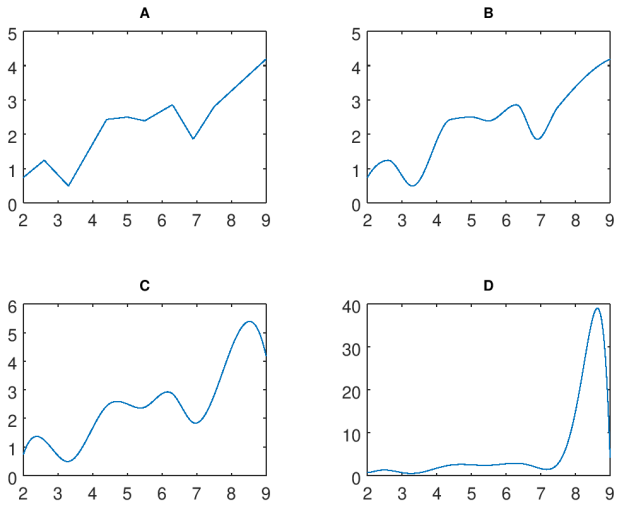
Practice final exam problems with solutions

1

The subplots below show different interpolating functions fit to the same set of 10 points, which have x coordinates ranging from 2 to 9. Identify each function plotted and briefly explain how you know. The options are polynomial interpolation, linear spline, cubic spline, PCHIP, or none of those (only select none if you are sure that the other options are all impossible).



Solution: A: linear spline (points are connected by straight-line segments)
B: PCHIP (points are connected by curves that don't over or undershoot the points)
C: spline (points are connected by a smoother curve than PCHIP; more oscillation than PCHIP evident near $x = 5$ and $x = 8$)
D: polynomial (the most oscillatory, as indicated by the much larger y axis range compared to the original points and the other interpolation methods)

2

Write a Python script to draw a smooth-looking graph of a function that goes through the following points:

x	0	1	3	4	6
y	7	8	5	3	2.

Solution: You need to use a smooth interpolation method, such as the cubic spline, to generate enough points that their graph looks smooth. For example:

```
from numpy import linspace
from scipy.interpolate import CubicSpline
from matplotlib.pyplot import plot
x = [0, 1, 3, 4, 6]
y = [7, 8, 5, 3, 2]
xx = linspace (0, 6, 200)
sp = CubicSpline(x, y, bc_type = 'natural')
yy = sp(xx)
plot (x, y, 'o', xx, yy)
```

3

Given the points

x_i	-2	-1	0	1	2
y_i	16	6	4	4	0,

estimate $y(x = 1.3)$ using (a) polynomial interpolation through all the given points, (b) polynomial interpolation through the three closest points, (c) piecewise linear interpolation.

Solution: (a) The interpolating polynomial is $y = -x^3 + x^2 + 4$. Its value at $x = 1.3$ is **3.493**
(b) The interpolating polynomial is $y = -2x^2 + 2x + 4$. Its value at $x = 1.3$ is **3.22**
(c) The interpolating polynomial piece is $y = -4x + 8$. Its value at $x = 1.3$ is **2.8**

4

Given the points

x_i	-2	-1	0	1	2
y_i	1	-2	1	-2	1,

estimate $y(x = 1.2)$ using (a) polynomial interpolation through all the given points, (b) polynomial interpolation through the three closest points, (c) piecewise linear interpolation.

Solution: (a) The interpolating polynomial through the points is $x^4 - 4x^2 + 1$, which at $x = 1.2$ has the value -2.6864
(b) The interpolating polynomial through the 3 closest points is $3(x-1)^2 - 2$, which at $x = 1.2$ has the value -1.88
(c) Linear interpolation using the $x = 1$ and $x = 2$ values gives -1.4

5

(a) Given the (x, y) values $(1, 2), (2, 4), (3, 10), (5, 7)$ write in matrix form the linear system to solve for the b_i coefficients in the interpolating cubic spline with natural boundary conditions.

