IPT 5260

Computational Methods for Optoelectronics

Ray-Kuang Lee†

Institute of Photonics Technologies,
Department of Electrical Engineering and Department of Physics,
National Tsing-Hua University, Hsinchu, Taiwan

†e-mail: rklee@ee.nthu.edu.tw

Course info: http://mx.nthu.edu.tw/∼rklee

Syllabus

1. Linear Algebraic Equations (March. 6)
2. Interpolation, Curve Fitting, and Integration (Mar. 13)
3. Ordinary Differential Equations (Mar. 20, 27)
   Homework # 1: two-weeks to finish, (deadline: Apr. 10)
4. Partial Differential Equations (Apr. 10, 17)
5. Nonlinear Equations and Nonlinear PDE (Apr. 24, May 1)
   Homework # 2: two-weeks to finish (deadline: May 15),
6. Eigenvalues and Eigenvectors (May 8)
7. Finite Element Method (May 15)
8. Monte Carlo Method (May 22)
9. Optimization (May 29)
   Project: one-month to finish (deadline: June 26),
10. Case studies (June 5, 19, 26)

Prelude

1. C/C++/Fortran simulation platform
   - gcc/f77,f90,f95, PGI, NAG, Visual C/Fortran, Borland C, Compaq C ...
   - gnuplot, tecplot, sigmaplot, origin ...
   - GNU Make and Concurrent Version System (CVS)
   - Parallel programming, MPI
2. Libraries/Standard subroutine packages
   - C++ Standard Template Library, STL
   - LINPACK/LAPACK
   - IMSL
   - Matlab
3. Mathematica, Maple, MathCAD, Perl ...
4. Numerical errors

Tips for using Matlab

- use *matrix/vector* operations rather than *loop* operations
- use *build-in* routine as often as possible
- build your self sub-routines, $.m$ files
- use adaptive input argument list
- use the *breakpoint* in the debug mode
- check *workspace* for the type/value of the variables
- remark the code as much as possible

Example

```
% Compute Fourier second derivative
[x,D] = fourdif(N,2);

% Rescale [0, 2pi] to [-L,L]
x = L*(x-pi)/pi;

u0 = 1.0*sech(x-x0).*exp(i*p0*(x-x0))+1.0*sech(x+x0).*exp(-i*p0*(x+x0))
tspan  = [0:tfinal/tnum:tfinal];

% Options for ODE45
options = odeset('RelTol',tol,'AbsTol',tol);

% Solve ODEs
[t,u] = ode45(@nlsrhs, tspan, u0, options, D);
```

IEEE 64-bit Floating-Point

In floating-point representation,

$$s \times M \times B^{e-E},$$

where

1. s: sign bit, $b_{63} = \pm$
2. B: the base of the representation (usually $B = 2$, but sometimes $B = 16$),
3. e: exact integer exponent, $(b_{62}b_{61}\cdots b_{52})$,
4. E: the bias of the exponent, $2^{11-1} - 1 = 1023$

$$e - E = -1022 \backsim +1023$$

5. M: exact positive integer mantissa,

$$\begin{aligned} M &= 0.b_{51}b_{50}\cdots b_1 b_0 = [b_{51}b_{50}\cdots b_1 b_0] \times 2^{-52}, \quad \text{un-normalized} \\ &= 1.b_{51}b_{50}\cdots b_1 b_0 = 1 + [b_{51}b_{50}\cdots b_1 b_0] \times 2^{-52}, \quad \text{normalized} \end{aligned}$$

Range of 64-bit Floating-Point

- Least Significant Digit:

$$\Delta M = 2^{-52} \approx 10^{-52*3/10} = 10^{-16}$$

- Least Significant Bit:

$$\Delta M \times 2^{e-E} = 2^{-52} \times 2^{e-E}$$

- minimum positive number:

$$(0 + 2^{-52}) \times 2^{-1022} = 2^{-1074} \approx 10^{-1074*3/10} = 10^{-321}$$

- maximum positive number:

$$(2 - 2^{-52}) \times 2^{1023} \approx 10^{308}$$

$$2^{10} = 1024 \approx 10^3$$

Errors

- Round-off Error: by finite bits,
- Truncation Error: by finite number of terms (operations),
- Overflow/Underflow: by too large or too small numbers,
- Negligible Addition: by adding two numbers differing by over 52 bits, "$+$"
- Loss of significance: by a "bad subtraction", "$-$"
- Error Magnification: by multiplying/dividing a number containing a small error, "$\times$" and "/"
- Errors by algorithms.

Tips for avoiding large errors

A quadratic equation, with real coefficients a, b, and c:

$$ax^2 + bx + c = 0$$

one can write the solution in *two* ways,

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \quad \text{or} \quad x = \frac{2c}{-b \pm \sqrt{b^2 - 4ac}}$$

If either a or c (or both) are small, $b^2 \gg ac$, the two roots are

$$x_1 = \frac{q}{a}, \quad \text{and} \quad x_2 = \frac{c}{q},$$

where

$$q \equiv \frac{-1}{2}[b + \text{sgn}(b)\sqrt{b^2 - 4ac}]$$

Normalization of coefficients

In order to decrease the magnitude of round-off errors and to lower the possibility of overflow/underflow errors, when computing

$$\frac{xy}{z},$$

- x(y/z) when y and z are close in magnitude,
- (x/z)y when z and z are close in magnitude,
- (xy)/z when x and y are very different in magnitude.

Normalization of coefficients, or reducing the equation to scaleless, is important.

Normalization of coefficients

Nonlinear Schrödinger equation:

$$i\frac{\partial U}{\partial z}+D_2\frac{\partial^2 U}{\partial t^2}+\chi_3|U|^2U=0$$

set $\eta=az$ and $\tau=bt$, then NLSE becomes

$$ia\frac{\partial U}{\partial \eta}+D_2b^2\frac{\partial^2 U}{\partial \tau^2}+\chi_3|U|^2U=0$$

then by choosing

$$\begin{aligned} a &= \chi_3, \\ b &= \sqrt{\frac{\chi_3}{2D_2}}, \end{aligned}$$

NLSE is reduced to scaleless form,

$$i\frac{\partial U}{\partial \eta}+\frac{1}{2}\frac{\partial^2 U}{\partial \tau^2}+|U|^2U=0$$

1, Linear Algebraic Equations

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b},$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ & \cdots & & \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}.$$

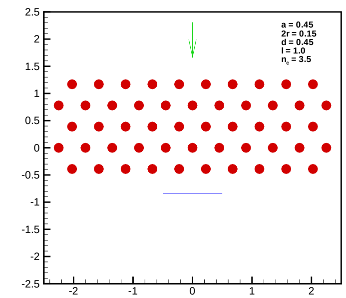
- Gauss-Jordan elimination
- LU decomposition
- Jacobi iteration
- Gauss-Seidel iteration

Systems for linear algebra equation

- Field decomposition by basis
- Finite Element Method (element basis)
- Differential matrix
- Many particle simulator
- . . .

Multiple scattering method for PBG

Helmholtz equation:

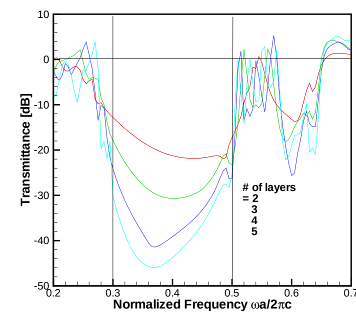


$$\bigtriangledown^2 E + \tilde{k}^2(M)E = 0$$

with

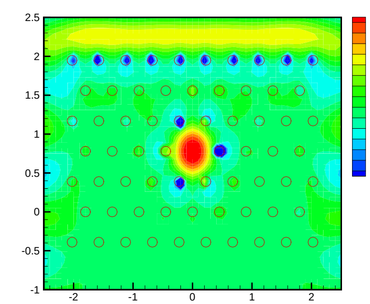
$$\tilde{k}^2(M) = k^2\tilde{\epsilon} = \begin{cases} k^2\epsilon_j & \text{if} \quad M \in C_j (j = 1, 2, \ldots, N) \\ k^2 & \text{if} \quad M \notin C_j (j = 1, 2, \ldots, N) \end{cases}$$

write down the total field in decomposition,



$$\begin{aligned} E(P) &= \sum_{m=-\infty}^{\infty} a_{l,m} J_m[kr_l(P)] e^{im\theta_l(P)} \\ &+ \sum_{m=-\infty}^{\infty} b_{l,m} H_m^{(1)}[kr_l(P)] e^{im\theta_l(P)}, \end{aligned}$$

then all we need is to solve



$$\hat{\mathbf{b}}_l - \sum_{j \neq l} \mathbf{S}_l \mathbf{T}_{l,j} \hat{\mathbf{b}}_j = \mathbf{S}_l \mathbf{Q}_l.$$

