



Пакеты научных вычислений

Лекция 5.

Класс разреженных матриц – Sparse. Эффективные решения СЛАУ, нелинейных систем и уравнений.

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



Содержание

Sparse

- Конструктор разреженных матриц. Связь с полной.
- Факторизация полных и разреженных матриц.
- Решение СЛАУ полных и разреженных систем.
- Оценка скорости решения алгоритмов.

Нелинейные системы (НС)

- Класс *Symbol*
- НС второго порядка. Прикладные задачи.
- Нелинейные системы высоких порядков. Методы решения в ML. Метод Ньютона.



Класс разреженных матриц. Sparse

Предпосылки создания Sparse:

- 1) Операции умножения затратны – да ещё и на ноль!
- 2) если $\text{nnz}(A) < n * m / 2$; $[n, m] = \text{size}(A) \rightarrow$ переходим в **sparse**
- 3) В результате дискретизации краевых задач получаются, в т.ч. симметричные, трехдиагональные, ленточные и существенно неполные матрицы систем (для разнородных сред – **ой, что это?**)
- 4) \exists эффективные алгоритмы факторизации систем, ориентированные на векторные, профильные формы хранения матриц (систем).
- 5) Возможность распараллеливания операций в ML в сочетании со **sparse** делает быстрыми алгоритмы для больших систем.
- 6) Функции идентификация ненулевых элементов матрицы и конвертирования «структуры» в **sparse** уже существовали:
 $[i, j, v] = \text{find}(A)$; элемент $A(i(\mathbf{1}), (j(\mathbf{1})))$ становится элементом $v(\mathbf{1})$
 $s = \text{spconvert}([i, j, v])$, $s \in \text{Sparse}$



Свойства и конструкторы объектов Sparse

s = sparse(A) – конструктор разреженной матрицы,
A – полная матрица

A = full(s) – конвертирование разреженной матрицы
в полную

issparse(s), isfull(A) – контроль типов

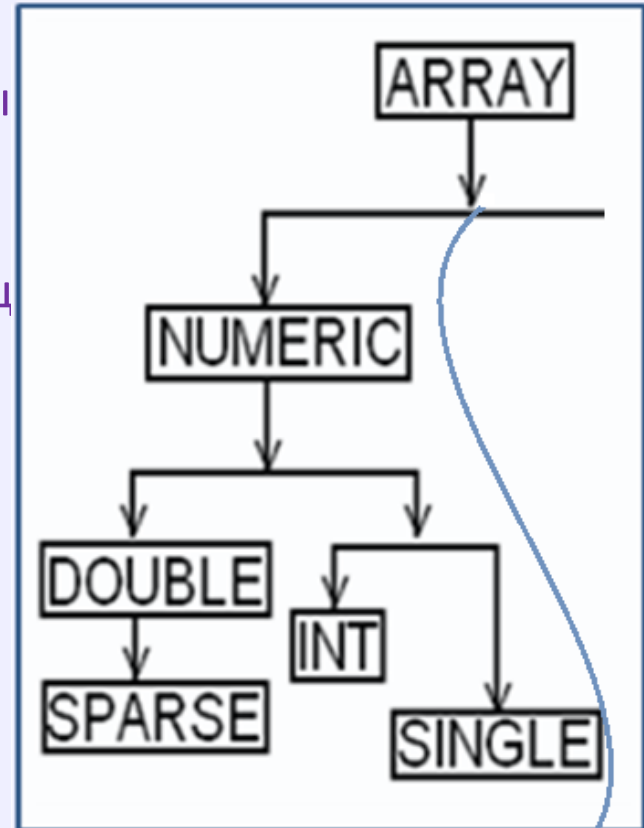
spones, speye, spdiags – конструкторы sparse-матриц
из единиц, единичной и диагональной

sprandn, sprand – генераторы sparse-матриц
нормально и равномерно распределенных на [0,1]
матриц

S = spalloc(m,n,nz) – выделяется память разрежен-
ной матрице S

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

Sparse матрицы наследуют методы классов Array,
Numeric, Double (в т.ч. операции арифметические,
логические, отношения, матричной алгебры и т.д.)





Конструктор `sprandsym` – факультативно!

