Practical Numerical Mathematics CTB2000(Q4)

To be filled in by student:

- *Full completion of this form is obligatory*
- *Every report must be handed in separately*
- ***Do not extend this form to add an extra (third) student***

Number (code) of assignment: 5R4 (one assignment only)
Description of activity: H2 & H3 (e.g. CT, group 2)

Report on behalf of (maximum two students):

Pieter van Halem	name
4597591	student number

Dennis Dane	name
4592239	student number

This must be the first page of your report, to be uploaded using Blackboard.

So, print the cover sheet, fill in and scan together with the (handwritten) theory part of your report.

Data of student taking the role of contact person:

Pieter van Halem	name
pietervanhalem@hotmail.com	email address

Not to be filled in by students:

Receipt:	
date of receipt:	x
corrected by:	x

x = to be filled in by administration
o = to be filled in by corrector

If approved:	
date and initials:	o

If to be adapted:	
date and initials:	o
date of return to student:	x
date of receipt adapted report:	x

Version May 2015

INHOUDSOPGAVE

1.0 PHASE I – THEORY

In this report the behaviour of the following differential equation is described:

$$EI * \frac{d^4y}{dx^4} = q(x), 0 \leq x \leq L \quad (1)$$

The boundary conditions are as follows:

$$y|_{x=0} = 0 \qquad \frac{d^2y}{dx^2}\Big|_{x=0} = 0$$
$$y|_{x=L} = 0 \qquad \frac{d^2y}{dx^2}\Big|_{x=L} = 0 \quad (1 - I)$$

The constant variables are:

L = 10 [m];
B = 0.04 [m];
d = 0.2 [m];
*E = 2*10^{11} [N/m^2];*
ρ = 7800 [kg/m^3];
g = 9.8 [m/s^2];
s = 2 [m];
m = 500 [kg].

1.1

Determine the mass of the beam. Is the additional mass of the same order?

The assignment description yields:

$$q(x) = q_{eig} + q_m(x) \quad (2)$$

In this assignment the q_m is given by:

$$q_m(x) = \begin{cases} 0, & 0 < x < \bar{x} - \frac{s}{2}, \bar{x} + \frac{s}{2} < x < L \\ \frac{m}{s}g, & \bar{x} - \frac{s}{2} < x < \bar{x} + \frac{s}{2} \end{cases}, \bar{x} = \frac{L}{2} \quad (2 - I)$$

The mass of both the beam and the additional load are q_m * L.

$$q_{eig} = \rho * B * d * g \quad (2)$$

Result: *Both masses are of the second Order.*

$$M = \frac{q_{eig} * L}{g} = 7800 * 0.04 * 0.2 * 10 = 624\ kg$$
$$m = 500\ kg \quad (2– II)$$

The mass of the beam is 624 kg this is of order c*10^2, the additional mass is 500 kg this is of the order k*10^2. This means that the additional mass and the weight of the beam of the same order.

1.2, 1.3, 1.4 & 1.5

For clarification a bar is made (fig. 1) with the individual step sizes (h) for visualizing the process.

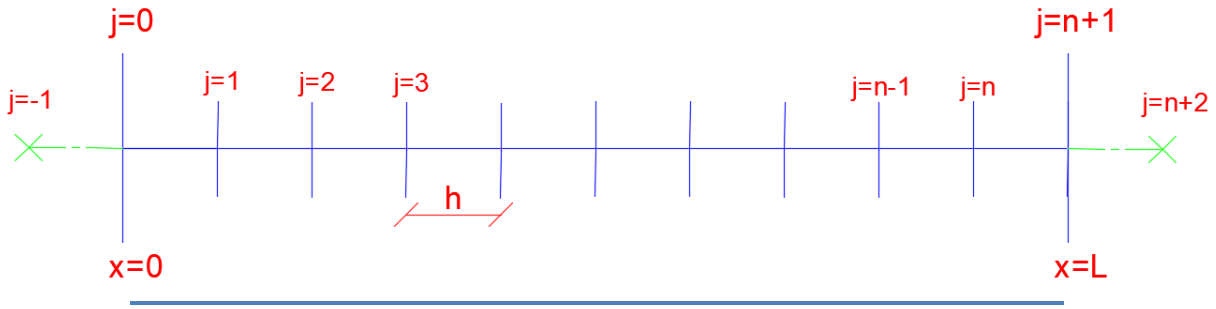


Figure 1 – Visualization of the steps.

