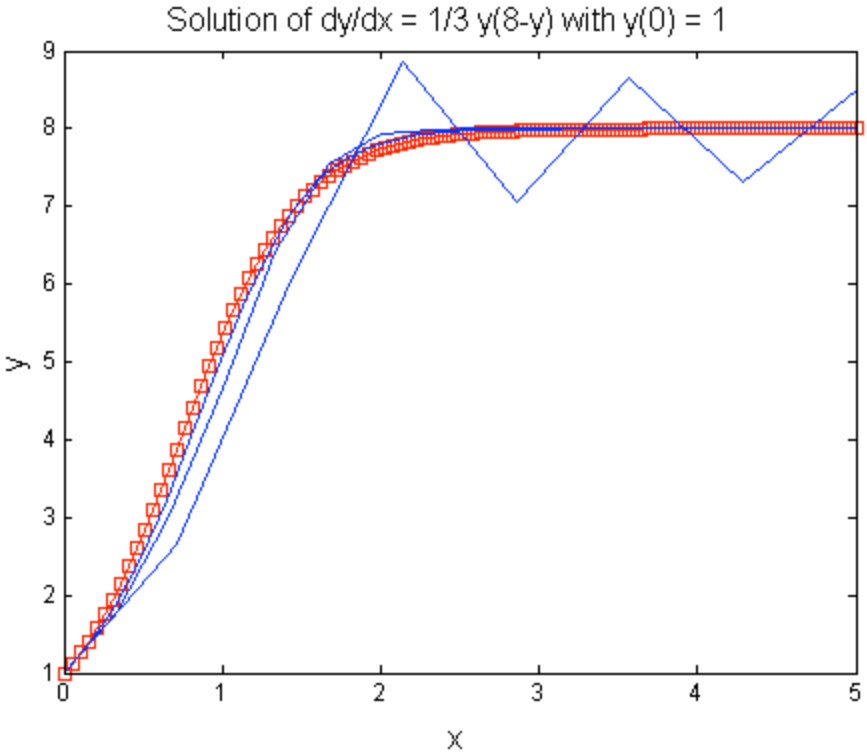
Mini-Matlab Lesson 8: Problem 1

```
clear;
close all;

nmat = [8 16 32];

f = @(x,y) 1/3*y*(8-y);
g = @(x)8./(1+7*exp(-8*x/3));

xfine = linspace(0, 5, 100);    % A nice, fine mesh

figure(1);
plot(xfine, g(xfine), 'rs-');
hold on;

for j = 1:3
    n = nmat(j);

    x = linspace(0, 5, n);
    h = x(2) - x(1);            % Discretization size

    y(1) = 1;                   % Initial value

    for k = 2:n
        y(k) = y(k-1) + f(x(k-1), y(k-1))*h;
    end

    plot(x, y, 'b');
    drawnow;

    err = norm(g(x) - y, inf);
    disp(['When n = ', num2str(n), ' the error is ', num2str(err)]);
end

hold off;
xlabel('x', 'FontSize', 16);
ylabel('y', 'FontSize', 16);
title('Solution of dy/dx = 1/3 y(8-y) with y(0) = 1', 'Fontsize', 16);
```

```
When n = 8 the error is 1.251
When n = 16 the error is 0.70714
When n = 32 the error is 0.36219
```



