nvkurbatova@sfedu.ru

Scientific Computer Software

(MatLab)

Topic 6. Class of sparse matrices – Sparse.

Topic 7. Effective methods for solving problems, nonlinear systems and equations.

Курбатова Наталья Викторовна, к.ф.-м.н.,
доцент кафедры математического моделирования, мехмат, ЮФУ

Contents:

Sparse

- The sparse matrix constructor. The connection with full matrix.
- Factorization of full and sparse matrices.
- Solving the SLAE with full and sparse matrices.
- Estimation of the algorithm solution speed.

Nonlinear systems (NS)

- Class Symbol
- Applied problems
- Nonlinear systems (high order)
- ML functions for solving NS

nvkurbatova@sfedu.ru

Class Sparse

Prerequisites :

1) Multiplication operations are expensive – and even by zero!
2) if **nnz(A)<n*m/2**; [n,m] =size(A) → **Sparse**
3) As a result of the discretization of boundary value problems, symmetric, tridiagonal, ribbon and substantially incomplete matrices of systems are obtained (for heterogeneous media – oh, what is it?)
4) ∃ effective algorithms for factorization of systems focused on vector, profile forms of matrix (systems) storage.
5) The ability to parallelize operations in ML in combination with **sparse** makes algorithms for large systems fast.
6) The functions of identifying non-zero matrix elements and converting the "structure" to sparse already existed:
7) [i, j, v] = find(A); element A**(**i(**1**), j(1)**)** → v(1)

   s=spconvert([i, j, v]), s∈Sparse

nvkurbatova@sfedu.ru

Sparse Consructors and Properties

s = sparse(A); s – Sparse → A – full

A = full(s); A – full → s – Sparse

issparse(s), isfull(A) – Type Control

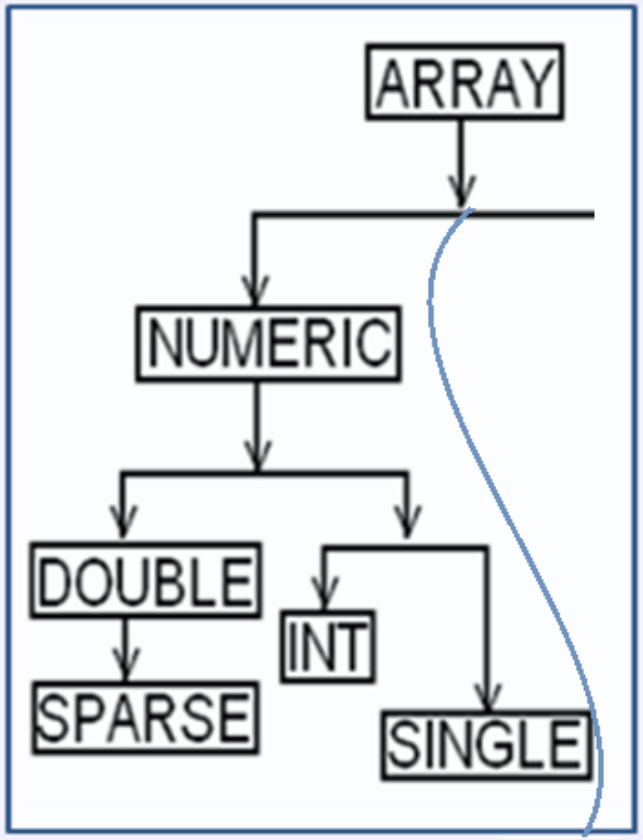
spones, speye, spdiags – constructors of sparse matrices of units, identity matrix and diagonal

sprandn, sprand – sparse matrix generator

S = spalloc(m,n,nz) – memory is reserved for the matrix **Sparse**

If D is a sparse matrix, then spconvert returns D!

Sparse matrices inherit methods of the Array, Numeric, Double classes (including arithmetic, logical, relations, matrix algebra, etc.)



nvkurbatova@sfedu.ru

Constructor **sprandsym**

- **R = sprandsym(S)** generates a symmetric matrix of the same structure as **S**
- **R = sprandsym(n,density)** , R - n-by-n, symmetric, density*n*n, density$\in$(0,1) - degree of sparsity
- **R = sprandsym(n,density,rc,kind)** - generate positiv matrix **R**

  rc – the number of conditionality (rc=norm(X,p) * norm(inv(X),p)), closeness to one means stability of the solution

  If **kind = 1**, **R** diagonal matrix **generates**.

