frac{\mathrm{d}}{\mathrm{d}x} \\ \left[f''(V_1')V_1''V_2' + f'(V_1')V_2''\right]\frac{\mathrm{d}}{\mathrm{d}x} + f'(V_1')V_2'\frac{\mathrm{d}^2}{\mathrm{d}x^2} & f'(V_1')V_1''\frac{\mathrm{d}}{\mathrm{d}x} + f(V_1')\frac{\mathrm{d}^2}{\mathrm{d}x^2} \\ (\cdot)|_L & 0 \\ 0 & (\cdot)|_L \\ \left(f'(V_1')V_2'\frac{\mathrm{d}}{\mathrm{d}x}\cdot\right)\Big|_0 & \left(f(V_1')\frac{\mathrm{d}}{\mathrm{d}x}\cdot\right)\Big|_0 \\ \hat{E}\left(\frac{\mathrm{d}}{\mathrm{d}x}\cdot\right)\Big|_0 & (1-\alpha)\,(\cdot)|_0 \end{pmatrix}
\end{aligned} \tag{19}$$

For Newton's Method, we want

$$A(V+W) \approx A(V) + A'(V)W = 0 \tag{20}$$
$$\Rightarrow \quad A'(V) \cdot (-W) = A(V). \tag{21}$$

So, our algorithm looks as follows:

1. Set $n = 0$ and $V_0 = 0 \in \mathcal{V}$.
2. Solve $A'(V_n) \cdot W_n = A(V_n)$ for $W_n \in \mathcal{V}$ by solving the underlying linear boundary value problem.
3. Set $V_{n+1} := V_n - W_n$.
4. Set $n \leftarrow n + 1$ and repeat from 2 until convergence.

The implementation in MATLAB for this using chebfun (Trefethen et al., 2008) is given in appendix A. Most parameters are taken similar to those proposed by Sobey and Wirth (2006) as

$$\begin{aligned}
\hat{E} &\approx 950 \frac{\mathrm{N}}{\mathrm{m}^2}, \\
\alpha &\approx 0.9, \\
L = c - a &\approx 7 \cdot 10^{-2} \mathrm{m}, \\
\hat{q} = \frac{q_0 \mu}{k_0} &= 100 \frac{\mathrm{N}}{\mathrm{m}^3}, \\
M &\approx 4.
\end{aligned}$$

To be comparable to results in Sobey and Wirth (2006), the flux should be much larger, but case c only has a solution for $\hat{q} \leq 2626 \frac{\mathrm{N}}{\mathrm{m}^3}$ and we want to stay well below this limit to avoid complications with Newton's method.

5 Results

5.1 Case a

Case a is a linear system of ordinary differential equations and has a polynomial solution for all parameter values, as we have seen in 3.1. Hence, this solution should be found exactly, up to almost machine precision, by chebops in the first Newton iteration. The chebfun solution almost satisfies this. After one Newton step, both, u and p, are quadratic functions, but the relative size of the error is of order 10^{-13}. Further Newton steps increase the order of u to cubic, but decrease the relative error, both in the L^2 and the L^∞ norm, to around 10^{-15}.

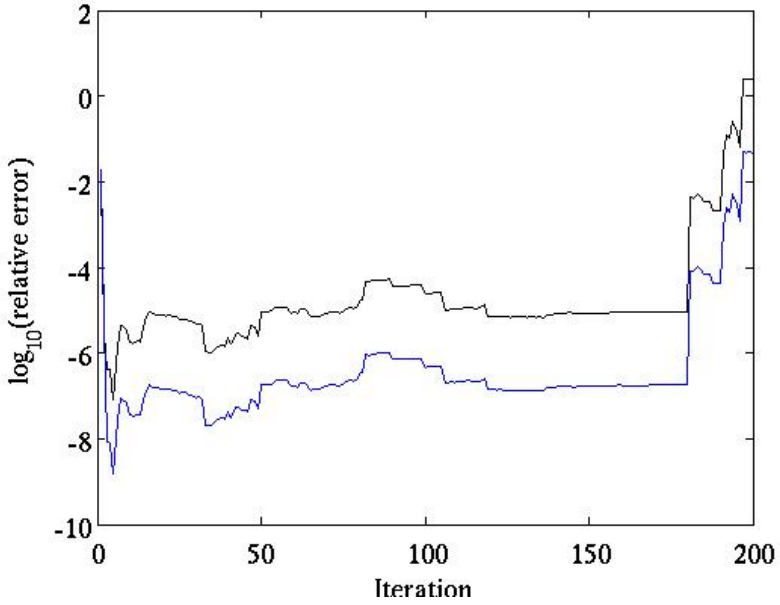

Figure 2: Relative error sizes of case b (u: black, higher, p: blue, lower)