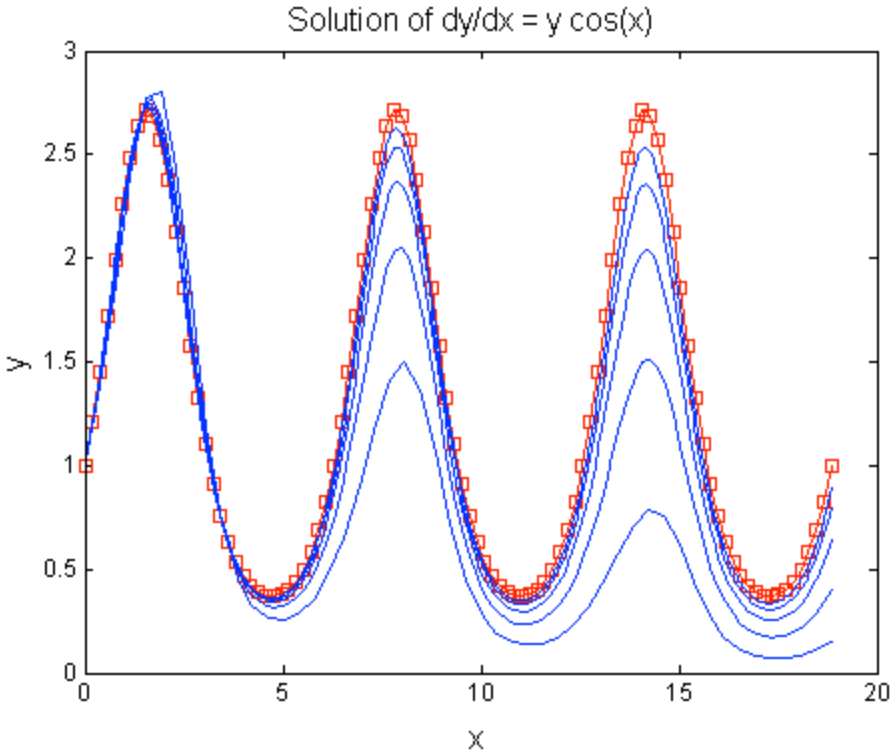
Published with MATLAB® 7.12

Contents

Mini-Matlab Lesson 8: Problem 2

```
clear;
close all;

nmat = [50 100 200 400 800];

f = @(x,y) y*cos(x);
g = @(x)exp(sin(x));

xfine = linspace(0, 6*pi, 100);    % A nice, fine mesh

figure(1);
plot(xfine, g(xfine), 'rs-');
hold on;

for j = 1:5
    n = nmat(j);

    x = linspace(0, 6*pi, n);
    h = x(2) - x(1);            % Discretization size

    y(1) = 1;                   % Initial value

    for k = 2:n
        y(k) = y(k-1) + f(x(k-1), y(k-1))*h;
    end

    plot(x, y, 'b');
    drawnow;

    err(j) = norm(g(x) - y, inf);
    disp(['When n = ', num2str(n), ' the error is ', num2str(err(j))]);
end

hold off;
xlabel('x', 'FontSize', 16);
ylabel('y', 'FontSize', 16);
title('Solution of dy/dx = y cos(x)', 'Fontsize', 16);
```

```
When n = 50 the error is 1.921
When n = 100 the error is 1.2167
When n = 200 the error is 0.68944
When n = 400 the error is 0.36792
When n = 800 the error is 0.19024
```

