UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Numerical Heat Transfer
for Applications

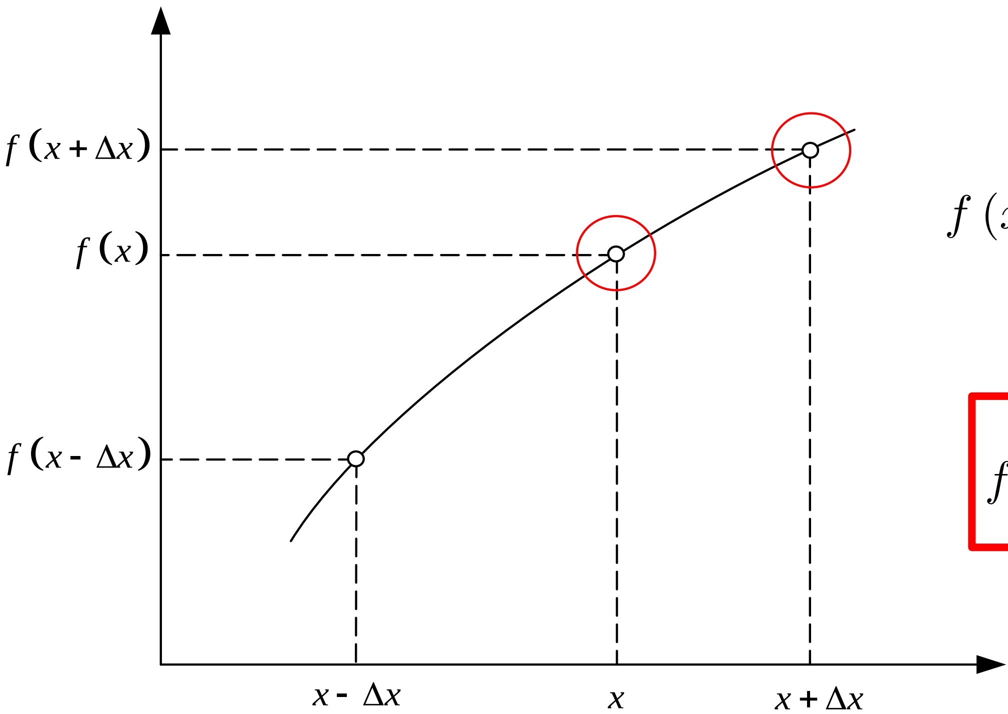
Finite Difference Method for Steady State Heat Conduction

Dr Valerio D'Alessandro

Finite difference method concept

A formal basis for developing finite difference approximation of derivatives is through the use of Taylor series expansion. Consider Taylor series expansion of a function f(x) about a given point in the forward (*i.e.*, positive x):

$$f\left(x+\Delta x\right)=f\left(x\right)+f'\left(x\right)\Delta x+f''\left(x\right)\frac{\Delta x^2}{2!}+f''\left(x\right)\frac{\Delta x^3}{3!}+\ldots$$



$$f\left(x+\Delta x\right)=f\left(x\right)+f'\left(x\right)\Delta x+O\left(\Delta x^2\right)$$

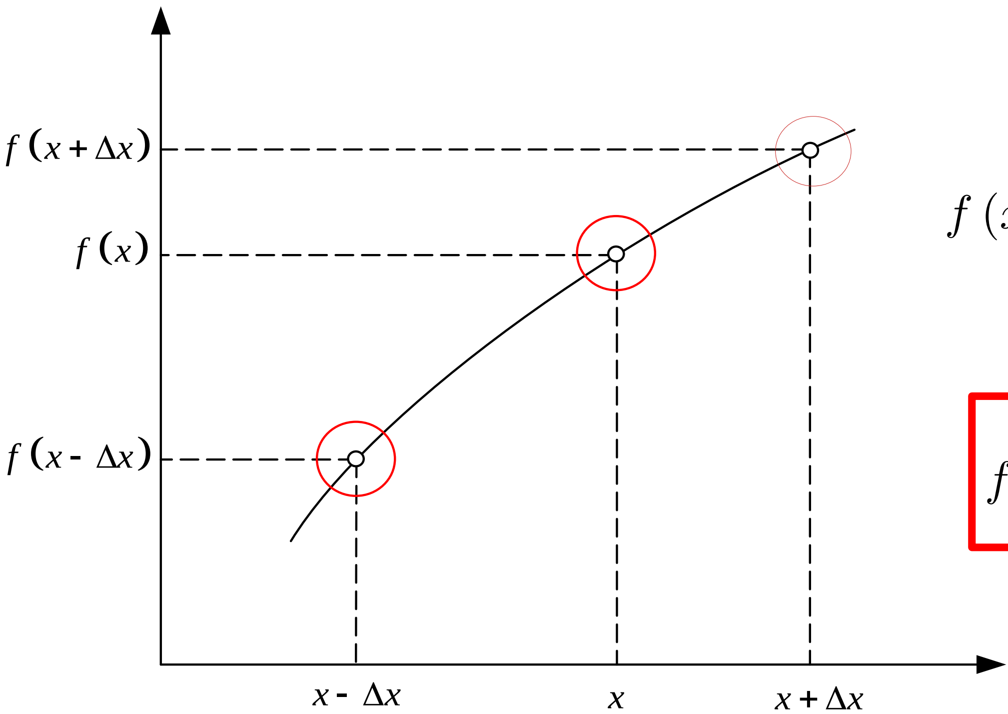
$$\boxed{f'\left(x\right)=\frac{f\left(x+\Delta x\right)-f\left(x\right)}{\Delta x}+O\left(\Delta x\right)}$$