  If **kind = 2**, for advanced problem programming programmers

Example for debugger: (let's feel like a debugger!):

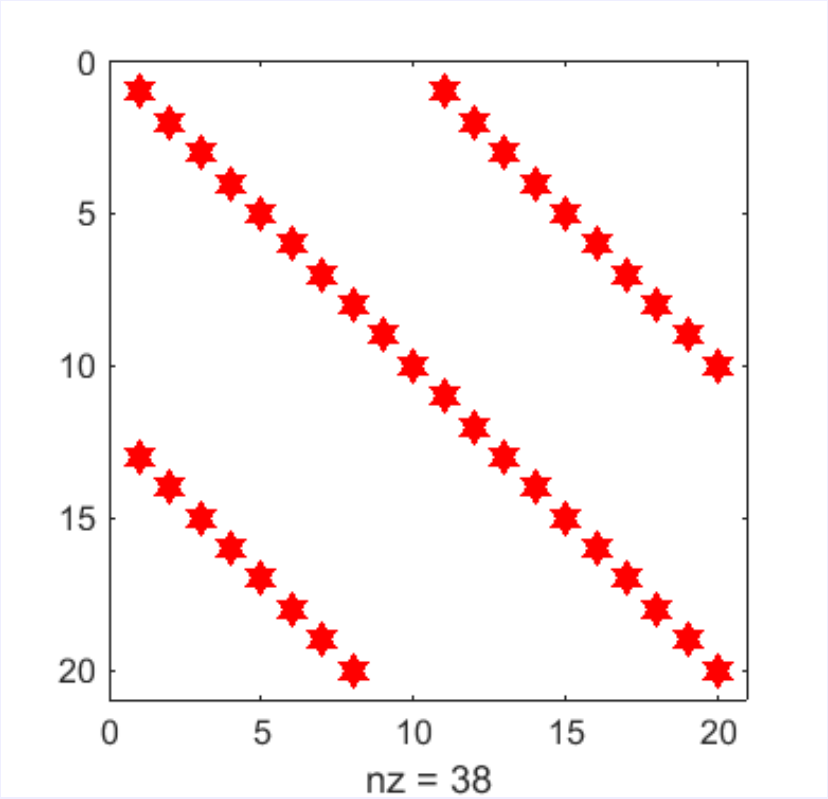
1) clear ; vu=rand(1,5); d=randn(1,6); vl=randn(1,5)+4;
2) S=diag(d)+diag(vu,1)+diag(vl,-1)
3) R = sprandsym(S,[],0.9,3)
4) spy(R); eig(R)
5) R==R'

nvkurbatova@sfedu.ru

Convert matrix to class Sparse?

Example:

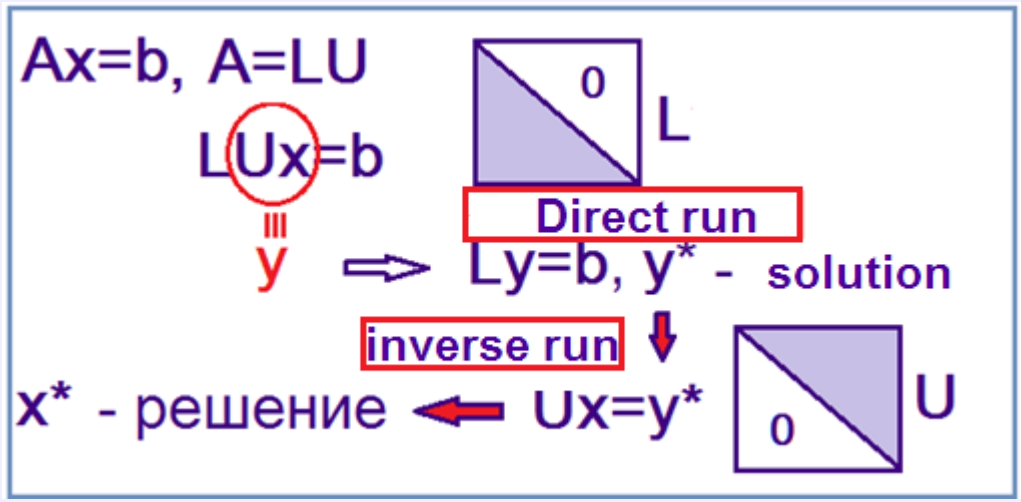
```
1)  b=randi([1,16],1,10);
2)  c=randi([1,16],1,20);
3)  l=randi([1,16],1,8);
4)  A=diag(c)...
5)    +diag(b,10)+diag(l,-12);
6)  nonzero=nnz(A)
7)  spy(A,'hr',12);
8)  set(get(gca,'children'),...
9)     'MarkerFaceColor','r');
10) [i, j, v] = find(A) ;
11) NumbersRow=feval('length',[i, j, v])
12) NumbersRow1=lenght([i, j, v]) % as 11)
13) sA=sparse(A) % converted in  Sparse
14) length(sA) %  equal to nnz!
15) B=A+diag(diag(A))
16) TrueFalse=all(A==B) %  Result ???
```



