OLD DOMINION UNIVERSITY

Ordinary Differential Equations III

A. Godunov

1. Boundary value problem: introduction
2. The shooting method
3. The equilibrium method
4. Eigenproblems (solving Schrodinger equation)

1

Part 1:

Introduction to the boundary value problem

2

Boundary value problem vs. initial value problem

The boundary value problem is more challenging than the initial value problem both analytically and numerically

For initial value problems

- Sufficient initial conditions determine unique solutions
- Same numerical methods work for both linear and nonlinear ODE

For boundary value problems

- The solution may not exist for given boundary conditions, or it may not be unique
- Some numerical methods work only for linear ODE

example: $y'' + y = 0.$

for $y(0) = 1,\ y(\pi/2) = 0,\quad y(x) = \cos x$

for $y(0) = 0,\ y(\pi) = 0,\quad y(x) = \mathrm{C}\sin x$ (not unique)

for $y(0) = 1,\ y(\pi) = 1\quad$ no solution

3

3

Major methods for the boundary value problem

Two classes of numerical methods for solving boundary value problems

I. Finite difference methods

a) The shooting method

b) The equilibrium method

II. Linear combination of trial functions

a) The finite element method

b) The Rayleigh-Ritz method

c) The Galerkin method

d) The collocation method

4

4

Second-order ODE with boundary conditions

Linear second-order ODE as an excellent starting point

$$y'' + p(x)y' + q(x)y = f(x)$$

General boundary conditions

$$a_1 y'(0) + b_1 y(0) = c_1$$

$$a_2 y'(l) + b_2 y(l) = c_2$$

Specific types of boundary conditions

1. Dirichlet boundary conditions
$$y(0) = y_0, \qquad y(l) = y_l$$
2. Neumann boundary conditions
$$y'(0) = y'_0, \qquad y'(l) = y'_l$$
3. Mixed boundary conditions – see general boundary conditions

5

5

Part 2:

The shooting method

6

Key idea for the shooting method

- The key idea of the shooting method is to transform the boundary value ODE into a system of first-order ODEs and solve as an initial value problem.
- Only boundary condition on one side is used as one of the initial conditions. The additional initial condition is assumed.
- Then an iterative approach is used to vary the assumed initial condition till the boundary condition on the other side is satisfied.

7

7

From boundary value problem to initial value problem

Consider the general nonlinear second-order boundary-value ODE with Dirichlet boundary conditions, written in the following form:

$$y'' = F(x, y, y'), \qquad y(x_1) = y_1, \qquad y(x_2) = y_2$$

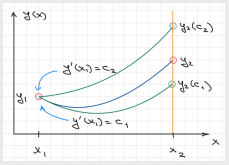
An initial-value problem is created by assuming a value $y'(x_1) = c_1$. Such guess gives a solution $y_{c_1}(x)$ than is most likely does not satisfy the given boundary condition on the right side: $y_{c_1}(x_2) \neq y(x_2) = y_2$.

8

8

From boundary value problem to initial value problem

A second guess $y'(x_1) = c_2$ gives another solution with $y_{c_2}(x_2) \neq y(x_2)$.



We can use the two solutions to initiate the iterative search to find such $y'(x_1)$ that the right boundary condition is satisfied.

For a non-linear ODE this is a zero finding problem for a nonlinear function of c such that we need to find c such that

$f(c) = y(x_2) - y_c(x_2) = 0$

9

9

Short review for solving a nonlinear problem $f(x) = 0$

For our problem in hand it is better to proceed with "open domain methods". For the Newton's method we have

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + (x - x_0)^2 \frac{f''(x_0)}{2!} + \cdots$$

Suppose that x is the solution for $f(x) = 0$. If we keep two first terms $f(x) = 0 = f(x_0) + (x - x_0)f'(x_0)$, then

$x = x_0 - \frac{f(x_0)}{f'(x_0)}$ or each next iteration is $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$

The method of secant estimates the derivative at x_k as

$$f'(x_k) = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

then

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

10

10

Applying the method of secants to $y(x_2) - y_c(x_2) = 0$

In the method of secants

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

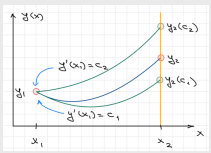
we let $f(c) = y(x_2) - y_c(x_2)$, and we consider c as a variable, then

$$c_{k+1} = c_k - \frac{y_2 - y_{c_n}(x_2)[c_k - c_{k-1}]}{y(x_2) - y_{c_k}(x_2) - y(x_2) + y_{c_{k-1}}}$$

or

$$c_{k+1} = c_k + \frac{\left(y_2 - y_{c_n}(x_2)\right)}{y_{c_k}(x_2) - y_{c_{k-1}}(x_2)}(c_k - c_{k-1})$$

Thus, we need two guesses to initiate the iterative process.



11

11

Special case: linear ODEs

For a linear ODE, the principle of superposition applies. First, compute two solutions for $y'(x_1) = c_1$ and $y'(x_1) = c_2$ denoted by $y_{c1}(x)$ and $y_{c2}(x)$ respectively.

Then form a linear combination of these two solutions:

$$y(x) = C_1 y_{c1}(x) + C_2 y_{c2}(x).$$

Applying the boundary conditions gives

$$y_1 = C_1 y_{c1}(x_1) + C_2 y_{c2}(x_1) = (C_1 + C_2) y_1$$

$$y_2 = C_1 y_{c1}(x_2) + C_2 y_{c2}(x_2).$$

Then solving for the unknows C_1 and C_2 gives

$$C_1 = \frac{y_2 - y_{c2}(x_2)}{y_{c1}(x_2) - y_{c2}(x_2)} \qquad C_2 = \frac{y_{c1}(x_2) - y_2}{y_{c1}(x_2) - y_{c2}(x_2)}$$

For linear ODEs no iterations are needed

12

12

Example: Matlab code using the shooting method

```
function [x,y,dy] = shoot2(f,x,y,dy,n,eps)
%{
! Solution of the boundary-value second-order 1D ODE
! d2y/dx2 = f(x,y,dy/dx) with Dirichlet boundary conditions
! y(xmin) = ..., and y(xmax) = ...
! Method: unilizes the shooting method based on the method of secants
! (calls 4th-order Runge-Kutta to solve the initial value problem)
! written by:  Alex Godunov (last revision: March 2022)
!--------------------------------------------------------------------
! input ...
!  f(x,y,dy)  - function d2y/dx2 (supplied by a user)
!  x(1), x(n) - boundary points
!  y(1), y(n) - boundary values (Dirichlet boundary conditions)
! dy(1),dy(2) - two guesses for y'(x(1))
!           n  - number of grid points
!         eps  - tolerance (abs value)
! output ...
!  y(i) and dy(i) solutions at points x(i) (i=1,...,n)
! note: dy corresponds to y' (the first derivative)
%}
```