- $R = \text{sprandsym}(S)$ генерирует симметричную матрицу, той же структуры, что S
- $R = \text{sprandsym}(n, \text{density})$, R - n -by- n , симметричная, степень разреженности, $\text{density} * n * n$, $\text{density} \in (0, 1)$
- $R = \text{sprandsym}(n, \text{density}, \text{rc}, \text{kind})$ - генерируется положительно определенная (ПО) R
 rc – число обусловленности ($\text{rc} = \text{norm}(X, p) * \text{norm}(\text{inv}(X), p)$), чем ближе к единице, тем устойчивее система (решение)
If $\text{kind} = 1$, R генерируется (ПО) матрица диагональной структуры.
If $\text{kind} = 2$, для продвинутых программистов проблемного программирования

Пример для отладчика: (почувствуем себя as a debugger!):

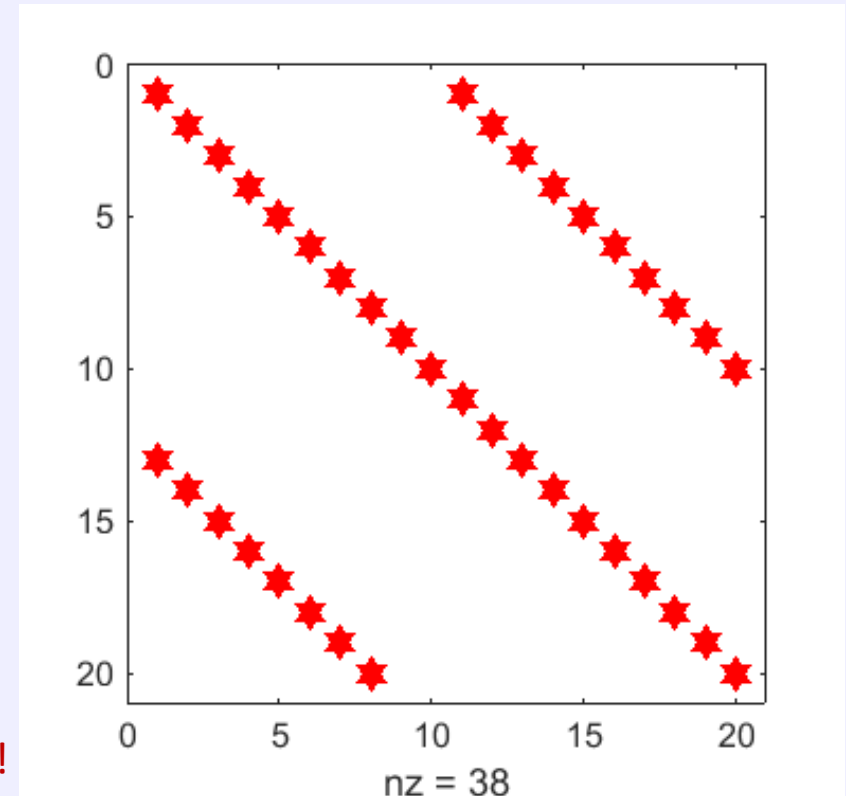
- 1) `clear ; vu=rand(1,5); d=randn(1,6); vl=randn(1,5)+4;`
- 2) `S=diag(d)+diag(vu,1)+diag(vl,-1)`
- 3) `R = sprandsym(S,[],0.9,3)`
- 4) `spy(R); eig(R)`
- 5) `R==R'`



Преобразуем матрицу в Sparse?

Пример создания разреженной матрицы:

