Problem 4

Problem 4 revisits Problem 2, where the optimisation problem was unconstrained and had one shared variable. Recall that the dual function was given by

$$\begin{aligned} q(\lambda) &= q_1(\lambda) + q_2(\lambda) \\ &= \inf_{x,\xi_1}[f_1(x,\xi_1) - \lambda\xi_1] + \inf_{y,\xi_2}[f_2(y,\xi_2) + \lambda\xi_2]. \end{aligned}$$

Dual decomposition and the subgradient method were applied to solve this problem, where we iteratively use a given λ to solve two separate sets of linear equations, $A_1x_1 = B_1$ and $A_2x_2 = B_2$, and obtain ξ_1 and ξ_2, then use ξ_1 and ξ_2 to update λ.

Consider now three agents Agent 1, Agent 2 and Agent 3. Agent 3 computes the updates $\lambda = \lambda - \alpha(\xi_1 - \xi_2)$, then sends λ to both Agent 1 and 2. Agents 1 and 2 respectively solve $A_1x_1 = B_1$ and $A_2x_2 = B_2$, then send ξ_1 and ξ_2 to Agent 3. This is repeated for some specified number of iterations. The figure below shows one iteration of this.

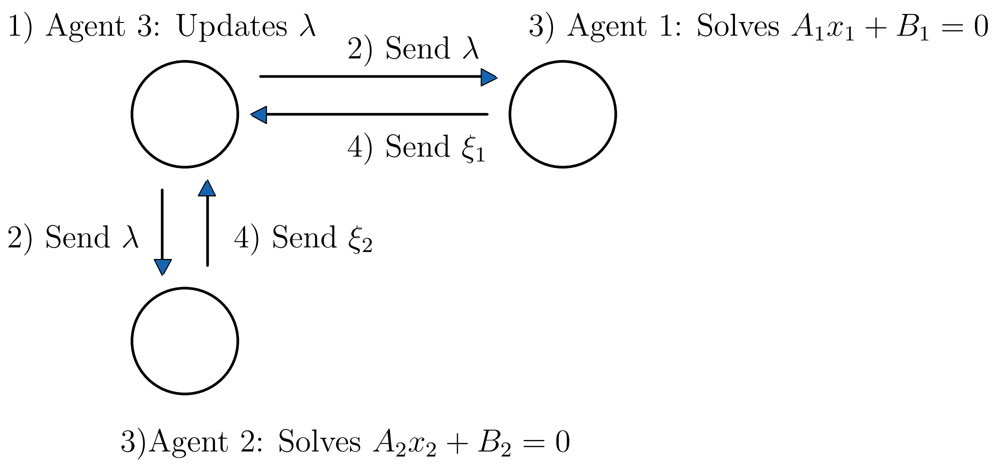
1) Agent 3: Updates λ
2) Send λ
3) Agent 1: Solves $A_1x_1 + B_1 = 0$
4) Send ξ_1
2) Send λ
4) Send ξ_2
3)Agent 2: Solves $A_2x_2 + B_2 = 0$

Figure 1: Steps 1, 2, 3, 4 show the procedure in a single iteration of the subgradient method.

Previously, messages were assumed to have unlimited size, i.e., there is no floating point error when λ, ξ_1 and ξ_2 are sent between Agents 1, 2 and 3. Let's now consider a limit on message size. Assume messages are sent in binary, then there is a limit on the number of bits that can be sent.

Word Limit

The following terms will be used to describe a binary number.

$$\underbrace{\pm}_{\text{sign}}\ \underbrace{10101010101010}_{\text{binary integer}}\ \underbrace{.}_{\text{binary point}}\ \underbrace{1010101010101010}_{\text{binary fraction}}$$

For the example problems generated, it so happens that the variable λ is in general larger than ξ_1 and ξ_2. Messages involving λ require at least 3 binary integer bits, while messages involving ξ_1 and ξ_2 generally converge to binary fractions, i.e., they converge to values smaller than decimal 1.

It makes sense to limit the number of bits used to represent the binary fraction, rather than vary the number of bits used to represent the binary integer. Thus, the word limit referred to below is the limit on the number of bits used to represent the binary fraction of a message.

Convergence

In order to measure convergence, we need a benchmark. Recall from Problem 2 the combined problem, where A_1 and A_2 are overlapped to form A, and B_1 and B_2 overlapped to form B. The combined problem can be solved, providing the required benchmark to measure convergence. Let x^* be the solution to the problem $Ax + B = 0$, let x_1^* be the solution to the problem $A_1x_1 + B_1 = 0$ and let x_1^* be the solution to the problem $A_2x_2 + B_2 = 0$. Overlap x_1^* and x_2^*

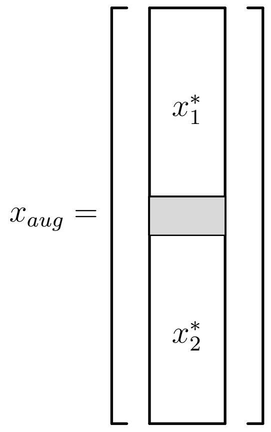

Figure 2: Create x_{aug}, a vector with x_1^* and x_2^* overlapping as shown. Only one element is in the grey overlapping region, and that element is equal to the **mean** of the elements of x_1^* and x_2^* that overlap.