nvkurbatova@sfedu.ru

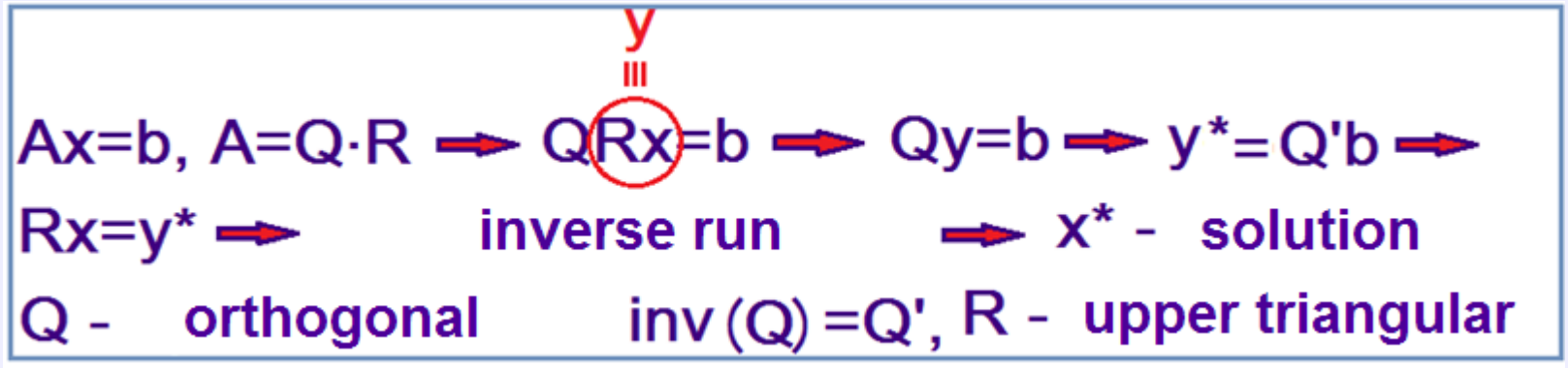
Motivation for matrix factorization

The matrix of the system is the product of the lower and upper triangular matrices:



Easy to solve!!!

The matrix of the system - product of orthogonal and upper triangular matrices:



Matrix factorization and SLAE solution

Functions are the same as for full matrices!

In earlier versions, the sparse matrix factorization functions were distinguished by the presence of sp* in the name, now they are not!

Factorization of full and sparse matrices :

	Description
[L,U]=lu(M)	**M=L·U, L, lower** triangular, **U** - upper triangular.
[Q,R]=qr(M)	**M=Q·R, R-**upper triangular, Q-orthogonal: inv(**Q**)=**Q'**,**Q·Q'**=eye(size(**Q**))
R=chol(M)	**R-**upper triangular., **M=R'·R, M –** positive **isequal(R'·R,M) %true!**

isequal(A,B,C) true – if all elements of marices A,B,C are equal to each other

Universal Solver

linsolve(A,B,opts)

A,B – the matrix and the column of free terms, respectively

opts.**UT** = true; opts.**LT** = true; (triangular matrices: L, U)

opts.**SYM** = true; (the matrix of the system is symmetric)

opts.**POSDEF** = true; (the matrix of the system is positive and symmetric)

Note that all options must be defined before finding the solution by linsolve(A,B,opts)

Evaluating the performance of algorithms

Functions for time calculation :