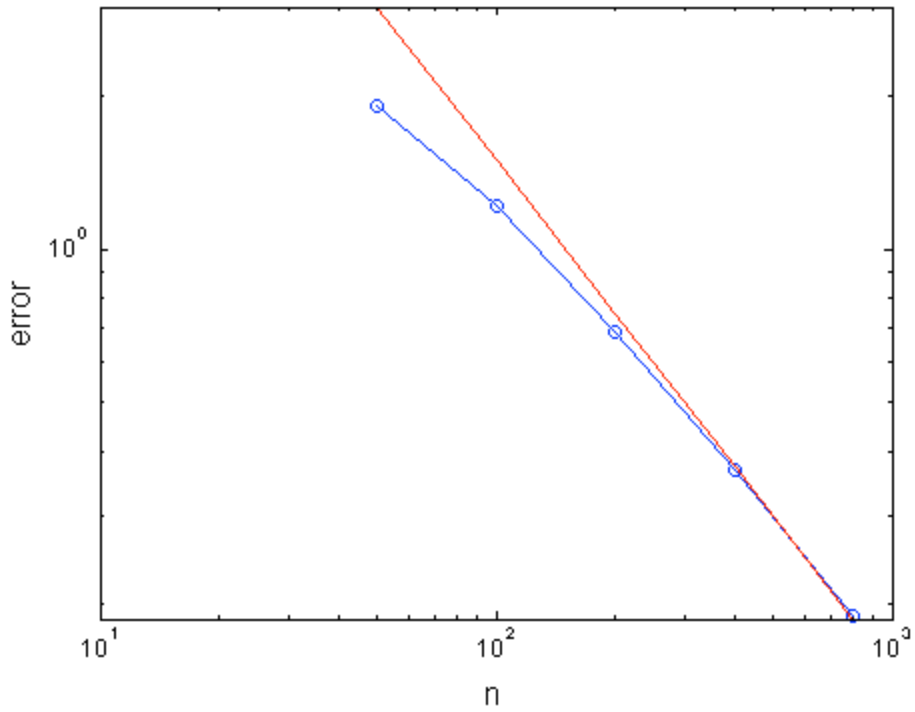


We can plot the error involved on a loglog plot

A very rough fit of the data shows that the $error \sim 150/n$ in the limit that n is large. (a more accurate fit can be achieved by doing least squares).

So if we want the error to be 10^{-12} (12 digits), this would require on the order of 1.5×10^{14} data points!

```
figure(2)
loglog(nmat, err, 'bo-');
hold on
loglog(nmat, 150./nmat, 'r');
hold off
xlabel('n', 'FontSize', 16);
ylabel('error', 'FontSize', 16);
```