EE.NTHU 國立清華大學 電機工程學系及研究所

Finite difference approximation

$$\frac{\mathrm{d}u(x)}{\mathrm{d}x} = x,$$

Second-order FD approximation for $u'(x_j)$

$$u'(x_j) \approx \frac{u_{j+1} - u_{j-1}}{2h}$$

in the matrix-vector form (with periodic boundary)

$$\begin{pmatrix} u'_1 \\ \vdots \\ \\ u'_j \\ \vdots \\ \\ u'_N \end{pmatrix} = h^{-1} \begin{pmatrix} 0 & \frac{1}{2} & & & & & -\frac{1}{2} \\ -\frac{1}{2} & 0 & & & & & \\ & & & \ddots & & & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} & & \\ & & & \ddots & & & \\ & & & & & 0 & \frac{1}{2} \\ \frac{1}{2} & & & & & -\frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ \\ u_j \\ \vdots \\ \\ u_N \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ \\ x_j \\ \vdots \\ \\ x_N \end{pmatrix}$$

Solution for a system of linear equations

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b},$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ & \cdots & & \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}.$$

- $M = N$, nonsingular case,
- $M < N$, underdetermined case: minimum-norm solution,
- $M > N$, overdetermined case, Least-Squares-Error solution,

$$\mathbf{e} = \mathbf{A}\mathbf{x} - \mathbf{b},$$

$$\frac{1}{2}||\mathbf{e}||^2.$$

Gaussian forward elimination

Consider $M = N = 3$,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix},$$

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & = & b_2 \\ a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & = & b_3 \end{array}$$

step 1: by *pivoting at* a_{11}

$$\begin{array}{ccccccc} a_{11}^{(0)}x_1 & + & a_{12}^{(0)}x_2 & + & a_{13}^{(0)}x_3 & = & b_1^{(0)} \\ & & a_{22}^{(1)}x_2 & + & a_{23}^{(1)}x_3 & = & b_2^{(1)} \\ & & a_{32}^{(1)}x_2 & + & a_{33}^{(1)}x_3 & = & b_3^{(1)} \end{array}$$

Gaussian forward elimination

step 2: by *pivoting at* a_{22}

$$
\begin{aligned}
a_{11}^{(0)} x_1 \quad + \quad a_{12}^{(0)} x_2 \quad + \quad a_{13}^{(0)} x_3 \quad &= \quad b_1^{(0)} \\
a_{22}^{(1)} x_2 \quad + \quad a_{23}^{(1)} x_3 \quad &= \quad b_2^{(1)} \\
a_{33}^{(2)} x_3 \quad &= \quad b_3^{(2)}
\end{aligned}
$$

$$
\begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} \\ 0 & 0 & a_{33}^{(2)} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1^{(0)} \\ b_2^{(1)} \\ b_3^{(2)} \end{bmatrix},
$$

- for subtraction $\frac{1}{3} N^3$ loops,
- for multiplication $\frac{1}{2} N^2 M$ loops.

Backsubstitution

step 3: by backward substitution

$$
\begin{aligned}
x_3 &= b_3^{(2)}/a_{33}^{(2)} \\
x_2 &= (b_2^{(1)} - a_{23}^{(1)} x_3)/a_2^{(1)}2 \\
x_1 &= (b_1^{(0)} - \sum_{n=2}^{3} a_{1n}^{(0)} x_n)/a_{11}^{(0)}
\end{aligned}
$$

in general form,

$$
x_m = \left(b_m^{(m-1)} - \sum_{n=m+1}^{M} a_{mn}^{(m-1)} x_n \right) / a_{mm}^{(m-1)}, \quad \text{for} \quad m = M, M-1, \ldots, 1
$$

- for backsubstituion $\frac{1}{2}N^2$ loops (one multiplication plus one subtraction),

Pivoting

If $a_{kk} = 0$, for example

$$\begin{bmatrix} 0 & 1 & 1 \\ 2 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1^{(0)} \\ b_2^{(1)} \\ b_3^{(2)} \end{bmatrix},$$

- apply *row switching* to avoid the zero division,
- interchange rows only: *partial pivoting*,
- interchange row and columns: *full pivoting*,
- simply pick the largest (in magnitude) element as the pivot,
- more useful to reduce the round-off error,

$$\text{Max}\{|a_{mk}|, k \le m \le M\}$$

- a better choice,

$$\text{Max}\{\frac{|a_{mk}|}{\text{Max}\{|a_{mn}, k \le n \le M\}}, k \le m \le M\},$$

EE.NTHU 國立清華大學 電機工程學系及研究所

LU decomposition

$$\mathbf{L} \cdot \mathbf{U} = \mathbf{A},$$

where **L** is *lower* triangular and **U** is upper triangular,

$$\begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} \cdot \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ 0 & \beta_{22} & \beta_{23} & \beta_{24} \\ 0 & 0 & \beta_{33} & \beta_{34} \\ 0 & 0 & 0 & \beta_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

We can use a decomposition to solve the linear set

$$\mathbf{A} \cdot \mathbf{x} = (\mathbf{L} \cdot \mathbf{U}) \cdot \mathbf{x} = \mathbf{L} \cdot (\mathbf{U} \cdot \mathbf{x}) = \mathbf{b}$$

by first solving for the vector **y**

$$\mathbf{L} \cdot \mathbf{y} = \mathbf{b}$$

then solving

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{y}$$

LU decomposition: forward/backward substitution

- forward substitution:

$$\begin{aligned} y_1 &= \frac{b_1}{\alpha_{11}} \\ y_i &= \frac{1}{\alpha_{ii}}[b_i - \sum_{j=1}^{i-1} \alpha_{ij} y_j], \quad i = 2, 3, \ldots, N \end{aligned}$$

- backward substitution:

$$\begin{aligned} x_N &= \frac{y_N}{\beta_{NN}} \\ x_i &= \frac{1}{\beta_{ii}}[y_i - \sum_{j=1+1}^{N} \beta_{ij} x_j], \quad i = N-1, N-2, \ldots, 1 \end{aligned}$$

Performing the LU decomposition: Crout's algorithm

$$\begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} \cdot \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ 0 & \beta_{22} & \beta_{23} & \beta_{24} \\ 0 & 0 & \beta_{33} & \beta_{34} \\ 0 & 0 & 0 & \beta_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

with

$$\alpha_{i1}\beta_{1j} + \cdots + \alpha_{ii}\beta_{jj} = a_{ij}$$

there are N^2 equations for the $N^2 + N$ unknown α's and β's.

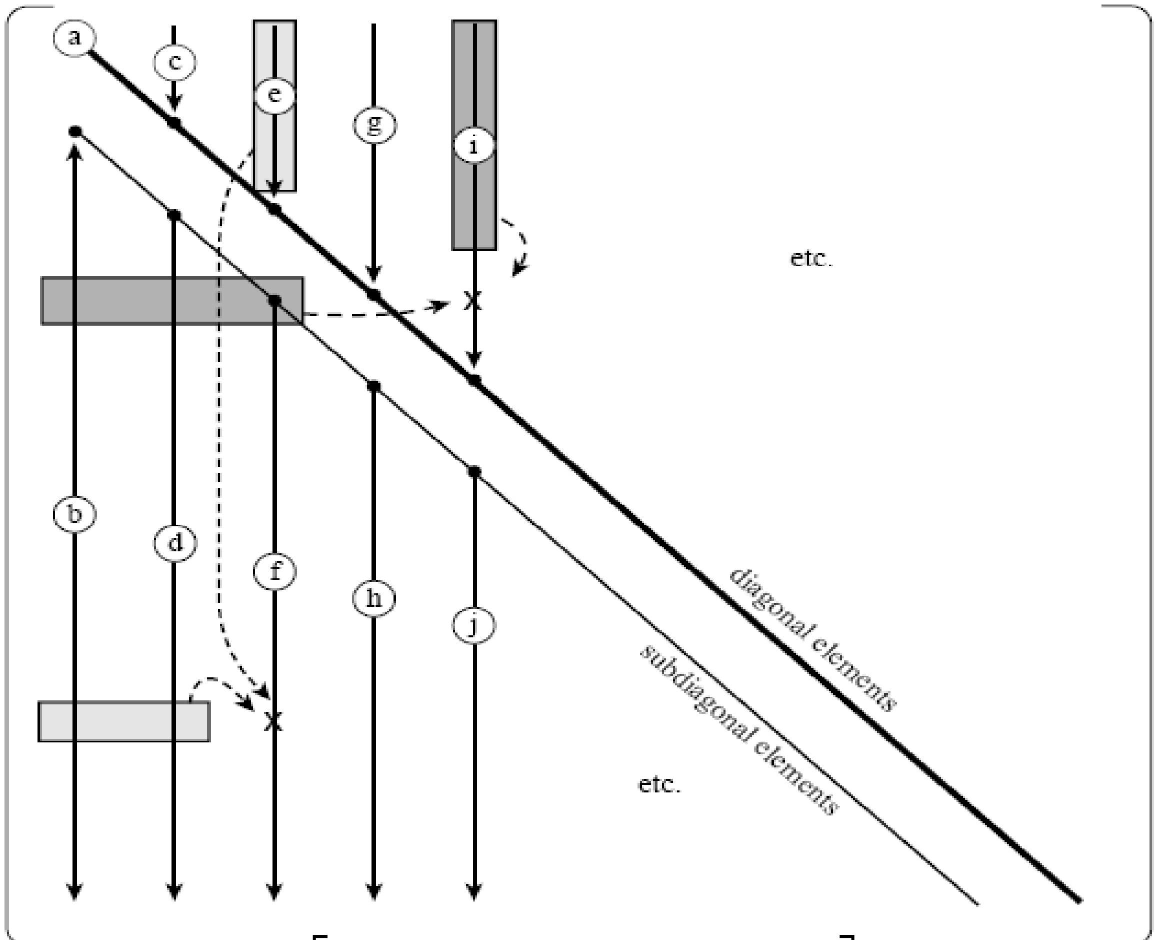
- set $\alpha_{ii} \equiv 1$, for $i = 1, \ldots, N$,
- for $i = 1, 2, \ldots, j$, solve β_{ij}

$$\beta_{ij} = a_{ij} - \sum_{k=1}^{i-1} \alpha_{ik}\beta_{kj}$$

- for $ji = j + 1, j + 2, \ldots, N$, solve α_{ij}

$$\alpha_{ij} = \frac{1}{\beta_{jj}}(\alpha_{ij} - \sum_{k=1}^{j-1} \alpha_{ik}\beta kj)$$