5.2 Case b

For case b, the relative error is of order 10^{-2} after one step and decreases to about 10^{-7} after the fifth step with a polynomial of order 128. The residual for the ordinary differential equations cannot be forced below 10^{-5}. Figure 2 shows the relative error size for the displacements u and the pressure p for up to 200 iterations. The size of the relative error of the pressure always stays about 2 orders of magnitude lower than that of the displacements. This is due to their different scaling. The maximal value for the displacement is about $3 \cdot 10^{-4}$m, the maximal value for the pressure only about $7 \frac{\mathrm{N}}{\mathrm{m}^2}$. After 180 iterations, both errors jump up several orders of magnitude. It is not clear whether this behaviour is due to instabilities in chebfun and chebops, solely a problem of Newton's algorithm or could already be solved by an appropriate scaling of the unknown functions.

5.3 Case c

For this case, we first had to find the value for parameter C_1 using `fminsearch` so that the flux boundary condition (5) is fulfilled. The first Newton iteration again solves the linear problem, yielding displacement and pressure about one per cent below the analytic solution (Fig. 3). In the second step, the matrices that are built in chebops for solving the boundary value problem are very badly conditioned. The displacements jump to about 50 times of the analytical solution and afterwards seems to converge to

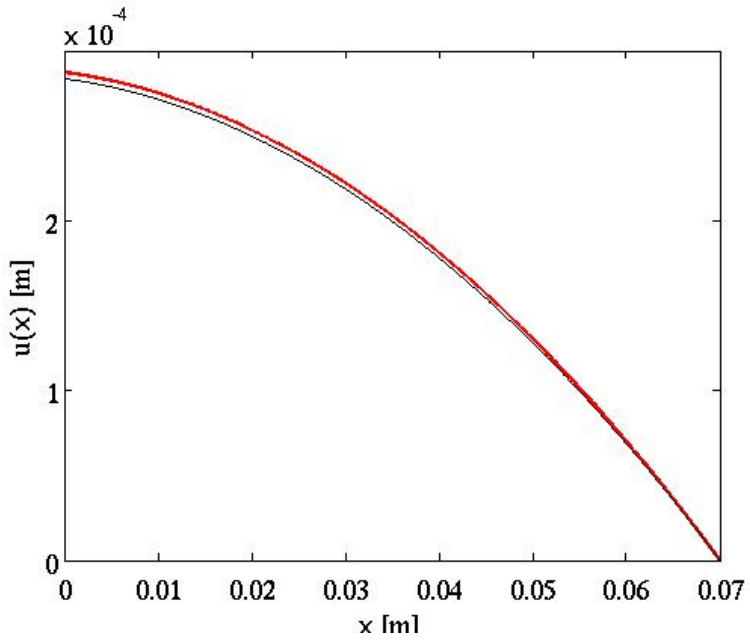

Figure 3: Displacement $u(x)$ for case c after the first Newton step (black) and the analytic solution (red)

the function given in Figure 4 with the pressure as in Figure 5, although those do not satisfy the ordinary differential equations (1) and (2). Testing for higher values of C_1, which correspond to smaller fluxes, pressures, and displacements, did not yield more satisfying results either.

6 Conclusion

Chebfun and chebops can very well be used to solve linear systems of ordinary differential equations. The main advantage is that this can be implemented in a very straightforward fashion. Furthermore, the built-in adaptivity is neat and user-friendly.

To be able to solve nonlinear problems, standard rootfinding techniques such as Newton's method can be applied. Problems these techniques have with global convergence persist in this setting as well. Also issues such as scaling and conditioning of the problem need to be addressed separately outside the chebfun code. Arbitrary precision cannot be expected from chebfun either. The chebfun representation of the analytic solution already only solves satisfies the ordinary differential equations to only eight digits precision.

Since chebfun and chebops are still rather new, there is not yet much theory known about their precision, especially in the case of solving systems of ordinary differential equations. Furthermore, they cannot yet deal with functions of multiple variables and