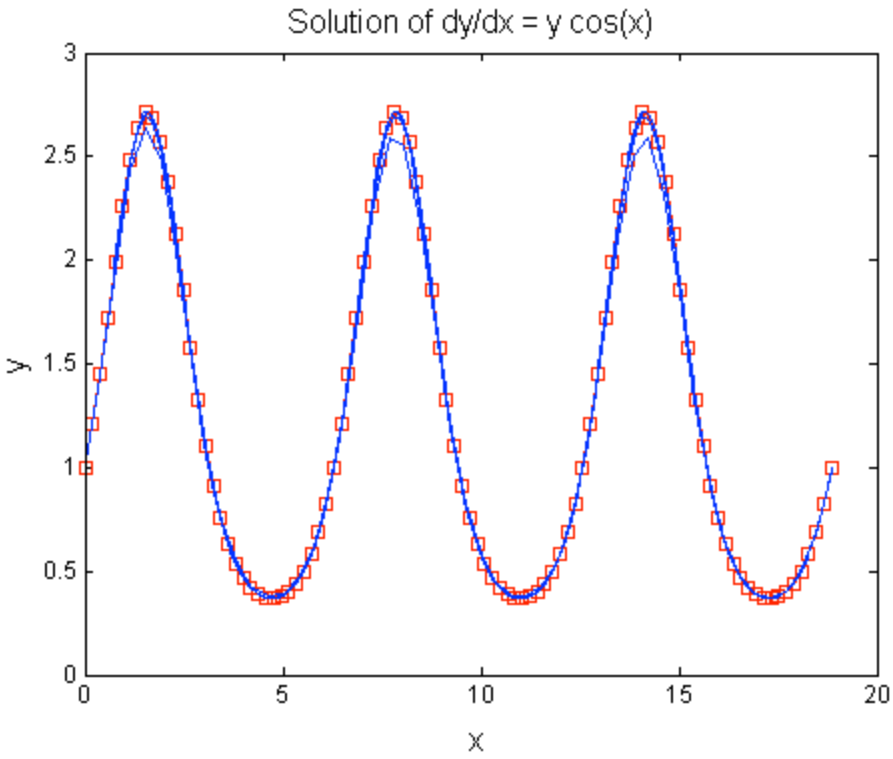
Published with MATLAB® 7.12

Contents

Mini-Matlab Lesson 8: Problem 3

```
clear;
close all;

nmat = [50 100 200 400];

f = @(x,y) y*cos(x);
g = @(x)exp(sin(x));

xfine = linspace(0, 6*pi, 100);    % A nice, fine mesh

figure(1);
plot(xfine, g(xfine), 'rs-');
hold on;

for j = 1:length(nmat)

    tic

    n = nmat(j);

    x = linspace(0, 6*pi, n);
    h = x(2) - x(1);            % Discretization size

    y(1) = 1;                   % Initial value

    for k = 2:n
        yy = y(k-1) + f(x(k-1), y(k-1))*h;   % Prediction
        nk = f(x(k), yy);                    % Prediction slope

        y(k) = y(k-1) + h/2*(f(x(k-1), y(k-1))+nk);     % Correction
    end

    plot(x, y, 'b');
    drawnow;

    err(j) = norm(g(x) - y, inf);
    disp(['When n = ', num2str(n), ' the error is ', num2str(err(j))]);

    time(j) = toc;
end

hold off;
xlabel('x', 'FontSize', 16);
ylabel('y', 'FontSize', 16);
title('Solution of dy/dx = y cos(x)', 'Fontsize', 16);
```

```
When n = 50 the error is 0.11593
When n = 100 the error is 0.023269
When n = 200 the error is 0.0052451
When n = 400 the error is 0.0012423
```

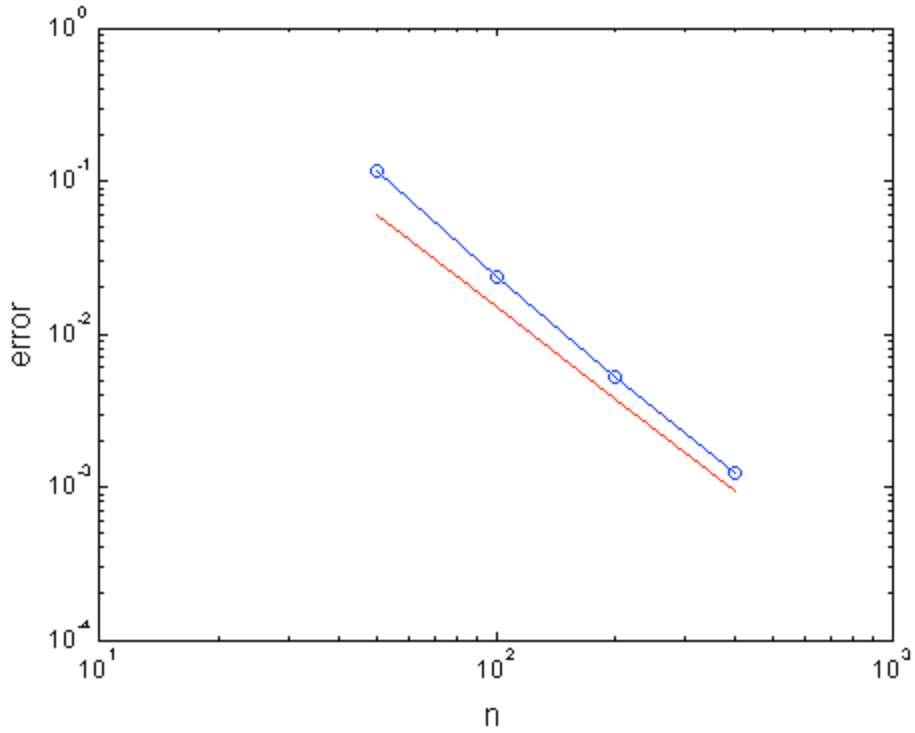


We can plot the error involved on a loglog plot

A very rough fit of the data shows that the $error \sim 150/n^2$ in the limit that n is large. (a more accurate fit can be achieved by doing least squares).