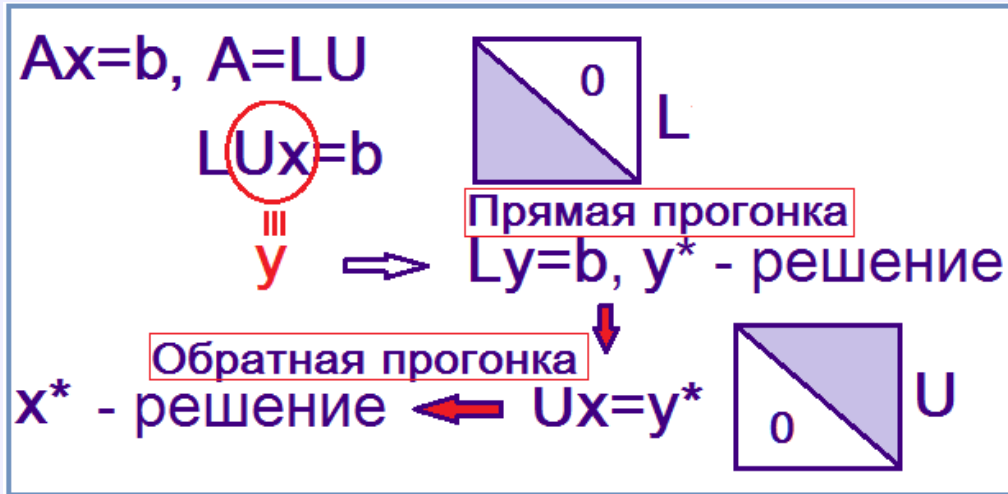
- 1) `b=randi([1,16],1,10);`
- 2) `c=randi([1,16],1,20);`
- 3) `l=randi([1,16],1,8);`
- 4) `A=diag(c)...`
- 5) `+diag(b,10)+diag(l,-12);`
- 6) `nonzero=nnz(A)`
- 7) `spy(A,'hr',12);`
- 8) `set(get(gca,'children'),...`
- 9) `'MarkerFaceColor','r');`
- 10) `[i, j, v] = find(A) ;`
- 11) `NumbersRow=feval('length',[i,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





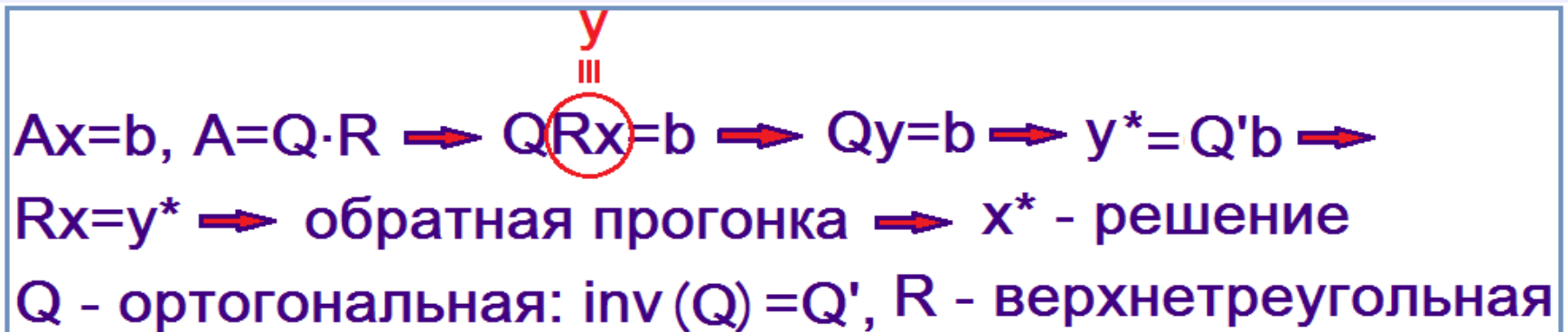
Мотивация факторизации матриц

Матрица системы – произведение ниже - и верхнетреугольной матриц:



Факторизованные системы легко решать!

Матрица системы – произведение ортогональной и верхнетреугольной матриц:



Факторизация матриц и решение систем

```

1 - n = 6;
2 - ipr='sp' % ipr='sp' или ipr=1
3 - %Положительно-определенные здесь:
4 - N = 50;
5 - A = gallery('poisson', N);
6 - A = pascal(6); M=A
7 - %
8 - if ipr=='sp', M=sparse(A), end
9 - R = chol(M)
10 - spy(R)
11 - isequal(R, 6x6 double =
12 - [Q,R] (1,1) 1
13 - det(Q) (1,2) 1
14 - (2,2) 1
15 - (1,3) ... 1

```

В относительно ранних версиях функции факторизации разреженных матриц отличались наличием sp^* в имени, теперь нет! Функции, что и для полных матриц!