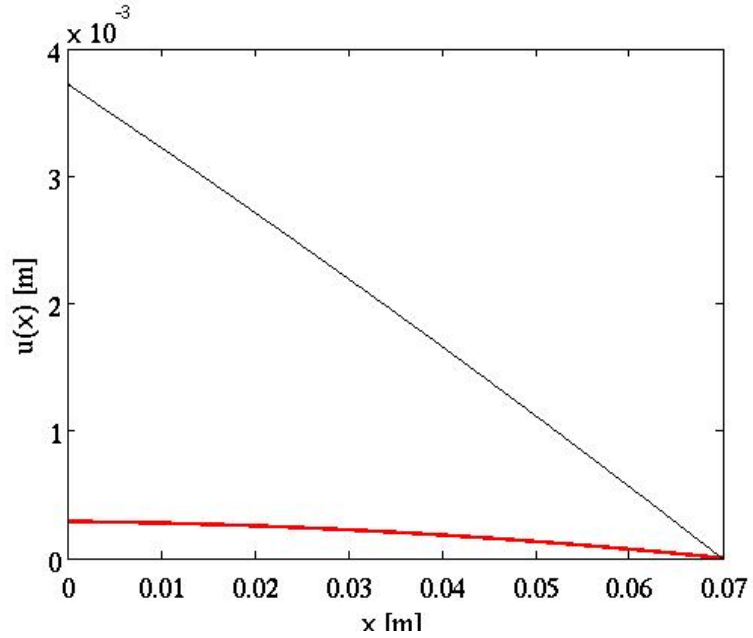

Figure 4: Displacement $u(x)$ for case c after 300 Newton steps (black) and the analytic solution (red)

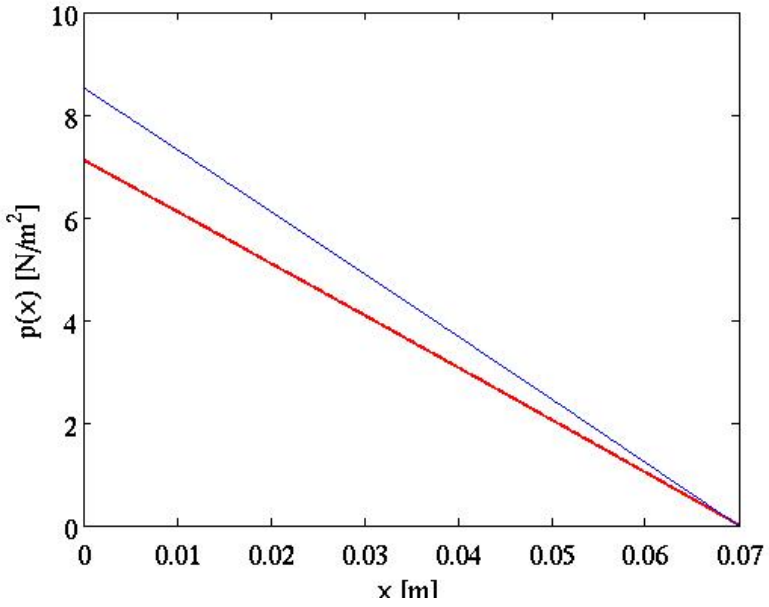

Figure 5: Pressure $p(x)$ for case c after 300 Newton steps (blue) and the analytic solution (red)

hence partial differential equations, but most problems in physics or engineering need two or three dimensions. Thus, conservative approaches such as finite differences or finite elements will be used in these cases or when knowledge about the error size is required. Nevertheless, the chebfun project is still growing and could yield very interesting results and algorithms in the future.

References

Biot, M. (1941). General theory of three dimensional consolidation. *Journal of Applied Physics*, 12:55–164.

Driscoll, T. A., Bornemann, F., and Trefethen, L. N. (2008). The chebop system for automatic solution of differential equations. To appear in Numerical Mathematics, available at `http://www.comlab.ox.ac.uk/projects/chebfun/publications/driscoll_born_tref.pdf`.

Eisenträger, A. (2009). Modelling flow of cerebrospinal fluid. Report for the transfer of status from Probationary Research Student to DPhil Student.

Sobey, I. and Wirth, B. (2006). Effect of non-linear permeability in a spherically symmetric model of hydrocephalus. *Mathematical Medicine and Biology*, 23:339–361.

Trefethen, L. N. (2007). Computing numerically with functions instead of numbers. *Mathematics in Computer Science*, 1:9–19. Available at `http://www.comlab.ox.ac.uk/projects/chebfun/publications/trefethen_functions.pdf`.

Trefethen, L. N., Pachón, R., Platte, R. B., and Driscoll, T. A. (2008). Chebfun version 2. Available at `http://www.comlab.ox.ac.uk/chebfun/`.

Wirth, B. (2005). A mathematical model for hydrocephalus. Master's thesis, Oxford University.