Crout's algorithm



$$\begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ \alpha_{21} & \beta_{22} & \beta_{23} & \beta_{24} \\ \alpha_{31} & \alpha_{32} & \beta_{33} & \beta_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \beta_{44} \end{bmatrix}$$

Complex system: quick-and-dirty

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b},$$

- if $\mathbf{b} = \mathbf{b} + i\mathbf{d}$, and **A** is real,
  1. *LU* decompose **A**,
  2. backsubstitution **b** to get the real part,
  3. backsubstitution **d** to get the imaginary part
- if **A** is complex,

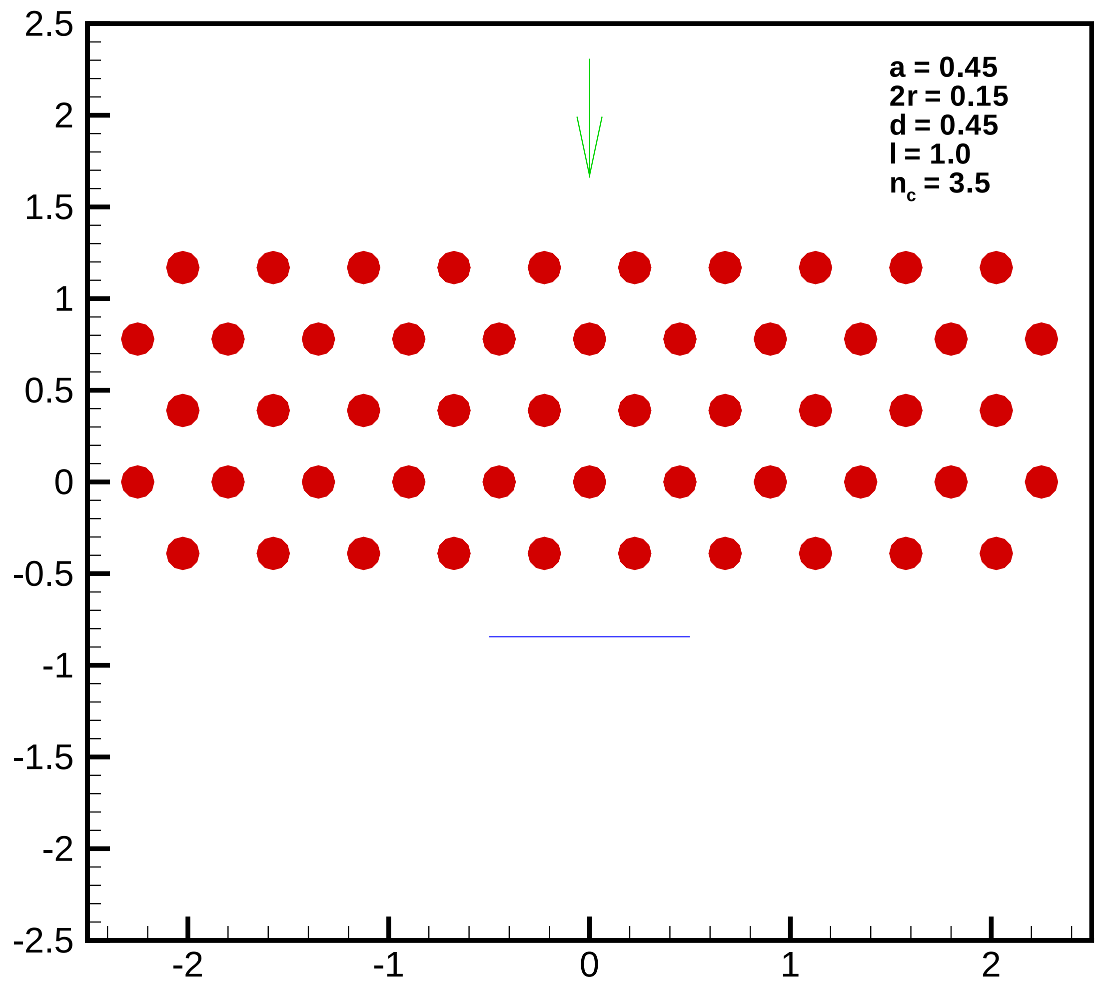
  $$(\mathbf{A} + i\mathbf{C}) \cdot (\mathbf{x} + i\mathbf{y}) = (\mathbf{b} + i\mathbf{d}),$$

  A *quick-and-dirty* way to solve complex system is

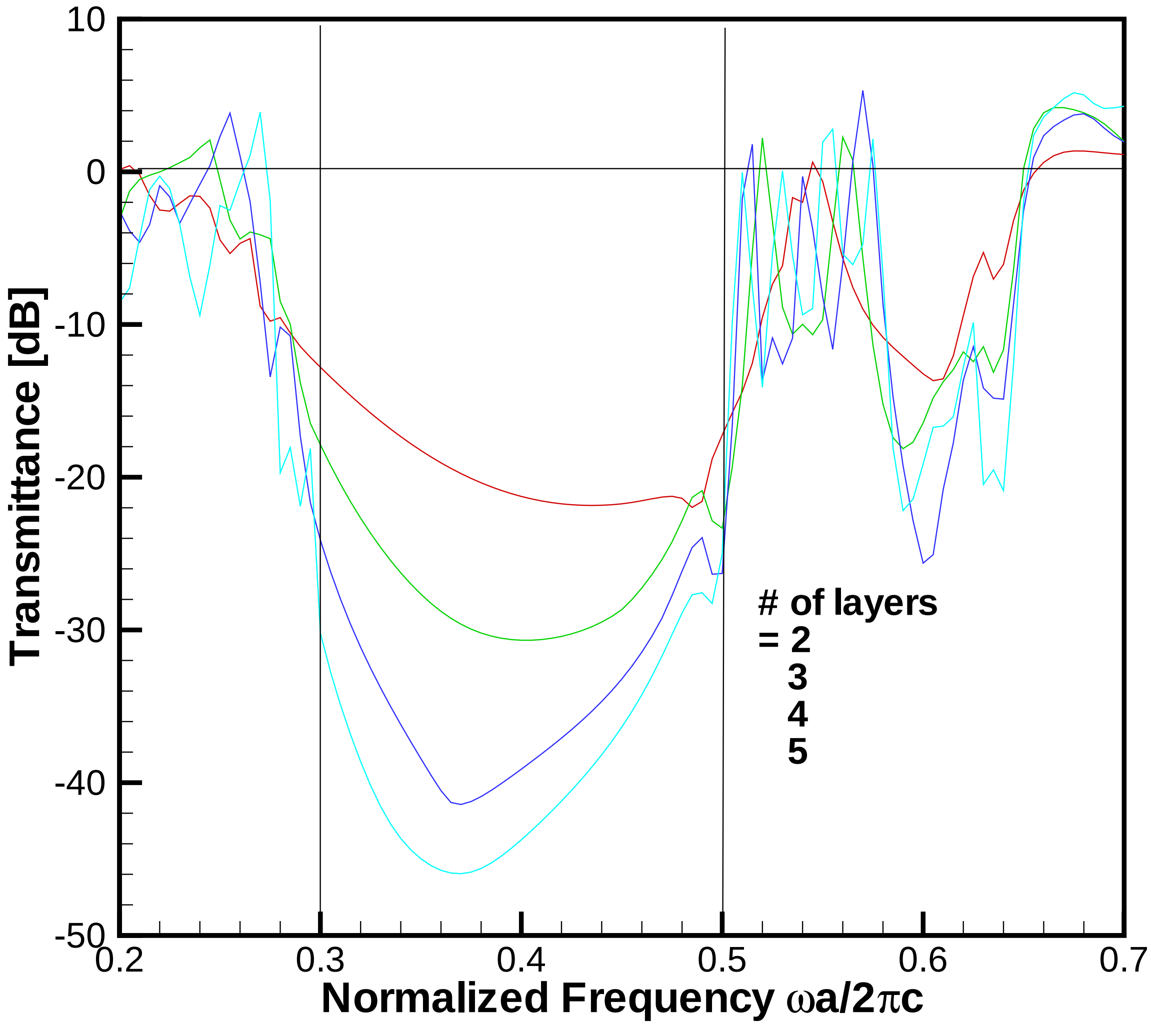
  $$\begin{pmatrix} \mathbf{A} & -\mathbf{C} \\ \mathbf{C} & \mathbf{A} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{d} \end{pmatrix}$$

  a $2N \times 2N$ set of real equations.