Факторизация полных и разреженных матриц:

isequal(A,B,C)

true – полном совпадении матриц A,B,C

| | Описание |
|--------------------|---|
| [L,U]=lu(M) | $M=L \cdot U$, L-нижнетреугольная матрица, U-верхнетреугольная. |
| [Q,R]=qr(M) | $M=Q \cdot R$, R-верхнетреугольная матрица, Q-ортогональная: $inv(Q)=Q'$, $Q \cdot Q' = eye(size(Q))$ |
| R=chol(M) | R-верхнетреугольная матрица, $M=R' \cdot R$, M – положительно определённая <u>isequal(R'·R,M) %true!</u> |



Универсальный решатель

`linsolve(A,B,opts)`

A, B – матрица и столбец свободных членов, соответственно

`opts.UT` – true; `opts.LT` – true; (треугольные матрицы LU)

`opts.SYM` – true; (матрица системы – симметричная)

`opts.POSDEF` – true; (матрица системы – положительно определенная и заведомо симметричная) и т.д.

Заметим, что все опции должны быть определены до решения СЛАУ с помощью `linsolve(A,B,opts)`



Оценка быстродействия алгоритмов

Функции оценки быстродействия:

- 1) `tic operators toc`
- 2) `clock operators etime`
- 3) `cputime`

Пример:

```
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 – разность арг.  
  
%% 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]
```

год 2023 месяц 3 число 9 час 22 минута 50 секунда 43 **(7) ???**



Класс Symbol и методы поддерживаемые в ML

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

Методы для класса SYM:

| | | |
|--------------|------------------|-----------------|
| <i>abs</i> | <i>ezpolar</i> | <i>nnz</i> |
| <i>acos</i> | <i>ezsurf</i> | <i>nonzeros</i> |
| <i>acosh</i> | <i>ezsurf</i> | <i>norm</i> |
| <i>acot</i> | <i>factor</i> | <i>not</i> |
| <i>acoth</i> | <i>factorial</i> | <i>null</i> |
| ... | | |

Декларирование

символьных переменных:

```
syms var1 var2 var3 % и т.д.
```

Конвертирование Char в SYM

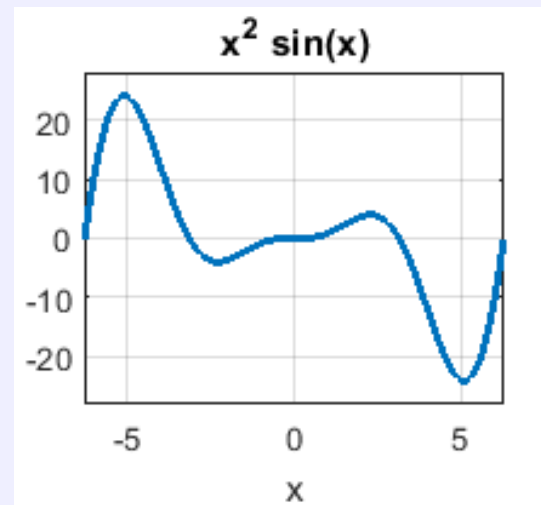
```
x=sym ('x') % переменной  
y=sym (expressionChar)
```

Пример

```
syms a x y % разделяются пробелами!!!  
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) – возможно, устареет
```

a =

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





Подходы к решению нелинейных уравнений и систем

- 1 Графическая визуализация уравнений (системы);
- 2 Выбор нулевых приближений и решение с помощью *fsolve*;
- 3 Выбор нулевых приближений интерактивно, используя *ginput*;
- 4 Решение уравнения вместо системы *fzero*;
- 5 Нахождение корней полиномов *roots*.