13

13

Example: Matlab code (cont.)

```
it=101;          % max number of iterations
% first guesses for g(it)
g(1) = dy(1);
g(2) = dy(2);
yn = y(n); %! remember the second boundary condition
dx = (x(n)-x(1))/(n-1); % generate base points x(i) from x(1), x(n) and n
for i=2:n
  x(i) = x(i-1)+dx;
end
% shooting iterations (for the first two - we use assumed values of dy(1))
for k=1:it
  dy(1) = g(k);
  [y,dy] = rk4_2d(f,x,y,dy,n);  % solves initial value ODE on n-points
  c(k) = y(n);
  if abs(yn-c(k)) < eps
      break
  end
  if k >= 2
      g(k+1)=g(k)-(c(k)-yn)*(g(k)-g(k-1))/(c(k)-c(k-1));
  end
end % end iterations
end % end function shoot2
```

14

14

Example: Matlab code (cont.)

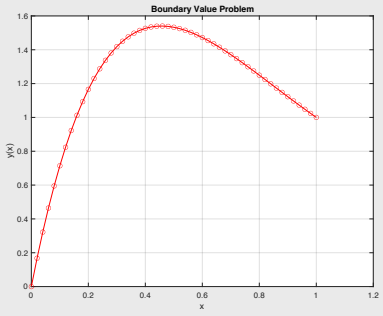
```
function [f] = f(x,y,dy)
% -----------------------------------------
%  the second derivative - use original ODE
%  d2y/dx2 = f(x,y,dy)
% -----------------------------------------
f = -4.0*dy - 6.25*y + exp(x);
end
```

15

15

Example 1a:

Solving $y'' = -4.0y' - 6.25y + e^x,\ \ y(0) = 0,\ y(1) = 1$

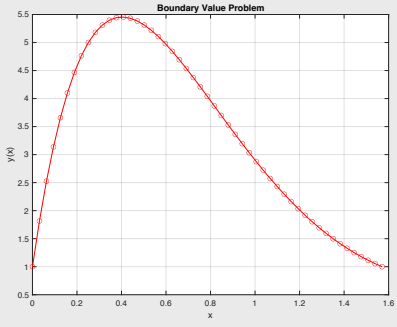


16

16

Example 1b

Solving $y'' = -4.0y' - 6.25y + e^x,\ \ y(0) = 1,\ y(\pi/2) = 1$

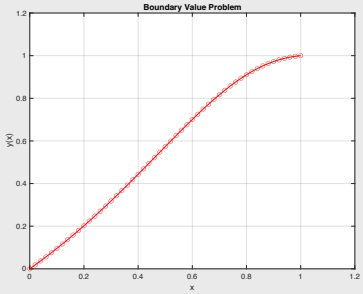


17

17

Example 2

$y'' = (-4xy) * y' - [6.25y^2 \sin(xy)] * y + e^x, \qquad y(0) = 0, \qquad y(1) = 1$



18

18

The shooting method for other boundary conditions

The boundary-value problems considered so far had Dirichlet (i.e., known function value) boundary conditions.

Many problems have derivative (i.e., Neumann) or mixed boundary conditions.

The shooting method for derivative boundary conditions is analogous to the shooting method for Dirichlet boundary conditions, except that we shoot for the value of the derivative instead of the value of the function at the boundary.

For the mixed boundary conditions we shoot for the mixed conditions.

19

Summary for the shooting method

Pro

- Solving as initial value problem
- Works very well for both linear and nonlinear ODEs
- Easy to implement fourth- or higher-order methods
- No solving a system of FDA equations

Con

- Iterative approach
- Shooting for more than one boundary condition is time-consuming

20

Part 3:

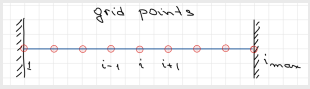
The equilibrium method

21

The equilibrium (boundary value) method

Key idea: construct a finite difference approximation of the exact ODE at every point on a discrete finite difference grid. Then a system of equations must be solved simultaneously. Here are the steps:

1. Discretizing the continuous solution domain into a discrete finite difference grid
2. Approximating the exact derivatives in the boundary-value ODE by algebraic finite difference approximations
3. Substituting the FDAs into the ODE to obtain an algebraic finite difference equation
4. Solving the resulting system of algebraic FDEs
   (for linear ODEs – a system of linear equations)



22

The finite difference approximation

"2. Approximating the exact derivatives in the boundary-value ODE by algebraic finite difference approximations"

$y(x_i) = y_i, \qquad y'(x_i) = y_i', \qquad y''(x_i) = y_i''$

$$y_{i+1} = y_i + y_i'\Delta x + \frac{1}{2}y_i''\Delta x^2 + \cdots$$

$$y_{i-1} = y_i - y_i'\Delta x + \frac{1}{2}y_i''\Delta x^2 + \cdots$$

The central difference approximation for y_i' and y_i'' are

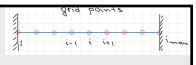
$$y_i' = \frac{y_{i+1} - y_{i-1}}{2\Delta x} + O(\Delta x^2)$$

$$y_i'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{2\Delta x^2} + O(\Delta x^2)$$

Higher-order approximations for y_i' and y_i'' can be derived using Taylor series (see lecture notes on Differentiation)

23

Linear second-order ODE b.v. problem



We consider linear second-order ODE on $[a, b]$ with Dirichlet boundary conditions

$$y'' + P(x)y' + Q(x)y = F(x)$$

$$y(a) = A, \qquad y(b) = B$$

Substituting the central difference approximations for y' and y'' at i gives

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{2\Delta x^2} + P_i\left(\frac{y_{i+1} - y_{i-1}}{2\Delta x}\right) + Q_i y_i = F_i$$

Multiplying all terms by Δx^2, and gathering terms yields:

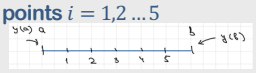
$$\left(1 - \frac{\Delta x}{2}P_i\right)y_{i-1} + (-2 + \Delta x^2 Q_i)y_i + \left(1 + \frac{\Delta x}{2}P_i\right)y_{i+1} = \Delta x^2 F_i$$

Applying this at each point in a discrete finite difference grid yields a tridiagonal system of FDEs, which can be solved by the Thomas algorithm (for solving linear tridiagonal systems of linear equations)