(b) If an interpolating cubic spline is as given below, find (1) what points it interpolates and (2) its value at $x = 0$.

$$-1.27(x+1)^3 + 6.54(x+1) - 2 \text{ for } -1 \le x \le 1$$
$$9.15(x-1)^3 - 7.57(x-1)^2 - 8.59(x-1) + 2 \text{ for } 1 \le x \le 2$$
$$-6.63(x-2)^3 + 19.89(x-2)^2 + 3.74(x-2) - 6 \text{ for } 2 \le x \le 3$$

Solution:

(a) Similar to a homework problem, the linear system can be written in the form

$$\begin{bmatrix} 1 & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 \\ 0 & h_2 & 2(h_2+h_3) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 3(\Delta_2 - \Delta_1) \\ 3(\Delta_3 - \Delta_2) \end{bmatrix}$$

which is

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 6 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 12 \\ -22.5 \end{bmatrix}$$

(b) (1) The x coordinates of the interpolated points are given by the segment endpoints and the y coordinates by the spline function values at those points. So the points are $(-1, -2), (1, 2), (2, -6), (3, 11)$.
(2) $x = 0$ is in the first segment, and the value of the first cubic polynomial there is **3.27**

6

Consider the problem of fitting a regression equation of the form $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 xz$ to the following data:

x	0	1	2	3	4
z	0.2	1	1.5	2	0.8
y	33	41	54	75	96

(a) Give the design matrix for the problem.
(b) Write code in Python to find the least-squares parameter values.

Solution: (a) The design matrix, which when multiplied by the vector of unknown parameters $\boldsymbol{\beta}$ gives the predicted y values for the given predictor values, is

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 3 & 9 & 6 \\ 1 & 4 & 16 & 3.2 \end{pmatrix}$$

(b)

```
from numpy import array, ones, transpose
from numpy.linalg import lstsq
x = array([0, 1, 2, 3, 4])
z = [0.2, 1, 1.5, 2, 0.8]
y = [33, 41, 54, 75, 96]
A = [ones(len(x)), x, x**2, x*z]
A = transpose(A)
beta = lstsq(A, y, rcond=None)
beta[0] #least-squares parameter values
```

7

You want to approximate the data below with the function $f(t) = c_1 + c_2 e^t + c_3 e^{-t}$.

t	0	1	2	3	4	5
y	0	2	5	2	9	1

Find $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$ for the system of normal equations $\mathbf{A}^T\mathbf{A}\mathbf{c} = \mathbf{A}^T\mathbf{b}$. You don't need to actually solve for the least-squares values of $\mathbf{c}$.

Solution: The design matrix $\mathbf{A}$ would have columns corresponding to the basis functions $\{1, e^t, e^{-t}\}$, while $\mathbf{b}$ would be the given y values. The normal equations then are

$$\begin{pmatrix} 6 & 234.20 & 1.5781 \\ 234.20 & 25474 & 6 \\ 1.5781 & 6 & 1.1565 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 19 \\ 722.35 \\ 1.6836 \end{pmatrix}.$$

8

An air pollution sensor emits a signal (voltage, V) that depends on the pollutant concentration c (parts per million, ppm) following the relationship $c \approx \beta_1 V +$

$\beta_2 V^2$. Given the calibration data below, use least squares regression to estimate β_1 and β_2.

V	0	1	2	3	4	5
c (ppm)	0	2	4	6	10	12

Solution:

We can find the least squares parameter values by solving the normal equations with the design matrix $\mathbf{A}$ having the V_i values in its first column and their squares in its second column. The system of normal equations $\mathbf{A}^T\mathbf{A}\beta = \mathbf{A}^T\mathbf{c}$ is $\begin{bmatrix} 55 & 225 \\ 225 & 979 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 128 \\ 532 \end{bmatrix}$. The result is $\beta_1 = 1.74, \beta_2 = 0.14$

9

Suppose that the data below are from an experiment on strength S of a concrete mix as a function of percent cement p.

% cement	1	2	3	5	7	8
Strength (MPa)	1	2	3	7	11	12

Estimate the strength at 6% cement using the least-squares first-degree polynomial fit to the given data.