1) **tic** *operators* **toc** 2) **clock** operators **etime**

3) **cputime**

Example:

```
clear,   x = rand(900000, 1);

%%  the first approach - stopwatch:
%(preferably! Opinion exists))
% fft – Fast Fourier transform (of signal)
tstart = tic;  fft(x);  toc(tstart) % 0.043874 seconds

%% the second approach - clock:
t1 = clock;  fft(x);  etime(clock, t1) % 0.023; etime – difference

%% the third approach - cputime:
tstart=cputime; fft(x); cputime-tstart % 0.0625 seconds
```

```
clock :1.0e+03 *[2.0230    0.0030    0.0090    0.0220    0.0500    0.0437]
                year-2023   month-3   day -9   hour -22 minute-50 seconds-43 (7) ???
```

Class Symbol and methods, supported in ML

```
>> methods(sym)
```

Methods of Class Symbol:

abs	ezpolar	nnz
acos	ezsurf	nonzeros
acosh	ezsurfc	norm
acot	factor	not
acoth	factorial	null

...

Converting to symbolic variables

```
syms  var1 var2 var3 % и т.д.
```

Converting

Char → SYM

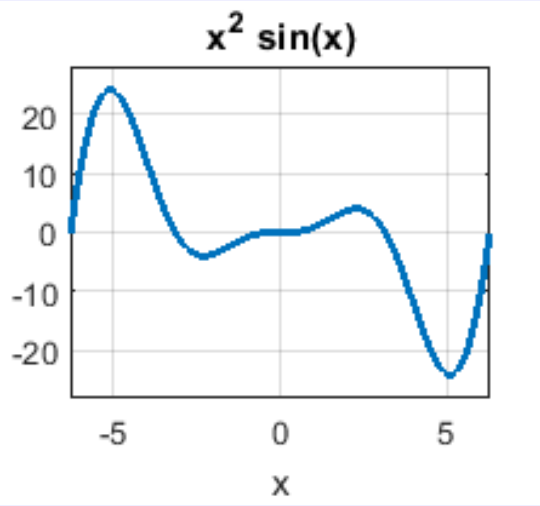
```
x=sym ('x')  % переменной
y=sym  (expressionChar)
```

Example

```
syms a x  y % separated by spaces!!!
y = sin(x) * x^2, t=ezplot(y)
set(t,'linewidth',2), grid on
a = expand(sym('(x+1)^2*(2*x+3)'))
% sym(Char) – It may be outdated
```

```
a =

2*x^3 + 7*x^2 + 8*x + 3
```



Step-by-step instructions for solving nonlinear equations and systems

I. Equations System Visualization

II. The choice of zero approximations and the solution using *fsolve*

III. Interactive selection of zero approximations using *ginput*

IV. Reducing the solution of a system of equations to the solution of a nonlinear equation by *fzero*

V. Finding the roots of polynomials, using *roots*

F(x)=0 and interactive selection of zero approximation or range:

[pointsX,pointsY]=ginput(n),

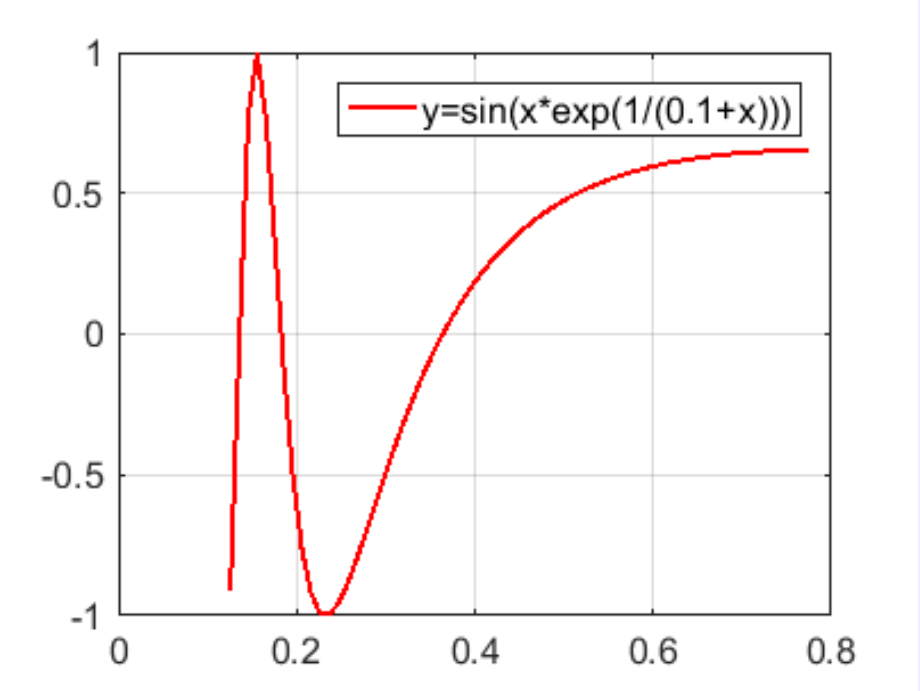
pointsX – n-dimension vector of abscissa points; pointsY – vector of ordinates points – zero approximation, coordinates are selected with the mouse

```
clear % F(x)=sin(x*exp(1/(0.1+x)))
set(0,'DefaultAxesFontSize',12,...
 'DefaultAxesFontName','Arial');
x=pi/25:0.01:pi/4;
y='sin(x.*exp(1./(0.1+x)))'
 plot(x,eval(y),'r-','linewidth',1.5)
 lg=legend('y=sin(x*exp(1/(0.1+x)))')
 set(lg,'fontsize',12), grid on,
[x0,y0]=ginput(3) % is waiting three zero approximation
sol=fsolve('sin(x.*exp(1./(0.1+x)))',x0)
```