Firstly the difference equation is given by:

$$Q(h) \approx EI * \frac{f(x+2h)-4f(x+h)+6f(x)-4f(x-h)+f(x-2h)}{h^4} \rightarrow \frac{EI}{h^4}\left(w_{j-2} - 4 * w_{j-1} + 6w_j - 4 * w_{j+1} + w_{j+2}\right) = q_{j(x)} \quad (3)$$

Secondly the central difference scheme for the second derivatives yields in combination with the boundary conditions w0 = 0 and w_{n+1}=0:

$$\left.\frac{d^2y}{dx^2}\right|_{x=0} = 0 \xrightarrow{yields} f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \rightarrow w_j = \frac{w_{j-1} - 2 * w_j + w_{j+1}}{h^2} \quad (4)$$

Result: $W_{-1} = -W_1$

$$y|_{x=0} = 0 \rightarrow w_0 = 0$$

$$j = 0 \rightarrow w_0 = \frac{w_{-1} - 2 * w_0 + w_1}{h^2} = 0 \quad (4\text{– I})$$

$$w_{-1} = -w_1$$

The same process is repeated for the second boundary condition:

$$\left.\frac{d^2y}{dx^2}\right|_{x=L} = 0 \xrightarrow{yields} f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \rightarrow w_j = \frac{w_{j-1} - 2 * w_j + w_{j+1}}{h^2} \quad (5)$$

Result: $W_{n+2} = -W_n$

$$y|_{x=L} = 0 \rightarrow w_{n+1} = 0$$

$$j = n + 1 \rightarrow w_{n+1} = \frac{w_n - 2 * w_{n+1} + w_{n+2}}{h^2} = 0 \quad (5\text{– I})$$

$$w_{n+2} = -w_n$$

The only thing left to do is to fill in the difference equation for the now known W_{-1}, W_0, W_{n+1} and W_{n+2}. This will be done for j = {1,2,3 … n-2, n-1 and n}.

$$\frac{EI}{h^4}(w_{j-2} - 4 * w_{j-1} + 6w_j - 4 * w_{j+1} + w_{j+2}) = q_j(x) \quad (6)$$

$j = 1 \rightarrow$	$\frac{EI}{h^4}(w_{-1} - 4 * w_0 + 6w_1 - 4 * w_2 + w_3) = q_1(x) \rightarrow \frac{EI}{h^4}(5w_1 - 4 * w_2 + w_3) = q_1(x)$	(6–I)
$j = 2 \rightarrow$	$\frac{EI}{h^4}(w_0 - 4 * w_1 + 6w_2 - 4 * w_3 + w_4) = q_2(x) \rightarrow \frac{EI}{h^4}(-4 * w_1 + 6w_2 - 4 * w_3 + w_4) = q_2(x)$	(6–II)
$j = 3 \rightarrow$	$\frac{EI}{h^4}(w_1 - 4 * w_2 + 6w_3 - 4 * w_4 + w_5) = q_3(x)$	(6–III)
$j = j \rightarrow$	$\frac{EI}{h^4}(w_{j-2} - 4 * w_{j-1} + 6w_j - 4 * w_{j+1} + w_{j+2}) = q_j(x)$	(6–IV)
$j = n - 2 \rightarrow$	$\frac{EI}{h^4}(w_{n-4} - 4 * w_{n-3} + 6w_{n-2} - 4 * w_{n-1} + w_n) = q_{n-2}(x)$	(6–V)
$j = n - 1 \rightarrow$	$\frac{EI}{h^4}(w_{n-3} - 4 * w_{n-2} + 6w_{n-1} - 4 * w_n + w_{n+1}) = q_{n-1}(x) \rightarrow$ $\frac{EI}{h^4}(w_{n-3} - 4 * w_{n-2} + 6w_{n-1} - 4 * w_n) = q_{n-1}(x)$	(6–VI)
$j = n \rightarrow$	$\frac{EI}{h^4}(w_{n-2} - 4 * w_{n-1} + 6w_n - 4 * w_{n+1} + w_{n+2}) = q_n(x) \rightarrow$ $\frac{EI}{h^4}(w_{n-2} - 4 * w_{n-1} + 5w_n) = q_n(x)$	(6–VII)

