Part I: Final exam

1. (10 points) Consider a tri-diagonal (diagonally dominant) system of linear equations of arbitrarily (very) large order *n*. Which method would you use to solve this system? Why? Assuming that *n=3*, show conceptually how you would use the proposed method.

$$\begin{bmatrix} a_1 & -b_1 & 0 & \cdots & 0 \\ -c_2 & a_2 & -b_2 & \ddots & \vdots \\ 0 & -c_3 & a_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -b_{n-1} \\ 0 & \cdots & 0 & -c_n & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}$$

Solution:
Since the order of the system is very large and the coefficient matrix is sparse the iterative methods are suitable for solving this type of systems. For direct methods we would face a fill-in effect that would increase significantly the cost of the computation. Since the system is diagonally dominant one can use Jacobi or Gauss-Seidel method.
If we apply the Jacobi method to the system:

$$\begin{bmatrix} a_1 & -b_1 & 0 \\ -c_2 & a_2 & -b_2 \\ 0 & -c_3 & a_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

we obtain

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = C \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + D, \text{where} \quad C = \begin{bmatrix} 0 & b_1/a_1 & 0 \\ c_2/a_2 & 0 & b_2/a_2 \\ 0 & c_3/a_3 & 0 \end{bmatrix}, D = \begin{bmatrix} d_1/a_1 \\ d_2/a_2 \\ d_3/a_3 \end{bmatrix}$$

2. (20 points) A space shuttle reentering the earth atmosphere decelerates in one hour from its orbital speed (7360.56 m/s) to its touchdown speed (96.11 m/s). Table 1 contains data of its measured velocity at equally spaced time-points.

Time (s)	-3600	-2400	-1200	0
Velocity (m/s)	7360.56	7245.61	6722.22	96.11

Table 1. Space shuttle descent velocity data

a) Based on the provided data, and using forward differences determine the acceleration values of the space shuttle. Detailed calculations and utilized expressions should be provided.
b) If the time vector is given in the variable *t*, and the accelerations are given in the vector *A*, write a sequence of Matlab commands that will give you information at which time the astronauts experienced their acceleration dropping below -5 m/s^2.
c) Using the composite trapezoid rule determine the total distance travelled by the shuttle from the first observation until the touchdown. Detailed calculations and utilized expressions should be provided.

Student name: Sciper no:

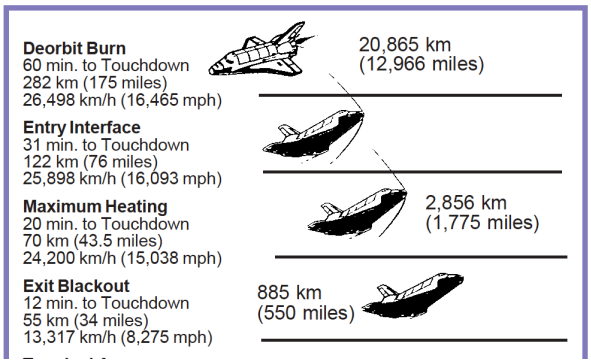


Figure 1: Space shuttle descent procedure (*source nasa.gov*)

Solution:

a) Forward differences: $f'(x_0) \approx \dfrac{f(x_0+h)-f(x_0)}{h}$

Using this formula with *h=1200* we get for the accelerations:
a_1=-0.096 m/s², a_1=-0.436 m/s², and a_1=-5.52175 m/s².

b) Code (one out of many alternative solutions):

```
t=[-3600 -2400 -1200];
A=[-0.096 -0.436 -5.52175];
tUnder5=t(find(A<-5));
```

c) We used the composite trapezoid rule

$$I_h = \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{n-1} f(a+ih) + f(b) \right]$$

and we obtained I=21350730km.

4. (20 points) Consider the following system of ODEs

$$\frac{dy_1}{dt} = -80.6y_1 + 119.4y_2$$

$$\frac{dy_2}{dt} = 79.6y_1 - 120.4y_2$$

The general solution of this system can be written in the form:

$$\mathrm{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = C_1 \begin{bmatrix} 3 \\ 2 \end{bmatrix} e^{-t} + C_2 \begin{bmatrix} -1 \\ 1 \end{bmatrix} e^{-200t}$$

a) Discuss which method you would apply to solve numerically the above-mentioned system of ODEs and what would be the appropriate step size.

b) For $C_1 = 1$ and $C_2 = 2$, apply the proposed method with the proposed step size to solve the system in the interval $t \in (0,1)$ (provide the details of the first step).

Solution: already provided.

5. (5 points) The Runge-Kutta method, as learned in the course, does it belong to the group of explicit or implicit methods? Would you use it to solve stiff systems? Why?

Solution: Runge-Kutta, as learned in the course, belongs to a class of explicit methods and it is not suitable for solving stiff systems as its step size is sensitive with respect to the dynamics of the system.

Student name: Sciper no:

6. (5 points) Explain which method you would choose for solving systems of nonlinear equations (continuous and continuously differentiable in the interval around the solution):
   a) to guarantee convergence to the solution;
   b) to have the fastest convergence rate.