Finally, our measure of convergence will be $||x_{aug} - x^*||_2$.

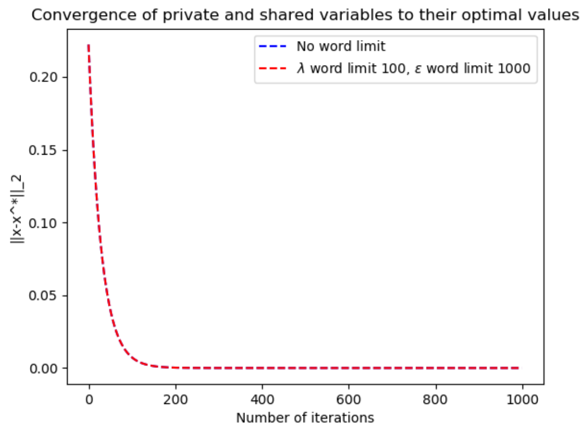

Figure 3: Large word limits.

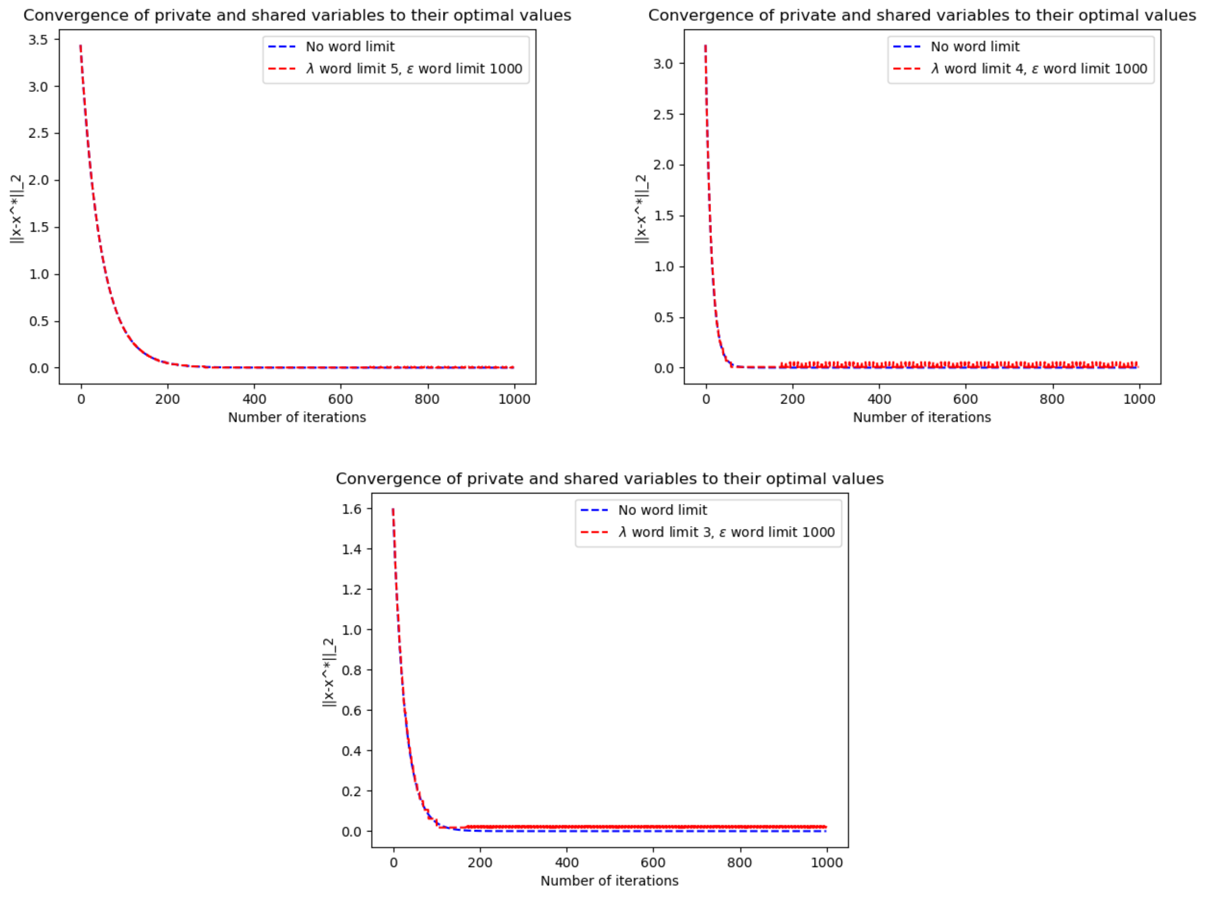

Figure 4: Decreasing the word limit on λ.