Now a matrix can be constructed (eq. 7):

$$\begin{bmatrix} 5 & -4 & 1 & 0 & 0 & 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ -4 & 6 & -4 & 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & & & & & \vdots & \vdots \\ 0 & 0 & & & & & & & & & \vdots & \vdots \\ \vdots & \vdots & & & & & & & & & \vdots & \vdots \\ \vdots & \vdots & & & & & & & & & \vdots & \vdots \\ \vdots & \vdots & & & & & & & & & \vdots & \vdots \\ \vdots & \vdots & & & & & & & & & 0 & 0 \\ \vdots & \vdots & & & & & 1 & -4 & 6 & -4 & 1 & 0 \\ 0 & 0 & & & & & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & 0 & \dots & \dots & \dots & \dots & 0 & 0 & 1 & -4 & 6 & -4 \\ 0 & 0 & \dots & \dots & \dots & \dots & 0 & 0 & 0 & 1 & -4 & 5 \end{bmatrix} * \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ \vdots \\ \vdots \\ w_j \\ \vdots \\ \vdots \\ w_{n-3} \\ w_{n-2} \\ w_{n-1} \\ w_n \end{bmatrix} = \begin{bmatrix} q_1(x) \\ q_2(x) \\ q_3(x) \\ q_4(x) \\ \vdots \\ \vdots \\ q_j(x) \\ \vdots \\ \vdots \\ q_{n-3}(x) \\ q_{n-2}(x) \\ q_{n-1}(x) \\ q_n(x) \end{bmatrix} \quad (7)$$

One can simply see that the matrix is symmetric and has n x n dimensions.

1.6

Consider the local truncation error for the difference equations associated with interior grid points, such as presented in question 2. Of what order in h is this local truncation error?

For the numerical method of solving the boundary problem is only one approximation used. This approximation is the approximation for the fourth derivative of y. The approximation that should be used is given in equation (8) and the error of this approximation is given in equation (9) of the assignment. The error of the method is $O(h^2)$. The fact that the method is exact except for this approximation means that the global truncation error of the method $O(h^2)$.

In equation (8) of the document we derived the relation between the local and the global truncation error. This derivations yield that the local truncation error is always smaller than the global truncation error.

In the equations below ε is defined as: **ε** ≤ |**A* y - A w**|, **y** as the vector containing the exact results and **w** the vector containing the approximate result find with the numerical method.

$$\boldsymbol{e = y - w} \tag{8}$$

Result:

$$\boldsymbol{e} \leq O(h^2)$$

$$\boldsymbol{e = y - w = A^{-1} * A * (y - w) = A^{-1} * (A * y - A * w)} \tag{8 – I}$$

$$\boldsymbol{A^{-1} * (A * y - A * w) = A^{-1} * \varepsilon}$$

$$\boldsymbol{\left||e|\right| = \left||A^{-1} * \varepsilon|\right| \leq \left||A^{-1}|\right| * \left||\varepsilon|\right|}$$

$$\boldsymbol{\left||A^{-1}|\right| * \left||\varepsilon|\right| = O(h^2)}$$

$$\boldsymbol{e} \leq O(h^2)$$

1.7

The matrix is already given in eq. 7. (*A short summery, the matrix is symmetric and the dimensions are n x n.*)

There are multiple ways to solve a system of linear equations. This can be done numerically by using the Newton-Rapson method for linear systems.

It can also be done by calculating the inverse of the matrix and multiplying by the results vector. This method is not a numerical approximation but the exact answer. However it becomes a numerical approximation if the inverse is calculated with a numerical method.

The exercise states that the system can be solved numerically, which we explained above. However didn't we create a new function in assignment 11 – 15. Writing such a function is very labour intensive and we think this is outside the curriculum of this practicum. Given the fact that the numpy package is a reliable method, which gives a fast and easy answer.

Which of the methods (algorithms) is used by lg.spsolve is unknown to us. A sparse matrix is a matrix mainly filled with zeros. Therefor it is likely that the function lg.spsolve(sparse matrix solve) is specialized in solving matrixes with many zeros. Hence it is most likely favourable to use lg.spsolve over the numpy.solve in case of large matrixes containing lots of zeros. However we used the method from the numpy package because we made the notebook earlier than this assignment. And the numpy method gives no difficulties. So we choose not to revise the notebook to prevent making small untraceable mistakes.