Solution: A first-degree polynomial would be of the form $f(p) = b_1 + b_2 p$, where p is the percent cement. To find the values of b_1, b_2 such that $f(p)$ is as close as possible (in a least squares sense) to the given strengths, we can solve the normal equations with the design matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 5 \\ 1 & 7 \\ 1 & 8 \end{pmatrix}$$

and right-hand side are the strength values,

$$\mathbf{S} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 7 \\ 11 \\ 12 \end{pmatrix}$$

. The system of normal equations $\mathbf{A}^T\mathbf{A}\mathbf{b} = \mathbf{A}^T\mathbf{S}$ is then

$$\begin{pmatrix} 6 & 26 \\ 26 & 152 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} 36 \\ 222 \end{pmatrix}, \text{with solution } \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} -1.27 \\ 1.68 \end{pmatrix}.$$

So the estimated value at $x = 6$ would be $-1.27 + (1.68)(6) = \mathbf{8.80}$ MPa

10

Suppose that the data below are from an experiment on strength S of a concrete mix as a function of curing time t.

Time	0	1	2	4	8	16	32
Strength	5	6	6	8	12	21	58

(a) Write Python code to find the least-squares values for b_1, b_2 in the linear regression model

$$S(t) = b_1 + b_2 e^{\sqrt{t}}.$$

(b) If instead we used the regression function

$$S(t) = \frac{10}{\beta_1 + e^{\beta_2(t-\beta_3)}},$$

and fitted parameter values $\beta_1 = 0.1, \beta_2 = -0.1, \beta_3 = 6$, find RSS, R^2, adjusted R^2, and AIC score.

Solution: (a)

```
from numpy import array, ones, exp, sqrt, transpose
from numpy.linalg import lstsq
t = [0, 1, 2, 4, 8, 16, 32]
S = [5, 6, 6, 8, 12, 21, 58]
A = [ones(len(t)), exp(sqrt(t))]
A = transpose(A)
beta = lstsq(A, S, rcond=None)
beta[0] #least-squares parameter values
```

(b) To find these, start by determining the regression function value $S^*(t)$ at the given points:

t	0	1	2	4	8	16	32
S	5	6	6	8	12	21	58
S^*	5.20	5.72	6.28	7.57	10.88	21.37	57.38

Now RSS is the sum of squares of the residual vector $S - S^*$, which is 2.15; R^2 is 1− RSS/TSS, which is 0.99898 (TSS is 2106). With $n = 7, m = 3$, we get $R_a^2 = 0.99847$ and AIC = 5.75

11

Suppose that the data below are from an experiment on strain of a composite material as a function of time under load.

Time	0	1	2	4	8	16	32
Strain	35	39	43	52	77	154	474

(a) Write Python code to find the least-squares values for b_1, b_2 in the linear regression model

$$S(t) = b_1 + b_2 e^{t/16}$$

.

(b) If instead these data were fitted with the regression function

$$S(t) = \frac{100}{\beta_1 + e^{\beta_2(t-\beta_3)}}$$

with $\beta_1 = 0.1, \beta_2 = -0.1, \beta_3 = 10$, find RSS, R^2, adjusted R^2, and AIC score.

Solution: (a)

```
from numpy import array, ones, exp, sqrt, transpose
from numpy.linalg import lstsq
t = array([0, 1, 2, 4, 8, 16, 32])
S = [35, 39, 43, 52, 77, 154, 474]
A = [ones(len(t)), exp(t/16)]
A = transpose(A)
beta = lstsq(A, S, rcond=None)
beta[0] #least-squares parameter values
```

(b) To find these, start by determining the regression function value $S^*(t)$ at the given points:

t	0	1	2	4	8	16	32
S	35	39	43	52	77	154	474
S^*	35.48	39.07	43.00	52.03	75.68	154.13	474.38

Now RSS is the sum of squares of the residual vector $S - S^*$, which is 2.15; R^2 is 1− RSS/TSS, which is 0.99999. With $n = 7, m = 3$, we get $R_a^2 = 0.99998$ and AIC = 5.72