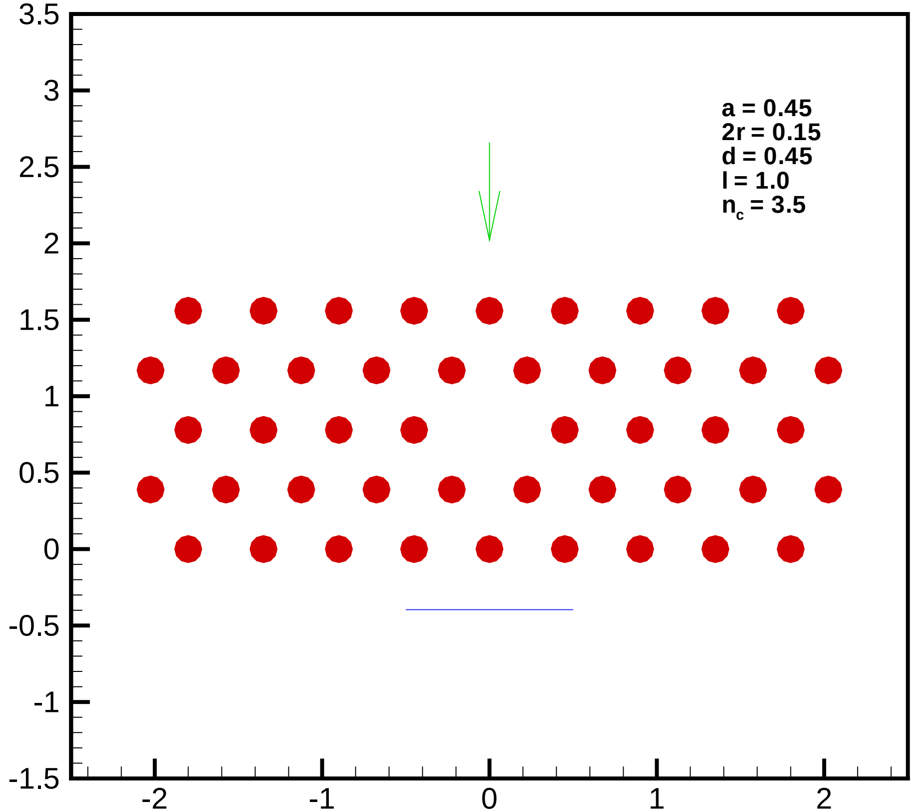
Photonic Crystals

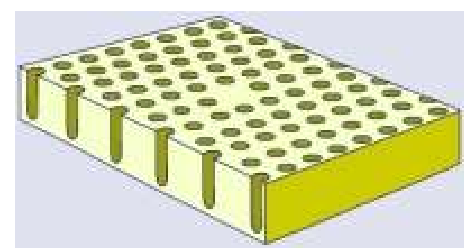


Transmittance

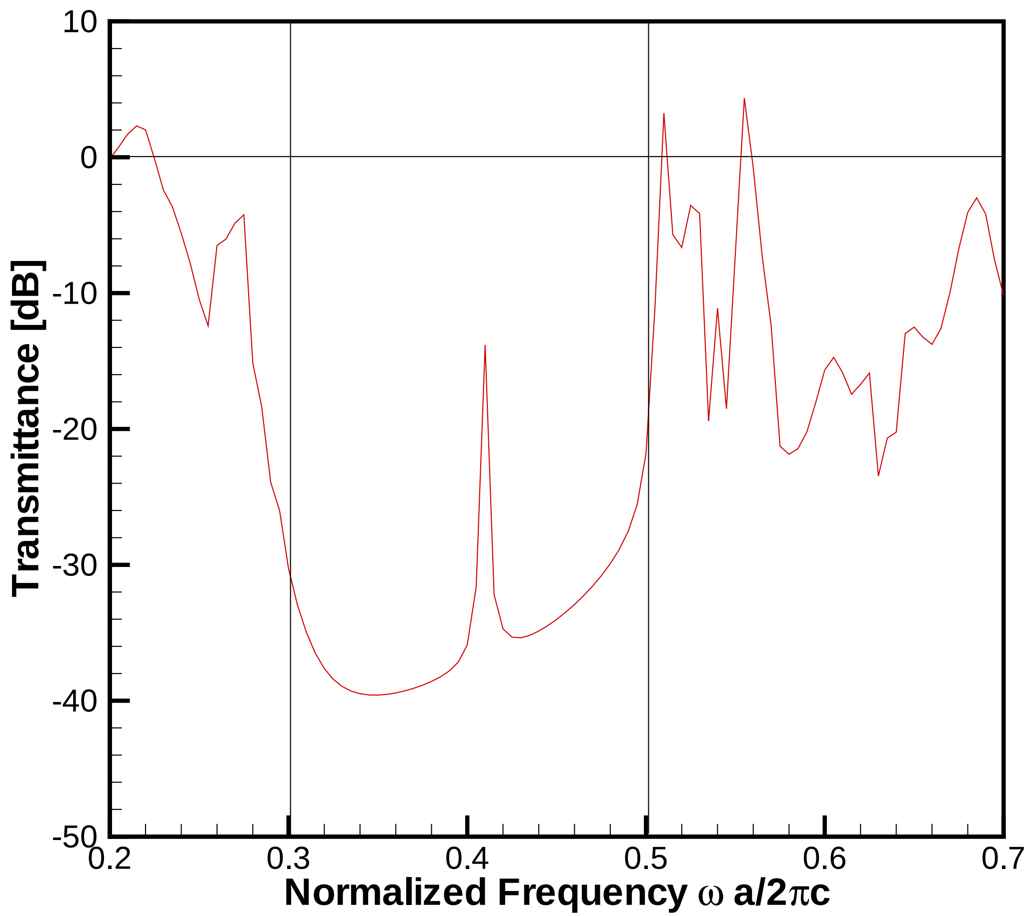


One-defect in photonic crystal

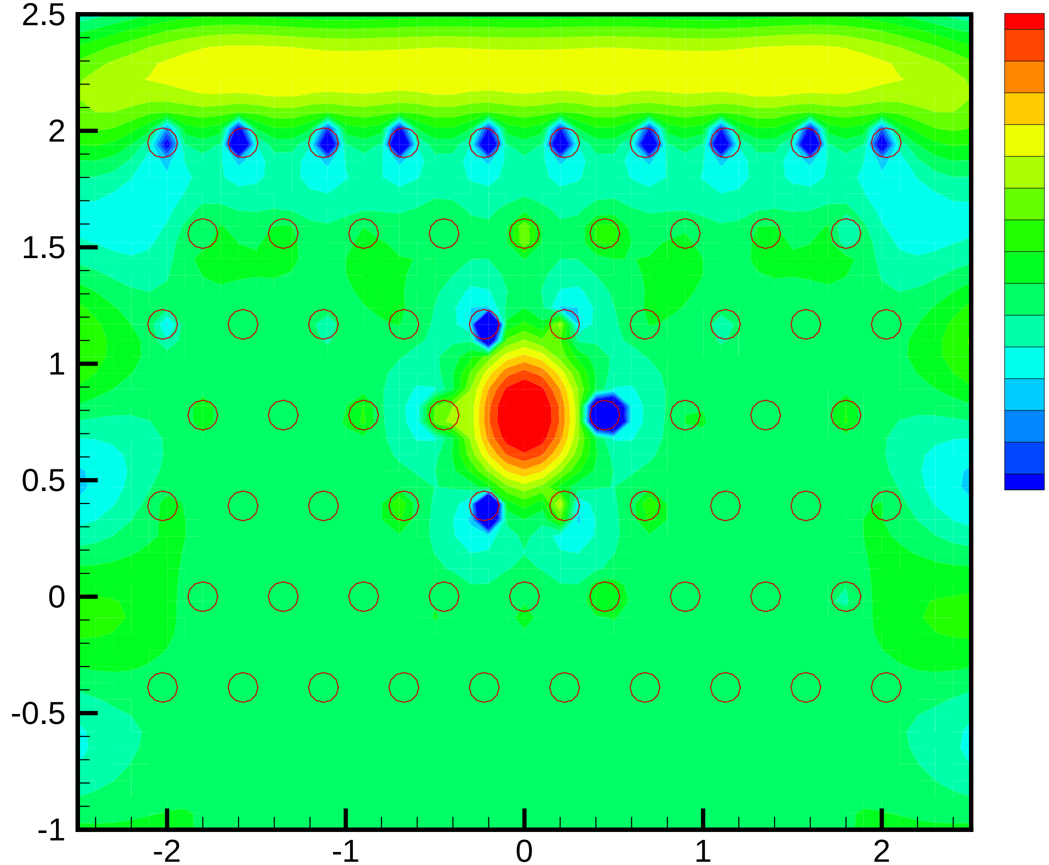




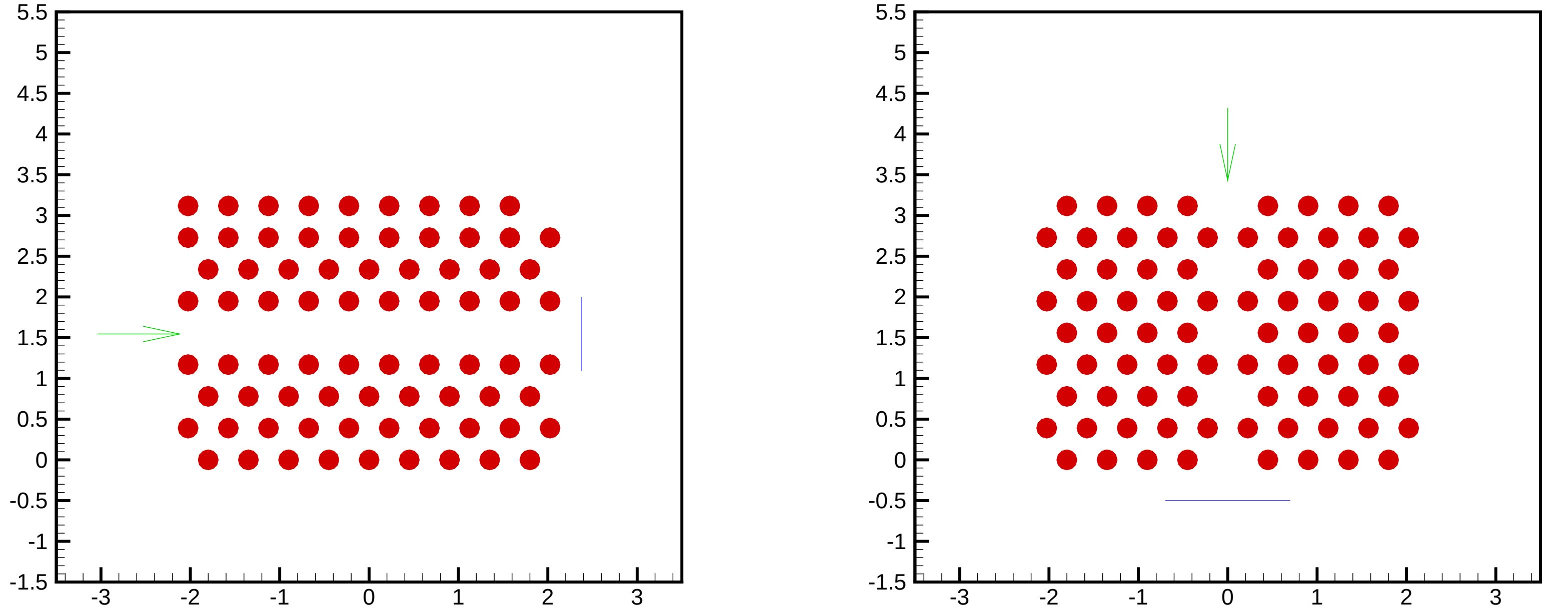
Transmittance: one-defect



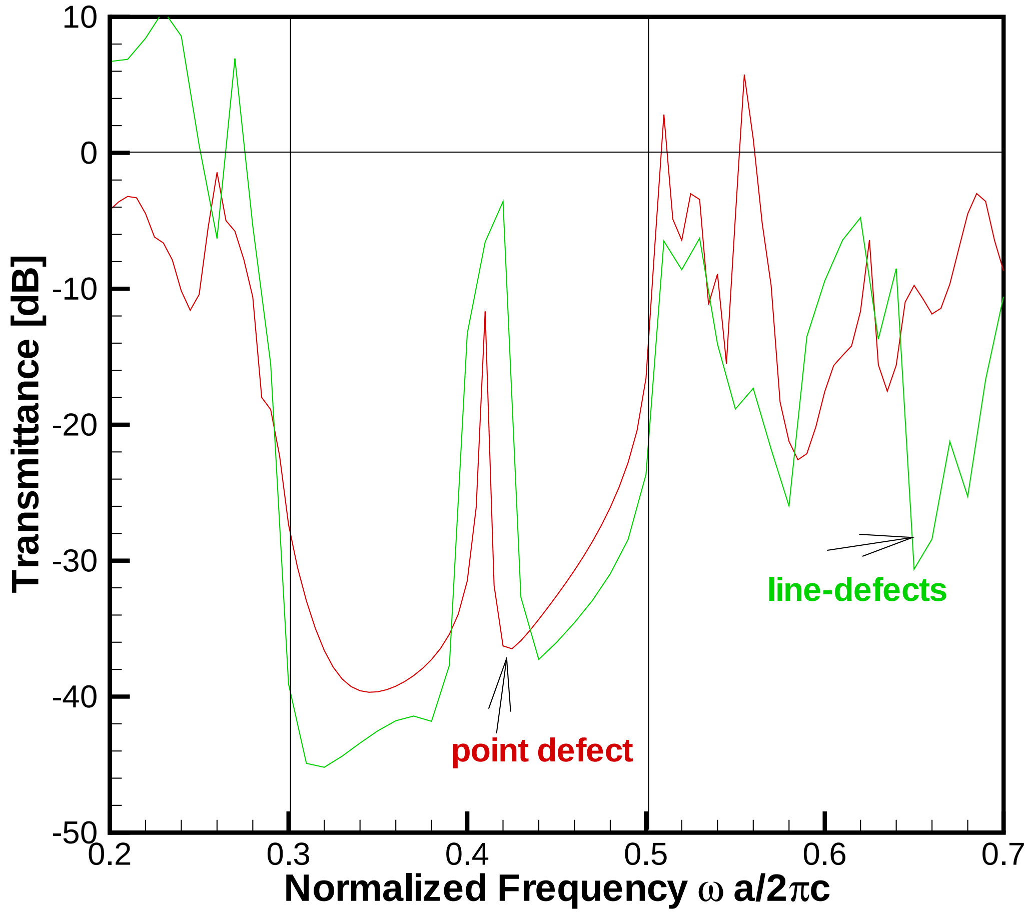
Localized field: one-defect



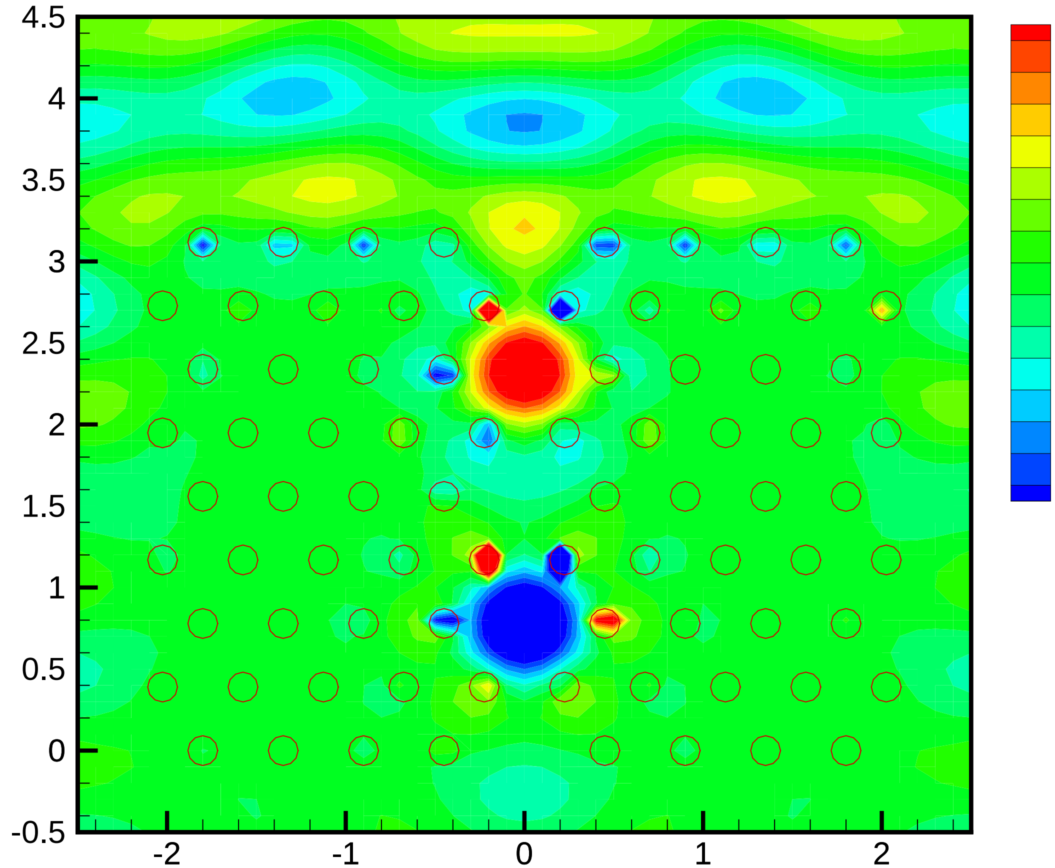
Line-defects in photonic crystal



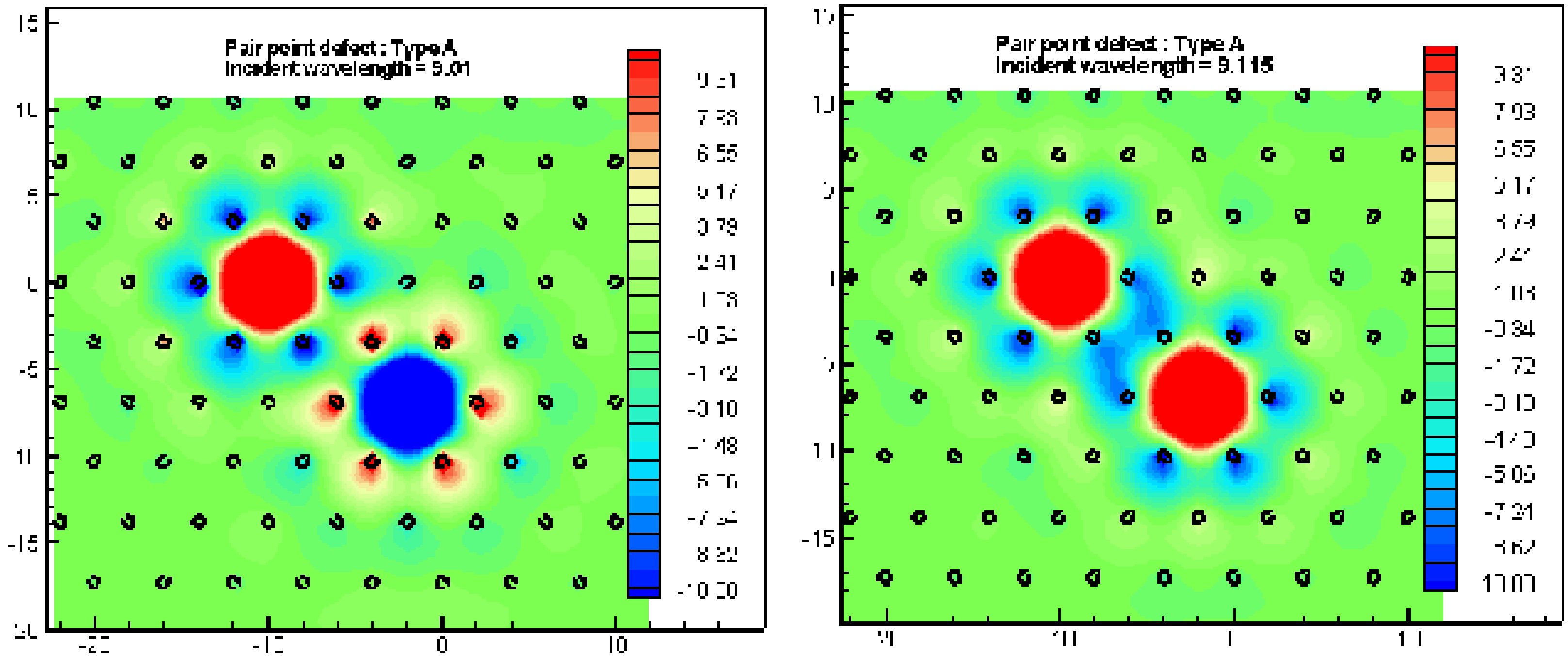