1.8

Determine the analytical value of Q_a of Q.

$$Q_a = \int_{\bar{x}-\frac{s}{2}}^{\bar{x}+\frac{s}{2}} q_m dx \qquad (8)$$

Result:

$$Q_a = \frac{h * m}{2s}$$

$$Q_a = \frac{h}{2} * 0 + \frac{h}{2} * \frac{m}{s} * g = \frac{h*m}{2s} \qquad (8 - I)$$

$$Q_a = \frac{h * m}{2s}$$

This answer is the exact answer to equation (8).

1.9

Determine the discrete value of Q_d of Q.

$$Q_d = 2 * \frac{h}{2} * q_m^i \qquad (9)$$

Result:

$$Q_d = \frac{h * m * g}{s}$$

$$Q_d = 2 * \frac{h}{2} * \frac{m}{s} * g \qquad (9 - I)$$

This is not the exact value but an approximation of the true integral.

1.10

A natural demand, resulting from a finite volume derivation of the discrete equations, is $Q_d = Q_a$. Verify that this implies that q_m^i must be chosen as the average of the values q_m on the left and right of the interface.

$$Q_a = Q_d \qquad (10)$$

Result:

$$q_m^i = \frac{m * g}{2 * s}$$

$$\int_{\bar{x}-\frac{s}{2}}^{\bar{x}+\frac{s}{2}} q_m dx = 2 * \frac{h}{2} * q_m^i \qquad (10–I)$$

$$q_m^i = \frac{m * g}{2 * s} \qquad (10–II)$$

Solving equation 7 – I yields $q_m^i = \frac{m*g}{2*s}$. This proves the statement in the assignment

Numerical evaluation of the deflection of the beam

Number (code) of assignment: 5R4

Description of activity: H2 & H3

Report on behalf of:

name : Pieter van Halem student number (4597591)

name : Dennis Dane student number (4592239)

Data of student taking the role of contact person:

name : Pieter van Halem

email address : pietervanhalem@hotmail.com

Function definition

In the first block are all the packages imported and constants defined. In the second block are all the functions for the numerical analyses defined. And the third block contains the two function for the bisect method. This is used in assignment 2.13.

In [1]:
```
import numpy as np
import scipy.sparse.linalg as sp_lg
import scipy.sparse as sp
import scipy as scp
import numpy.linalg as lg
import matplotlib.pyplot as plt
%matplotlib inline

EI = 2 * 10 ** 11 * (1/12) * 0.04 * 0.2 ** 3
L = 10
s = 2
xleft = 0.0
xright = L
yleft = 0.0
yright = 0.0
g = 9.8
```

In [2]:
```
def A(h, N):
    d0 = np.ones(N)
    d1 = np.ones(N-1)
    d2 = np.ones(N-2)
    A = (6*np.diag(d0,0) + -4*np.diag(d1,-1) + -4*np.diag(d1,1) + 1*np.diag(d2,-
2) + 1*np.diag(d2,2))
    A[0,0] = 5
    A[N-1,N-1] = 5
    return A * EI/(h ** 4)

def beig(h,N,x,yleft,yright, qM):
    result = qM*np.ones(N)
    return result

def bm(h,N,x,yleft,yright, qm):
    result = np.zeros(N)
    if(((L/2-s/2)/h).is_integer() == True):
        for i in range(int((L/2-s/2)/h - 1),int((L/2+s/2)/h)):
            if (i==int((L/2-s/2)/h - 1) or i == int((L/2+s/2)/h - 1)):
                result[i] = result[i] + qm/2
            else:
                result[i] = result[i] + qm
    return result

def bn(h,N,x):
    result = np.zeros(N)
    for i in range(int((L/2-s/2)/h -1),int((L/2+s/2)/h -1)):
        result[i] = result[i] + 125 * np.pi* g * np.sin(np.pi*((h*(i+1)-4)/2))
    return result


def solve(h,N,x,yleft,yright, k, qM, qm):
    AA = A(h,N)
    if k == 1:
        bb = beig(h,N,x,yleft,yright, qM)
    elif k == 2:
        bb = bm(h,N,x,yleft,yright, qm)
    elif k==3:
        bb = beig(h,N,x,yleft,yright, qM)
        bb = bb + bm(h,N,x,yleft,yright, qm)
    elif k == 4:
        bb = beig(h,N,x,yleft,yright, qM)
        bb = bb + bn(h,N,x)
    y = lg.solve(AA,bb)
    result = np.concatenate(([yleft],y,[yright]))
    return result

def main(N, k, qM = 611.52, qm = 2450.0):
    h = (xright  - xleft)/(N+1)
    x = np.linspace(xleft,xright,N+2)
    y = solve(h,N,x,yleft,yright,k, qM, qm)
    return x,y

def plot(x,y):
    plt.figure("Boundary value problem")
    plt.plot(x,y,"k")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.title("De graph of the function y")
    plt.legend("y", loc="best")

def table(x,y,N):
    print ("{:>4}{:>11}{:>21}".format("k", "x_k", "y(x_k)"))
    for k in range(0, N+2):
        print ("{:4.0f}{:11.2f}{:23.7e}".format(k, x[k], y[k]))
```