<u>Forward finite difference approx.</u>

Finite difference method concept

Similarly, it is possible to consider Taylor series expansion of a function f(x) about a given point in the backward (*i.e.*, negative x):

$$f\left(x-\Delta x\right)=f\left(x\right)-f'\left(x\right)\Delta x+f''\left(x\right)\frac{\Delta x^2}{2!}-f''\left(x\right)\frac{\Delta x^3}{3!}+\ldots$$



$$f\left(x-\Delta x\right)=f\left(x\right)-f'\left(x\right)\Delta x+O\left(\Delta x^2\right)$$

$$\boxed{f'\left(x\right)=\frac{f\left(x\right)-f\left(x-\Delta x\right)}{\Delta x}+O\left(\Delta x\right)}$$

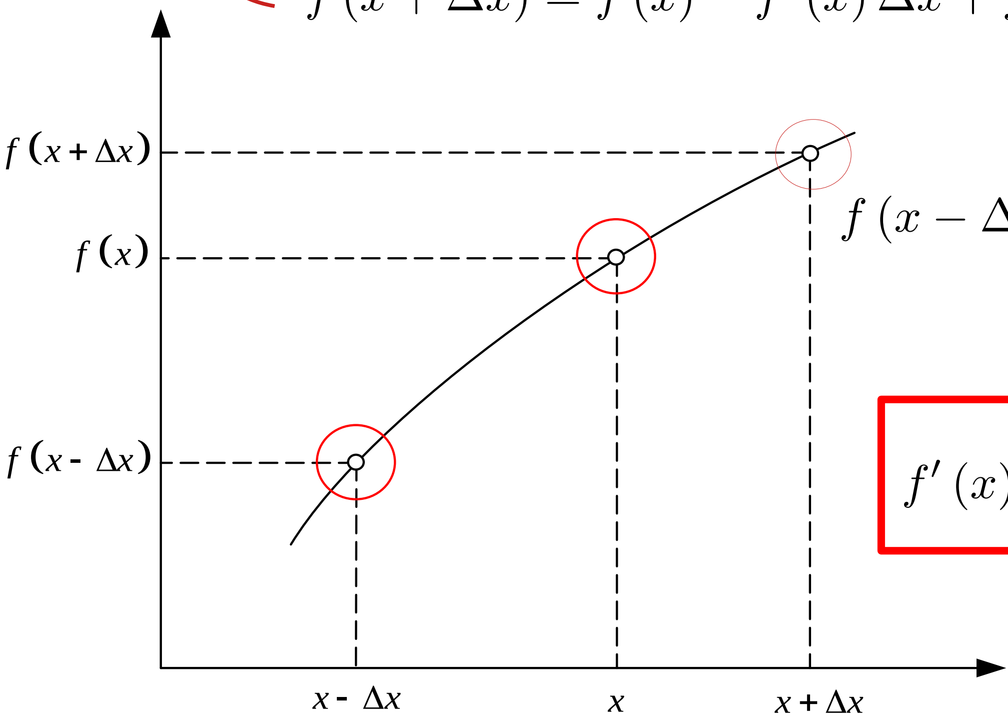
Backward finite difference approx.

Finite difference method concept

Considering both forward and backward differentiation approach is possible to obtain a third kind of differentation: the **central approach**.

$$f\left(x+\Delta x\right)=f\left(x\right)+f'\left(x\right)\Delta x+f''\left(x\right)\frac{\Delta x^2}{2!}+O\left(\Delta x^3\right)$$

$$f\left(x+\Delta x\right)=f\left(x\right)-f'\left(x\right)\Delta x+f''\left(x\right)\frac{\Delta x^2}{2!}+O\left(\Delta x^3\right)$$



$$f\left(x-\Delta x\right)-f\left(x-\Delta x\right)=2f'\left(x\right)\Delta x+O\left(\Delta x^3\right)$$

$$\boxed{f'\left(x\right)=\frac{f\left(x+\Delta x\right)-f\left(x-\Delta x\right)}{2\Delta x}+O\left(\Delta x^2\right)}$$

Central finite difference approx.

Finite difference method concept

Considering both forward and backward differentiation approach is possible to obtain also a specific approximation for second derivative:

$$f\left(x+\Delta x\right)=f\left(x\right)+f'\left(x\right)\Delta x+f''\left(x\right)\frac{\Delta x^2}{2!}+f'''\left(x\right)\frac{\Delta x^3}{3!}+O\left(\Delta x^4\right)$$