Transmittance: line-defect



Field distribution: line-defect

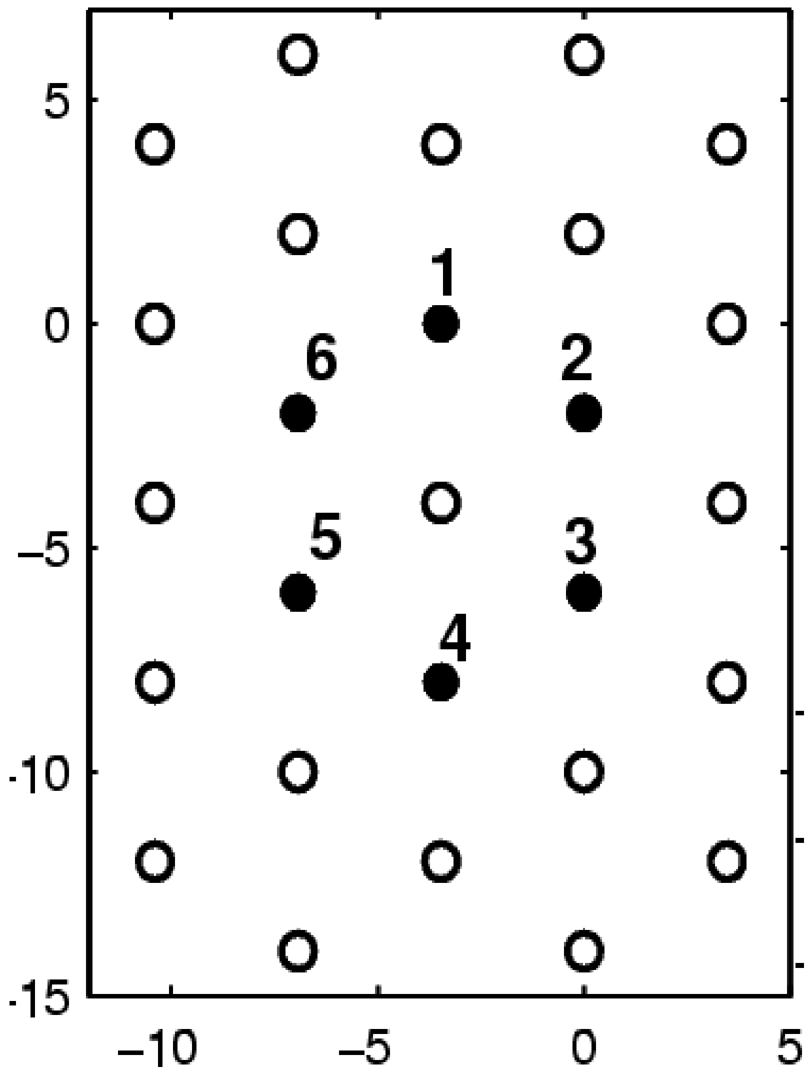


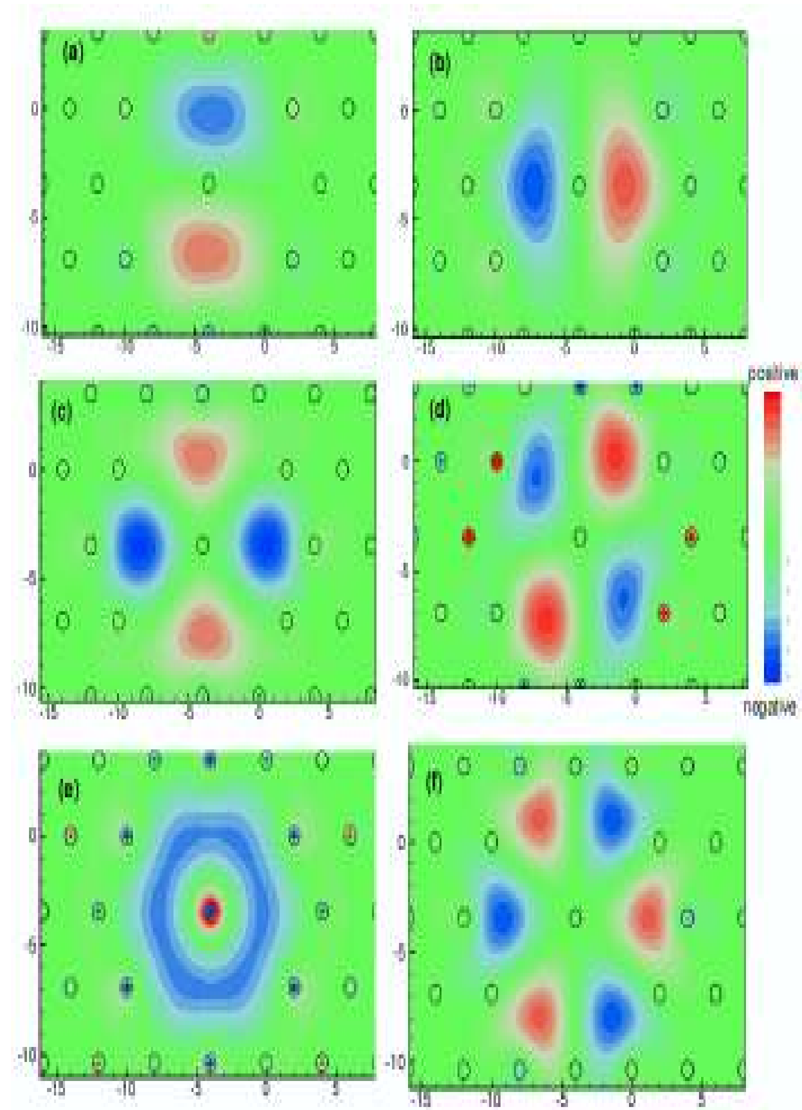
two-defects in photonic crystal



G. Tayeb and D. Maystre, *J. Opt. Soc. Am. A* **14**, 3323 (1997).

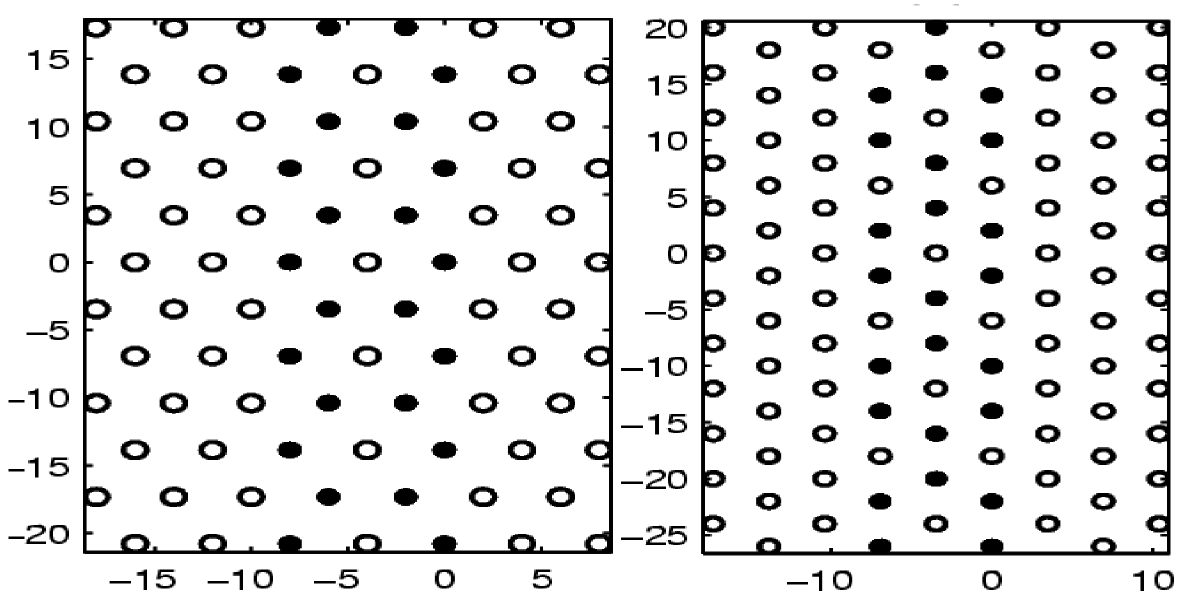
Optical molecule: benzene

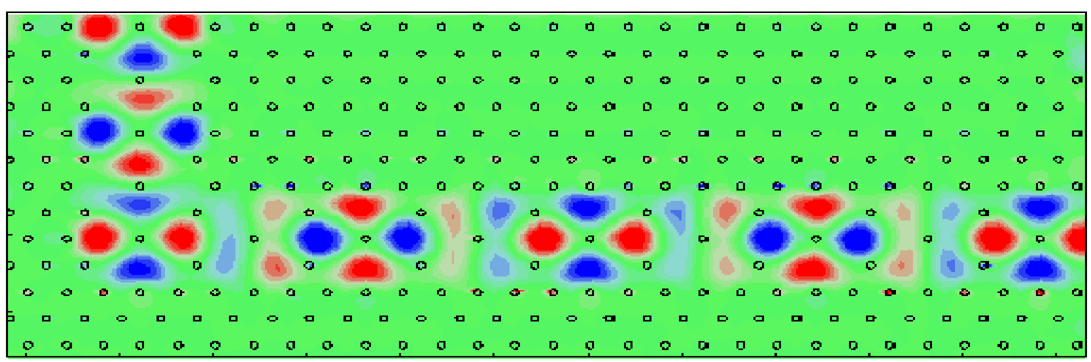




Courtesy: B. S. Lin

Optical molecule waveguide





Courtesy: B. S. Lin

Theoretical Backgrounds

the Helmholtz equation:

