Методические указания по SciLab
для практикума по дисциплине «Теория систем и системный анализ»

Цель методички — дать студентам пошаговые инструкции для выполнения практических работ в SciLab. Все примеры подобраны так, чтобы запускались без дополнительных пакетов в стандартной установке SciLab.

Предварительные требования: установлен SciLab (Windows/Linux/macOS), базовые навыки работы с компьютером. Рекомендуется использовать редактор Scinotes и сохранять скрипты с расширением .sce.

Установка и первый запуск

1) Установите SciLab по умолчанию.
2) Запустите SciLab → Applications → Scinotes.
3) Создайте новый файл, вставьте код из примеров и сохраните в папку проекта.

Проверьте работу среды простым примером:

```
// hello.sce
disp("Hello, SciLab!");
a = sin(%pi/4);
disp(a); // ожидаемо ~0.7071
```

Практическая работа №1. Знакомство с SciLab и базовые операции

Цель: Освоить интерфейс, консоль, работу со скриптами, базовые арифметические операции и функции.

Шаги:

1) Откройте Scinotes. Создайте файл p1_basics.sce.

2) Выполните приведённые ниже команды и проанализируйте вывод.

Код для запуска:

```
// p1_basics.sce
// Переменные и базовые функции
x = 3; y = 4;
r = sqrt(x^2 + y^2);    // 5
disp(r);

disp(sin(%pi/6));       // 0.5
disp(exp(1));           // e
disp(log(10));          // ln(10)

// Векторы и диапазоны
v = 1:5;        // [1 2 3 4 5]
u = linspace(0, 1, 6); // 6 точек от 0 до 1
disp(v); disp(u);

// Элементные операции
a = [1 2 3]; b = [4 5 6];
disp(a + b);
disp(a .* b);           // поэлементное умножение
```

Проверка результата: Ожидается корректный вывод чисел без ошибок. Убедитесь, что работают диапазоны и элементные операции.

Практическая работа №2. Построение графиков (2D/3D)
Цель: Научиться строить 2D и 3D графики, настраивать подписи и легенды.

Шаги:

1) Создайте файл p2_plots.sce. Постройте 2D-график функции $y=4x^2+5x+8$.

2) Добавьте легенду, сетку и заголовок. Затем постройте 3D-поверхность.

Код для запуска:

```
// p2_plots.sce
// 2D график
x = -10:0.1:10;
y = 4*x.^2 + 5*x + 8;
clf();
plot(x, y);
xtitle("y = 4x^2 + 5x + 8", "x", "y");
xgrid();
legend("полином");

// Параметрические кривые (пример Лиссажу)
t = 0:0.01:%pi*2;
x1 = sin(3*t); y1 = sin(4*t);
figure();
plot(x1, y1);
xtitle("Фигура Лиссажу", "x(t)", "y(t)");
xgrid();

// 3D поверхность z = x^2 - y^2
xv = -2:0.1:2;
yv = -2:0.1:2;
[X,Y] = meshgrid(xv, yv);
Z = X.^2 - Y.^2;
figure();
plot3d(xv, yv, Z);      // сетка по векторам xv,yv и матрице Z
xtitle("z = x^2 - y^2", "x", "y");
```

Проверка результата: Должны открыться окна с 2D-графиками и 3D-поверхностью. Проверьте подписи осей и легенды.

Практическая работа №3. Векторы, матрицы и СЛАУ

Цель: Освоить матричные операции и решение линейных систем Ax=b.

Шаги:

1) Создайте файл p3_linear_algebra.sce.

2) Сформируйте матрицу A и вектор b. Решите Ax=b и проверьте точность.

Код для запуска:

```
// p3_linear_algebra.sce
A = [3 2 -1; 2 -2 4; -1 0.5 -1];
b = [1; -2; 0];
x = A \ b;              // решение СЛАУ
disp(x);

// Проверка невязки
res = A*x - b;
disp("Невязка:"); disp(res);

// Собственные значения/векторы (дополнительно)
[vecs, vals] = spec(A);  // eigenvectors/eigenvalues
disp(vals);
```

Проверка результата: Невязка должна быть близка к нулю.
Собственные значения должны выводиться корректно.

