

Boundary Conditions: Dirichlet BC

```
clear, close all, clc
set_defaults()
```

Dimensional problem

Initially we neglect the spherical cap geometry and simply assume a linear domain (for now!). A steady confined groundwater aquifer across the entire southern highlands leads following governing equations

$$-\frac{d}{dx} \left[bK \frac{dh}{dx} \right] = f_s \text{ for } x \in [0, l],$$

where b is the thickness of the aquifer and f_s is the precipitation. The boundary conditions are given by:

1. $\left. \frac{dh}{dx} \right|_0 = 0$ - due to the symmetry at the south pole
2. $h(x=l) = h_o$ - ocean level at dichotomy

The analytical solution is given by

$$h(x) = h_o + \frac{f_s l^2}{2bK} \left(1 - \left(\frac{x}{l} \right)^2 \right)$$

Assume the following problem parameters:

```
yr2s = 60^2*24*365.25; % second per year
rho = 1e3;             % [kg/m^3] density of water
grav = 3.711;          % [m/s^2] grav. acceleration on Mars
k = 1e-11;             % [m^2] permeability (Hanna & Phillips 2005)
mu = 1e-3;            % [Pa s] water viscosity
b = 5e3;
precip = 0.13;         % [mm/yr] equatorial precipitation (Andrews-Hanna 2010)
theta_bnd = acos(1/3); % [rad] co-latitude of dichotomy bnd
R = 3389508;          % [m] radius of mars
ho = -500;            % [m] sealevel (for now)
```

Derived quantities

```
K = k*rho*grav/mu      % [m/s] hydraulic conductivity
```

```
K = 3.7110e-05
```

```
fp = precip/1e3/yr2s   % [m/s] precipitation
```

```
fp = 4.1195e-12
```

```
l = R*(pi-theta_bnd)   % [m] distance from south pole to dichotomy bnd
```

```
l = 6.4761e+06
```

```
xdim = linspace(0,1,1e2);  
hdim = @(x) ho + fp*l^2/(2*b*K)*(1-(x/l).^2);  
  
subplot 121  
plot(xdim/1e3,hdim(xdim))  
pbaspect([1 .8 1])  
xlabel('x [km]')  
ylabel('h [m]')  
title('Dimensional solution')
```

Dimensionless problem

Introducing the following scales for the variables

$$h' = \frac{h - h_o}{f_s l^2 / bK} \text{ and } x' = \frac{x}{l}$$

we have the following dimensionless governing equation

$$-\frac{d^2 h'}{dx'^2} = 1 \text{ for } x' \in [0, 1]$$

with the boundary conditions

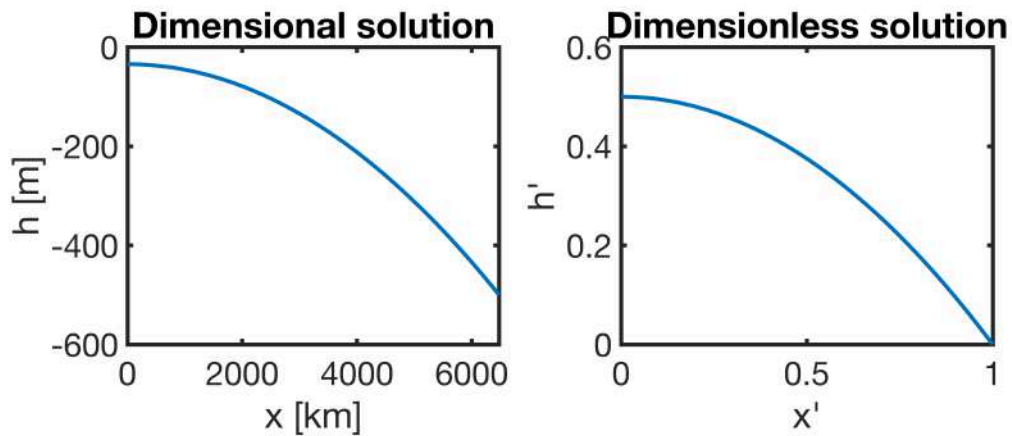
1. $\left. \frac{dh'}{dx'} \right|_{x'=0} = 0$
2. $h'(x' = 1) = 0$