12

Suppose a bridge's displacement due to vibration as a function of time follows the equation

$$y(t) = a_1 \sin(\omega t) + a_2 \cos(\omega t).$$

(a) Estimate a_1 and a_2 using linear least squares fitting to the data below (and, e.g., solving the normal equations) if it is known that $\omega = 0.7$
(b) Find RSS, R^2, adjusted R^2, and AIC score for this fit.
(c) Based on your regression model, estimate $y(t = 7)$.
(d) Write Python code to estimate ω along with a_1 and a_2 using nonlinear least squares fitting.

t	1	2	3	4	5
y	71	160	175	107	-11

Solution: (a) The design matrix has as columns $\sin(\omega \boldsymbol{t})$ and $\cos(\omega \boldsymbol{t})$, as follows:

$$\mathbf{A} = \begin{pmatrix} 0.64422 & 0.76484 \\ 0.98545 & 0.16997 \\ 0.86321 & -0.50485 \\ 0.33499 & -0.94222 \\ -0.35078 & -0.93646 \end{pmatrix}$$

The normal equations for the least-square parameter values $\boldsymbol{a}$ are $(\boldsymbol{A}^T \boldsymbol{A})\boldsymbol{a} = (\boldsymbol{A}^T \boldsymbol{y})$, which work out to

2.36652	0.23729	394.175
0.23729	2.63348	-97.366

Solved, this gives $a_1 \approx \mathbf{172}, a_2 \approx \mathbf{-52}$.

(b) The fit is very good. To find RSS, we first need to find $\boldsymbol{r} \equiv \boldsymbol{y} - \boldsymbol{A}\boldsymbol{a} = (0.43\ -0.41\ 0.20\ 0.02\ 0.15)^T$. The sum of squares of this is RSS $= \mathbf{0.41}$. The total sum of squares is that of $\boldsymbol{y} - \bar{y}$, which works out to 22435. $R^2 = 1 - \text{RSS/TSS} = \mathbf{0.99998}$. The adjusted R^2 is $1-(4/3)\times\text{RSS/TSS} = \mathbf{0.99998}$. AIC $= n \log(\text{RSS}/n) + 2pn/(n-p-1) = \mathbf{-2.4801} (n = 5,\ p = 2)$.

(c) $y(7) \approx a_1 \sin(7\omega) + a_2 \cos(7\omega) = \mathbf{-179}$.

(d)

```
from numpy import array, sin, cos, transpose, dot
from numpy.linalg import lstsq, norm
from scipy.optimize import fmin
t = array([1, 2, 3, 4, 5])
y = [71, 160, 175, 107, -11]
A = lambda omega, t: transpose([sin(omega*t), cos(omega*t)])
afit = lambda A: lstsq(A, y, rcond=None)[0]
RSS = lambda omega: norm(y - dot(A(omega, t), afit(A(omega, t)))) **
2
omegafit = fmin(func=RSS, x0=0.7)
```

13

If fluidity f of a concrete mix is supposed to follow temperature T as $\log(f) = b_0 + b_1 T$, estimate b_0 and b_1 using least squares linear regression (setting up and solving the system of normal equations) with the following measurements:

T	10	20	30	40	50
f	17	45	172	453	1572

Solution: The design matrix $\mathbf{A}$ has ones as its first column and $\mathbf{T}$ as its second column. The predicand vector $\mathbf{y}$ is equal to $\log(\mathbf{f})$. The resulting system of normal equations $\mathbf{A}^T\mathbf{A}\mathbf{b} = \mathbf{A}^T\mathbf{y}$ therefore has the augmented matrix

$$\left[\begin{array}{cc|c} 5 & 150 & 25.26 \\ 150 & 5500 & 871.53, \end{array}\right.$$

with solution $b_0 = \mathbf{1.644}, b_1 = \mathbf{0.1136}$

14

An equation from mechanics is $M = A - e\sin(A)$. If M is 26 and e is 0.1, use 2 iterations of Newton's method from a reasonable initial guess to estimate A.

