A linear-elastic-nonlinear-swelling theory for hydrogels. Part 2. Displacement formulation

Electronic supplementary material

Joseph J. Webber[1] †, Merlin A. Etzold[1] and M. Grae Worster[1]

[1]Department of Applied Mathematics and Theoretical Physics, Centre for Mathematical Sciences, Wilberforce Road, Cambridge CB3 0WA, UK

1. Experimental details

To carry out the motivating experiments for the drying cylinders, cubes of commercially-available poly–(acrylamide–potassium–acrylate) copolymer (*Deco Cubes*™, JRM Chemical Inc, 4881 NEO Parkway, Cleveland, Ohio 44128, USA) were placed in deionised water and left to swell until they reached a steady size. These were then removed, and placed in a dish of deionised water, which was kept topped up through the process of the experiment such that it never dried out fully. No quantitative measurements were taken at any point, and the experiments were used solely to motivate the subsequent theoretical analysis that followed, with the key qualitative features (curved top and bottom interfaces) apparent.

2. A numerical scheme for solving equation (5.36)

Since the size of the domain on which this differential equation is to be solved changes in time, introduce the scaled variable $Y = Z/\mathcal{H}(T)$ and instead solve on $Y \in [0,\ 1]$. Then

$$\frac{\partial}{\partial T} \to \frac{\partial}{\partial T} - \frac{Y}{\mathcal{H}}\frac{\partial \mathcal{H}}{\partial T}\frac{\partial}{\partial Y} \quad \text{and} \quad \frac{\partial}{\partial Z} \to \frac{1}{\mathcal{H}}\frac{\partial}{\partial Y}, \tag{2.1}$$

and equation (5.36) thus becomes, after rearrangement,

$$\begin{aligned}\frac{\partial \Phi}{\partial T} - \Phi^{1/3}\frac{\partial \Phi}{\partial Y}\int_0^Y \frac{\partial \Phi^{1/3}}{\partial T}\,\mathrm{d}Y' - \frac{\Phi^{1/3}}{\mathcal{H}}\frac{\partial \mathcal{H}}{\partial T}\frac{\partial \Phi}{\partial Y}\int_0^Y \Phi^{1/3}\,\mathrm{d}Y' = \\ \frac{\Phi}{\mathcal{H}^2}\frac{\partial}{\partial Y}\left[f(\Phi)\frac{\partial \Phi}{\partial Y}\right] - \frac{1}{\mathcal{H}}\left[\frac{2f(\Phi)}{3\mathcal{H}}\frac{\partial \Phi}{\partial Y} + Y\left(\phi^{2/3}-1\right)\frac{\partial \mathcal{H}}{\partial T}\right]\frac{\partial \Phi}{\partial Y} + 2\Phi^{4/3}U_s,\end{aligned} \tag{2.2}$$

with the height $\mathcal{H}(T)$ set as

$$\mathcal{H}(T) = \left(\int_0^1 \Phi^{1/3}\,\mathrm{d}Y'\right)^{-1}, \tag{2.3}$$

and the Neumann boundary condition replaced by

$$f(\Phi)\frac{\partial \Phi}{\partial Y} = \mathcal{H}U_t \quad \text{at } Y = 1. \tag{2.4}$$

Now, discretising the domain $[0,\ 1]$, and sampling at $Y = 0,\ \Delta,\ \ldots,\ N\Delta$, let $\boldsymbol{U}$ be the $(N+1)$-dimensional vector with i^{th} component

$$U_i = \Phi((i-1)\Delta,\ T), \tag{2.5}$$

† Email address for correspondence: j.webber@damtp.cam.ac.uk

for $\Delta = 1/N$. Similarly define $\boldsymbol{V}$ to have i^{th} component

$$V_i = \Phi((i-1)\Delta, T)^{-2/3} \left.\frac{\partial \Phi}{\partial T}\right|_{Z=(i-1)\Delta}. \tag{2.6}$$

Also introduce the diagonal matrices $\boldsymbol{U}$ with entries $U_i^{2/3}$ and $\boldsymbol{D}$ with entries $D_i = \Phi^{1/3}\, \partial\Phi/\partial Y$ evaluated at $Y = (i-1)\Delta$,

$$\boldsymbol{U} = \operatorname{diag}(U_1^{2/3}, U_2^{2/3}, \ldots, U_{N+1}^{2/3}) \quad \text{and} \quad \boldsymbol{D} = \operatorname{diag}(D_1, D_2, \ldots, D_{N+1}). \tag{2.7}$$