The dimensionless analytic solution is given by

$$h'(x') = \frac{1}{2}(1 - x'^2).$$

Below we will be solving the dimensionless system.

```
hprime = @(xprime) .5*(1-xprime.^2);  
xprime = linspace(0,1,1e2);  
  
subplot 122  
plot(xprime,hprime(xprime))  
pbaspect([1 .8 1])  
xlabel('x''')  
ylabel('h''')  
title('Dimensionless solution')
```



Imposing Dirichlet BC's by eliminating constraints

Building the system matrix \mathbf{L}

Using discrete operators the partial differential equation can be discretized as follows

$$\mathbf{L} \cdot \mathbf{u} = \mathbf{f_s}$$

where $\mathbf{L} = -\mathbf{D} \cdot \mathbf{G}$ is the discrete Laplacian operator, \mathbf{u} is the unknown vector of temperatures and $\mathbf{f_s}$ is the right hand side vector. Today we will discuss how to discretize the boundary conditions.

Without boundary conditions the problem is ill-posed and does not have a solution. This is reflected in the condition number of the discrete Laplacian operator, \mathbf{L} .

```
Grid.xmin = 0; Grid.xmax = 1; Grid.Nx = 20;
Grid = build_grid(Grid);
[D,G,I] = build_ops(Grid);
L = -D*G;
condest(L)
```

```
ans = Inf
```

A matrix with infinite condition number has no inverse. This is because there is an infinite number of possible solutions to the Laplace equation, only the boundary conditions (BC's) make the solution unique.

Dirichlet BC's as a linear system

Dirichlet BC's prescribe the solution on the boundary. In the discrete solution they prescribe the solution in the cells neighboring the boundaries. This constraint can be formulated as a linear system,

$$\mathbf{B}^* \mathbf{u} = \mathbf{g}$$

where \mathbf{B} is the constrain matrix, \mathbf{u} is the vector of unknowns (temperature), and \mathbf{g} is a right hand side vector. The constrain matrix \mathbf{B} is N_C by N_x , where N_C is the number of constraints, i.e., cells along Dirichlet boundaries with prescribed temperatures. This means that Dirichlet BC's provide constraints that reduce the overall number of unknown we need to solve for.

Therefore the boundary value problem is described by two linear systems

- 1) $\mathbf{L}^* \mathbf{u} = \mathbf{f}\mathbf{s}$, arising from the PDE, where \mathbf{L} is the N_x by N_x **system matrix**
- 2) $\mathbf{B}^* \mathbf{u} = \mathbf{g}$, arising from the BC's, where \mathbf{B} is the N_C by N_x **constraint matrix**

Neither \mathbf{L} nor \mathbf{B} is invertible, both allow infinite solutions. To find the unique solution to the boundary value problem, the constraints in \mathbf{B} must be eliminated from the system matrix \mathbf{L} .

⇒ we need to understand how to eliminate constraints

Building the constraint matrix

Suppose we have the following two N_C by 1 column vectors:

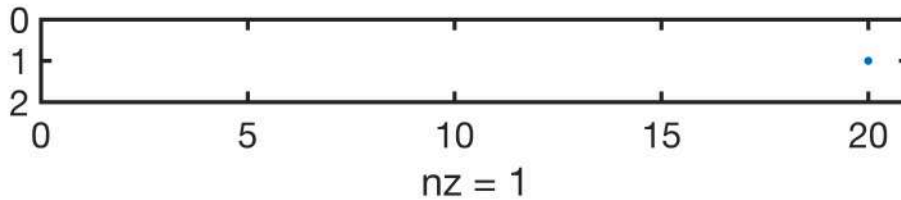
1. `dof_dir`: contains the degrees of freedom (dofs), i.e., cell numbers, of all cells along the Dirichlet boundary.
2. `g`: contains the prescribed values the unknown is set to along the Dirichlet boundary.