So if we want the error to be 10^{-12} (12 digits), this would require on the order of 1.2×10^7 data points!

```
figure(2)
loglog(nmat, err, 'bo-');
hold on
loglog(nmat, 150./nmat.^2, 'r');
hold off
xlabel('n', 'FontSize', 16);
ylabel('error', 'FontSize', 16);
```



How long is it all taking?

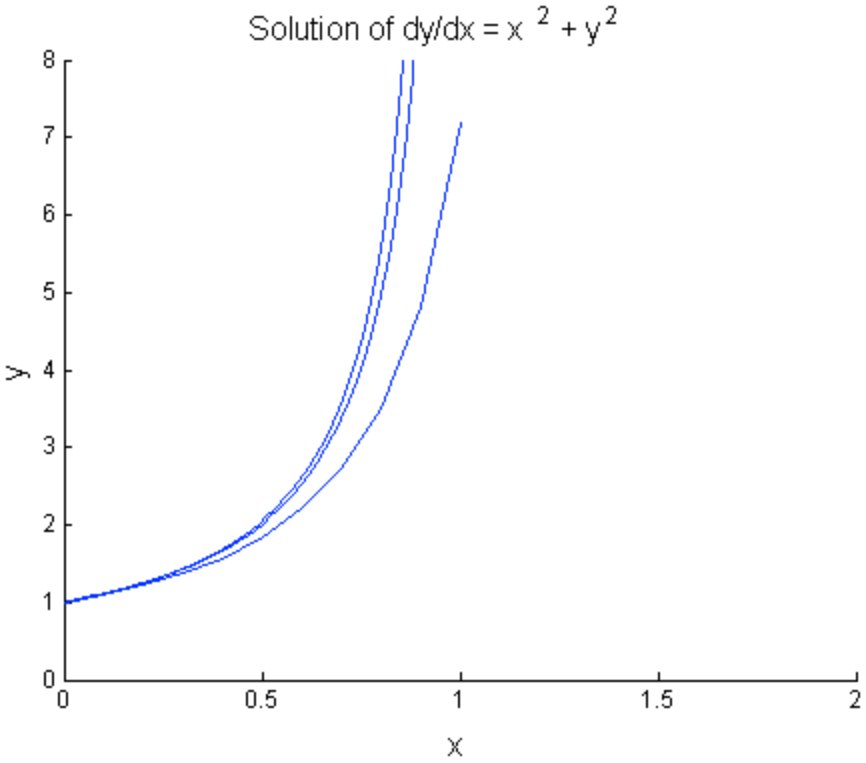
```
disp(['The last computation took ', num2str(time(end)/nmat(end)), ' seconds per step
disp(['Assuming that we require 1.2e7 steps, we would need ', num2str(time(end)/nmat

% Performing 1e7 steps impractical for many reasons. Here are two:
% with so many steps, roundoff error (due to not keeping enough digits)
% will become a problem. Second, storage of 1e7 is a big pain. What we need
% is a higher-order method!
```

```
The last computation took 5.744e-05 seconds per step
Assuming that we require 1.2e7 steps, we would need 11.488 minutes
```

Published with MATLAB® 7.12

Mini-Matlab Lesson 8: Problem 4

```
clear;
close all;

hmat = [0.1 0.02 0.005];

f = @(x,y) x.^2 + y.^2;

figure(1);
hold on;

for j = 1:length(hmat)

    h = hmat(j);
    x = [0:h:1];


    y(1) = 1;                    % Initial value


    for k = 2:length(x)
        y(k) = y(k-1) + f(x(k-1), y(k-1))*h;
    end

    plot(x, y, 'b');
    drawnow;
end

hold off;
xlabel('x', 'FontSize', 16);
ylabel('y', 'FontSize', 16);
title('Solution of dy/dx = x^2 + y^2', 'Fontsize', 16);
axis([0 2 0 8]);
```



Published with MATLAB® 7.12