Solution: a) bisection method guarantees the convergence albeit a very slow one as its algorithm is not dependent on the form of the function. b) among the methods that have been presented in the course, the Newton-Raphson methods has the fastest (quadratic) convergence rate.

7. (20 points) Consider the following nonlinear equation on the interval (pi/2,pi):

$$y = \sin x - \frac{x}{2}$$

   a) Using the Newton-Raphson method find numerically the root of this equation with the precision of 10^{-3}. Take as initial point $x_0=2$; the exact solution is $x^*=1.89549$.
   b) Estimate the number of Newton-Raphson iterations needed to reach this root with a precision of 10^{-9}.
   c) How many iterations would be needed to attain the accuracy of 10^{-9} if we use the bisection method?

   Detailed calculus and utilized expressions should be provided as well.

Solution:

a) 1st derivative:

$$f'(x) = \cos x - \frac{1}{2}$$

2nd derivative:

$$f''(x) = -\sin x$$

From initial condition: $x_0=2$, we obtain

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 1.9$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 1.89511$$

Accuracy of the solution: $\left|x^* - x_2\right| = 3.75 \cdot 10^{-4} < 10^{-3}$

b) From the Newton-Raphson convergence analysis we have that:

$$\left|x^* - x_{k+1}\right| \le M\left|x^* - x_k\right|^2 \quad \text{with} \quad M = \frac{1}{2}\frac{\max\limits_{x^*-\varepsilon \le x \le x^*+\varepsilon}\left|f''(x)\right|}{\min\limits_{x^*-\varepsilon \le x \le x^*+\varepsilon}\left|f'(x)\right|}$$

From above we can write:

$$|x^* - x_k| \le M^k |x^* - x_0|^{2^K}$$

In the interval (x0,x*) and knowing the 1st and 2nd derivative of *f(x)* we estimate *M* from the above equation as *M*=0.5172. Therefore, to ensure $|x^* - x_k|$ smaller than 10^{-9} we have to perform: K=3.0505, i.e. 4 iterations.

c) Error of the bisection method:

$$\left|\varepsilon\right| \le \frac{b-a}{2^n} \;\Rightarrow\; 2^n \le \frac{b-a}{\left|\varepsilon\right|} \;\Rightarrow\; n = \log_2\left(\frac{b-a}{\left|\varepsilon\right|}\right) = 30.55$$

therefore for *n=31*, the obtained error will be under 10^{-9}.

Student name: Sciper no:

8. (8 points) The *Matlab* function below *poly_function.m* evaluates the following polynomial $f(x) = 5x^3 + 3x^2 + 2x + 6$. We want to compute this polynomial for the given vector of coefficients *a* at the value *x=3*. The output argument of this function should be a scalar *pol*. Try to identify the four errors in this function.

```
>> a = [5 3 2 6];
>> x = 3;

>> polVal = poly_function(a, x);

% function
function pol = poly_function(a,Y)
     Y = 5;
     potency = Y.^(length(a):1);
     pol = sum(a*potency);
end
```

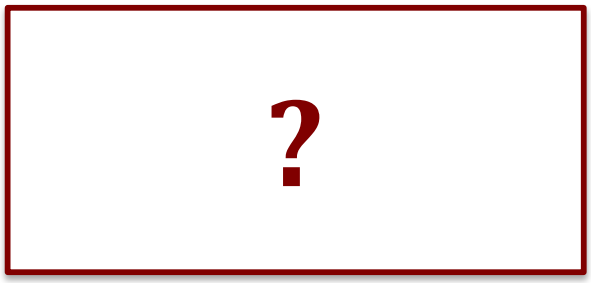
Solution:

1. `Y = 5` **redefines the purpose of the function.**
2. In `length(a):1` **a step is missing (-1)**
3. The degree of the polynomial is 3, but in potency we are defining 4 elements
4. In `sum(a*potency)` we need a dot product, i.e `sum(a.*potency)`

9. Analyze the *Matlab* code given below where *A* is a matrix of size *C*x*C* and *b* is a vector of size *C*x*1*.
(i) (6 points) Try to recognize and explain what is the purpose of this code. In which numerical method that you have seen in the class could it be used?
(ii) (6 points) A couple of lines of code is missing (indicated by the red rectangle) so that the code is not giving the expected outputs according to its intended use. Identify the purpose of the missing part and try to write the missing lines.

```
for col=1:C
    max = abs(A(col,col));
    max_pos = col;
    for i = col+1:C
        if abs(A(i,col)) > max
            max = abs(A(i,col));
            max_pos = i;
        end
    end

    temp = A(col,:);
    A(col,:) = A(max_pos,:);
    A(max_pos,:) = temp;
```



Student name: Sciper no:

```
    for row=col+1:C
        mult_factor = A(row,col)/A(col,col);
        A(row,:) = A(row,:) - mult_factor * A(col,:);
        b(row) = b(row) - mult_factor * b(col);
    end
end
```

Solution:

(i) This code performs first Gaussian elimination with partial pivoting.

(ii) The transformation of the vector *b* is missing:

```
Temp=b(col);
b(col)=b(max_pos);
b(max_pos)=Temp;
```