x0 =	y0 =	sol =
0.1266	-0.2041	0.1359
0.1798	-0.1983	0.1826
0.3234	-0.2041	0.3639

Solution F(x) = 0

[sol,accuracy,exitSol,InDetail]=fsolve(F,x0,opts)



Solving a second-order nonlinear system ($F_1(x)=0$, $F_2(x)=0$)

1) To Solve the system - find the intersection points of the curves $F_1(x)=0$ and $F_2(x)=0$ intersection

```
fplot(@(x) sin(8*x).*x',[-1,1],'linewidth',2),
hold on
fplot(@(x) x.^5-x+0.5',[-1,1],'linewidth',2)
x0=[-0.9 0.2 0.8]              % visually
```

2) Find roots of $F_1(x)-F_2(x)=0$:

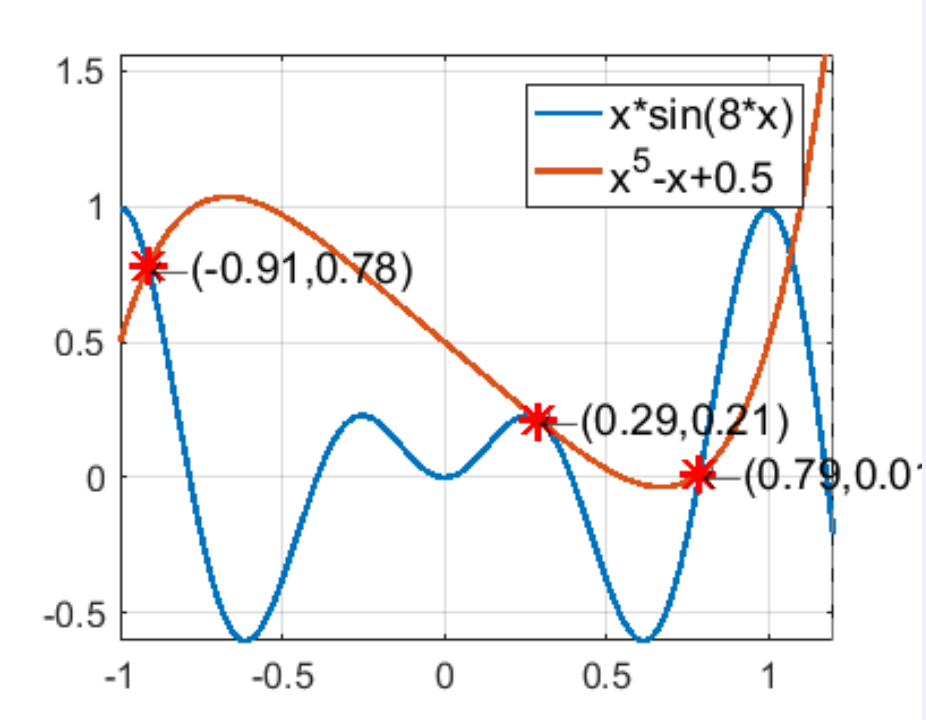
```
x=fsolve('x.*sin(8*x)-(x.^5-x+0.5)',x0)
```

3) Determine ordinates of intersection points:

```
y=eval('x.*sin(8*x)')
plot(x,y,'r*','linewidth',2)
```

4) Form labels:

```
for i=1:length(x)
   xy{i}=['(',num2str(x(i),2),',',num2str(y(i),2),')']
   text(x(i),y(i),['\leftarrow' xy{i}])
end
```



Why are we doing the step 2)?