The constraint matrix, \mathbf{B} , needs to set the unknown in `dof_dir` to `g`. The matrix \mathbf{B} therefore comprises the rows of the N_x by N_x identity matrix, \mathbf{I} , that correspond to `dof_dir`. Therefore \mathbf{B} can be built as follows

```
dof_dir = [Grid.dof_xmax];
B = I(dof_dir,:);
size(B)
```

```
ans = 1x2
      1      20
```

```
clf
spy(B)
```



The resulting constraint matrix has one row for every cell that is set to a prescribed value by the Dirichlet BC's. In the 1D case with Dirichlet BC's at one ends $N_C = 1$ and \mathbf{B} only has 1 rows. This simple construction will remain the same, even in higher dimensions.

Homogeneous constraints

Initially, we consider a problem with homogeneous constraints, i.e., $T_0 = 0$. The discrete problem is then given by

$$\text{PDE: } \mathbf{L}^* \mathbf{u} = \mathbf{f} \mathbf{s}$$

$$\text{BC's: } \mathbf{B}^* \mathbf{u} = \mathbf{0}$$

Reduced linear system

Given that the constraints in \mathbf{B} reduce the number of unknown we expect to solve a smaller or reduced linear system of size $(N_x - N_C)$ by $(N_x - N_C)$

$$\mathbf{L} \mathbf{r}^* \mathbf{u} \mathbf{r} = \mathbf{f} \mathbf{s} \mathbf{r}.$$

Here the variables are:

1. $\mathbf{u} \mathbf{r}$ is the $(N_x - N_C)$ by 1 reduced vector of unknowns.
2. $\mathbf{f} \mathbf{s} \mathbf{r}$ is the $(N_x - N_C)$ by 1 reduced r.h.s. vector.
3. $\mathbf{L} \mathbf{r}$ is the $(N_x - N_C)$ by $(N_x - N_C)$ reduced system matrix.

Projection matrix

What is the relation between \mathbf{u} and $\mathbf{u_r}$, $\mathbf{f_s}$ and $\mathbf{f_{sr}}$, and \mathbf{L} and $\mathbf{L_r}$? Two vectors of different length are related by a rectangular matrix

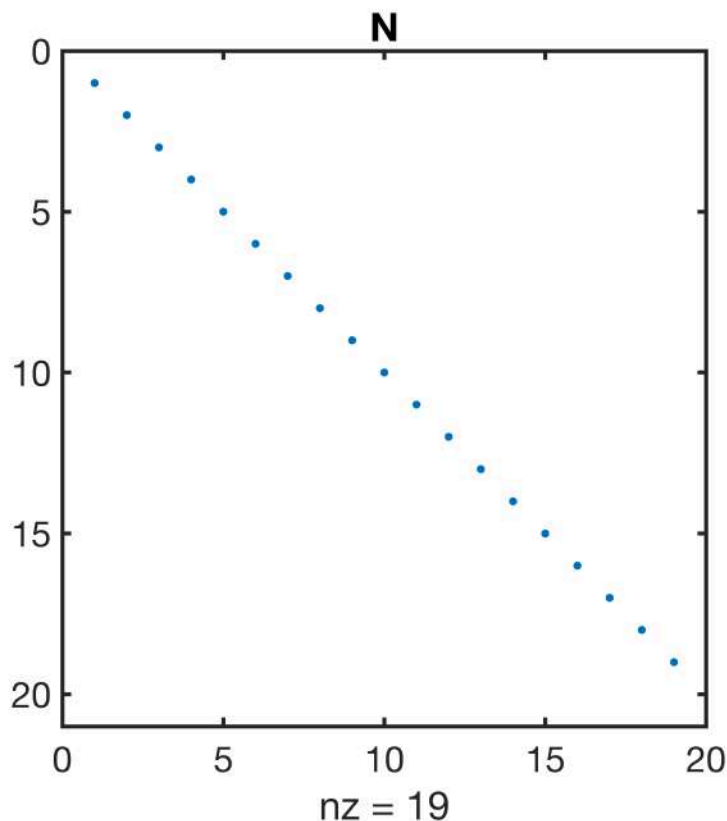
$$\mathbf{u} = \mathbf{N}^* \mathbf{u_r} \text{ and } \mathbf{f_s} = \mathbf{N}^* \mathbf{f_{sr}}$$