$$+$$

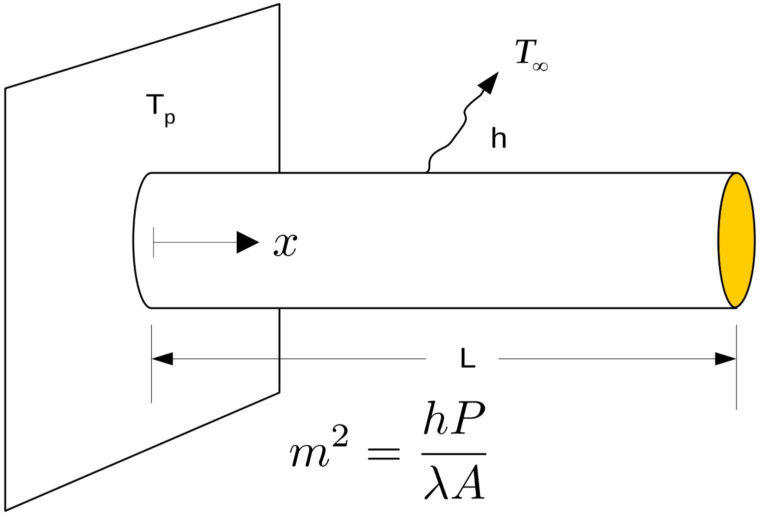
$$f\left(x-\Delta x\right)=f\left(x\right)-f'\left(x\right)\Delta x+f''\left(x\right)\frac{\Delta x^2}{2!}-f'''\left(x\right)\frac{\Delta x^3}{3!}+O\left(\Delta x^4\right)$$

$$f\left(x+\Delta x\right)+f\left(x-\Delta x\right)=2f\left(x\right)+f''\left(x\right)\Delta x^2+O\left(\Delta x^4\right)$$

$$\Downarrow$$

$$\boxed{f''\left(x\right)=\frac{f\left(x+\Delta x\right)-2f\left(x\right)+f\left(x-\Delta x\right)}{\Delta x^2}+O\left(\Delta x^2\right)}$$

Finite difference solution of fin equation



$$m^2 = \frac{hP}{\lambda A}$$

$$\frac{d^2T}{dx^2} - m^2\left(T - T_\infty\right) = 0$$

$$T\left(x = 0\right) = T_b$$

$$\left.\frac{dT}{dx}\right|_{x=L} = 0$$

$$x^* = \frac{x}{L}$$

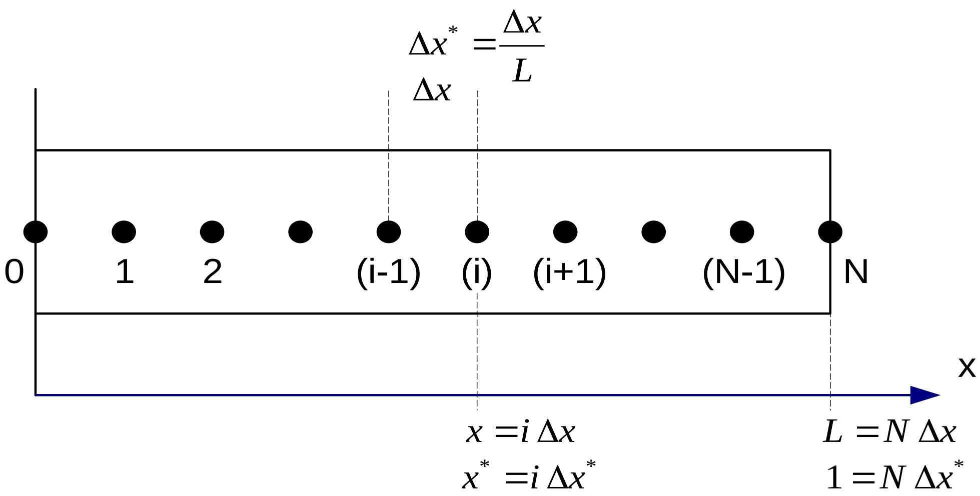
$$T^* = \frac{T - T_\infty}{T_b - T_\infty}$$

$$\frac{d^2T^*}{dx^{*2}} - \left(mL\right)^2 T^* = 0$$

$$T^*\left(x^* = 0\right) = 1$$

$$\left.\frac{dT^*}{dx^*}\right|_{x^*=1} = 0$$

Finite difference solution of fin equation



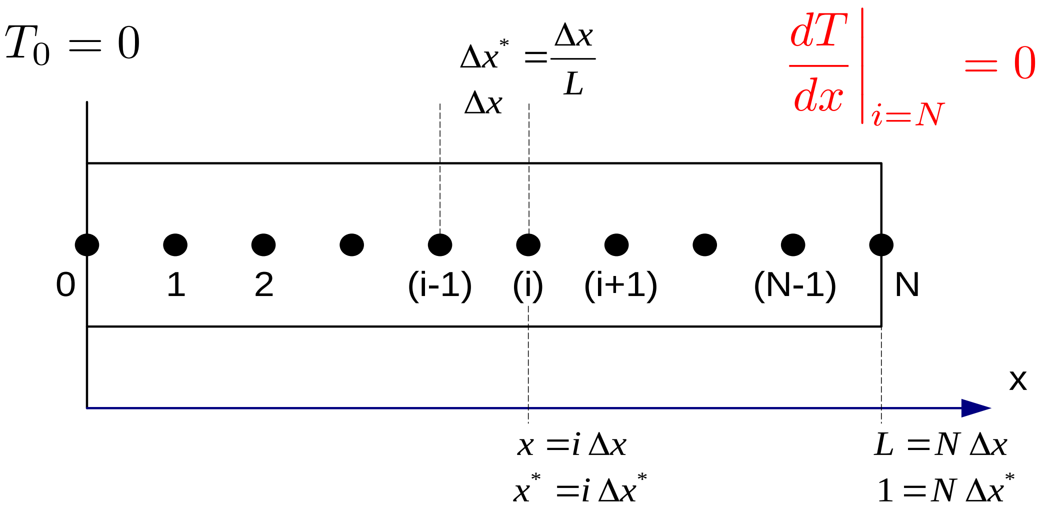
$$\boxed{\begin{aligned} &\frac{d^2T^*}{dx^{*2}} - (mL)^2\,T^* = 0 \\ &T^*\left(x^* = 0\right) = 1 \\ &\left.\frac{dT^*}{dx^*}\right|_{x^*=1} = 0 \end{aligned}}$$