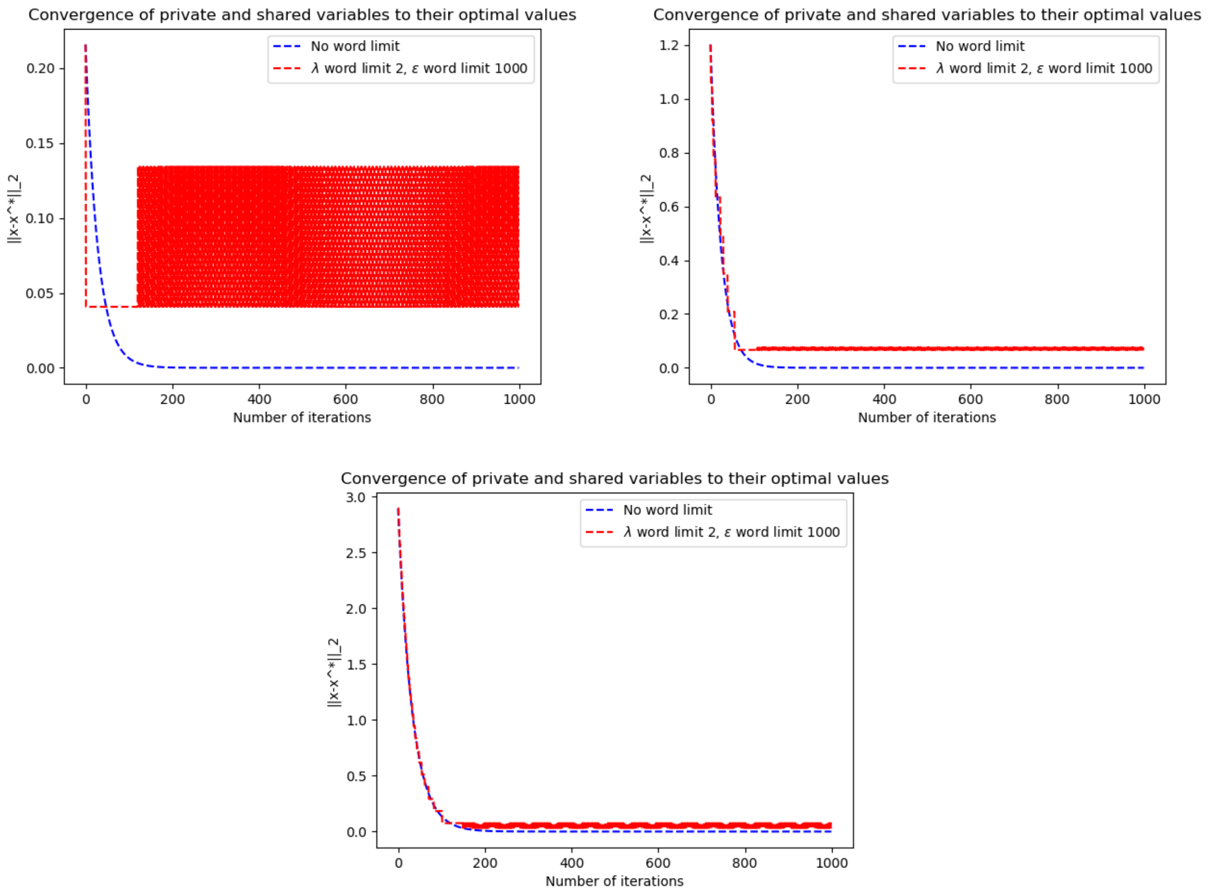

Figure 5: Convergence for a given limit on λ is somewhat dependent on the coefficients of the problem.