A MATLAB Code

```
function [u,p]=main(N)

    global relerru relerrp
    relerru=0;relerrp=0;

    close all;     tic;
    [Eh,alpha,L,qh,findex,M,tol]=Parameters;
    if not(CheckExistence(Eh,alpha,L,qh,findex,M));
        return
    end
    [d,x]=domain([0,L]); u=0*x; p=0*x; V=[u,p];
    [uana,pana,qh]=AnalyticSolution(Eh,alpha,L,qh,M,findex);
    Residual([uana,pana],Eh,alpha,L,qh,findex,M,tol)

    ToleranceMet=0;
    k=0;
    if (N>0)
        for k=1:N
            disp(' ');
            disp(['Step ',int2str(k),':']);
            V=NewtonStep(V,Eh,alpha,L,qh,findex,M);
            res=Residual(V,Eh,alpha,L,qh,findex,M,tol);
            relerru(k)=norm(uana-V(:,1))/norm(uana);
            relerrp(k)=norm(pana-V(:,2))/norm(pana);
            disp(['Tolerance not met by ',res]);
            disp(['Relative error u = ',num2str(relerru(k),'
                %10.2e\n')]);
            disp(['Relative error p = ',num2str(relerrp(k),'
                %10.2e\n')]);
        end
    else
        while not(ToleranceMet)
            k=k+1;
            disp(' ');
            disp(['Step ',int2str(k),':']);
            V=NewtonStep(V,Eh,alpha,L,qh,findex,M);
            res=Residual(V,Eh,alpha,L,qh,findex,M,tol);
            relerru(k)=norm(uana-V(:,1))/norm(uana);
            relerrp(k)=norm(pana-V(:,2))/norm(pana);
            disp(['Tolerance not met by ',res]);
            disp(['Relative error u = ',num2str(relerru(k),'
```

```
                %10.2e\n')]);
            disp(['Relative error p = ',num2str(relerrp(k),'
                %10.2e\n')]);
            ToleranceMet=strcmp(res,'');
        end
    end
     u=V(:,1);       p=V(:,2);

     figure;plot(uana,'r','linewidth',2);hold on;plot(u,'k');
         set(gca,'FontSize',15,'FontName','Times')
         xlabel('x [m]')
         ylabel('u(x) [m]')
     figure;plot(pana,'r','linewidth',2);hold on;plot(p,'b');
         set(gca,'FontSize',15,'FontName','Times')
         xlabel('x [m]')
         ylabel('p(x) [N/m^2]')
     figure;plot((uana-u)/norm(uana),'k');hold on;
         plot((pana-p)/norm(pana),'b');
         set(gca,'FontSize',15,'FontName','Times')
         xlabel('x [m]')
         ylabel('relative error sizes')

     figure;plot(log(relerru)/log(10),'k');hold on;
         plot(log(relerrp)/log(10),'b');
         set(gca,'FontSize',15,'FontName','Times')
         ylabel('log_{10}(relative error)')
         xlabel('Iteration')

     disp(' ');toc,disp(' ');
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parameters
function [Eh,alpha,L,qh,findex,M,tol]=Parameters
    Eh=950;
    alpha=0.9;
    L=7e-2;
    qh=100;
        findex='c';
    M=4;
     tol=1e-8;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for Existence of Solutions % ok
function SolExists=CheckExistence(Eh,alpha,L,qh,findex,M)
    SolExists=true;
    if findex=='a' || findex=='b'
        return
    end
    if findex=='c'
        if ( alpha<1 )
            qhmax=Eh/M/alpha/L* ...
                (exp( (1-alpha)*log(1-alpha)/alpha ) ...
                 -exp( log(1-alpha)/alpha ) );
            if (qh> qhmax)
                disp('There exists no solution for this problem.');
                disp(['You need qh < ', num2str(qhmax)]);
                SolExists=false;
            end
        %else % Existence of a unique solution proven
        end
        return
    end

    % other findex
    error('findex must be a, b or c');

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DiffOperator AV=A(V)
function [AVode,AVbc]=AV(V,Eh,alpha,L,qh,findex,M)
    V1=V(:,1);        V2=V(:,2);
    V1p=diff(V1);     V2p=diff(V2);
    V1pp=diff(V1p); V2pp=diff(V2p);
    fV1p=f(V1p,M,findex);