Solution: The equation shows that A differs from M only by the term $e\sin(A)$, which is no more than $|e|$ in absolute value. A good initial guess for A is therefore M. The iterations on $f(A) = A - e\sin(A) - M$ then give $26, 26.081530, \mathbf{26.081255}$

15

(a) Use 5 iterations of Newton's method to find a root of the function $f(x) = x^5 - 2x + 1$ starting with $x_0 = 0$. (b) Estimate the fractional error of your result.

Solution: (a) $f(x) = x^5 - 2x + 1$, $f'(x) = 5x^4 - 2$, and successive iterations give
0
0.5
$0.\overline{518}$
0.518790000907635
0.518790063675881
0.518790063675884
(b) Using the difference between the last two iterations as a measure of the absolute error, we get $\frac{|x_i - x_{i-1}|}{x_i} = 6 \times 10^{-15}$. Note though that even if the last two values are identical to our computer/calculator precision, we should not assume that the fractional error is smaller than machine epsilon or (assuming double precision) $\sim 10^{-16}$.

16

Apply 3 iterations of the false position method to estimate where $x^3 - 3x^2 - x = -1$, for x between 0 and 1.

Solution: In standard form, the problem is $f(x) = x^3 - 3x^2 - x + 1 = 0$. Applying the false position method,
Initialize: $a \leftarrow 1, b \leftarrow 0$ (f is negative at a and positive at b).
Iteration 1: $s = -3, c \leftarrow 1/3, f(c) > 0, b \leftarrow c$
Iteration 2: $s = -3.5556, c \leftarrow 0.4375, f(c) > 0, b \leftarrow c$
Iteration 3: $s = -3.6836, c \leftarrow \mathbf{0.45705}, f(c) > 0, b \leftarrow c$

17

Given the system of equations

$$x^3 = y^2$$
$$x + y = 7,$$

carry out an iteration of Newton's method to estimate x and y. Use the initial values $x_0 = 2, y_0 = 4$

Solution: Writing the system in standard form as
$x^3 - y^2 = 0$
$x + y - 7 = 0$,
the Jacobian matrix is

$$\mathbf{J} = \begin{pmatrix} 3x^2 & -2y \\ 1 & 1 \end{pmatrix}.$$

Setting $\mathbf{x} \equiv \begin{pmatrix} x \\ y \end{pmatrix}$, one iteration involves

$$\mathbf{x}_1 = \mathbf{x}_0 - \mathbf{J}(\mathbf{x}_0) \backslash \mathbf{f}(\mathbf{x}_0) = \begin{pmatrix} 2 \\ 4 \end{pmatrix} - \begin{pmatrix} 12 & -8 \\ 1 & 1 \end{pmatrix} \backslash \begin{pmatrix} -8 \\ -1 \end{pmatrix} = \begin{pmatrix} 2.8 \\ 4.2 \end{pmatrix}$$

18

Set up and do 1 iteration of Newton's method for finding an intersection point of the curves $x^2 + y^3 = 10$ and $y + y^2 = \sin(x)$, starting from $(x, y) = (1, 2)$.

Solution: With $v_1 \equiv x, v_2 \equiv y$, the system of equations to solve can be written as $\mathbf{f}(\mathbf{v}) = \begin{pmatrix} v_1^2 + v_2^3 - 10 \\ v_2 + v_2^2 - \sin(v_1) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, and the Jacobian matrix is $\mathbf{J}(\mathbf{v}) = \begin{pmatrix} 2v_1 & 3v_2^2 \\ -\cos(v_1) & 1 + 2v_2 \end{pmatrix}$. Using the given starting point of $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$, we

have $\mathbf{v}_1 = \mathbf{v}_0 - \mathbf{J}(\mathbf{v}_0)\backslash\mathbf{f}(\mathbf{v}_0) = \begin{pmatrix} 1 \\ 2 \end{pmatrix} - \begin{pmatrix} 2 & 12 \\ -0.54030 & 5 \end{pmatrix}^{-1} \begin{pmatrix} -1 \\ 5.1585 \end{pmatrix} = \begin{pmatrix} \mathbf{5.0587} \\ \mathbf{1.4069} \end{pmatrix}$.