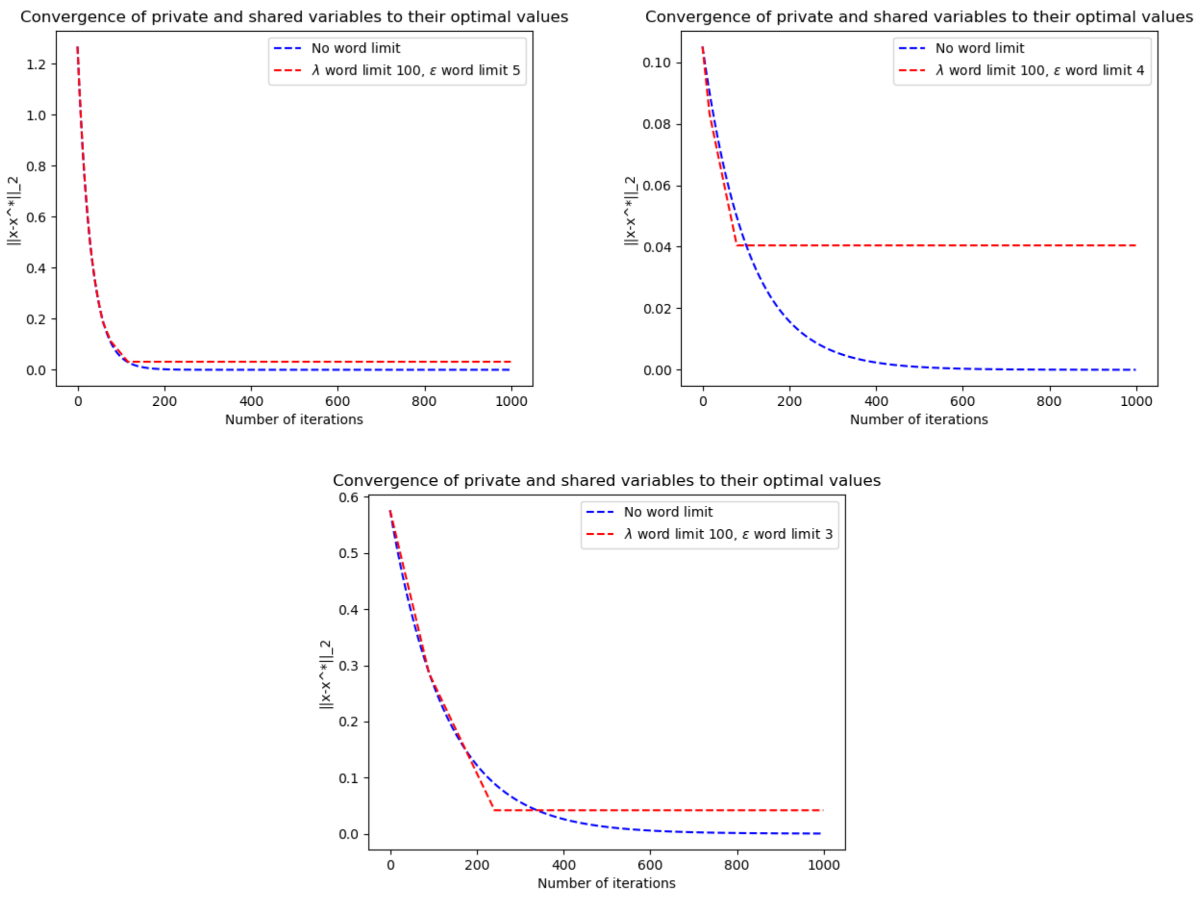

Figure 6: Similarly, convergence for a given limit on ξ is also somewhat dependent on the coefficients of the problem.

Notice low precision on λ causes the steady state error to oscillate between values, while low precision on ξ causes the error to be constant in steady state.

How to Run

The above examples

Make sure you are in the correct directory. Then to run the tests that generated the above plots, execute the **main.py** file, i.e. use the command

```
>>>python main.py
```

Vary lambda_word_limit and eps_word_limit to generate plots with different binary message limits.

Function Descriptions

The function **conversions.float2bin**, description.

Syntax: float2bin(number, places)

Parameter values:

- number, Required. A float variable that is to be converted to binary representation.
- places, Required. Number of bits used to represent the binary fraction of the input number. There is no limit on the number of bits used to represent the binary integer of the input number.

Outputs:

- Output 1. Binary string representation of the input number.

The function **conversions.bin2float**, description.

Syntax: bin2float(bin_str)

Parameter values:

- bin_str, Required. A binary string representation that is to be converted to float.

Outputs:

- Output 1. Float representation of input binary string.

The function **imprecise.do_imprecise**, description.

Syntax: do_imprecise(max_iter,alpha,A1,A2,b1,b2,xstar,

lambda_word_limit,xi_word_limit,verbose=False)

Parameter values:

- max_iter, Required. Number of iterations for the subgradient method.
- alpha, Required. Step size for the subgradient method.
- A1, Required. The matrix of coefficients A_1 as described above.

- A2, Required. The matrix of coefficients A_2 as described above.
- b1, Required. The matrix of coefficients B_1 as described above.
- b2, Required. The matrix of coefficients B_2 as described above.
- xstar, Required. True solution to the problem.
- lambda_word_limit, Required. Word limit of λ.
- xi_word_limit, Required. Word limit of ξ.
- verbose, Default False. Print results to screen.

Outputs:

- Output 1. The measure of convergence described above, i.e., $||x_{aug} - x^*||_2$.