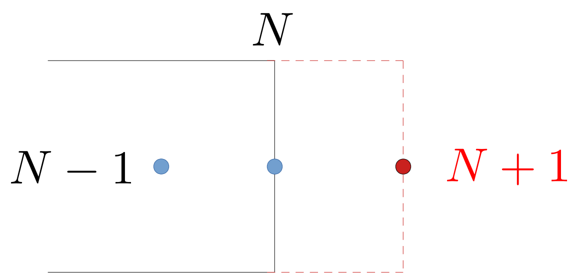
$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} - (mL)^2\,T_i = 0 \qquad 2 \le i \le N-1$$

$$\Downarrow$$

$$-T_{i-1} + \left[2 + (mL\;\Delta x)^2\right]T_i - T_{i+1} = 0$$

Finite difference solution of fin equation





$$\left.\frac{dT}{dx}\right|_{i=N} = 0 \Rightarrow \frac{T_{N+1} - T_{N-1}}{2\Delta x} = 0$$

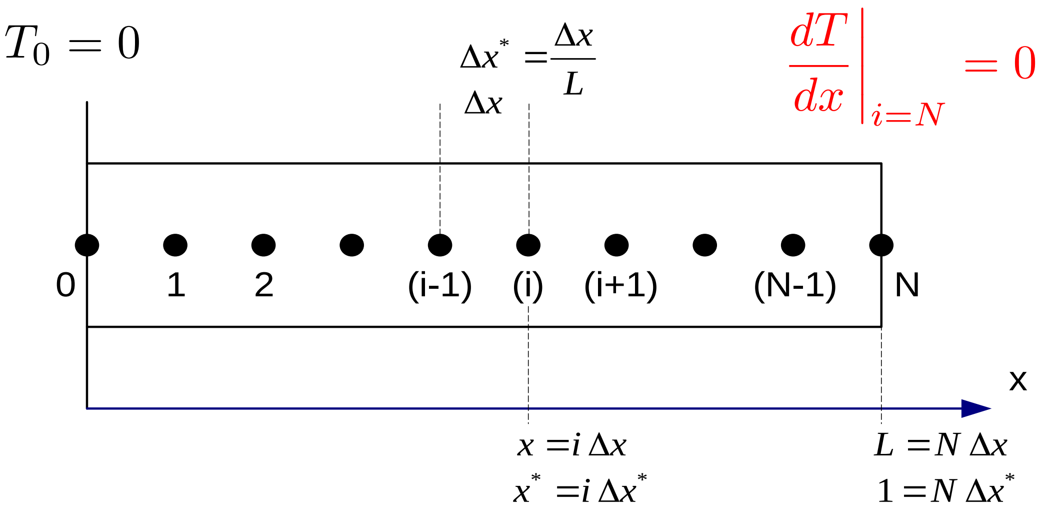
$$\boxed{\mathbf{T_{N+1} = T_{N-1}}}$$

The B.C. at fin head requires a particular treatment. A possible strategy is consider a so called ghost-node which allows to use to discretized the boundary condition itself.

$$-T_{N-1} + \left[2 + (mL\,\Delta x)^2\right]T_N - T_{N+1} = 0 \qquad i = N$$

$$\boxed{-2T_{N-1} + \left[2 + (mL\,\Delta x)^2\right]T_N = 0}$$

Finite difference solution of fin equation



$$\frac{d^2T^*}{dx^{*2}} - (mL)^2\,T^* = 0$$

$$T^*\,(x^* = 0) = 1$$

$$\left.\frac{dT^*}{dx^*}\right|_{x^*=1} = 0$$

$$\left[2 + (mL\,\Delta x)^2\right]T_1 - T_2 = 1 \qquad i = 1$$

$$-T_{i-1} + \left[2 + (mL\,\Delta x)^2\right]T_i - T_{i+1} = 0 \qquad 2 \le i \le N-1$$

$$-2T_{N-1} + \left[2 + (mL\,\Delta x)^2\right]T_N = 0 \qquad i = N$$

$$c = \left[2 + (mL\,\Delta x)^2\right]$$

Finite difference solution of fin equation

$$\left[2+(mL\ \Delta x)^2\right]T_1 - T_2 = 1 \qquad i=1$$

$$-T_{i-1} + \left[2+(mL\ \Delta x)^2\right]T_i - T_{i+1} = 0 \qquad 2 \le i \le N-1$$

$$-2T_{N-1} + \left[2+(mL\ \Delta x)^2\right]T_N = 0 \qquad i=N$$

$$c = \left[2+(mL\ \Delta x)^2\right]$$

$$N = 4$$

$$cT_1 - T_2 = 1$$

$$-T_1 + cT_2 - T_3 = 0$$

$$-T_2 + cT_3 - T_4 = 0$$

$$-2T_3 + cT_4 = 0$$

$$\begin{bmatrix} c & -1 & 0 & 0 \\ -1 & c & -1 & 0 \\ 0 & -1 & c & -1 \\ 0 & 0 & -2 & c \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Tri-diagonal linear system