24

Example: application to five points $i = 1,2 \ldots 5$

Introducing the following notations



$$a_i = 1 - \frac{\Delta x}{2} P_i, \qquad b_i = -2 + \Delta x^2 Q_i, \qquad c_i = 1 + \frac{\Delta x}{2} P_i, \qquad d_i = \Delta x^2 F_i$$

the system of equations can be rewritten as

$$a_i y_{i-1} + b_i y_i + c_i y_{i+1} = d_i$$

$$\begin{array}{ccccccc} b_1 y_1 & +c_1 y_2 & +0 & +0 & +0 & = & d_1 - a_1 y_0 \\ a_2 y_1 & +b_2 y_2 & +c_2 y_3 & +0 & +0 & = & d_2 \\ 0 & +a_3 y_2 & +b_3 y_3 & +c_3 y_4 & +0 & = & d_3 \\ 0 & +0 & +a_4 y_3 & +b_4 y_4 & +c_4 y_5 & = & d_4 \\ 0 & +0 & +0 & a_5 y_4 & +b_5 y_5 & = & d_5 - a_5 y_6 \end{array}$$

where $y_0 = y(a),\ y_6 = y(b)$.

This is clearly a tri-diagonal system.

25

Example: Matlab code – the equilibrium method

```
%{
Solving linear boundary value problem with Dirichlet boundary conditions
y'' + P(x)y' +Q(x)y = F(x)

METHOD: Equilibrium + Thomas algorithm for solving 3-diagonal system

CALLS: external functions: Thomas.m

INPUT:
P(x), Q(x), F(x) - external functions
a and y(a), b and y(b) - boundary conditions
N - total number of points in the grid, including two boundary points

OUTPUT:
Y(X) - solution as arrays Y and X (size N)

Last revision: AG March 2022
%}

function [X,Y] = ODEbvE(P, Q, F, a, ya, b, yb, N)
```


26

Example: Matlab code – cont.

```
function [X,Y] = ODEbvE(P, Q, F, a, ya, b, yb, N)
n = N-2;
h = (b-a)/(n+1);
for k = 1:n
    x(k) = a + k*h;
    c(k,1) = 1.0 - h*P(x(k))/2.0;
    c(k,2) = (1.0)*h*h*Q(x(k)) - 2.0;
    c(k,3) = 1.0 + h*P(x(k))/2.0;
    d(k)   = h*h*F(x(k));
end
d(1)   = d(1) - c(1,1)*ya;
c(1,1) = 0.0;
d(n)   = d(n) - c(n,3)*yb;
c(n,3) = 0.0;
[y] = Thomas(c,d,n);
X(1)  = a;
X(N)  = b;
Y(1)  = ya;
Y(N)  = yb;
for k = 1:n
    X(k+1) = x(k);
    Y(k+1) = y(k);
end
end
```


27

Example: Matlab code – the Thomas method

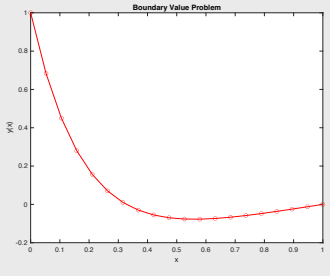
```
function [x] = Thomas(c,b,n)
% !============================================================
% ! Solutions to a system of tridiagonal linear equations c*x=b
% ! Method: the Thomas method
% !------------------------------------------------------------
% ! input ...
% ! c(n,3) - array of coefficients for matrix where c(n,2) - diagonal
% ! b(n)   - vector of the right hand coefficients b
% ! n      - number of equations
% ! output ...
% ! x(n)   - solutions
% !============================================================
%step 1: forward elimination
for k = 2:n
  coeff=c(k,1)/c(k-1,2);
  c(k,2)=c(k,2)-coeff*c(k-1,3);
  b(k)=b(k)-coeff*b(k-1);
end
%step 2: back substitution
x(n) = b(n)/c(n,2);
for k = n-1:-1:1
    x(k) = (b(k)- c(k,3)*x(k+1))/c(k,2);
end
end % Thomas
```


28

Example 3:

$$y'' + (\sqrt{x} + 5)y' + (x - 1)y = \cos(x)\, e^{-\frac{x}{4}} + 1$$

$$y(0) = 1, \qquad y(1) = 0$$



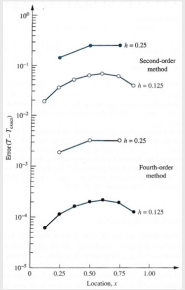
29

Accuracy of the second-order method

Example for the heat-transfer problem

$$T'' - \alpha^2 T = \alpha^2 T_e, \qquad T(0) = 100, \qquad T(1) = 0, \qquad T_e = 0$$

While the fourth-order methods demonstrates better accuracy it involves solving a penta-diagonal system of equations.



30

Derivative boundary conditions on the right

When the equilibrium method is used to solve a boundary-value problem with a derivative boundary condition, a finite difference procedure must be developed to solve for the value of the function at the boundary where the derivative boundary condition is imposed.

The finite difference approximation for Dirichlet boundary conditions

$$\left(1-\frac{\Delta x}{2}P_i\right)y_{i-1}+(-2+\Delta x^2 Q_i)y_i+\left(1+\frac{\Delta x}{2}P_i\right)y_{i+1}=\Delta x^2 F_i$$

Assume that point n is the last point of interest, and the $n+1$ is the right boundary point.

With the Dirichlet boundary condition we have y_{n+1}, now we have y'_{n+1}.

How to update the FDA for given y'_{n+1}?

31

From y'_{n+1} to y_{n+1}

We can use Newton's backward difference polynomial for the first derivative at $n+1$ to connect it to the value of the function at the same point.

1. First-order

   $$y_{n+1}=y_n+\Delta x y'_{n+1}$$

   this is $O(\Delta x)$ accuracy, but if the solution is well behaved then there is no problem
2. Second-order

   $$y_{n+1}=\frac{1}{3}(4y_n-y_{n-1}+2\Delta x y'_{n+1})$$
3. Third-order

   $$y_{n+1}=\frac{1}{11}(18y_n-9y_{n-1}+2y_{n-2}+6\Delta x y'_{n+1})$$

Generally the second-order method provides the best results since it's accuracy is the same as the second-order FDA – same order $(\Delta x)^2$

32

Derivative boundary conditions on the left

On the left side we can use Newton's forward difference polynomial for the first derivative at $i=0$ to connect it to the value of the function at the same point.

