8.3 Leukocyte Migration in a Capillary

a

Matlab Function to compute SSE:

```
function SSE = piecewisefunc(params)
global t x
slope = params(1);
intercept = params(2);
t0 = params(3);
SSE = 0;
for i=1:length(t)
    if t(i)>t0
        SSE = SSE + (x(i) - (slope*t(i) + intercept))^2;
    else
        SSE = SSE + (x(i) - (slope*t0 + intercept))^2;
    end
end
```

b

Matlab Script for main function:

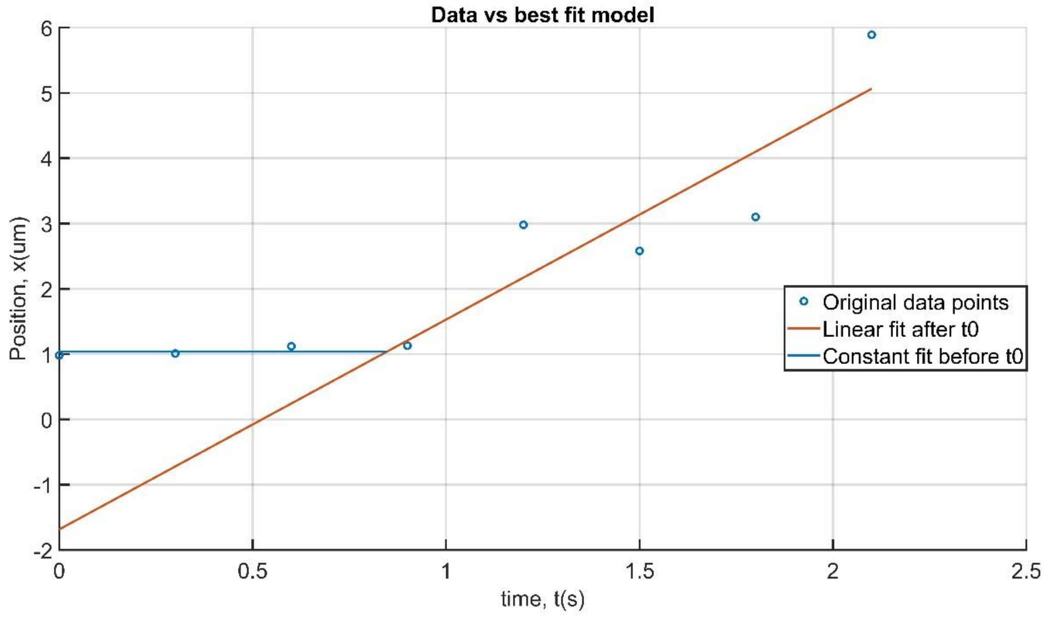
```
clc; close all; clear all;
global t x
t = [0 0.3 0.6 0.9 1.2 1.5 1.8 2.1];
x = [0.981 1.01 1.12 1.13 2.98 2.58 3.1 5.89];

paramsopt = fminsearch('piecewisefunc', [3 -1.7 1]);
```

c

Optimized Function Parameters:

Slope = 3.2134   intercept = -1.6841   t0 = 0.8468


Data vs best fit model
Position, x(um)
time, t(s)
6
5
4
3
2
1
0
-1
-2
0
0.5
1
1.5
2
2.5
Original data points
Linear fit after t0
Constant fit before t0

8.5 Friction Co-efficient of an ellipsoid moving through a fluid:

Here, the function,

$$f = \frac{4\pi\mu a}{\ln\left(\frac{2a}{b}\right) - \frac{1}{2}}$$

where, $\mu = 1\ \mu g/(s .\ \mu m)$

$b = 1 \mu m$

ⓐ Golden search section method:

$$f = \frac{4\pi a}{\ln(2a) - 0.5} \quad ; \text{ since } \mu, b = 1$$

iter #	a	b	x_1	x_2	f(a)	f(b)	x_1	x_2
0	1.5	2.5	1.882	2.118	31.489	28.317	28.65	28.206
1	1.882	2.5	2.118	2.264	28.65	28.317	28.206	28.161
2	2.118	2.5	2.264	2.354	28.206	28.317	28.161	28.192
3	2.118	2.354	2.208	2.264	28.206	28.192	28.161	28.168