Ex1_FD - Finite difference solution of fin equation

Matlab code

```
clc
clear all

%Fin dimensions
%-----------------------------------
%
 L    = 0.1112; % 111.2 mm
 t    = 0.0012; %  1.2 mm
 b    = 0.0206; % 20.6 mm
%
 p    = 2*(t+b);
 Af   = t*b;
%
 h    = 150;    % [W/m2K]
 lam  = 202.6;  % [W/mK ]

 N= 30;  % nodes' number
%-----------------------------------
%
 m    = ((h*p)/(lam*Af))^0.5;
 mL   = m*L;
%
 dx    = L/N;
 dx    = dx/L;  % dimensionless dx
 c     = 2 + (m*L*dx)^2;
%
%-----------------------------------
```

$$c = \left[2 + (mL\,\Delta x)^2\right]$$

$$cT_1 - T_2 = 1 \qquad i = 1$$

$$-T_{i-1} + cT_i - T_{i+1} = 0 \qquad 2 \le i \le N-1$$

$$-2T_{N-1} + cT_N = 0 \qquad i = N$$

$$A = \begin{pmatrix} C & -1 & 0 & 0 & 0 \\ -1 & C & -1 & 0 & 0 \\ 0 & -1 & C & -1 & 0 \\ 0 & 0 & -1 & C & -1 \\ 0 & 0 & 0 & -2 & C \end{pmatrix}$$

Ex1_FD - Finite difference solution of fin equation

Matlab code

$$c = \left[2 + (mL\,\Delta x)^2\right]$$

```
 A      = zeros(N,N);

 A(1,1)  =  c;
 A(1,2)  = -1;

 for i=2:(N-1)
     A(i,i-1)  = -1;
     A(i,i  )  =  c;
     A(i,i+1)  = -1;
 end

A(N,N-1) = -2;
A(N,N  ) =  c;
%
%--------------------------------
%
  b=zeros(N,1);
  b(1) = 1;
%
% Linear system solution
%
  x =A\b;

%--------------------------------
%Boundary value (x=0)
 x =[1;x];
%
```

$$cT_1 - T_2 = 1 \qquad i = 1$$

$$-T_{i-1} + cT_i - T_{i+1} = 0 \qquad 2 \le i \le N-1$$

$$-2T_{N-1} + cT_N = 0 \qquad i = N$$

$$A = \begin{pmatrix} C & -1 & 0 & 0 & 0 \\ -1 & C & -1 & 0 & 0 \\ 0 & -1 & C & -1 & 0 \\ 0 & 0 & -1 & C & -1 \\ 0 & 0 & 0 & -2 & C \end{pmatrix}$$

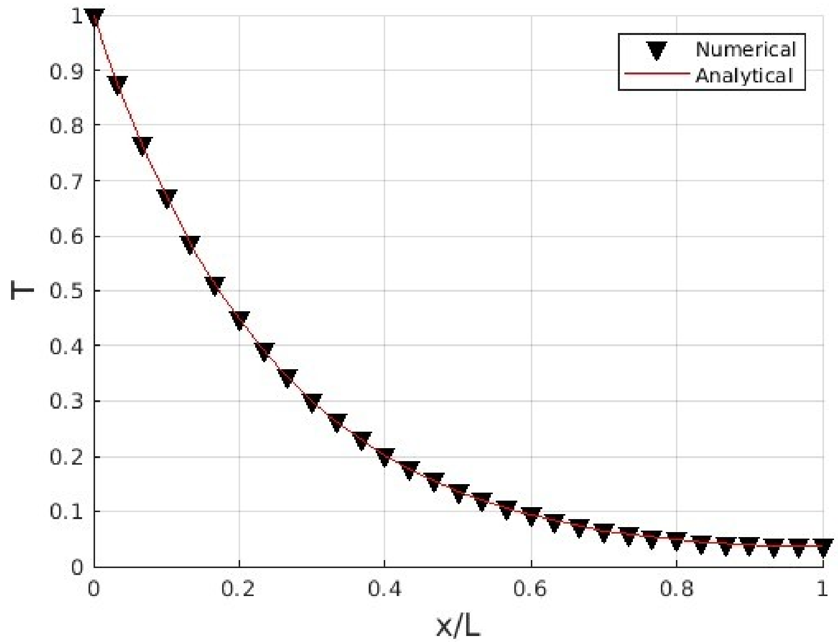
$$b = [1, 0, 0, \cdots, 0]^T$$

Ex1_FD - Finite difference solution of fin equation

Matlab code

```
% Analytic solution
%
%
  dx1   = L/50;
  dx1   = dx1/L;
  lf = dx1*[0:50]';
  xan  = (cosh(mL*(1-lf)))/(cosh(mL));
%

%  Numerical vs Analytical
%
 lfn   = dx*[0:N]'; %dimensionless x
 grid on
%
 hold on
 plot(lfn, x,
'kv','markerfacecolor','k',
'markersize',8);
 plot(lf,  xan, 'r-');
 hold off
 xlim ([0 lfn(end)]);

 xlabel('x/L' ,    'fontsize', 14);
 ylabel('T'     ,    'fontsize', 14);
 legend ('Numerical', 'Analytical');
%
```

$$\frac{\theta(x)}{\theta_b} = \frac{1-\cosh\left(mL\left(1-\frac{z}{L}\right)\right)}{\cosh(mL)}$$