where \mathbf{N} is a N_x by $(N_x - N_c)$ matrix. Here \mathbf{N} is any basis for the [nullspace](#) of the constraint matrix \mathbf{B} . The nullspace of \mathbf{B} is simply the set of all solutions that satisfy $\mathbf{B}^* \mathbf{u} = \mathbf{0}$, i.e., all the possible solutions that satisfy the homogeneous boundary conditions. If we search for solutions to $\mathbf{L}^* \mathbf{u} = \mathbf{f_s}$ in the nullspace of \mathbf{B} , then the BC's are automatically satisfied. In Matlab the nullspace of a matrix can be found with the function [null\(\)](#) or [spnull\(\)](#) for sparse matrices.

```
N = spnull(B);  
size(N)
```

```
ans = 1x2  
      20      19
```

```
spy(N), title 'N'
```



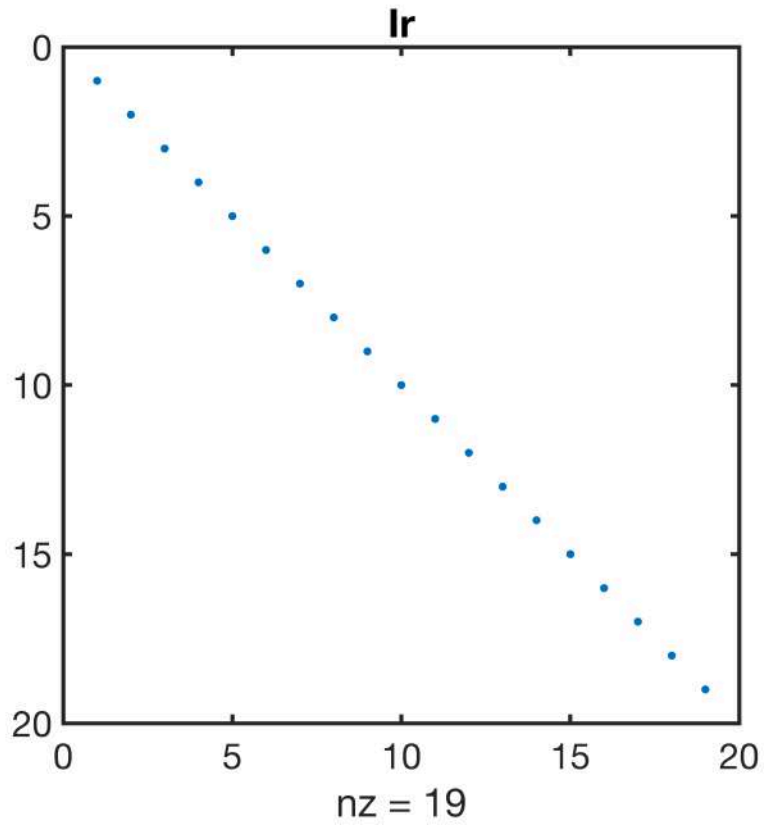
Assume that \mathbf{N} is orthonormal, i.e., that the dot product between all columns is unity. then it follows that

1. $\mathbf{N}'^* \mathbf{N} = \mathbf{I_r}$, where $\mathbf{I_r}$ is the $(N_x - N_c)$ by $(N_x - N_c)$ identity matrix in the reduced space.
2. $\mathbf{N}^* \mathbf{N}' = \mathbf{I_c}$, where $\mathbf{I_c}$ is the N_x by N_x "identity matrix" with N_c zeros on the diagonal.

```
Ir = N'*N;  
size(Ir)
```

```
ans = 1x2  
    19    19
```

```
spy(Ir), title 'Ir'
```