Then, the time derivative $\partial\Phi/\partial T$ in equation (2.2) can be replaced by $\boldsymbol{U}\boldsymbol{V}$. It is also now possible to discretise the integrals, noticing that

$$\int_0^Y \Phi^{-2/3} \frac{\partial \Phi}{\partial T}\, \mathrm{d}Y \to \frac{1}{2N} \underbrace{\begin{pmatrix} 0 & 0 & \ldots & 0 \\ 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 0 \\ 1 & 1 & \ldots & 1 \end{pmatrix}}_{(N+1)\times N} \underbrace{\begin{pmatrix} 1 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 1 & 0 \\ 0 & 0 & \ldots & 0 & 1 & 1 \end{pmatrix}}_{N\times(N+1)} \boldsymbol{V}, \tag{2.8}$$

where the first matrix represents the integral and the second matrix product gives the values of $\Phi^{-2/3}\, \partial\Phi/\partial T$ in the centres of each grid space. This can be more concisely written $\boldsymbol{M}\boldsymbol{V}$, for

$$\boldsymbol{M} = \frac{1}{2N} \begin{pmatrix} 0 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 1 & 2 & 1 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 2 & 2 & 2 & \ldots & 2 & 1 \end{pmatrix}, \tag{2.9}$$

an $(N+1)\times(N+1)$ matrix. Then, equation (2.2) can be rewritten in discretised form with

$$\left(\boldsymbol{U} - \frac{2}{3}\boldsymbol{D}\boldsymbol{M}\right)\boldsymbol{V} = \boldsymbol{F}\left(U, \mathcal{H}, \dot{\mathcal{H}}; \mathcal{M}, \mathcal{Q}\right), \tag{2.10}$$

where spatial derivatives are approximated as finite differences and $\boldsymbol{F}$ represents the right-hand side terms. Then,

$$\frac{\partial \boldsymbol{U}}{\partial T} = \left[\left(\boldsymbol{U} - \frac{2}{3}\boldsymbol{D}\boldsymbol{M}\right)^{-1} \boldsymbol{F}\left(U, \mathcal{H}, \dot{\mathcal{H}}; \mathcal{M}, \mathcal{Q}\right)\right]\boldsymbol{U}. \tag{2.11}$$

This is then solved with a predictor-corrector method. Let $\boldsymbol{U}^{(n)} = \boldsymbol{U}(n\Delta T)$, and use the same superscript notation for other quantities evaluated at $T = n\Delta T$, then

$$\begin{aligned} \boldsymbol{U}^{(n+\frac{1}{2})} &= \boldsymbol{U}^{(n)} + \frac{\Delta T}{2}\left[\left(\boldsymbol{U}^{(n)} - \frac{2}{3}\boldsymbol{D}\boldsymbol{M}^{(n)}\right)^{-1} \boldsymbol{F}\left(\boldsymbol{U}^{(n)}, \mathcal{H}^{(n)}, \dot{\mathcal{H}}^{(n)}; \mathcal{M}, \mathcal{Q}\right)\right]\boldsymbol{U}^{(n)} \quad \text{and} \\ \boldsymbol{U}^{(n+1)} &= \boldsymbol{U}^{(n)} + \Delta T\left[\left(\boldsymbol{U}^{(n+\frac{1}{2})} - \frac{2}{3}\boldsymbol{D}\boldsymbol{M}^{(n+\frac{1}{2})}\right)^{-1} \boldsymbol{F}\left(\boldsymbol{U}^{(n+\frac{1}{2})}, \mathcal{H}^{(n+\frac{1}{2})}, \dot{\mathcal{H}}^{(n+\frac{1}{2})}; \mathcal{M}, \mathcal{Q}\right)\right]\boldsymbol{U}^{(n+\frac{1}{2})}. \end{aligned} \tag{2.12}$$

We impose the Dirichlet boundary condition on the base of the gel by requiring $U_1^{(n)} = 1$ and the evaporative Neumann condition at the top by requiring

$$\frac{U^{(n)}_{N+1} - U^{(n)}_{N}}{\Delta} = \frac{\mathcal{H}(T)U_t}{1 + (4\mathcal{M}/3)\left(U^{(n)}_{N+1}\right)^{1/3}}. \tag{2.13}$$

This system was solved using MATLAB for all of the plots and numerical results in this paper, carrying out the matrix inversion using `mldivide`. An example function which solves this equation is shown below for illustrative purposes.