Finite difference solution of fin equation – Jacobi method

$$cT_1 - T_2 = 1$$

$$-T_1 + cT_2 - T_3 = 0$$

$$-T_2 + cT_3 - T_4 = 0$$

$$-2T_3 + cT_4 = 0$$

$$T_1^{(p+1)} = \frac{1 + T_2^{(p)}}{c}$$

$$T_2^{(p+1)} = \frac{T_1^{(p)} + T_3^{(p)}}{c}$$

$$T_3^{(p+1)} = \frac{T_2^{(p)} + T_4^{(p)}}{c}$$

$$T_4^{(p+1)} = \frac{2}{c} T_3^{(p)}$$

	T_1	T_2	T_3	T_4
0	0	0	0	0
1	$1/c$	0	0	0
2	$1/c$	$1/c^2$	0	0
3	$(c+1)/c^3$	$1/c^2$	$1/c^3$	0

The Jacobi method exhibits slow converge properties. It is very clear that in our problem are required 4 iterations to change, for the first time, T value from its initial guess in each node.

Finite difference solution of fin equation – Gauss-Siedel method

$$cT_1 - T_2 = 1$$

$$-T_1 + cT_2 - T_3 = 0$$

$$-T_2 + cT_3 - T_4 = 0$$

$$-2T_3 + cT_4 = 0$$

$$T_1^{(p+1)} = \frac{1 + T_2^{(p)}}{c}$$

$$T_2^{(p+1)} = \frac{T_1^{(p+1)} + T_3^{(p)}}{c}$$

$$T_3^{(p+1)} = \frac{T_2^{(p+1)} + T_4^{(p)}}{c}$$

$$T_4^{(p+1)} = \frac{2}{c} T_3^{(p+1)}$$

	T_1	T_2	T_3	T_4
0	0	0	0	0
1	$1/c$	$1/c^2$	$1/c^3$	$1/c^4$
2	$\cdots$	$\cdots$	$\cdots$	$\cdots$
3	$\cdots$	$\cdots$	$\cdots$	$\cdots$

The Gauss-Siedel method is certainly faster than Jacobi one. After only 1 iteration, T values are different from their initial guess in each node.

Ex2_FD - Finite difference solution of fin equation – Jacobi vs. Gauss-Siedel

Matlab code

```
function x2=jacobi(A,b,N,n_max)
%
   x1 = zeros(N,1);  % guess vector
   x2 = zeros(N,1);  % updated vector

for k=1:n_max
%
      app = 1/A(1,1);
      x2(1) = app*(b(1) - A(1,2:N)*x1(2:N));
%
   for i=2:N
      app = 1/A(i,i);
      ri1 = A(i,1:(i-1))*x1(1:(i-1));
      ri2 = A(i,(i+1):N)*x1((i+1):N);
      x2(i) = app*(b(i) -ri1 -ri2);
   end
%
r=b- A*x2;
rMag = sqrt(sum(r.*r))/sqrt(sum(b.*b));
%
   x1=x2;
%
   if ( rMag <= 1e-12)
         break
   end

   fprintf('%s\n',['It. ', num2str(k), ' residual
', num2str(rMag)]);
 end
```

$$\mathbf{A}\mathbf{x} = \mathbf{b} \Rightarrow a_{ij}x_j = b_i$$

$$a_{ii}x_i + \sum_{i \neq j} a_{ij}x_j = b_i$$

$$\Downarrow$$

$$\boxed{x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{i \neq j} a_{ij}x_j^{(k)}\right)}$$

Ex2_FD - Finite difference solution of fin equation – Jacobi vs. Gauss-Siedel

Matlab code

```
function x2=jacobi(A,b,N,n_max)
%
   x1 = zeros(N,1);  % guess vector
   x2 = zeros(N,1);  % updated vector

for k=1:n_max
%
      app = 1/A(1,1);
      x2(1) = app*(b(1) - A(1,2:N)*x1(2:N));
%
   for i=2:N
      app = 1/A(i,i);
      ri1 = A(i,1:(i-1))*x1(1:(i-1));
      ri2 = A(i,(i+1):N)*x1((i+1):N);
      x2(i) = app*(b(i) -ri1 -ri2);
   end
%
r=b- A*x2;
rMag = sqrt(sum(r.*r))/sqrt(sum(b.*b));
%
   x1=x2;
%
   if ( rMag <= 1e-12)
         break
   end

   fprintf('%s\n',['It. ', num2str(k), ' residual
', num2str(rMag)]);
 end
```