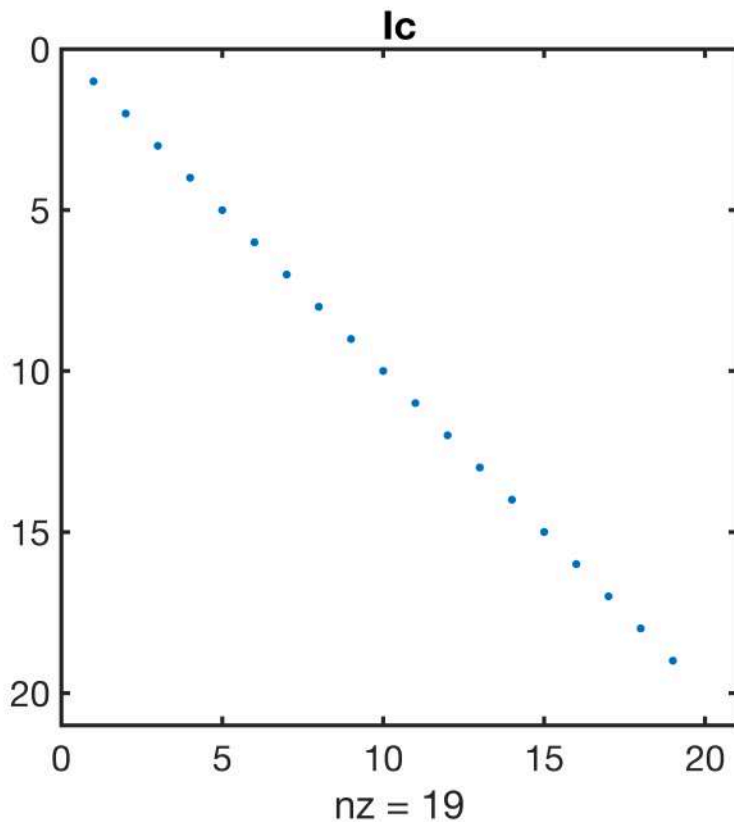
```
non_0_entries_Ir = full(sum(diag(Ir)))
```

```
non_0_entries_Ir = 19
```

```
Ic = N*N';  
size(Ic)
```

```
ans = 1x2  
    20    20
```

```
spy(Ic), title 'Ic'
```



```
non_0_entries_Ic = full(sum(diag(Ic)))
```

```
non_0_entries_Ic = 19
```

In this case, we have the following relationship, $\mathbf{N}' * \mathbf{u} = \mathbf{N}' * \mathbf{N} * \mathbf{ur} = \mathbf{I_r} * \mathbf{ur} = \mathbf{ur}$, so that \mathbf{N} and \mathbf{N}' allow us to go forth and back between \mathbf{u} and \mathbf{ur} :

$$\mathbf{u} = \mathbf{N} * \mathbf{ur}$$

$$\mathbf{ur} = \mathbf{N}' * \mathbf{u}$$

Of course, the same relationship exists between \mathbf{fsr} and \mathbf{fs} , $\mathbf{fsr} = \mathbf{N}' * \mathbf{fs}$.

The matrix \mathbf{N}' *projects* the vector of unknowns into the nullspace of \mathbf{B} . Note that a proper [projection matrix](#) is square, it would simply zero out the entries that are not in the nullspace. Instead, our \mathbf{N}' matrix eliminates these entries, but the idea is the same.

Reduced system matrix

Given the properties of \mathbf{N} , defined above, the expression for the reduced system matrix is derived as follows

$$\mathbf{L} * \mathbf{u} = \mathbf{fs}$$

$$\mathbf{N}' * \mathbf{L} * \mathbf{u} = \mathbf{N}' * \mathbf{fs}$$

$$\mathbf{N}' * \mathbf{L} * \mathbf{I} * \mathbf{c} * \mathbf{u} = \mathbf{N}' * \mathbf{f} * \mathbf{s}$$

$$\mathbf{N}' * \mathbf{L} * (\mathbf{N} * \mathbf{N}') * \mathbf{u} = \mathbf{N}' * \mathbf{f} * \mathbf{s}$$

$$(\mathbf{N}' * \mathbf{L} * \mathbf{N}) * (\mathbf{N}' * \mathbf{u}) = \mathbf{N}' * \mathbf{f} * \mathbf{s}$$