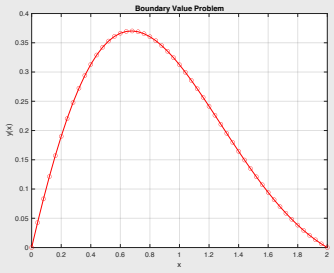
Good practice exercises:

1. Update the FDA system of equations when the derivative boundary condition is given for the right boundary point.
2. Update the FDA system when the derivative condition is given for the left boundary point.

33

Example 4a: Dirichlet boundary condition

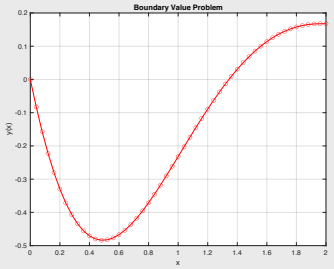
$$y''+2y'+5y=e^{-x},$$
$$y(0)=0,\qquad y(2)=0$$



34

Example 4b: The derivative condition on the right

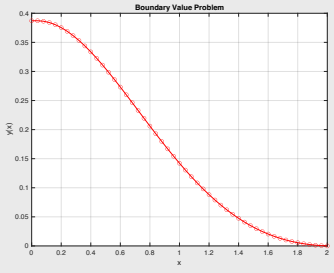
$$y''+2y'+5y=e^{-x},$$
$$y(0)=0,\qquad y'(2)=0$$



35

Example 4c: The derivative condition on the left

$$y''+2y'+5y=e^{-x},$$
$$y'(0)=0,\qquad y(2)=0$$



36

Boundary conditions at infinity

Occasionally one boundary condition is given at infinity.

In such a case, the boundary conditions might be

$$y(0) = A, \qquad y(\infty) = B.$$

There are two procedures for implementing boundary conditions at infinity: finite domain and asymptotic solution

37

37

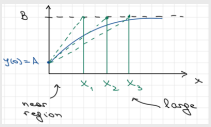
Boundary conditions at infinity

Finite domain:

In this approach, the boundary condition at $x = \infty$ simply replaced by the same boundary condition applied at a finite location, $x = X$.
Thus $y(\infty) = B \rightarrow y(X)$.

The major problem with this approach is determining what value of X, if any, yields a reasonable solution to the original problem.

In most cases, our interest is in the near the region far away from infinity. In that case, successively larger values of X, denoted by X_1, X_2, etc., can be chosen, until successive solutions in the region of interest change by less than some prescribed tolerance.



38

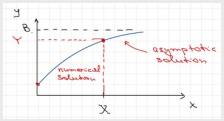
38

Boundary conditions at infinity

Asymptotic solution:

A second approach for implementing boundary conditions at infinity is based on an asymptotic solution for large values of x. In many problems, the behavior of the solution near $x \rightarrow \infty$ is much simpler than the behavior in the near region, and the simplified differential equation can be solved exactly, including the boundary condition at infinity, to yield the solution $y_{asymp.}(x) = F(x)$.

The boundary condition for the solution of the original differential equation is determined by choosing a finite location, $x = X$, and using it as boundary condition at X as $F(X) = Y$



39

39

Nonlinear ODEs

While the shooting method works well for both linear and non-linear ODE, the equilibrium method is only practical for linear ODE.

Otherwise we need to solve a system of nonlinear FDA equations using, for example, the Newton's method. In this case we need to have a good initial guess.

40

40

Summary for the equilibrium method

Pro

- Boundary conditions are automatically satisfied
- The method is good for complicated or delicate boundary conditions

Con

- A system of FDA equations should be solved
- Achieving higher than second-order accuracy demands solving a system of many FDA equations
- Non-linear ODEs yields a system of non-linear FDA equations (hard to solve)
- The method needs special handling for non-uniform grids.

41

41

Part 4:

Eigenvalue problem

42

Major methods for the boundary value problem

Eigenproblems arise in equilibrium problems in which the solution exists only for special values (i.e., eigenvalues) of a parameter of the problem.

Eigenproblems occur when homogeneous boundary-value ODEs also have homogeneous boundary conditions.

The eigenvalues are to be determined in addition to the solutions.

Example:

$$\frac{d^2y}{dx^2} + k^2 y = 0, \qquad y(0) = 0, \qquad y(1) = 0$$

The solutions for $y(x)$ exist only for $k = \pm n\pi,\ n = 1, 2, \ldots$

There are two principal methods for solving eigenproblems

1. Equilibrium method (most general)
2. Shooting method (less powerful than equilibrium methods, but works well with higher accuracy for some problems in physics)

43

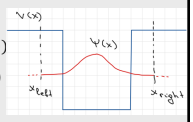
More on the stationary Schrodinger equation

Stationary Schrodinger equation

$$-\frac{\hbar^2}{2m}\frac{d^2\psi(x)}{dx^2} + (V(x) - E)\psi(x) = 0$$

Two principal types of solutions

Type 1: Bound states - a particle is bound, i.e. it is confined to some finite region of space.



For short-range potentials*: $\psi(x) \to e^{-\sqrt{2|E|}}\ (x \to \infty)$

For large $|x|$ we can set $\psi(x_{left}) = 0, \psi(x_{right}) = 0$

*A short-range potential $V(x) \sim C/x^{2+\mu}\ (\mu \geq 0)$.
A Coulomb potential is a long-range potential with different asymptotic behavior.

Type 2: Continuum states (oscillating wave functions)

We will concentrate on solutions for bound states.

44

Part 4:

Methods based on FDM

45

Part 4a:

Equilibrium method

46

The equilibrium method for Schrodinger equation

While the equilibrium method can be easily applied to a general linear ODE, we will concentrate on solving the stationary Schrodinger equation

Using atomic units (also called Hartree units) we can write

$$\frac{d^2y}{dx^2} + 2[E - V(x)]y = 0$$

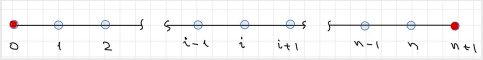
For bound states with imply the homogeneous boundary conditions

$$y(x_{left}) = 0, \qquad y(x_{right}) = 0.$$

47

Using the finite difference approximation

Discretizing the continuous domain into a discrete finite difference grid



and using the second-order central difference for the derivative

$$\frac{d^2y}{dx^2} \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$$

we have for the stationary Schrodinger equation

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + [E - V(x_i)]y_i = 0$$

with the homogeneous boundary conditions

$$y_0 = 0, \qquad y_{n+1} = 0.$$

48

Transforming to the eigenvalue problem

Rearranging the terms we get classical eigenvalue problem $Ax = \lambda x$ in linear algebra

$$-\frac{y_{i+1}}{2h^2} + y_i\left(\frac{1}{h^2} + V(x_i)\right) - \frac{y_{i-1}}{2h^2} = Ey_i$$

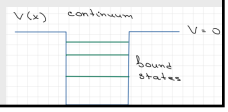
where the diagonal elements are $d_{ii} = \frac{1}{h^2} + V(x_i)$

and non-diagonal elements $a_{i-1} = a_{i+1} = -\frac{1}{2h^2}$

Then we can use one of methods for solving the eigenvalue problem to find values of E_i $(i = 1, n)$ and corresponding eigenfunctions.