How many strings are concatenated in a step 4)?

Roots of polynomials. Functions FZERO and ROOTS

!!! Consider the function yourself **fminbnd** –
finding the minimum value on a segment

Syntax:

sol=fzero(@(x) express, x0)

sol=fzero(Fun, x0)

```
%% fzero
x1=fzero(@(x) x^5+x^2-10*x-4.5,1.5)
x2=fzero(@(x) x^5+x^2-10*x-4.5,0.5)
x3=fzero(@(x) x^5+x^2-10*x-4.5,2)
      err=x1^5+x1^2-10*x1-4.5
%% roots
p=[1 0 0 1 -10 4.5]
allroots=roots(p)
polyval(p,allroots(4))
```

```
allroots =

  -1.9432 + 0.0000i
  -0.0327 + 1.7827i
  -0.0327 - 1.7827i
   1.5336 + 0.0000i
   0.4750 + 0.0000i

ans =

  -6.2172e-15
```

```
err = 0
```

allroots=roots(p) – are determined roots of polynomials

polyval(p,value) calculated the value of the polynomial at the points

p is the vector of coefficients of the polynomial

High accuracy of calculations by default for functions **roots** and **fzero**!!!

The algorithm of the simple iteration method

1. Задаём точность ε (≤0.001, по фантазии) ~ setting the accuracy

2. Преобразуем систему к нормализованному виду:

~ convert to the normal form:

$$X=\Phi(X), \text{ где } \Phi(X)=\begin{cases}\varphi_1(x_1, x_2, \dots, x_n)\\ \dots\\ \varphi_n(x_1, x_2, \dots, x_n)\end{cases}$$

$$\varphi_i = x_i + f_i, \qquad i = 1, 2, \dots, n.$$

Choosing the zero approximation

Выбираем начальное приближение (геометрически) $X^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})$

3. Вводим k – счетчик итераций~ introduce the iteration counter

4. Defining an iterative rule :

Определяем формулу итерационного процесса: $X^{(k+1)} = \Phi(X^{(k)})$

5. Calculating the k+1 approximation ~

Вычисляем $(k+1)$ приближение

6. Сравниваем $(k+1)$ приближение с предыдущим Compare

$$\max_{1<i<n} \left|x_i^{(k+1)} - x_i^{(k)}\right| < \varepsilon$$

Если условие выполнено, то решение найдено, если нет, то вычисляем следующее приближение (шаг 5) If the inequality holds, then the solution is with the specified accuracy!

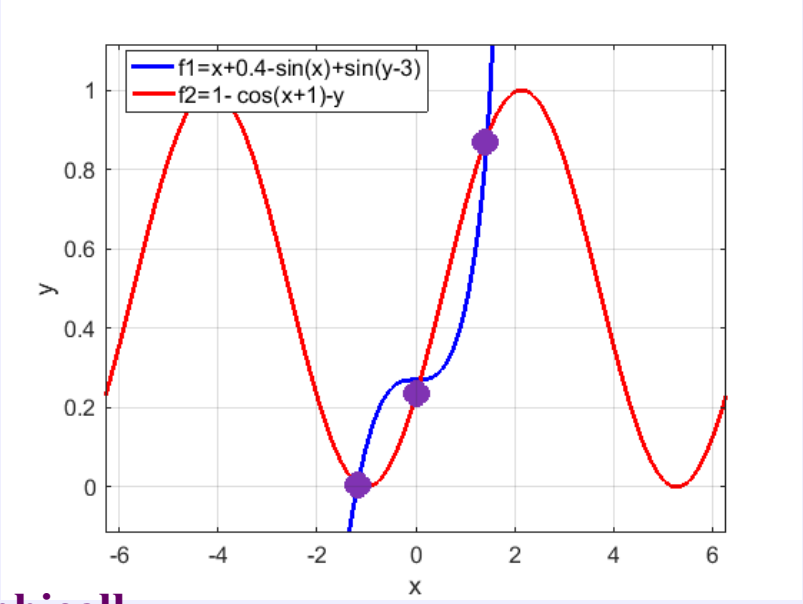
An example of solving a SNLE using simple iterations