$$\bigtriangledown^2 E + \tilde{k}^2(M)E = 0$$

with

$$\tilde{k}^2(M) = k^2\tilde{\epsilon} = \begin{cases} k^2\epsilon_j & \text{if} \quad M \in C_j (j = 1, 2, \ldots, N) \\ k^2 & \text{if} \quad M \notin C_j (j = 1, 2, \ldots, N) \end{cases}$$

write the total field into $E = E^i + E^s$, then

$$\bigtriangledown^2 E^i + k^2(M)E^i = 0$$

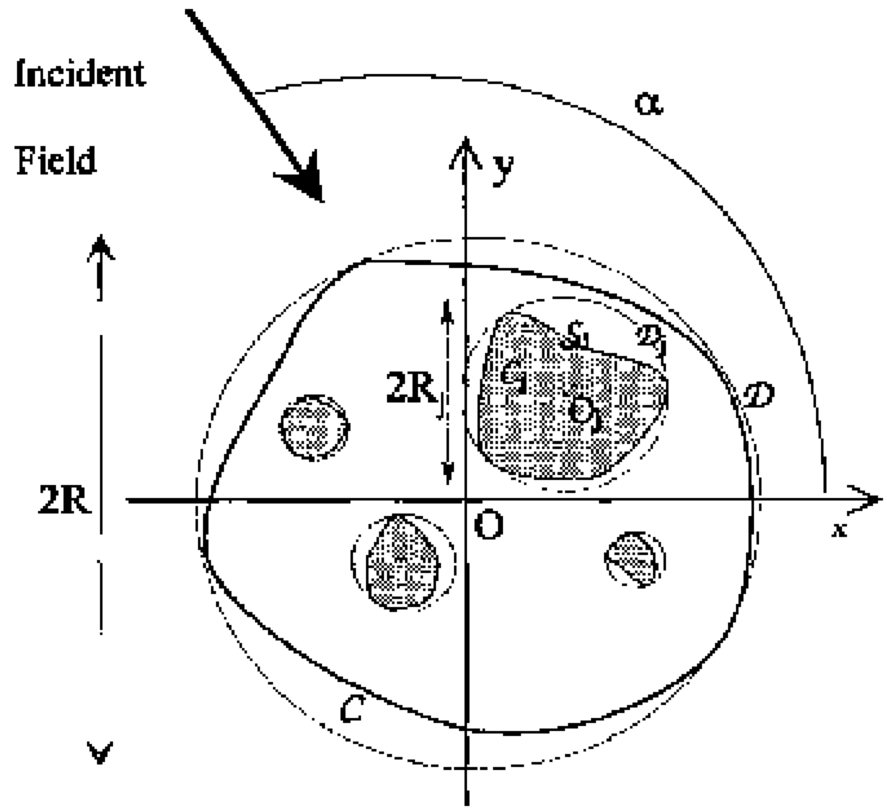
$$\bigtriangledown^2 E^s + k^2(M)E^s = [k^2 - \tilde{k}^2(M)]E$$

D. Felbacq, C. Tayeb, and D. Maystre, *J. Opt. Soc. Am. A* **11**, 2526 (1994).