Note that n is the number of grid points.

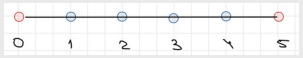
Attention: Solutions for $E_i < V(x)|_{x\to\infty}$ corresponds to bound states. The rest represents pseudo-continuum states*.



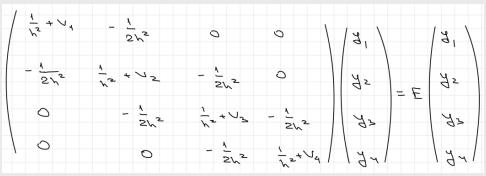
49

Example for n=4

Assume that we need to find a solution using this grid



with $y_0 = 0$ and $y_5 = 0$. Then the system is



We can use the Jacoby method for symmetric matrices, or QR method, or a method from well-established numerical library

50

50

Example: MatLab code

```
function [x,y,Ei,nstates] = QMSch01(Vp,Xmin,Xmax,n)
% ============================================================
% Solver for stationary Schrodinger equation
% Version: November 2021
% ============================================================
%{
IN:
    Vp      - external function (potential)
    Xmin    - left end-point
    Xmax    - right end-point
    n       - number of endpoint
OUT:
    x       - set of grid points
    y       - eigenvectors corresponding to eigenvalues
    Ei      - diagonal matrix of eigenvalues
    nstates - number of bound states
%}
% 1: preparation
h = (Xmax - Xmin)/(n-1);
a = zeros(n,n);
x = zeros(n,1);
S = zeros(n,1);
Vplot = zeros(n,1);
```

51

51

Example: MatLab code

```
% 2: Calculate the matrix a(i,j)
% grid points and the diagonal elements
for k = 1:n
    x(k) = Xmin + h*(k-1);
    a(k,k)   = 1.0/h^2 + Vp(x(k));
    Vplot(k)   = Vp(x(k));
end
%  +/- non-diagonal elements
for k=1:n-1
    a(k,k+1) = (-1.0)/(2*h^2);
    a(k+1,k) = a(k,k+1);
end

% 3. Calculating eigenvalues and eigenvectors
% === solver - eigenvalues and eigenvector (Matlab function)
[y,Ei] = eig(a);
% === end solver

% 4. sorting eigenvalues (and eigenvectors) in in ascending order
% attention - Matlab function eig may not always sort
[dwork,ind] = sort(diag(Ei));
EiSort = Ei(ind,ind);
ySort  = y(:,ind);
```

52

52

Example: MatLab code

```
Ei = EiSort;
y  = ySort;
% end of sorting

% 5. count bound states
Emax = Vp(Xmax);
nstates = 0;
for k = 1:n
    if Ei(k,k) > Emax
        break
    end
    nstates = nstates+1;
end
```

53

53

Example: MatLab code

```
% 6. Normalization
% Normalization integral so that |y(x)|^2 dx =1 (method: Simpson)
% note: normalization for the first m functions
for i = 1:nstates
    S(i) = 0.0;
    for k = 2:2:n-1
        S(i) = S(i) + 4.0*y(k,i)*y(k,i);
        S(i) = S(i) + 2.0*y(k+1,i)*y(k+1,i);
    end
    S(i) = S(i) + y(1,i)*y(1,i) - y(n,i)*y(n,i);
    S(i) = sqrt(S(i)*h/3.0);
end
% Normalization
for i=1:nstates
    for k=1:n
        y(k,i) = y(k,i)/S(i);
    end
end
```

54

54

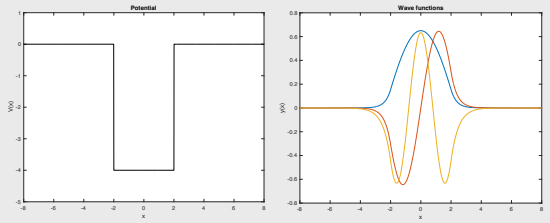
Example: MatLab code

```
% 7. plot the potential
Vmin = min(Vplot);
Vmax = max(Vplot);
figure (1)
plot(x,Vplot,'k','LineWidth',1.2);
title('Potential');
xlabel('x');
ylabel('V(x)');
ylim([Vmin-1. Vmax*1.2+1.])
end
```

55

55

Example 5a: Stationary Schrodinger equation

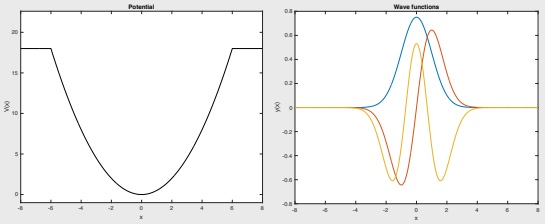



```
Energies of bound states
    i      Energy          Analytic solutions
    1    -3.778856         -3.77791
    2    -3.123640         -3.11995
    3    -2.065921         -2.05806
    4    -0.706421         -0.69467
```

56

56

Example 5b: Stationary Schrodinger equation

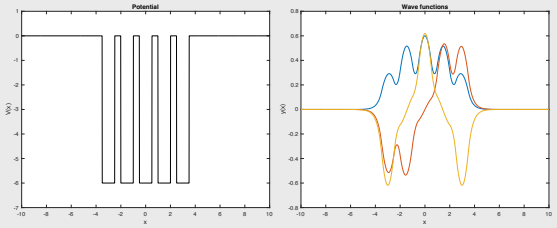



```
first 7 out of   18 states
    i      Energy
    1     0.499997
    2     1.499984
    3     2.499959
    4     3.499922
    5     4.499872
    6     5.499809
    7     6.499734
```