$$\mathbf{L} * \mathbf{r} * \mathbf{u} = \mathbf{f} * \mathbf{s} * \mathbf{r}$$

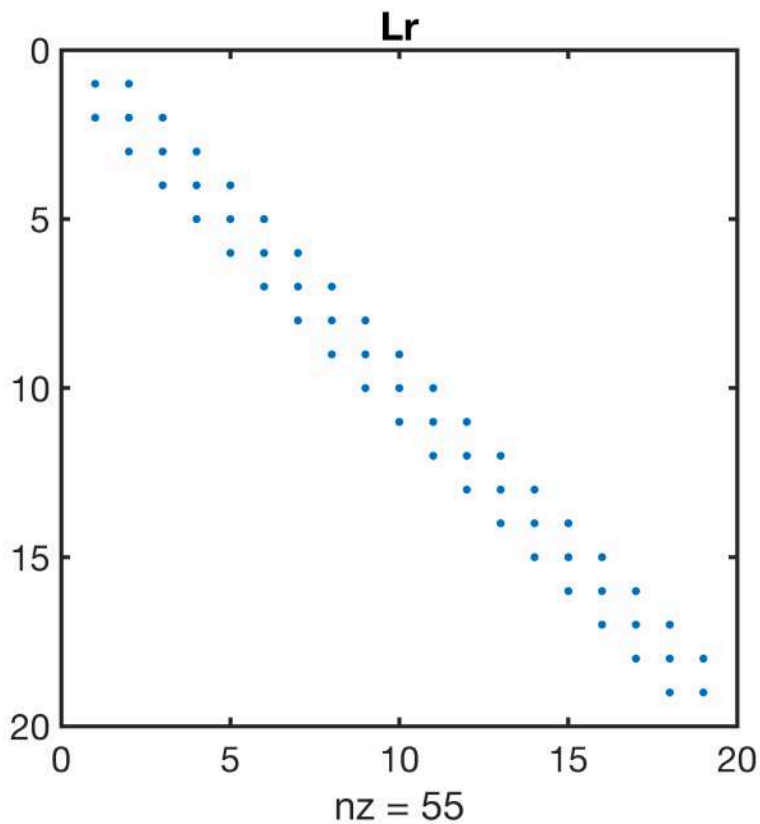
where

1. $\mathbf{L} * \mathbf{r} = \mathbf{N}' * \mathbf{L} * \mathbf{N}$
2. $\mathbf{u} * \mathbf{r} = \mathbf{N}' * \mathbf{u}$
3. $\mathbf{f} * \mathbf{s} * \mathbf{r} = \mathbf{N}' * \mathbf{f} * \mathbf{s}$

```
Lr = N'*L*N;  
size(Lr)
```

```
ans = 1x2  
      19      19
```

```
spy(Lr), title 'Lr'
```



The reduced system matrix $\mathbf{L} * \mathbf{r}$ is not singular anymore, because the constraints have been incorporated. This can be checked by estimating the condition number

```
condest(Lr)
```