In [3]:
```
def func(qm):
    N = 199
    x,y = main(N, 3,611.52, qm)
    return np.max(y) - 0.03

def bisection(func, x1, x2, tol=0.01, nmax=10):
    i = 0
    for i in range(nmax):
        xm = (1/2)*(x1 + x2)
        fm = func(xm)
        if func(xm) * func(x2) <= 0:
            x1 = xm
        else:
            x2 = xm
        i += 1
        if np.abs(func(x1)) < tol:
            break
    if i == nmax:
        a = str('Warning: the nmax is exeeded')
        print(a)
    return x1
```

Assignment 2.11

Choose $h = 1.0$ as grid size and make a table f the obtained numerical approximation of y. The table must give the deflection in 8-digit floating point format.

In [4]:
```
N = 9
x,y = main(N, 3)
table(x,y,len(y)-2)
```

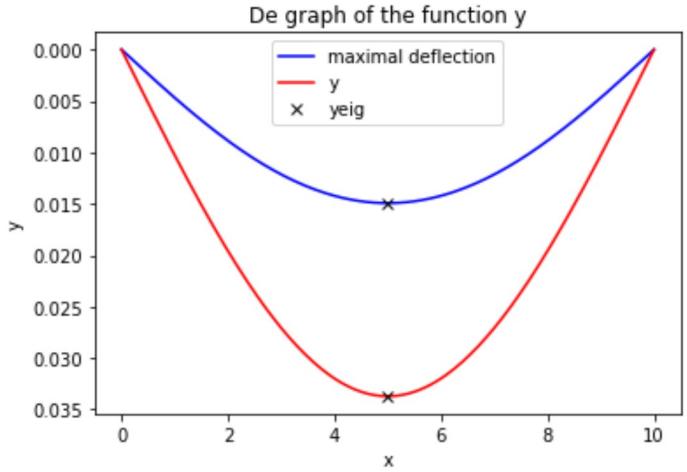
k	x_k	y(x_k)
0	0.00	0.0000000e+00
1	1.00	1.0357069e-02
2	2.00	1.9738792e-02
3	3.00	2.7284486e-02
4	4.00	3.2248125e-02
5	5.00	3.3998344e-02
6	6.00	3.2248125e-02
7	7.00	2.7284486e-02
8	8.00	1.9738792e-02
9	9.00	1.0357069e-02
10	10.00	0.0000000e+00

Assignment 2.12

Take $h = 0.05$. Compute both y and $yeig$. plot the obtained approximation of y as a function of x. plot $yeig$ in the same picture, distinguishing the different graphs visually. Where is the maximal deflection attain? With values take y and yeig at the midpoint of the beam?

In [5]:
```
N = 199
x,y = main(N, 1)
x2,y2222 = main(N, 3)
plt.figure("Boundary value problem")
plt.plot(x,y,"b", x2,y2222,"r")
plt.plot(x[np.argmax(y)],np.max(y),"xk", x2[np.argmax(y2222)],np.max(y2222),"xk"
)
plt.xlabel("x")
plt.ylabel("y")
plt.title("De graph of the function y")
plt.legend({"yeig","y", "maximal deflection"}, loc="best")
plt.gca().invert_yaxis()
plt.show()

print("The maximal deflection of yeig occurs at: x=",x[(np.argmax(y))])
print("The maximal deflection of y occurs at: x=",x2[(np.argmax(y2222))])
print()
print("The deflection in the midpoint of the beam is: yeig(5)= {:.7e}".format(np
.max(y)))
print("The deflection in the midpoint of the beam is: y(5)= {:.7e}".format(np.ma
x(y2222)))
```




```
The maximal deflection of yeig occurs at: x= 5.0
The maximal deflection of y occurs at: x= 5.0

The deflection in the midpoint of the beam is: yeig(5)= 1.4929986e-02
The deflection in the midpoint of the beam is: y(5)= 3.3707370e-02
```

assignment 2.13

Determine the maximal mass m allowed for, i.e. the mass leading to a deflection in the midpoint of the beam with a magnitude 0.03 (see original goal, formulated at the beginning of the assignment).