57

57

Example 5c: Stationary Schrodinger equation

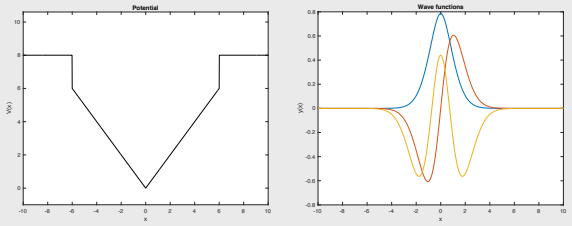



```
first 7 out of   7 states (note that eigenlevels are in two bands)
    i      Energy
    1    -4.543076
    2    -4.394130
    3    -4.162973
    4    -3.885428
    5    -3.635264
    6    -0.496734
    7    -0.139594
```

58

58

Example 5d: Stationary Schrodinger equation




```
first 8 out of   12 states
    i      Energy
    1     0.808594
    2     1.855739
    3     2.578056
    4     3.244557
    5     3.825692
    6     4.382126
    7     4.895134
    8     5.401913
```

59

59

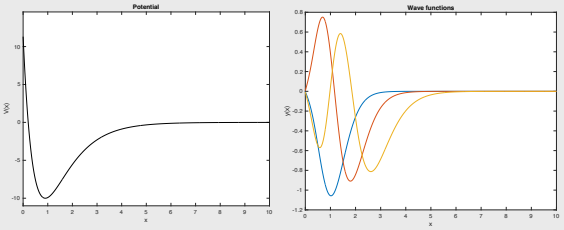
Observation

1. For a symmetric potential, $V(x) = V(-x)$ the solutions are either even or odd, that is, the wave function has definite parities.
2. Accuracy for states closer to the continuum is lower due to mixing with pseudo-continuum states

60

60

Example 5d: Stationary Schrodinger equation




```
first  4 out of    4 states
   i     Energy
   1    -7.872870
   2    -4.376419
   3    -1.895208
   4    -0.439901
```

* Morse potential

61

61

Part 4b:

Shooting method

62

62

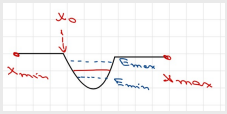
The shooting method for Schrodinger equation

The idea is close to the shooting method for a regular boundary value problem.

We choose some x_{min} and x_{max} as interval of integration of the ODE.

We set the boundary conditions as

$y(x_{min}) = 0, \quad y(x_{max}) = 0$



Since for bound states asymptotically $y(x) \sim e^{-\mu|x|}$ then for $y'(x) \sim \mu e^{-\mu|x|}$ where $\mu = \sqrt{2E}$ (E is the energy)

Attention (watch the signs of the derivatives for even and odd states)

Then we guess a value of E and integrate from x_{min} to x_{max}, and check if we satisfy the boundary condition on the right. And we keep adjusting E till it works.

63

63

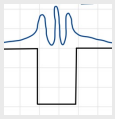
Serious complication

There are two mathematical solutions at $x \to \infty$,

$$y(x) \sim Ae^{-\mu x} + Be^{\mu x}$$

one is exponentially decaying (physical solutions), and the second solution is exponentially increasing (numerically dominant).

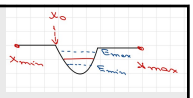
A small error (accuracy or round-off) will be exponentially magnified.



64

64

Algorithm for the shooting method



1. Choose x_{min}, x_{max} and some x_0
2. Guess some E_{min} and E_{max} (looking for E in between)
3. And solve the ODE for E and E_{max}
   a) from the left: x_{min} to x_0 to get $y_l(x_0)$ and $y_l'(x_0)$
   b) then from the right: x_{max} to x_0 to get $y_r(x_0)$ and $y_r'(x_0)$
4. We want a smooth connection between the two solution, therefore instead of the logarithmic derivatives $y_l'(x_0)/y_l(x_0) = y_r'(x_0)/y_r(x_0)$ it is preferable to use the difference in logarithmic derivatives as

$$\Delta = \frac{y_l'(x_0)/y_l(x_0) - y_r'(x_0)/y_r(x_0)}{y_l'(x_0)/y_l(x_0) + y_r'(x_0)/y_r(x_0)}$$

5. If $\Delta_E \cdot \Delta_{E_{max}} > 0$ then $E_{max} = E$ otherwise $E_{min} = E$
   and repeat steps 1-4 till given tolerance.

Note: The Numerov algorithm is a very good choice for solving ODEs

65

65

Example: MatLab code

```
function [Ei] = Qwell02(Xmin,Xmax,X0,N,Emin,Emax,eps,state)
%{
Solving stationary Schrodinger equation
y'' + 2(E-V(x))y = 0
METHOD: shooting method + Runge-Kutta 4th as initial value problem
CALLS: rk4_2d(x,y,dy,n), ypp(x,y,dy), V(x) (potential)
INPUT:
  V(X) - a potential (as a function, so far the name is fixed)
  XMIN, XMAX - left and right endpoints for integration
  X0 - "meeting point" (results should not depend on it)
  N  - number of points for the whole interval
  Emin, Emax - energy interval for searching an energy level
  eps - tolerance on matching the log derivative at X0
  state - parity (+1 for odd states and -1 for even states
OUTPUT:
  Ei - eigenvalue (energy)
Additional output (in the function)
Plot 1: matching (left/right) wavefunctions and derivatives
Plot 2: Wavefunction (normalized)
%}
global Ee
MaxIter = 128;            % Max number of iterations
% Working arrays
Delta = zeros(MaxIter,1);
E     = zeros(MaxIter,1);
Vp    = zeros(N,1);
```

66

66

Example: MatLab code

```
% Phase 1: Prepare arrays x,y xl,yl, xr,yr(left and righ)
h = (Xmax - Xmin)/(N-1);
N0 = ceil((X0-Xmin)/h) + 1;
Nl = N0;
Nr = N - N0 +1;

% reserve arrays
x  = zeros(N,1);
y  = zeros(N,1);

xl  = zeros(Nl,1);
yl  = zeros(Nl,1);
dyl = zeros(Nl,1);

xr  = zeros(Nr,1);
yr  = zeros(Nr,1);
dyr = zeros(Nr,1);

% grid points for x, xl and xr
for k = 1:N
    x(k) = Xmin + h*(k-1);
end
for k=1:N0
    xl(k) = x(k);
end
for k=N:-1:N0
    kr = N-k+1;
```


67

Example: MatLab code

```
% Phase 1: Prepare arrays x,y xl,yl, xr,yr(left and righ)
h = (Xmax - Xmin)/(N-1);
N0 = ceil((X0-Xmin)/h) + 1;
Nl = N0;
Nr = N - N0 +1;

% grid points for x, xl and xr
for k = 1:N
    x(k) = Xmin + h*(k-1);
end
for k=1:N0
    xl(k) = x(k);
end
for k=N:-1:N0
    kr = N-k+1;
    xr(kr) = x(k);
end

% Phase 2: Prepare first three energies (initialization)
E(1) = Emin;
E(2) = Emax;
E(3) = (Emin+Emax)/2;
```