    AVode=[Eh*V1pp-alpha*V2p,...
           fp(V1p,M,findex).*V1pp.*V2p+fV1p.*V2pp];
    y=fV1p.*V2p+qh;
    z=Eh*V1p+(1-alpha)*V2;
    AVbc=[V1(L),V2(L),y(0),z(0)];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Derivative of A
```

```
function A=ApV(V,BC,Eh,alpha,findex,M)
    V1=V(:,1); V2=V(:,2); d=domain(V1);
    Z=zeros(d); I=eye(d); D=diff(d); D2=diff(d,2);
    V1p=D*V1;   V2p=D*V2;
    V1pp=D2*V1; V2pp=D2*V2;

    diagfV1p=diag(f(V1p,M,findex));
    diagfpV1p=diag(fp(V1p,M,findex));

    ApV11= Eh*D2;
    ApV12=-alpha*D;
    ApV21=    diag(fpp(V1p,M,findex))*diag(V1pp)*diag(V2p)*D...
              +diagfpV1p*diag(V2pp)*D  +diagfpV1p*diag(V2p)*D2;
    ApV22=diagfpV1p*diag(V1pp)*D +diagfV1p*D2;
    A=[ApV11 ApV12;ApV21 ApV22];

    A.rbc(1)={[I Z],BC(1)};
    A.rbc(2)={[Z I],BC(2)};
    A.lbc(1)={[diagfpV1p*diag(V2p)*D diagfV1p*D],BC(3)};
    A.lbc(2)={[Eh*D (1-alpha)*I],BC(4)};

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NewtonIteration
function V=NewtonStep(V,Eh,alpha,L,qh,findex,M)
    [RHS,bcrhs]=AV(V,Eh,alpha,L,qh,findex,M);
    ODE=ApV(V,bcrhs,Eh,alpha,findex,M);
    W=ODE\RHS;
    V=V-W;

    disp(['change u = ',num2str(norm(W(:,1))/norm(V(:,1)))]);
    disp(['change p = ',num2str(norm(W(:,2))/norm(V(:,2)))]);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Residual
function str=Residual(V,Eh,alpha,L,qh,findex,M,tol)

    [AVode,AVbc]=AV(V,Eh,alpha,L,qh,findex,M);

    str='';
    if (norm(AVode(:,1))>tol)
        str=[str , 'ODE1, '];
```

```
    end
    if (norm(AVode(:,2))>tol)
        str=[str , 'ODE2, '];
    end
    if (norm(AVbc(1))>tol)
        str=[str , 'BC1, '];
    end
    if (norm(AVbc(2))>tol)
        str=[str , 'BC2, '];
    end
    if (norm(AVbc(3))>tol)
        str=[str , 'BC3, '];
    end
    if (norm(AVbc(4))>tol)
        str=[str , 'BC4, '];
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Permeability function
function fy=f(y,M,findex)
    if findex=='a'
        fy=0*y+1;
    elseif findex=='b'
        fy=1./(1-M*y);
    elseif findex=='c'
        fy=exp(M*y);
    else
        error('findex must be a, b or c');
    end
end
function fpy=fp(y,M,findex)
    if findex=='a'
        fpy=0*y;
    elseif findex=='b'
        fpy=M./(1-M*y).^2;
    elseif findex=='c'
        fpy=M*exp(M*y);
    else
        error('findex must be a, b or c');
    end
end
function fppy=fpp(y,M,findex)
```

```
    if findex=='a'
        fppy=0*y;
    elseif findex=='b'
        fppy=2*M*M./(1-M*y).^3;
    elseif findex=='c'
        fppy=M*M*exp(M*y);
    else
        error('findex must be a, b or c');
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Analytic Solution
function [uana,pana,qh]= AnalyticSolution(Eh,alpha,L,qh,M,findex)
    [d,x]=domain([0,L]);

    switch findex
        case 'a'
            uana=-alpha*qh/2/Eh*x.^2-(1-alpha)*qh/Eh*L*x+(1-alpha/2)*qh*L*L/Eh;
            pana=-qh*(x-L);
        case 'b'
            C1=-alpha*qh/Eh;
            C2=-alpha/(1-   (1-alpha)*exp(alpha*qh*M*L/Eh));
            C3=-C2*Eh*exp(-C1*M*L)/M/alpha;
            C4=(C2*exp(-C1*M*L)-C1*M*L)/C1/M/M;

            uana=-C2*exp(-C1*M*x)/C1/M/M+x/M+C4;
            pana=Eh*C2/alpha/M*exp(-C1*M*x)+C3;
        case 'c'
            C1=1.503425216674805e+02;
            logC1=log(abs(C1));
            logC1mM=log(abs(C1-M));
            C3=-Eh/alpha/M*logC1mM;
            C2=-logC1/M/alpha   +logC1mM/M/alpha*(1-alpha);
            pana=Eh/alpha/M*log(abs(C1-M*x/L))+C3;
            epsilonc=log(abs(C1-M*x/L))/M+C2;
            uana=cumsum(epsilonc);
            uana=uana-uana(L);
            qhfunction=-f(epsilonc,M,'c').*diff(pana);
            qh=qhfunction(0);
    end
end
```