3. MATLAB code

```
function [phiReturns, dPhiReturns, hReturns] = numericalDry(M, Ut, Us, times, N, dt)
    %NUMERICALDRY Numerically solve the governing equation for the polymer fraction
    %   field when drying of a cylinder from the top and sides. Returns phiReturns, the polymer
    %   fraction field; dPhiReturns, the Y-derivative of the polymer fraction field; hReturns,
    %   the height of the cylinder \mathcal{H}(T) -- all at the specified timesteps.

    % M: the material parameter M
    % Ut: the non-dimensional flux Ut from the top
    % Us: the non-dimensional flux Us from the sides

    % times: the times to save values at

    % N: the number of spatial grid steps
    % dt: the timestep

    % ---

    % Number of steps to save at
    savedSteps = max(size(times, 2), size(times, 1)); % allow for row/column vectors for time

    % *** Variables to return
    phiReturns = NaN*ones(N+1, savedSteps);
    dPhiReturns = NaN*ones(N+1, savedSteps);
    hReturns = NaN*ones(1, savedSteps);

    t = 0;
    currentIndex = 1;

    % *** Key variables used at each timestep
    p = ones(N+1, 1); % polymer fraction, evaluated at edges of cells
```

```
    pMids = ones(N, 1); % polymer fraction, evaluated at middles of cells

    dpMids = zeros(N, 1); % dPhi/dY, evaluated at middles of cells
    dp = zeros(N+1, 1); % dPhi/dY, evaluated at edges of cells
    dp(end) = Ut/(1+4*M/3); % (straight away, set evaporation flux from top)

    h = 1; % scaled height of gel
    hDot = 0; % dH/dT

    % *** Construct the matrix M
    MBase = zeros(N+1, N+1);
    for k = 2 : N+1
        MBase(k, 1) = 1/(2*N);
        if(k~=2)
            for j = 2:(k-1)
                MBase(k, j) = 1/N;
            end
        end
        MBase(k, k) = 1/(2*N);
    end

    % *** Loop until we've reached the final timestep
    while (currentIndex <= savedSteps)
        holdP = p; % introduce a holding variable for polymer fraction to allow the predictor-corrector steps

        % Step forward by dt/2 and correct to improve stability
        for timestep = [dt/2 dt]
            % *** Diffusive bracket term
            bktMids = (1 + (4*M/3)*pMids.^(-2/3)).*dpMids;
            baseBkt = (1+(4*M/3)*p(1)^(-2/3))*dp(1);
            endBkt = Ut*h;

            % d[]/dY
            dBkt = N*[2*(bktMids(1)-baseBkt); bktMids(2:end)-bktMids(1:end-1); 2*(endBkt - bktMids(end))];

            % prefactor is the matrix U-(2/3)DM
            prefactor = diag(holdP.^(2/3)) - (2/3)*diag(dp.*holdP.^(1/3))*MBase;

            % *** Advective terms on the right-hand side
            advect = (2/(3*h^2))*(1+(4*M/3)*holdP.^(-2/3)).*dp.^2 + ...
                (hDot/h)*linspace(0, 1, N+1)'.*(holdP.^(2/3)-1).*dp - ...
                (hDot/h)*MBase*holdP.^(1/3);
```

```

            % The right-hand side
            rhs = holdP.*dBkt/h^2 + 2*holdP.^(4/3)*Us - advect;

            % Calculate d(Phi)/dT using mldivide
            dpdt = (prefactor\rhs)./holdP.^(-2/3);

            % *** TIMESTEP
            holdP = p + timestep*dpdt;
            holdP(1) = 1; % Dirichlet BC on base

            % Get midpoint phis
            pMids = (holdP(1:end-1)+holdP(2:end))/2;

            % Calculate d(Phi)/dY, including applying Neumann BC on top
            dp = N*[2*(pMids(1)-holdP(1)); pMids(2:end)-pMids(1:end-1); h*Ut/(N*(1+(4*M/3)*holdP(end)^(-2/3)))];
            dpMids = (dp(1:end-1)+dp(2:end))/2;

            % Calculate H and Hdot
            hNew = N/sum(pMids.^(1/3));
            hDot = (hNew-h)/dt;
            h = hNew;
        end

        % Catch numerical errors
        if(isnan(holdP))
            break;
        end

        p = holdP;

        % Save the values if we're at a reporting step
        if(t >= times(currentIndex))
            phiReturns(:, currentIndex) = p;
            dPhiReturns(:, currentIndex) = dp;
            hReturns(currentIndex) = h;

            % Increment next counter
            currentIndex = currentIndex + 1;
        end

        t = t+dt;
    end
end
```