The method was evaluated in the Matlab. The results for different tolerance values are attached below:

Matlab Script:

```
function f = frictionminimization(a)
% Calculates the friction coefficient function

u=1;
b=1;

n = 4*pi*u;
d = log(2*a/b) - 0.5;

f = n*a/d;
```

Golden Search Result:

```
>> goldensectionsearch('frictionminimization', [1.5 2.5], 0.01)

a = 2.237621 b = 2.245751 f(a) = 28.159312 f(b) = 28.159350

number of iterations = 10

>> goldensectionsearch('frictionminimization', [1.5 2.5], 0.001)

a = 2.240273 b = 2.241006 f(a) = 28.159284 f(b) = 28.159283

number of iterations = 15

>> goldensectionsearch('frictionminimization', [1.5 2.5], 0.0001)

a = 2.240793 b = 2.240859 f(a) = 28.159283 f(b) = 28.159283

number of iterations = 20

>> goldensectionsearch('frictionminimization', [1.5 2.5], 0.00001)

a = 2.240839 b = 2.240849 f(a) = 28.159283 f(b) = 28.159283

number of iterations = 24
```

(b) Newton's Method :

Here, $f(a) = \dfrac{4\pi\mu a}{\ln\left(\frac{2a}{b}\right) - 0.5}$

$$f'(a) = \frac{4\pi\mu a}{\ln\left(\frac{2a}{b}\right) - 0.5} - \frac{4\pi\mu}{\left[\ln\left(\frac{2a}{b}\right) - 0.5\right]^2}$$

$$f''(a) = -\frac{4\pi\mu}{\left[\ln\left(\frac{2a}{b}\right) - 0.5\right]^2 \cdot a} + \frac{2\times 4\pi\mu}{\left[\ln\left(\frac{2a}{b}\right) - 0.5\right]^3 \cdot a}$$

Here with intial $a = 1.5$ and tolerance $= 0.01$,

iter #	x_i	$f'(x_i)$	$f''(x_i)$	x_{i+1}
0	1.5	-14.076	54.732	1.757
1	1.757	-5.334	20.505	2.017
2	2.017	-1.649	9.605	2.19
3	2.19	- 0.309	6.308	2.238

The method was evaluated in Matlab. Corresponding matlab script and results are shown below

Matlab Script:

```
function [df ddf] = frictionminimization(a)
% Calculates the first and second derivatives of the friction
coefficient function

u=1;
b=1;

n = 4*pi*u;
d = log(2*a/b) - 0.5;

df = n/d - n/(d^2);
ddf = -n/(d^2*a) + (2*n)/(d^3*a);
```

Newton’s Method Results:

```
>> newtons1Doptimization('frictionminimization', 1.5, 0.01)

 i  x(i)   f'(x(i))   f''(x(i))

 1  1.757183  -5.333807  20.505374

 2  2.017301  -1.649017  9.605132

 3  2.188982  -0.308540  6.308003

 4  2.237894  -0.016601  5.644948

 5  2.240835  -0.000054  5.607994
```

```
>> newtons1Doptimization('frictionminimization', 1.5, 0.001)

 i  x(i)   f'(x(i))   f''(x(i))

 1  1.757183  -5.333807  20.505374

 2  2.017301  -1.649017  9.605132

 3  2.188982  -0.308540  6.308003

 4  2.237894  -0.016601  5.644948

 5  2.240835  -0.000054  5.607994

 6  2.240845  -0.000000  5.607873
```

```
>> newtons1Doptimization('frictionminimization', 1.5, 0.0001)

 i   x(i)    f'(x(i))    f''(x(i))

 1  1.757183  -5.333807  20.505374

 2  2.017301  -1.649017  9.605132

 3  2.188982  -0.308540  6.308003

 4  2.237894  -0.016601  5.644948

 5  2.240835  -0.000054  5.607994

 6  2.240845  -0.000000  5.607873
```