Example

$$\begin{cases} 2y + \cos(x+1) = 1 \\ \sin(x) - \sin(y-3) = 0.4 \end{cases}$$

Normal system

$$\begin{cases} x = \varphi_1(x,y) = x + 0.4 - \sin(x) + \sin(y-3) \\ y = \varphi_2(x,y) = (1 - \cos(x+1))/2 \end{cases}$$



Programming using built-in functions:

1. Plot the functions of the original system graphically.
2. Visually identify zero approximations.
3. Set the accuracy of the solutionε
4. Enter a counter for the number of iterations (k=0 - initial)
5. To check whether the current solutions provide the specified accuracy
6. No, accuracy is not achieved – we change the number of iterations and use the current value as a zero approximation, repeat from step 5
7. Yes, accuracy is achieved, we celebrate the victory!

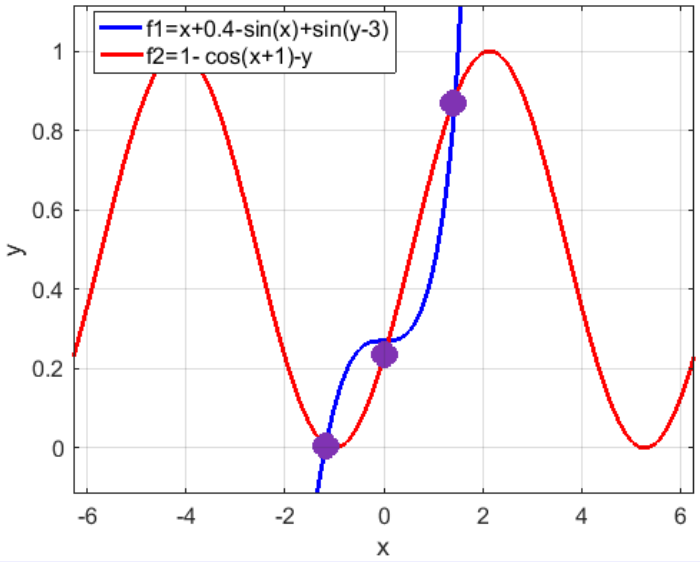
An example of solving nonlinear system using MatLab

Example

$$\begin{cases} 2y + \cos(x+1) = 1 \\ \sin(x) - \sin(y-3) = 0.4 \end{cases}$$

Normal system

$$\begin{cases} x = \varphi_1(x,y) = x + 0.4 - \sin(x) + \sin(y-3) \\ y = \varphi_2(x,y) = (1 - \cos(x+1))/2 \end{cases}$$



Программирование с использованием встроенных функций:

1. Plot the functions of the original system graphically.
2. Step1. Use ML functions: ezplot, fplot, implicitplot
3. For system solving use ML function fplot (set the equations of the system as an external function) with the Tolerance X: TolX, for example, with accuracy 10^{-3}:
   optimset('TolX',1.0e-3)
4. For clarity, edit the properties of the axes

An example of SNAE solution for high-order systems

```
Clear        % high order nonlinean System
x0=[0.9 0.9 0.9]
Fun=@FunNL
   options.StepTolerance = 0.0001;
   options.Display = 'iter'
[X,FVAL,EXITFLAG,output,jacobian] = fsolve(Fun,x0,options)

FVAL     %1.0e-07 *[-0.0001   -0.1225    0.0101]
EXITFLAG % 1 - system solved
output   % iterations: 3
         % funcCount: 16
         % algorithm: 'trust-region-dogleg'
         % firstorderopt: 3.6043e-08
         % message: 'Equation solved....'
```

```
% подфункция (где расположена?)
function F=FunNL(x)
F(1)=3*x(1)+x(2)-x(3)-1;
F(2)=sqrt(x(1))-x(2)+pi*x(3)-pi;
F(3)=x(1)+x(2)+x(3)^2-3
end
```

Explanation: Trust-region methods are stable, can be used, including for poorly conditioned systems, and have very decent convergence characteristics!

The method takes into account the step when searching for roots and the direction!!!

nvkurbatova@sfedu.ru

Thanks for your attention!

«No one will embrace the immensity! » - Kozma P. Prutkov
(L. M. Zhemchuzhnikov, A. E. Beideman, L. F. Lagorio)!

Remind me of Eugene Azhar and Romain Gary!