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{i\neq j} a_{ij}x_j^{(k)}\right)$$

$$r = \frac{\| \mathbf{b} - \mathbf{A}\mathbf{x} \|}{\| \mathbf{b} \|}$$

$$\| \mathbf{x} \| = \sqrt{\sum_{k=1}^{N} x_k^2}$$

Ex2_FD - Finite difference solution of fin equation – Jacobi vs. Gauss-Siedel

Matlab code

```
function x2=gs(A,b,N,n_max)
%
   x1 = zeros(N,1);  % guess vector
   x2 = zeros(N,1);  % updated vector

for k=1:n_max
%
      app = 1/A(1,1);
      x2(1) = app*(b(1) - A(1,2:N)*x1(2:N));
%
   for i=2:N
      app = 1/A(i,i);
      ri1 = A(i,1:(i-1))*x2(1:(i-1));
      ri2 = A(i,(i+1):N)*x1((i+1):N);
      x2(i) = app*(b(i) -ri1 -ri2);
   end
%
r=b- A*x2;
rMag = sqrt(sum(r.*r))/sqrt(sum(b.*b));
%
   x1=x2;
%
   if ( rMag <= 1e-12)
        break
   end

   fprintf('%s\n',['It. ', num2str(k), ' residual
', num2str(rMag)]);
 end
```

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - r_1 - r_2\right)$$

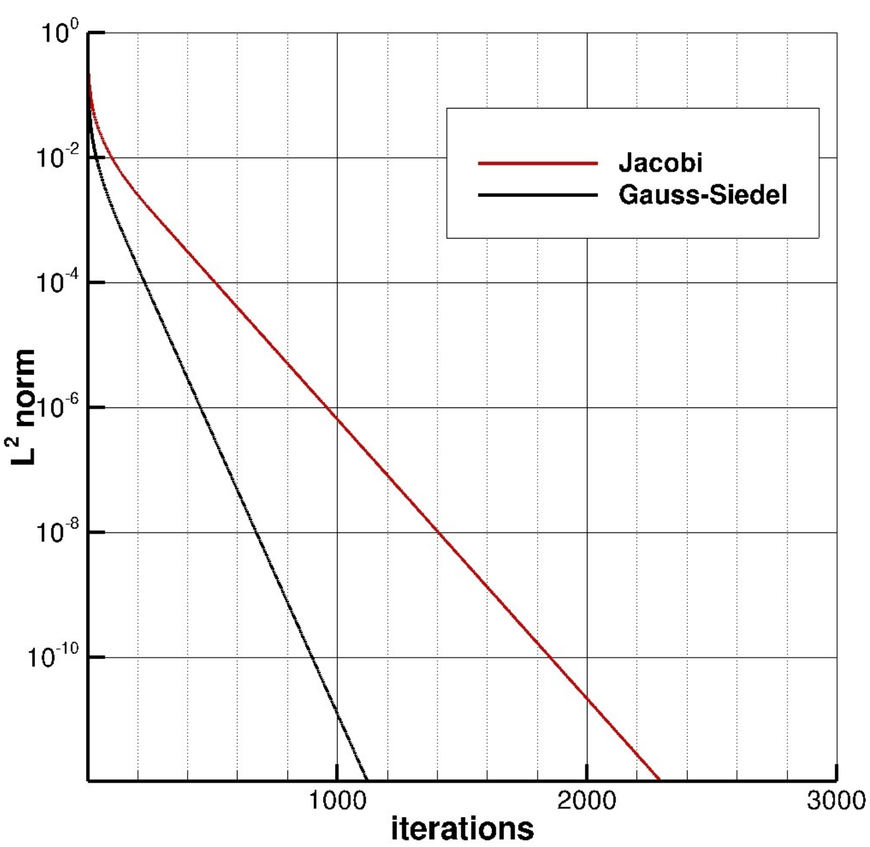
$$r_1 = \sum_{i<j} a_{ij} x_j^{(k+1)}$$

$$r_1 = \sum_{i>j} a_{ij} x_j^{(k)}$$

Ex2_FD - Finite difference solution of fin equation – Jacobi vs. Gauss-Siedel

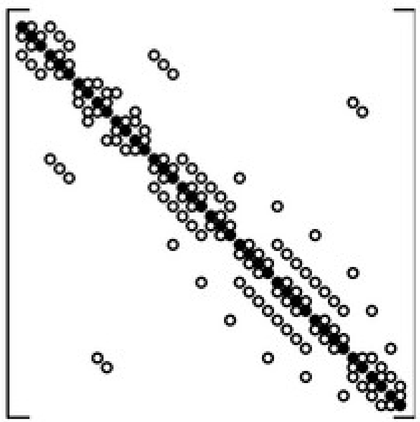
Matlab code

```
 A      = zeros(N,N);

 A(1,1)  =  c;
 A(1,2)  = -1;

 for i=2:(N-1)
     A(i,i-1)  = -1;
     A(i,i  )  =  c;
     A(i,i+1)  = -1;
 end

A(N,N-1) = -2;
A(N,N  ) =  c;
%
%---------------------------------
%
  b=zeros(N,1);
  b(1) = 1;
%
% Linear system solution
%
  %x=A\b;
  %x= gs(A,b,N,30000);
  x = jacobi(A,b,N,30000);

%---------------------------------
%Boundary value (x=0)
 x =[1;x];
```



Matrix sparsity

In the previous exercise we solve a linear system with a matrix having rank equal to 30. However, the matrix has only **88 non-null entries of its overall 900**. The matrix **A** is sparse, *i.e.* the majority of its coefficients are zero.



The sparsity is due to each node interacts only with adjacent ones. For example, also a more complex stencil with 5 nodes, produces up to 5 coefficients per matrix row, with one diagonal coefficient corresponding to a particular cell and 4 off-diagonal coefficients for the neighbour cells.