In [6]:
```
qmopt = bisection(func, 1000, 30000, tol = 1e-15, nmax = 100)
```

In [7]:
```
x,y = main(N, 3, qm =  qmopt)
qmopt = qmopt*2/g
ymaxx = np.max(y)
print("The max value for m is:{:.7e}[kg] the deflection for this m is:{:.7e}".fo
rmat(qmopt, ymaxx))
print("The truncation error is smaller than: 1e-15")
```

```
The max value for m is:4.0128099e+02[kg] the deflection for this m is:3.000000
0e-02
The truncation error is smaller than: 1e-15
```

The maximal load m is obtained with the bisect method. In this method we choose a tolerance of $1e-15$, such that the error is not visible in this notebook. We choose the Bisect method because it always converges. We couldn't use the Newton Raphson method because the derivatives of the function are not known.

the defined functions are given in $ln[3]$

Assignment 2.14

Determine Am such that the total additional mass is again 500 kg. sketch the original load qm, i.e. (6) with m = 500, and the load qm in one figure.

To determine the value of Am we need to solve the following equation:

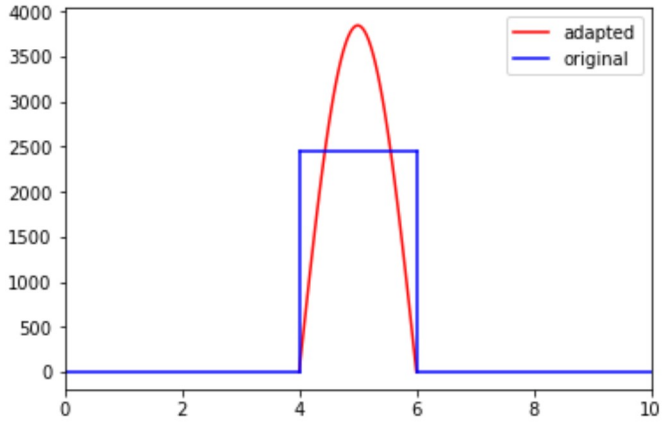
$$\int_{L/2-s/2}^{L/2+s/2} (Am * \sin(\pi * \frac{x-(L/2-s/2)}{s}))dx = 500$$

solving this equation results in:

$$\frac{4}{\pi}Am = 500$$
$$Am = 125\pi$$

In [8]:
```
x = np.linspace(4,6,100)
x2 = 4*np.ones(100)
x3 = 6*np.ones(100)
x4 = np.linspace(0,4,100)
x5 = np.linspace(6,10,100)
y1 = (500 * g / s)*np.ones(100)
y2 = 125 * np.pi* g * np.sin(np.pi*((x-4)/2))
y3 = np.linspace(0,500 * g / s,100)
y4 = np.zeros(100)

plt.plot(x,y2, 'r', x,y1, 'b', x2,y3,'b',x3,y3,'b', x4,y4,'b',x5,y4,'b')
plt.legend({"original","adapted"}, loc="best")
plt.xlim(0, 10);
```



Assignment 2.15

Determine (using $h = 0.05$) the maximal deflection of the beam with the new load. Check whether this value is significantly different from the one obtained in exercise 12.

In [9]:
```
N=199
x,y = main(N, 4)
print("y(L/2) = {:.7e}".format(np.max(y)))
print("y2(L/2)-y1(L/2) = {:.7e} [m] = {:.7e} %".format(np.max(y) - np.max(y2222)
,(np.max(y) - np.max(y2222))/np.max(y) * 100 ) )
```

```
y(L/2) = 3.3853569e-02
y2(L/2)-y1(L/2) = 1.4619928e-04 [m] = 4.3185782e-01 %
```

The deflection increases with approximately 0.14 mm with is 0.43% witch is not significant. However it is very logical that the deflection increases. because the load concentrates more in the centre, a larger moment is caused. This increases the deflection of the beam.