Green's theorem

$$E^s(P) = \frac{-i}{4}k^2 \int\int dxdy\, H_0^{(1)}(kPM)[1-\tilde{\epsilon}(M)]E(M)$$

$$= \sum_{j=1}^{N} \frac{ik^2(\epsilon_j - 1)}{4} \int\int_{C_j} dxdy\, H_0^{(1)}(kPM)E(M)$$

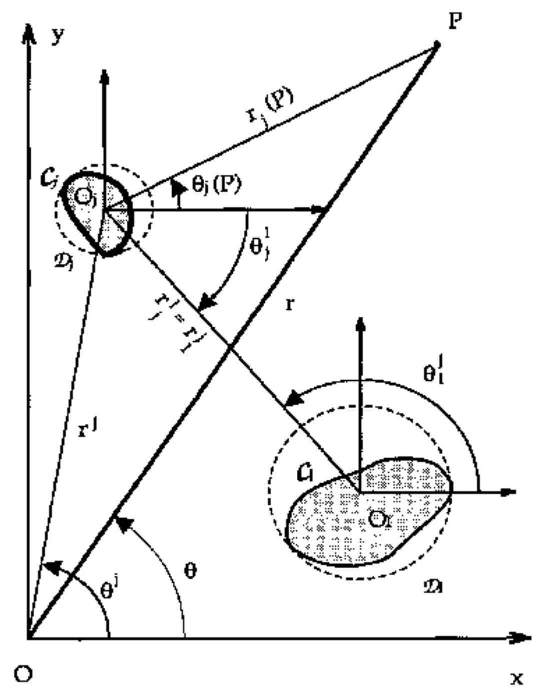


Graf's formula

$$H_0^{(1)} = \sum_{m=-\infty}^{\infty} e^{-im\theta_j(M)} J_m[kr_j(M)] H_m^{(1)}[kr_j(P)] e^{im\theta_j(P)}$$

$\forall P$ such that $r_j(P) \geq R_j$

$$E_j^s(P) = \sum_{m=-\infty}^{\infty} b_{j,m} H_m^(1)[kr_j(P)] e^{im\theta_j(P)}$$