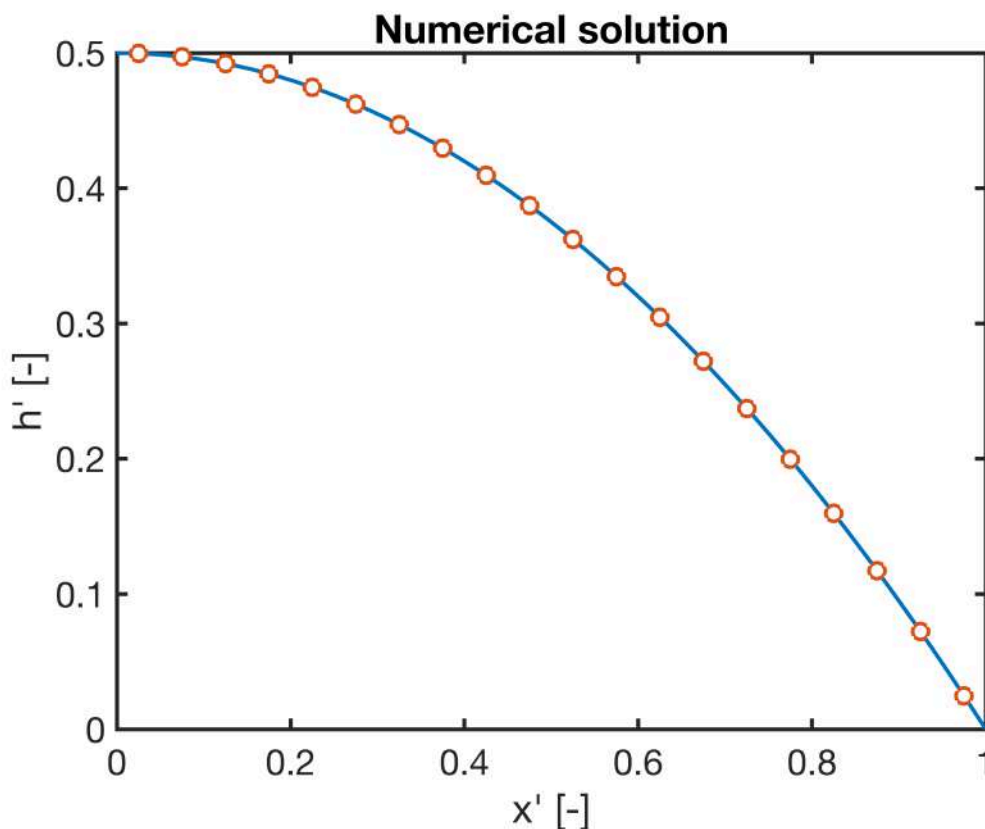
ans = 760

Solving problem with homogeneous boundary conditions

Solving a steady problem with homogeneous boundary conditions therefore requires 3 steps:

1. Compute \mathbf{N} and obtain \mathbf{Lr} and \mathbf{fsr} .
2. Solve reduced problem: $\mathbf{ur} = \mathbf{Lr} \backslash \mathbf{fsr}$.
3. Obtain full solution: $\mathbf{u} = \mathbf{N} * \mathbf{ur}$.

```
fs = ones(Grid.Nx,1);  
fsr = N'*fs;  
ur = Lr\fsr;  
u = N*ur;  
u = u+hprime(Grid.xc(Grid.Nx)); % shift to account for BC  
  
clf  
plot(xprime,hprime(xprime)), hold on  
plot(Grid.xc,u,'o','markerfacecolor','w','markersize',8)  
  
xlabel('x' ' [-]')  
ylabel('h' ' [-]')  
title('Numerical solution')
```



```

hprime = function_handle with value:
    @(xprime).5*(1-xprime.^2)
xprime = 1x100
    0    0.0101    0.0202    0.0303    0.0404    0.0505    0.0606    0.0707 ...

```

Note, the boundary condition is set at the center of the first cell, which makes the solution look bad (first order error).

If we shift the solution upward by the appropriate amount the fit to the analytic solution is quite good.

Heterogeneous constraints

We are interested in solving for the geotherm which requires heterogeneous, i.e., non-zero, BC's. In this case

$$\mathbf{B}^* \mathbf{u} = \mathbf{g},$$

where $\mathbf{g} = [T_0]$ is a vector containing the two boundary conditions.

To obtain the solution of a problem with heterogeneous boundary conditions, we split the solution into a homogeneous and a particular solution as follows

$$\mathbf{u} = \mathbf{u}_0 + \mathbf{u}_p,$$

where the homogeneous solution solves $\mathbf{B}^* \mathbf{u}_0 = 0$ as before and the particular solution solves $\mathbf{B}^* \mathbf{u}_p = \mathbf{g}$. The solution then proceeds in three steps

1. Find a particular solution that satisfies $\mathbf{B}^* \mathbf{u}_p = \mathbf{g}$.
2. Find the associated homogeneous solution, \mathbf{u}_0 .
3. Find total solution $\mathbf{u} = \mathbf{u}_0 + \mathbf{u}_p$.