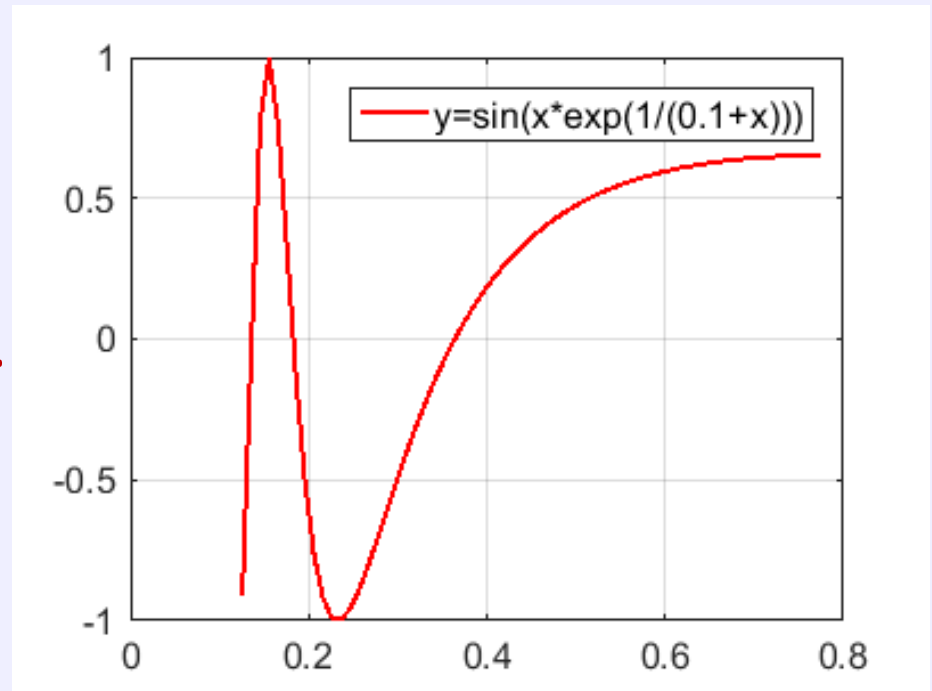
$F(x)=0$ и интерактивный выбор нулевого приближения или диапазона:

[pointsX,pointsY]=ginput(n),

pointsX - вектор абсцисс n- точек, pointsY –вектор ординат точек-нулевых приближений, в которых щёлкните мышкой

```
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) % ждёт три нул.прибл.
sol=fsolve('sin(x.*exp(1./(0.1+x)))',x0)
```

Решение $F(x) = 0$
[sol,accuracy,exitSol,InDetail]=fsolve(F,x0,opts)



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



Решение нелинейной системы второго порядка ($F_1(x)=0$, $F_2(x)=0$)

1) Решение системы – точки пересечения кривых $F_1(x)=0$ и $F_2(x)=0$:

```
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) Сначала ищем (абсциссы) корни $F_1(x)-F_2(x)=0$
 $x=\text{fsolve}('x.*\sin(8*x)-(x.^5-x+0.5)',x0)$

3) Ищем значения ординат точек пересечения:

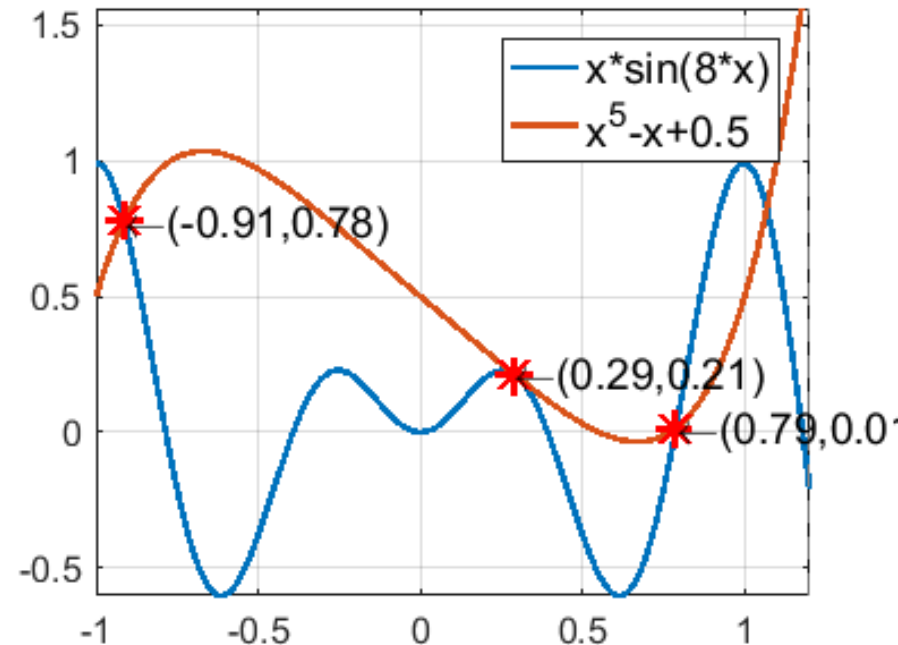
```
y=eval('x.*sin(8*x)')  
plot(x,y,'r*','linewidth',2)
```

4) Формируем надписи:

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

Почему выполняем шаг 2)?

Сколько строк конкатенирует на шаге 4)?





Корни полиномов . FZERO и ROOTS

!!! Рассмотрите самостоятельно функцию **fminbnd** – поиск минимального значения на отрезке

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