Практическая работа №4. Решение алгебраических уравнений и систем

Цель: Научиться находить корни полиномов и решать системы нелинейных уравнений.

Шаги:

1) Создайте файл p4_equations.sce.

2) Найдите корни квадратного полинома и решите систему нелинейных уравнений методом fsolve.

Код для запуска:

```
// p4_equations.sce
// Корни полинома x^2 - 5x + 6
p = [1 -5 6];
r = roots(p);
disp("Корни:"); disp(r);

// Система нелинейных уравнений:
// x^2 + y^2 = 1, x - y = 0
function F = sys(u)
    x = u(1); y = u(2);
    F(1) = x^2 + y^2 - 1;
    F(2) = x - y;
endfunction

[sol, fval] = fsolve([0.5; 0.5], sys);
disp("Решение:"); disp(sol);
disp("Проверка:"); disp(fval); // должно быть близко к [0;0]
```

Проверка результата: Проверьте, что корни полинома равны 2 и 3, а система решается с малой невязкой.

Практическая работа №5. Численное дифференцирование и интегрирование

Цель: Освоить численное дифференцирование и вычисление определённых интегралов.

Шаги:

1) Создайте файл p5_diff_int.sce.

2) Вычислите производную sin(x) по сетке и интеграл ∫0^π sin(x) dx.

Код для запуска:

```
// p5_diff_int.sce
// Численная производная sin(x)
x = 0:0.01:%pi;
```

```
y = sin(x);
dy = diff(y) ./ diff(x);
xm = (x(1:$-1) + x(2:$)) / 2;
clf(); plot(xm, dy);
xtitle("Численная производная sin(x)", "x", "dy/dx"); xgrid();

// Численный интеграл ∫0^π sin(x) dx
function val = f(x)
    val = sin(x);
endfunction
I = intg(0, %pi, f);
disp("Интеграл sin(x) от 0 до π:"); disp(I); // ожидаемо ~2
```

Проверка результата: График производной должен быть близок к cos(x). Интеграл должен быть ≈ 2.0.

Практическая работа №6. Интерполяция и аппроксимация, регрессия

Цель: Научиться интерполировать и аппроксимировать данные, строить линейную регрессию.

Шаги:

1) Создайте файл p6_fit.sce.

2) Сгенерируйте данные, аппроксимируйте линейной моделью y=a0+a1*x и оцените ошибку.

Код для запуска:

```
// p6_fit.sce
// Синтетические данные
x = linspace(0, 10, 50)';
y_true = 2 + 0.8*x;
noise = grand(50, 1, "nor", 0, 0.5);
y = y_true + noise;
```

```
// Линейная регрессия (reglin): y ≈ a0 + a1*x
a = reglin(x, y);  // a(1)=a0, a(2)=a1
y_hat = a(1) + a(2)*x;

// Качество (RMSE)
rmse = sqrt(mean((y - y_hat).^2));
disp("Оценки:"); disp(a);
disp("RMSE:"); disp(rmse);

// Интерполяция
xi = 0:0.1:10;
yi = interp1(x', y', xi, "linear"); // линейная интерполяция
clf(); plot(x, y, "o"); plot(x, y_true, "-"); plot(xi, yi, "--");
legend("данные","истинная модель","интерполяция"); xgrid();

// Полиномиальная аппроксимация (2-й степени) через МНК
V = [ones(size(x,1),1) x x.^2];
coef = V \ y;
y2 = V*coef;
disp("Коэф. полинома степени 2:"); disp(coef);
```

Проверка результата: График должен показывать точки, истинную линию и интерполяцию. Коэффициенты и RMSE — в консоли.

Практическая работа №7. Случайные величины и распределения
Цель: Смоделировать выборки из распределений, построить гистограммы и оценить параметры.

Шаги:

1) Создайте файл p7_random.sce.