[Doing a few more iterations will in fact result in convergence to the solution $x = 3.162279, y = -0.021132$]

19

Estimate where

$$f(x) = 2e^x - 3x$$

reaches a minimum, using 3 iterations of golden section search. Assume that the minimum lies between $x = 0.4$ and $x = 1$

Solution: The iterations can be done as

	a	b	c	d	$f(c)$	$f(d)$
1	0.4	1	0.77082	0.62918	2.0106	1.8646
2	0.4	0.77082	0.62918	0.54164	1.8646	1.8127
3	0.4	0.62918	0.54164	0.48754	1.8127	1.7872
4	0.4	0.54164	0.48754,			

giving as our best guess the current c, **0.48754**, with a possible range of $0.4 - 0.542$

20

The deflection of a beam under loading is described by $y(x) = x^5 - 2x^3 + x$. Use 4 iterations of golden section search with starting values $a = 0.1, b = 0.8$ to estimate for what x the deflection is maximum.

Solution: With $a = 0.1, b = 0.8$, 4 iterations give $a = 0.37, b = 0.47$, and the position of maximum displacement can be estimated as 0.42 ± 0.05

21

(a) Estimate where $f(x) = -2e^{-x} - 2x^2$ reaches a maximum using 6 iterations of golden section search. Assume that the maximum lies between $x = 0.2$ and $x = 0.6$
(b) Use 1 iteration of Newton's method to improve your estimate from (a).
(c) Write Python code to solve this problem using `fmin`, with an initial guess of $x = 0.4$

Solution: (a) The iterations can be done as

	a	b	c	d	$f(c)$	$f(d)$
1	0.2	0.6	0.44721	0.35279	−1.6788	−1.6544
2	0.2	0.44721	0.35279	0.29443	−1.6544	−1.6633
3	0.44721	0.29443	0.35279	0.38885	−1.6544	−1.6581
4	0.29443	0.38885	0.35279	0.33050	−1.6544	−1.6556
5	0.38885	0.33050	0.35279	0.36656	−1.6544	−1.6550
6	0.33050	0.36656	0.35279		−1.6544,	

giving as our best guess the current c, **0.35279**, with a possible range of $0.330 - 0.367$
(b) Using our result from (a) as a starting point, we have

$$x = c - \frac{2e^{-x} - 2 \cdot 2 \cdot c}{-2e^{-x} - 2 \cdot 2} = \mathbf{0.35173}$$

(c)

```
from math import exp
from scipy.optimize import fmin
f = lambda x: 2*exp(-x) + 2 * x**2
x = fmin(func=f, x0=0.4)
```

22

(a) Estimate the minimum of $f(x) = e^x - 2x$ using 5 iterations of golden section search assuming that it lies between $x = 0$ and $x = 2$.
(b) Estimate the same minimum using 2 iterations of Newton's method.

Solution: (a) For $f(x) = e^x - 2x, a = 0, b = 2$, the iterations can be done as

	a	b	c	d	$f(c)$	$f(d)$
1	0	2	1.2361	0.76393	0.96992	0.61884
2	0	1.2361	0.76393	0.47214	0.61884	0.65914
3	1.2361	0.47214	0.76393	0.94427	0.61884	0.68240
4	0.47214	0.94427	0.76393	0.65248	0.61884	0.61534
5	0.47214	0.76393	0.65248	0.58359	0.61534	0.62528
6	0.76393	0.58359	0.65248		0.61534,	

giving as our best guess the current c, **0.65248**, with a possible range of 0.58359 to 0.76393
(b) Starting with the best guess from (a) and with $x_{i+1} = x_i - f'(x_i)/f''(x_i) = x_i - (e^{x_i} - 2)/e^{x_i} = x_i - 1 + 2/e^{x_i}$, we have

	x
0	0.65248
1	0.69399
2	**0.69315**.