```
clear 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))
```

err = 0

```
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
```

allroots=roots(p) – вычисляются корни полинома

polyval(p,value) - вычисляется значение полинома в точке **value**

p-вектор коэффициентов полинома

Точность вычисления функций:

roots – точность высокая default

fzero – точность высокая default



Решение систем нелинейных алгебраических уравнений. Метод простых итераций (МПИ) для СНАУ

Система НАУ порядка n может быть представлена в общем виде:

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ \dots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases}$$

f_1, f_2, \dots, f_n – в т.ч. нелинейные функции переменных $X \equiv x_1, x_2, \dots, x_n$.

Пусть $|f_i(X)|$ – невязки системы на векторе X , тогда

если X^* – решение системы, то $|f_i(X^*)| = 0, i = 1, 2, \dots, n$.

Процесс нахождения решения итеративный; строится k -е приближение, и оно является решением с заданной точностью ε , если

$$|f_i(X^k)| < \varepsilon, i = 1, 2, \dots, n.$$



Алгоритм метода простых итераций

1. Задаём точность ε (≤ 0.001 , по фантазии)
2. Преобразуем систему к нормализованному виду:

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

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

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

Если невозможно, то н.п. - нулевой вектор или случайный

3. Вводим k – счетчик итераций.
4. Определяем формулу итерационного процесса: $X^{(k+1)} = \Phi(X^{(k)})$
5. Вычисляем $(k + 1)$ приближение, формула 4-го шага
6. Сравниваем $(k + 1)$ приближение с предыдущим

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

Если условие выполнено, то решение найдено, если нет, то вычисляем следующее приближение (шаг 5)



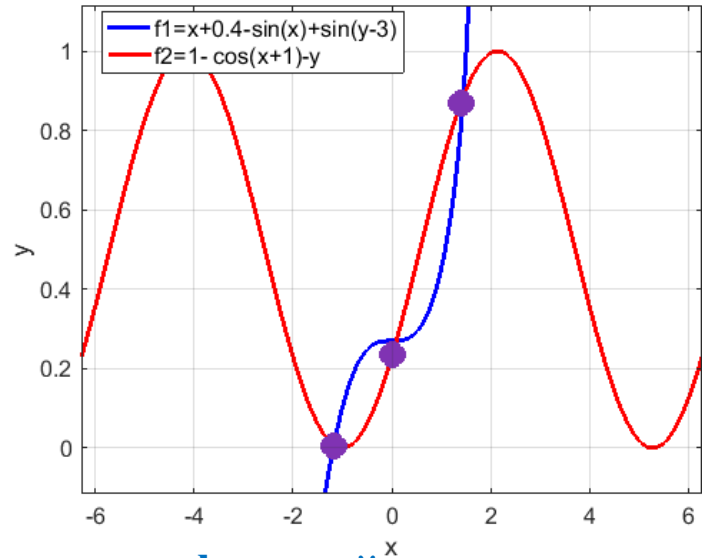
Пример решения **СНАУ** методом простых итераций

Пример:

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

нормализованная система:

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



Программирование с использованием встроенных функций:

1. Построить функции исходной системы графически.
2. Визуально определить нулевые приближения.
3. Задать точность решения ε
4. Ввести счетчик количества итераций ($k=0$ в начале)
5. Проверить, обеспечивают ли текущие решения заданную точность
6. Нет, точность не достигается – изменяем количество итераций и текущее значение используем как нулевое приближение, повторяем с шага 5
7. Да, точность достигается, празднуем победу!



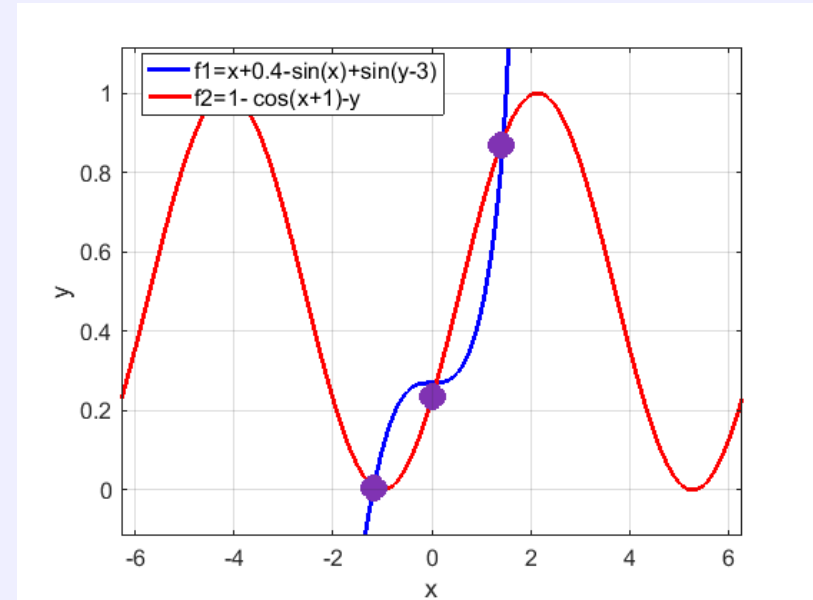
Пример решения **СНАУ** с помощью функционала MatLab

Пример:

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

нормализованная система:

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



Программирование с использованием встроенных функций:

1. Построить функции исходной системы графически.
2. На шаге 1. используем `ezplot`, `fplot`, `implicitplot`
3. Для решения системы используем `fplot` (уравнения системы задать как внешнюю функцию) с опцией регулирования `ToleranceX: TolX`, например, с точностью 10^{-3} :
`optimset('TolX',1.0e-3)`
4. Следует для наглядности отредактировать свойства осей



Пример решения **СНАУ** для систем высоких порядков