Multiple-Scattering

incident field:

$$E^i(P) = e^{-i\mathbf{k}\cdot\mathbf{OP}}$$
$$= e^{-ikr^l cos(\alpha-\theta^l)} \sum_{m=-\infty}^{\infty} (-i)^m e^{-im\alpha} J_m[kr_l(P)] e^{im\theta_l(P)}$$

total field:

$$\begin{aligned} E(P) &= \sum_{m=-\infty}^{\infty} a_{l,m} J_m[kr_l(P)] e^{im\theta_l(P)} \\ &+ \sum_{m=-\infty}^{\infty} b_{l,m} H_m^{(1)}[kr_l(P)] e^{im\theta_l(P)} \end{aligned}$$

where

$$a_{l,m} = (-i)^m e^{-ikr^l cos(\alpha-\theta^l)-im\alpha} + \sum_{j\neq l}\sum_{q=-\infty}^{\infty} b_{j,q} e^{i(q-m)\theta_l^j} H_{m-q}^{(1)}(kr_l^j)$$

write into matrix form:

$$\hat{\mathbf{a}}_l = \hat{\mathbf{Q}}_l + \sum_{j\neq l} \mathbf{T}_{l,j}\hat{\mathbf{b}}_j$$

then, with the boundary condition

$$\hat{\mathbf{b}}_l = \mathbf{S}_l \hat{\mathbf{a}}_l$$

EE.NTHU 國立清華大學 電機工程學系及研究所

all we need is to solve

$$\hat{\mathbf{b}}_l - \sum_{j \neq l} \mathbf{S}_l \mathbf{T}_{l,j} \hat{\mathbf{b}}_j = \mathbf{S}_l \mathbf{Q}_l$$

or

$$\begin{bmatrix} \mathbf{I} & -\mathbf{S}_1\mathbf{T}_{1,2} & -\mathbf{S}_1\mathbf{T}_{1,3} & \dots & \dots \\ -\mathbf{S}_2\mathbf{T}_{2,1} & \mathbf{I} & -\mathbf{S}_2\mathbf{T}_{2,3} & \dots & \dots \\ -\mathbf{S}_3\mathbf{T}_{3,1} & -\mathbf{S}_3\mathbf{T}_{3,2} & \mathbf{I} & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}}_1 \\ \hat{\mathbf{b}}_2 \\ \hat{\mathbf{b}}_3 \\ \dots \\ \dots \end{bmatrix} = \begin{bmatrix} \mathbf{S}_1\mathbf{Q}_1 \\ \mathbf{S}_2\mathbf{Q}_2 \\ \mathbf{S}_3\mathbf{Q}_3 \\ \dots \\ \dots \end{bmatrix}$$

Examples in Matlab: Ax=b

```
A=[1 2 5; 0.2 1.6 7.4; 0.5 4 8.5];
b=[2 0 1]';
```

- Direct solve:

  ```
  x = A\b %Denotes the solution to Ax = b,
  x = b/A %Denotes the solution to xA = b.
  ```

- Gauss Elimination:

  ```
  x= gauss(A, b)
  ```

```
x = [2.1000 0.2000 -0.1000]'
```

Examples in Matlab: LU

- LU decomposition:

```
[L, U, P] = lu(A) % L*U = P*A.
L = 1.0000         0         0
    0.5000    1.0000         0
    0.2000    0.4000    1.0000
U =  1     2     5
     0     3     6
     0     0     4
P =  1     0     0
     0     0     1
     0     1     0
```

```
x = U^-1*L^-1*P*b % P'LUx=b
```

Jacobi iteration

consider the equation: $3x + 1 = 0$

- $2x = -x + 1, \rightarrow x_{k+1} = -\frac{1}{2}x_k - \frac{1}{2},$

$$x_k = \frac{-1/2[1-(-1/2)^k]}{1-(-1/2)} + (-1/2)^k x_0,$$
$$= \frac{-1}{3}, \quad \text{as} \quad k \rightarrow \infty$$

- $x = -2x + 1, \rightarrow x_{k+1} = -2x_k - 1,$

$$x_k = \frac{-[1-(-2)^k]}{1-(-2)} + (-2)^k x_0,$$
$$= \infty, \quad \text{as} \quad k \rightarrow \infty$$

EE.NTHU 國立清華大學 電機工程學系及研究所

Jacobi iteration

$$\mathbf{A}\cdot\mathbf{x}=\mathbf{b}, \quad \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}\cdot\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}=\begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

Jacobi's method for iteration

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix}=\begin{bmatrix} 0 & -2/3 \\ -1/2 & 0 \end{bmatrix}\cdot\begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix}+\begin{bmatrix} 1/3 \\ -1/2 \end{bmatrix},$$

$$\mathbf{x}_{k+1}=\tilde{\mathbf{A}}\cdot\mathbf{x}_k+\tilde{\mathbf{b}},$$

Jacobi iteration

```
A = [3 2; 1 2]; b = [1 -1]';
x0 = [0 0]';

jacobi(A, b, x0, 20)

X =
    [0.3333 -0.5000]' %01 [0.6667 -0.6667]' %02
    [0.7778 -0.8333]' %03 [0.8889 -0.8889]' %04
    [0.9259 -0.9444]' %05 [0.9630 -0.9630]' %06
    [0.9753 -0.9815]' %07 [0.9877 -0.9877]' %08
    [0.9918 -0.9938]' %09 [0.9959 -0.9959]' %10
    [0.9973 -0.9979]' %11 [0.9986 -0.9986]' %12
    [0.9991 -0.9993]' %13 [0.9995 -0.9995]' %14
    [0.9997 -0.9998]' %15 [0.9998 -0.9998]' %16
    [0.9999 -0.9999]' %17 [0.9999 -0.9999]' %18
    [1.0000 -1.0000]' %19 [1.0000 -1.0000]' %20
```

Jacobi iteration

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b},$$

Jacobi's method for iteration

$$\mathbf{x}_{k+1} = \tilde{\mathbf{A}} \cdot \mathbf{x}_k + \tilde{\mathbf{b}},$$

where

$$\tilde{\mathbf{A}} = \begin{bmatrix} 0 & -a_{12}/a_{11} & \cdots & -a_{1N}a_{11} \\ -a_{21}/a_{22} & 0 & \cdots & -a_{2N}/a_{22} \\ & \cdots & & \\ -a_{N1}/a_{NN} & -a_{N2}/a_{NN} & \cdots & 0 \end{bmatrix}, \quad \tilde{\mathbf{b}} = \begin{bmatrix} b_1/a_{11} \\ b_2/a_{22} \\ \vdots \\ b_N/a_{NN} \end{bmatrix}.$$

$$x_m^{(k+1)} = -\sum_{n \neq m}^{N} \frac{a_{mn}}{a_{mm}} x_n^{(k)} + \frac{b_m}{a_{mm}}, \quad \text{for} \quad m = 1, 2, \ldots, N$$

Gauss-Seidel iteration

Gauss-seidel iteration:

$$\begin{aligned} x_{1,k+1} &= -\frac{2}{3}x_{2,k} + \frac{1}{3}, \\ x_{2,k+1} &= -\frac{1}{2}x_{1,k+1} - \frac{1}{2}. \end{aligned}$$

for $m = 1, 2, \ldots, N$,

$$\begin{aligned} x_m^{(k+1)} &= -\sum_{n=1}^{m-1} \frac{a_{mn}}{a_{mm}} x_n^{(k+1)} - \sum_{n=m+1}^{N} \frac{a_{mn}}{a_{mm}} x_n^{(k)} + \frac{b_m}{a_{mm}}, \\ &= \frac{b_m - \sum_{n=1}^{m-1} a_{mn} x_n^{(k+1)} - \sum_{n=m+1}^{N} a_{mn} x_n^{(k)}}{a_{mm}} \end{aligned}$$

convergy more fast!

Example in Matlab: Gauss-Seidel iteration

```
A = [3 2; 1 2]; b = [1 -1]';
x0 = [0 0]';

gauseid(A, b, x0, 10)

X =
    [0.3333 -0.6667]' %01 [0.7778 -0.8889]' %02
    [0.9259 -0.9630]' %03 [0.9753 -0.9877]' %04
    [0.9918 -0.9986]' %05 [0.9973 -0.9986]' %06
    [0.9991 -0.9995]' %07 [0.9997 -0.9998]' %08
    [0.9999 -0.9999]' %09 [1.0000 -1.0000]' %10
```

convergence condition:

$$|a_{mm}| > \sum_{n \neq m}^{N} |a_{mn}|, \quad \text{for} \quad m = 1, 2, \ldots, N$$

EE.NTHU 國立清華大學 電機工程學系及研究所

Example in Matlab: nonlinear equations

nonlinear equations:

$$\begin{aligned} x_1^2 + 10x_1 + 2x_2^2 - 13 &= 0, \\ 2x_1^3 - x_2^2 + 5x_2 - 6 &= 0, \end{aligned}$$

with the Gauss-Seidel scheme,

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} (13 - x_1^2 - 2x_2^2)/10 \\ (6 - 2x_1^3 + x_2^2)/5 \end{bmatrix}.$$

with the conditions

$$|\mathbf{x}_{k+1} - \mathbf{x}_k| < \epsilon, \quad \text{or} \quad \frac{|\mathbf{x}_{k+1} - \mathbf{x}_k|}{|\mathbf{x}_k + \text{eps}|} < \epsilon$$

Relaxation technique

relaxation technique:

$$x_m^{(k+1)} = (1-\omega)x_m^{(k)} + \omega \frac{b_m - \sum_{n=1}^{m-1} a_{mn} x_n^{(k+1)} - \sum_{n=m+1}^{N} a_{mn} x_n^{(k)}}{a_{mm}}$$

with $0 < \omega < 2$

- $1 < \omega < 2$: SOR, Successive Over-Relaxation,
- $0 < \omega < 1$: successive under-relaxation.