68

Example: MatLab code

```
% Phase 3: the main loop (iterations)

for k = 1:MaxIter
    Ee = E(k);
    b = sqrt(2.0*abs(Ee));

%3a Integration from the left to X0
    % initial position and the derivative on the left
    yl(1)  = exp(-b*abs(xl(1)));
    dyl(1) = b*yl(1);
    [yl,dyl] = rk4_2d(xl,yl,dyl,Nl);

%3b Integration from the right to X0
    % initial position and the derivative on the right
    yr(1) = state*exp(-b*abs(xr(1)));
    dyr(1) = (-1.0)*b*yr(1);
    [yr,dyr] = rk4_2d(xr,yr,dyr,Nr);

%3C Calculating the log derivative difference between the two solutions
    Lelt  = dyl(N0)/yl(N0);
    Right = dyr(Nr)/yr(Nr);
    Delta(k) = (Lelt-Right)/(Lelt + Right);
```


69

Example: MatLab code

```
%3d when k=2 check if there is a solution
    if k == 2
        if Delta(1)*Delta(2) > 0
            fprintf(' No solution for given Emin and Emax \n')
            Ei = 0.0;
            return
        end
    end
%3E bisectional search for the root
% Attention: the bisectional method may converge not only to a root
% of a function f(x)=0 but to a discontinuity, then change the matching
% point X0.
    if k >= 3
        if Delta(1)*Delta(k) < 0.0
            E(2) = E(k);
        else
            E(1) = E(k);
            Delta(1) = Delta(k);
        end
        E(k+1) = (E(1)+E(2))/2.0;
        if abs(Delta(k)) < eps
%        if abs(E(1)-E(2)) < eps
            Ei = E(k);
            break
        end
    end   % end bisectional search
end % end iterations
```


70

Example: MatLab code

```
if k == MaxIter
    fprintf(' Max iterations - still no solution \n')
    Ei = 0.0;
    return
end

figure (1)
plot(xl,yl,'b',xr,yr,'r',xl,dyl,'-m',xr,dyr,'-k')
title('Matching functions and derivatives')
grid

% Phase 4: Assembling the whole wave function + normalization

% 4a  One function y(x) as a sum of two solutions (left + right)
Ynorm = yr(Nr)/yl(N0);
for k = 1:N0
    x(k) = xl(k);
    y(k) = yl(k)*Ynorm;
end
for k = 1:Nr
    x(N-k+1) = xr(k);
    y(N-k+1) = yr(k);
end
```


71

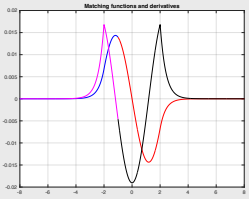
Example: MatLab code

```
% 4b  Calculating Integral |y(x)|^2dx for normalization
Sn = 0.0;
for k = 2:2:N-1
    Sn = Sn + 4.0*y(k)*y(k);
    Sn = Sn + 2.0*y(k+1)*y(k+1);
end
Sn = Sn + y(1)*y(1) - y(N)*y(N);
Sn = sqrt(Sn*(h/3.0));
% 4c Normalization
for k = 1:N
    y(k) = y(k)/Sn;
    Vp(k) = V(x(k));
end

figure (2)
plot(x,y,'r')
%plot(x,y,'r',x,Vp,'b')
str = sprintf('Wave function for Ei = %6.4f',Ei);
title(str);
grid

end %end function
```


72

Example 6: Potential well

```
function Vx = V(x)
a = 2.0;
if abs(x) <= a
    Vx = -4.0;
else
    Vx = 0.0;
end
```



i	Analytic	FDM	Shooting
1	-3.77791	-3.778856	-3.778227
2	-3.11995	-3.123640	-3.121178
3	-2.05806	-2.065921	-2.060667
4	-0.69467	-0.706421	-0.698546

73

73

The Numerov method

While Runge-Kutta methods (RK- 4th order or RKF45) works very well for solving ODEs, there is a powerful method for solving second-order ODEs that don't have first derivative.

We consider equation

$$\frac{d^2y}{dx^2} + k^2(x)y = S(x)$$

The power of the Numerov method is to get extra precision in the second derivative by taking advantage of there being no first derivative in equation above.

74

74

The Numerov method

The Taylor series for the function $y(x)$ at the points $i+1$ and n $i-1$

$$y_{i+1} = y_i + y_i'h + \frac{1}{2}y_i''h^2 + \frac{1}{6}y_i'''h^3 + \frac{1}{24}y_i''''h^4 + \cdots$$

$$y_{i-1} = y_i - y_i'h + \frac{1}{2}y_i''h^2 - \frac{1}{6}y_i'''h^3 + \frac{1}{24}y_i''''h^4 - \cdots$$

$$y_{i+1} + y_{i-1} = 2y_i + y_i''h^2 + \frac{1}{12}y_i''''h^4 + \cdots$$

Then

$$y_i'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - \frac{1}{12}y_i''''h^2$$

At the same time from the differential equation $\frac{d^2y}{dx^2} + k^2(x)y = S(x)$

$$y_i'''' = \frac{d^2}{dx^2}(-k^2(x)y + S(x))\Big|_{x=x_i}$$

75

75

The Numerov method

Using the second-order central difference for the second-order derivatives for $-k^2(x)y + S(x)$ we have

$$y_i'''' = -\frac{(k^2y)_{i+1} - 2(k^2y)_i + (k^2y)_{i-1}}{h^2} + \frac{S_{i+1} - 2S_i + S_{i-1}}{h^2}$$

then

$$y_i'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - \frac{1}{12}y_i''''h^2 =$$

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + \frac{1}{12}h^2\left[\frac{(k^2y)_{i+1} - 2(k^2y)_i + (k^2y)_{i-1}}{h^2} + \frac{S_{i+1} - 2S_i + S_{i-1}}{h^2}\right]$$

and the equation $y'' = -k^2(x)y + S(x)$ reads

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + \frac{1}{12}h^2\left[\frac{(k^2y)_{i+1} - 2(k^2y)_i + (k^2y)_{i-1}}{h^2} + \frac{S_{i+1} - 2S_i + S_{i-1}}{h^2}\right]$$
$$= -(k^2y)_i + S_i$$

Rearranging the terms we have

76

76

The Numerov method

$$y_{i+1}\left(1 + \frac{h^2}{12}k_{i+1}^2\right) - 2y_i\left(1 - \frac{5h^2}{12}k_i^2\right) + y_{i-1}\left(1 + \frac{h^2}{12}k_{i-1}^2\right)$$
$$= \frac{h^2}{12}(S_{i+1} + 10S_i + S_{i-1}) + O(h^6)$$

We see that the Numerov method uses the values of $y(x)$ at the two previous steps x_i and x_{i-1} to move y forward to x_{i+1}.