```
Clear      % high order nonlinear 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 методы стабильны, могут применяться в т.ч. для плохо обусловленных систем и имеют очень достойные характеристики сходимости!

В методе учитывается шаг при поиске корней и направление!!!



Бонус

nvkurbatova@sfedu.ru

Solve Sudoku Puzzles Via Integer Programming

This example shows how to solve a Sudoku puzzle using binary integer programming

Initial Puzzle

Here is a data matrix $|B|$ of clues. The first row, $|B(1,2,2)|$, means row 1, column 2 has a clue 2. The second row, $|B(1,5,3)|$, means row 1, column 5 has a clue 3. Here is the entire matrix $|B|$.

SudokuExample.m % Exampes ML

```
B=[1,2,2; 1,5,3; 1,8,4;...  
  2,1,6; 2,9,3; 3,3,4; ...  
  3,7,5; 4,4,8; 4,6,6;...  
  5,1,8; 5,5,1; 5,9,6;...  
  6,4,7; 6,6,5; 7,3,7;...  
  7,7,6; 8,1,4; 8,9,8;...  
  9,2,3; 9,5,4; 9,8,2];  
drawSudoku(B)  
set(get(gcf,'children'),'fontsize',18)  
...
```

| | | | | | | | | |
|---|---|---|---|---|--|---|---|---|
| | 2 | | | 3 | | | 4 | |
| 6 | | | | | | | | 3 |
| | | 4 | | | | 5 | | |
| | | | 8 | 6 | | | | |
| 8 | | | | 1 | | | | 6 |
| | | | 7 | 5 | | | | |
| | | 7 | | | | 6 | | |
| 4 | | | | | | | | 8 |
| | 3 | | | 4 | | | 2 | |



Спасибо за внимание!

«Никто не обнимет необъятного!» - Козьма П. Прутков
(Л. М. Жемчужников, А. Е. Бейдеман, Л. Ф. Лагорио)

Напомните о Эжене Ажаре и Ромен Гари!