Find a particular solution

Note there are many possible particular solutions, here we just find the simplest one. Also note that \mathbf{u}_p does not need to satisfy $\mathbf{L}^* \mathbf{u}_p = \mathbf{f}_s$ it only needs to satisfy the boundary conditions $\mathbf{B}^* \mathbf{u}_p = \mathbf{g}$. Since the system is not square and $N_x > N_c$ and \mathbf{B} has only N_c entries we can again project into a reduced space of size N_c .

It is natural to use \mathbf{B} as projection matrix, so that $\mathbf{u}_{pr} = \mathbf{B}^* \mathbf{u}_p$ and $\mathbf{u}_p = \mathbf{B}'^* \mathbf{u}_{pr}$. We derive the reduced system as follows

$$\mathbf{B}^* \mathbf{u}_p = \mathbf{g}$$

$$\mathbf{B}^* (\mathbf{B}'^* \mathbf{u}_{pr}) = \mathbf{g}$$

$$(\mathbf{B}^* \mathbf{B}')^* \mathbf{u}_{pr} = \mathbf{g}$$

$$\mathbf{B}_r^* \mathbf{u}_{pr} = \mathbf{g}$$

where the reduced constraint matrix is $\mathbf{B}_r = \mathbf{B}^* \mathbf{B}'$ is N_c by N_c . For the simple constraints we use here \mathbf{B}_r is simply the N_c by N_c identity matrix, so that $\mathbf{u}_{pr} = \mathbf{g}$. However, our definition is also valid for more general constraints so we'll stick with that. Once \mathbf{u}_{pr} is known the full particular solution can be recovered, $\mathbf{u}_p = \mathbf{B}'^* \mathbf{u}_{pr}$.

Find associated homogeneous solution

Once up is known we find the associated homogeneous solution, h_0 , as follows

$$\mathbf{L}^* \mathbf{u} = \mathbf{f}_s$$

$$\mathbf{L}^* (\mathbf{u}_0 + \mathbf{u}_p) = \mathbf{f}_s$$

$$\mathbf{L}^* \mathbf{u}_0 = \mathbf{f}_s - \mathbf{L}^* \mathbf{u}_p$$

$$\mathbf{L}^* \mathbf{u}_0 = \mathbf{f}_s + \mathbf{f}_d$$

where $\mathbf{f}_d = -\mathbf{L}^* \mathbf{u}_p$ is a new source term due to heterogeneous Dirichlet BC's. But the problem can be solved with the nullspace projection for homogeneous problems as above. Combining the r.h.s. $\mathbf{f} = \mathbf{f}_s + \mathbf{f}_d$ we solve as follows

Auxillary functions

This implementation of `spnull()` is taken from [Bruno Luong](#), thanks man!

```
function Z = spnull(S, varargin)
% Z = SPNULL(S)
% returns a sparse orthonormal basis for the null space of S, that is,
% S*Z has negligible elements, and Z'*Z = I
%
% If S is sparse, Z is obtained from the QR decomposition.
% Otherwise, Z is obtained from the SVD decomposition
%
% Bruno Luong <brunoluong@yahoo.com>
% History
% 10-May-2010: original version
%
% See also SPORTH, NULL, QR, SVD, ORTH, RANK

if issparse(S)
    [m n] = size(S);
    try
        [Q R E] = qr(S. '); %#ok %full QR
        if m > 1
            s = diag(R);
        elseif m == 1
            s = R(1);
        else
            s = 0;
        end
        s = abs(s);
        tol = norm(S, 'fro') * eps(class(S));
        r = sum(s > tol);
        Z = Q(:, r+1:n);
    catch %#ok
```

```

        % sparse QR is not available on old Matlab versions
        err = lasterror(); %#ok
        if strcmp(err.identifier, 'MATLAB:maxlhs')
            Z = null(full(S), varargin{:});
        else
            rethrow(err);
        end
    end
else % Full matrix
    Z = null(S, varargin{:});
end

end

```

set_defaults()

```

function [] = set_defaults()
    set(0, ...
        'defaultaxesfontsize', 18, ...
        'defaultaxeslinewidth', 2.0, ...
        'defaultlinelength', 2.0, ...
        'defaultpatchlinewidth', 2.0, ...
        'DefaultLineMarkerSize', 12.0);
end

```