The Numerov methods is a three-point recursion relation.

It is stable and has a local error $\sim O(h^6)$ the same as RKF45. We need six calls for RKF45 and only one call for the Numerov method.

77

77

The Numerov method

Two issues:

1) the method is not self-starting, however, we can use the asymptotic behavior
2) the method does not provide first derivatives on its own. But we need them when matching the wave functions. We can calculate the first derivative using the central difference formula or more precisely

$$y_i' = \frac{1}{2h}\left[\left(1 + \frac{h^2}{12}k_{i+1}^2\right)y_{i+1} - \left(1 + \frac{h^2}{12}k_{i-1}^2\right)y_{i-1}\right] + O(h^4)$$

Summary for the Numerov method:

The speed gain for shooting with Numerov's method is significant. We can use it to extend calculations to systems requiring large number of grid points.

78

78

Finite difference approximation or the Shooting method?

1. The shooting method results are more accurate than the FDM results, especially for states closer to continuum.
   Note that the accuracy either RK45 or Numerov's is higher than second-order central difference for FDM
2. The shooting method can be used for non-homogeneous boundary conditions
3. But the FDM is much simpler to use and can produce bound states en masse.

79

79

Part 5

Other methods (not based on FDM)

80

80

More methods I: Final Element Method

Final element method - very powerful for solving Partial Differential Equations.

It can be used for solving Schrodinger equation too.
FEM breaks space up into multiple geometric objects (elements), determine approximate solution for each element, and then match the solutions up at the element edges.
Much more powerful than FDM but MUCH more work required.

81

81

More methods II: Basic expansion method

The idea – expand the unknown function on a finite basis set

The method is very popular for structure calculations in multi-electron systems. It's often called as Configuration Interaction method.

There are very many variants of the method.

82

82

More methods III: Variational methods

The idea – the exact wavefunction gives the lowest energy for the ground state

$$E_0 = \frac{\langle \psi_0(r) | \hat{H} | \psi_0(r) \rangle}{\langle \psi_0(r) | \psi_0(r) \rangle}$$

The variational method can be adapted to give bounds on the energies of excited states (under certain conditions).

There are many versions of the method: Hartree-Fock method, Variational Monte-Carlo method

83

83

Variational Monte Carlo method

The objective is finding $\psi(x)$ that minimize

$$E_0 = \frac{\langle \psi_0(x) | \hat{H} | \psi_0(x) \rangle}{\langle \psi_0(x) | \psi_0(x) \rangle} = \frac{\left\langle \psi_0(x) \left| -\frac{1}{2}\nabla^2 + V(x) \right| \psi_0(x) \right\rangle}{\langle \psi_0(x) | \psi_0(x) \rangle}$$

Steps:

1. Choose a trial function $y_t(x)$ and discretize space into bins Δx size
2. Choose randomly a "bin i" (or x_i value) and create a provisional function $y_p(x)$ by changing $y_t(x)$ function in x_i location by an amount chosen randomly $\pm dy_k$ using Monte Carlo
3. Calculate E_p. If it is lover that E_t with y_t then accept the provisional function, if it is higher that E_t then discard the provisional function
4. Keep doing 2 and 3 till desired tolerance is reached

Note: Use Metropolis method to accept/reject solutions with $E_p > E_t$

84

84

Example: MatLab code

```
% !====================================================================
% ! Monte Carlo interactive procedure to minimize energy
% ! by varying randomly f(i) and random point x(i)
% !====================================================================
function [f,energy, hits] = norfolk(x,f,df,ei,n,tests)

hits = 0;

for i = 1: tests
  %k = 2 + floor(rand*(n-2));
  k = 1+ randi(n-3);
  fold = f(k);
  f(k) = f(k) + 2.0*(rand-0.5)*df;
  ef =  hamilton(x,f,n);
  if ef < ei
      ei = ef;
      f = fnorm(x,f,n);
      hits = hits + 1;
      else
      f(k) = fold;
  end
end
energy = ei;
end % norfolk
```


85

Example: MatLab code

```
%================================================
% compute <f|H|f>/<f|f> for a given x(i) and f(i)
%================================================
function energy = hamilton(x,f,n)

energy = 0.0;
sum = 0.0;
dx = x(2)-x(1);

for i=2:n-1
  potential = V(x(i));
  energy = energy + dx*potential*f(i)*f(i);
  energy = energy - dx*(0.5/dx*dx)*f(i)*(f(i+1)-2.0*f(i)+f(i-1));
  sum = sum + f(i)*f(i)*dx;
end
energy = energy / sum;
end % hamilton
```


86

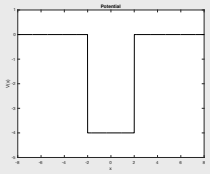
Example: MatLab code

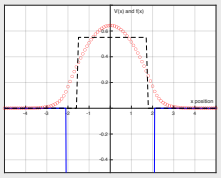
```
%================================================
% normalization, so that |<f|f>|**2 = 1.0
%================================================
function f = fnorm(x,f,n)

%f=zeros(n,1);

sum = 0.0;
dx  = x(2)-x(1);

for i=2:n-1
  sum = sum + dx*f(i)*f(i);
end

for i=2:n-1
  f(i) = f(i)/ sqrt(sum);
end

end
```


87

Example: Potential well

```
function Vx = V(x)
a = 2.0;
if abs(x) <= a
    Vx = -4.0;
else
    Vx = 0.0;
end

Analytic solutions: -3.77791

Final energy      -3.78169
Variations:        2*10^6
Success variations  0.007
```

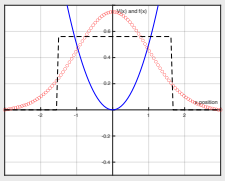




88

Example: Harmonic oscillator

```
Analytic solutions: 0.50000
Final energy        0.50039
Variations:         2*10^6
Success variations  0.012
```



The method has lower accuracy but it can be
1. accelerated by better guesses for trial wave functions
2. extended to multidimension (multiparticle) systems

89

More methods IV: Density Functional Theory

Nobel Prize 1998

The method gives energy levels (mostly ground state energies) without calculating wave functions!

90