2) Сгенерируйте нормальную выборку, постройте гистограмму, вычислите среднее и СКО.

Код для запуска:

```
// p7_random.sce
n = 1000;
X = grand(1, n, "nor", 0, 1);  // нормальное(0,1)
m = mean(X); s = stdev(X);
clf(); histplot(30, X);
xtitle("Нормальное распределение (0,1)","X","частота");
disp("Выборочное среднее:"); disp(m);
disp("Выборочное СКО:"); disp(s);
```

Проверка результата: Гистограмма должна быть колоколообразной; среднее ближе к 0, СКО — ближе к 1.

Практическая работа №8. Решение ОДУ: одиночные и системы

Цель: Научиться решать ОДУ с помощью ode и анализировать результаты на графиках.

Шаги:

1) Создайте файл p8_ode.sce.

2) Решите dx/dt = -k x, затем систему Лотки–Вольтерры.

Код для запуска:

```
// p8_ode.sce
// 1) Одно уравнение dx/dt = -k x
function dx = fdecay(t, x)
   k = 0.5;
   dx = -k*x;
endfunction
t = 0:0.1:10;
x0 = 10;
x = ode(x0, 0, t, fdecay);
clf(); plot(t, x);
xtitle("Затухающая экспонента","t","x(t)"); xgrid();

// 2) Система Лотки–Вольтерры
```

```
function dx = fLV(t, X)
    a = 1.0; b = 0.1; c = 1.5; d = 0.075;
    prey = X(1); pred = X(2);
    dx = [a*prey - b*prey*pred;  -c*pred + d*prey*pred];
endfunction
t2 = 0:0.05:50;
x0 = [10; 5];
X = ode(x0, 0, t2, fLV);
figure();
plot(t2, X(1,:)); plot(t2, X(2,:));
legend("Жертвы","Хищники"); xgrid();
xtitle("Модель Лотки–Вольтерры","t","популяции");
```

Проверка результата: Первый график — плавное экспоненциальное затухание. Второй — колебания жертв и хищников.

Практическая работа №9. Нелинейная оптимизация (без ограничений)

Цель: Минимизировать функцию (пример Розенброка) и интерпретировать результат.

Шаги:

1) Создайте файл p9_optim.sce.

2) Минимизируйте функцию Розенброка из произвольной начальной точки.

Код для запуска:

```
// p9_optim.sce
// Функция Розенброка
function y = rosen(x)
    y = (1 - x(1))^2 + 100*(x(2) - x(1)^2)^2;
endfunction

x0 = [-1.2; 1];
```

```
/*
optim: [fopt, xopt] = optim(f, x0)
f — функция, принимающая вектор x и возвращающая скаляр
*/
[fopt, xopt] = optim(rosen, x0);
disp("Минимум: fopt="); disp(fopt);
disp("В точке: xopt="); disp(xopt);
```

Проверка результата: Ожидаемо минимум близок к 0 при x≈[1;1].
Значения должны сходиться к окрестности (1,1).

Соответствие тем с методичками по Mathcad

SciLab (данная методичка)	Mathcad (ваши методички)
ПР1. Базовые операции	Практическая №1 (Основы)
ПР2. Графики 2D/3D	Практическая №2
ПР3. Векторы, матрицы, СЛАУ	Практическая №3
ПР4. Уравнения и системы	Практическая №4; ЛР №12
ПР5. Дифф./интегрирование	Практическая №5
ПР6. Интерполяция/регрессия	Практическая №6
ПР7. Распределения	Практическая №7
ПР8. ОДУ и системы ОДУ	ЛР №14
ПР9. Оптимизация (НЛО)	ЛР №13 (экстремумы)

Чек-лист для студента перед сдачей

• Скрипт запускается без ошибок (Ctrl+L в Scinotes).
• Графики имеют подписи осей и заголовки.
• Для задач оптимизации и решения уравнений приведены проверочные расчёты (невязка/значение функции).
• Все ответы и графики сохранены (File → Export).