Typically current accademic and industrial applications require huge meshes to obtain reliable solutions. Nowadays, $N=10^6$ is the common scale. It is easy to understand that for similar problems the non-zero terms a really lower than zeros ones. For efficiency reasons, zero coefficients are not stored in the computer's memory. Instead the storage is based on an array of non-zero coefficients and addressing arrays of the corresponding row and column indices for each coefficient.

Ex3_FD - Finite difference solution of fin equation – Thomas algorithm

The Thomas algorithm is a variant of Gauss elimination method which is specifically developed for tri-diagonal matrices. It is also known also ad Tri-Diagonal Matrix Algorithm (TDMA). It reduces the computational costs from O(N^3) to O(N).

$$\begin{aligned} b_1x_1 \quad + c_1x_2 &= d_1 \\ a_ix_{i-1} + b_ix_i + c_ix_{i+1} &= d_i\,, \\ a_nx_{n-1} + b_nx_n \quad &= d_n\,. \end{aligned}$$

(coeff. x_{i-1} of eq i-1)(eq i) - (coeff. x_i of eq i) (eq i -1)

$$\begin{aligned} b_1x_1 + c_1x_2 &= d_1 \\ a_2x_1 + b_2x_2 + c_2x_3 &= d_2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} a_2b_1x_1 + a_2c_1x_2 &= a_2d_1 \\ b_1a_2x_1 + b_1b_2x_2 + b_1c_2x_3 &= b_1d_2 \end{aligned}$$

$$(b_1b_2 - a_2c_1)\,x_2 + b_1c_2x_3 = b_1d_2 - a_2d_1$$

Using this strategy (**forward substitution**) the coeffs become more complex. However, we cancel all the contributions deriving from sub-diagonals terms.

Ex3_FD - Finite difference solution of fin equation – Thomas algorithm

$$a_{N-1}x_{N-2}+b_{N-1}x_{N-1}+c_{N-1}x_N=d_{N-1} \qquad \times\ a_N$$

$$a_N x_{N-1}+b_N x_N=d_N \qquad \times\ b_{N-1}$$

$$\boxed{(b_{N-1}b_N-c_{N-1}a_N)\,x_N=b_{N-1}d_N-a_N b_{N-1}}$$

The forward substution procedure allows to obtain the **last equation** with **only one unknown**. This condition is highly appealing since in this way with a **backward substitution** is possible the finalize the solution of the overall linear system.

Basically, forward substitution is used to compute the coeffs of the new system without sub-diagonal terms. Backward substitution is used to compute the final solution.

$$c'_i=\begin{cases}\dfrac{c_i}{b_i}, & i=1,\\[2ex] \dfrac{c_i}{b_i-a_i c'_{i-1}}, & i=2,3,\dots,n-1\end{cases} \qquad d'_i=\begin{cases}\dfrac{d_i}{b_i}, & i=1,\\[2ex] \dfrac{d_i-a_i d'_{i-1}}{b_i-a_i c'_{i-1}}, & i=2,3,\dots,n.\end{cases}$$

forward substution

$$x_n=d'_n,$$
$$x_i=d'_i-c'_i x_{i+1}, \quad i=n-1,n-2,\dots,1.$$

backward substution

Ex3_FD - Finite difference solution of fin equation – Thomas algorithm

Matlab code

```
% Matrix - vectors;
%
al      = -ones(N-1,1);
al(end) = -2;

ad      =   ones(N,1)*c;

au      =  -ones(N-1,1);

%---------------------------------
%
  b=zeros(N,1);
  b(1) = 1;
%
%  Linear system solution
%
  %x =A\b;
  x = tridiagonal_vector(al,ad,au,b);

%
%---------------------------------
```

$$
\begin{array}{ll}
A = \begin{pmatrix} C & -1 & 0 & 0 & 0 \\ -1 & C & -1 & 0 & 0 \\ 0 & -1 & C & -1 & 0 \\ 0 & 0 & -1 & C & -1 \\ 0 & 0 & 0 & -2 & C \end{pmatrix} & \mathbf{a_u} \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{a_l} \qquad \mathbf{a_d} &
\end{array}
$$

The first three entries of the *tridiagonal_vector* are stored as showed in the above skecth.

In this specific case we rely on the so-called **opportunistic code-reuse strategy**. In other words, we use existing software to build new software.

https://github.com/tamaskis/tridiagonal-MATLAB

Diagonal dominance

The behaviour of this problem indicates a convergence condition: <u>the magnitude of the diagonal coefficient in each matrix row must be greater than or equal to the sum of the magnitudes of the other coefficients in the row:</u>

$$|a_{i,i}| \geq \sum_{i \neq j} |a_{i,j}|$$

where the $>$ condition must be satisfied for at least one row. *This condition is mathematically "sufficient" which means that convergence may occur when the condition is not met.*

$$A = \begin{pmatrix} C & -1 & 0 & 0 & 0 \\ -1 & C & -1 & 0 & 0 \\ 0 & -1 & C & -1 & 0 \\ 0 & 0 & -1 & C & -1 \\ 0 & 0 & 0 & -2 & C \end{pmatrix}$$

$$c = \left[2 + (mL\ \Delta x)^2\right]$$