Jordi Castro Pérez

Pràctiques d'Optimització de Processos

Universitat Rovira i Virgili

Pràctiques d'Optimització de Processos

Jordi Castro Pérez

Estadística i Investigació Operativa

Dept. d'Enginyeria Química

Universitat Rovira i Virgili

Introducció

Aquest document presenta les diferents pràctiques de la part d'Optimització de l'assignatura Simulació i Optimització de Processos. Aquestes pretenen ajudar a fixar les nocions adquirides a les classes de teoria, tot experimentant amb diversos paquets i algorismes d'Optimització. Les pràctiques es realitzaran en grups de tres alumnes i hauran de ser lliurades abans de la data que s'indicarà a les classes de teoria.

El primer que cal fer notar és que no totes elles han de ser realitzades obligatòriament pels alumnes. Cada pràctica val una sèrie de punts (aquests s'indiquen a l'inici de l'exposició de cada una d'elles), i serà avaluada sobre 10 (per exemple, una pràctica que valgui 3 punts i on l'alumne obtingui una nota de 5 li suposarà 1.5 punts). L'alumne pot fer el nombre d'elles que consideri necessari, sempre i quan obtingui una nota final de pràctiques superior a 5 punts. Les notes de pràctiques inferiors a 5 punts no tindran dret a fer promig amb la nota de teoria. La nota màxima que es pot obtenir és de 10 punts. La única restricció que hi ha sobre les pràctiques a fer és que, com a mínim, heu de lliurar una de les dues pràctiques següents: "Optimització d'un procés d'alquilació" o "Optimització d'un sistema turbo-generador".

Cada una de les pràctiques ha de ser realitzada amb un d'aquests tres paquets: Lingo, Octave o lp_solve. Lingo és un paquet comercial especialment adequat per a la modelització i solució de problemes generals d'Optimització. El seu ús és molt senzill. La versió que haureu d'usar corre sobre MS-Windows. Octave, per la seva banda, és un paquet de lliure distribució (no cal pagar per ell) pensat per realitzar (i programar) operacions amb vectors i matrius de forma molt senzilla. Un programa de càlcul matricial que en Fortran o C portaria moltes hores de feina pot fer-se en qüestió de minuts. Funciona només sobre entorns Unix. Finalment, lp_solve és també un paquet de lliure distribució (per usos acadèmics només), que només necessita d'un compilador de C per poder funcionar (pot ser instal·lat, doncs, en qualsevol tipus de sistema pràcticament). És un paquet que permet solucionar problemes de programació lineal i entera. Al final dels enunciats de les pràctiques trobareu tres annexes on es fa una breu descripció d'aquests tres paquets per tal que pogueu utilitzar-los.

L'estructura i continguts de l'informe a presentar dependrà de cada pràctica. Tot i així, i en la mesura que us sigui possible, es recomana que en general tots els informes tinguin els següents apartats:

- Un primer full on s'indiqui el nom de la pràctica, el nom i cognoms dels tres components del grup, i la data de lliurament. Si disposeu d'adreça de correu electrònic, indiqueu-la també.
- Un primer apartat on es faci un molt breu resum dels objectius de la pràctica.
- Un segon apartat on es detalli la tasca realitzada i el com i per què heu fet les coses.
- Un nou apartat on presenteu un llistat dels programes o codi introduït al paquet usat en la pràctica.
- Un apartat on presenteu la sortida/solució del paquet, tot comentant els resultats obtinguts, i qualsevol cosa que creieu convenient destacar.
- Unes conclusions finals.

Tarragona, febrer de 1997.

Índex

PRÀCTICA 1. Optimització d'un procés d'alquilació (4.5 punts).

En tot procés d'obtenció industrial de productes químics existeix una escala òptima determinada per la quantitat de producte a fabricar i les mides i qualitat de l'aparellatge (reactors, fraccionadors, dipòsits etc.) a utilitzar, de forma que els beneficis econòmics obtinguts (diferència entre el valor del producte obtingut i els costos de producció) siguin màxims. Els valors òptims de les quantitats de producte, i mides i qualitat dels aparells poden ser obtinguts resolent un problema d'optimització que cal plantejar i on cal introduir com a restriccions les equacions que descriuen el procés químic que s'està tractant.

1.1 Presentació del problema.

Efectuarem aquí la determinació dels valors òptims a emprar dins un procés d'alquilació, usat per obtenir gasolina. Més concretament, al procés d'alquilació es combinen hidrocarburs amb pocs carbonis (per exemple, una olefina i isobutà) per tal d'obtenir un producte amb un més gran nombre de carbonis (l'alquilat). Com a exemple d'una reacció d'aquest tipus tenim: *Isobutà + Isobutilè → Isooctà*. Un àcid és normalment emprat com a catalitzador positiu per afavorir aquesta reacció.

Una característica important de l'alquilat obtingut, la qual modifica el seu valor, és l'octanatge. L'octanatge és un índex que reflexa la facilitat que té l'alquilat per detonar a l'interior del cilindre d'un motor quan és comprimit abans de saltar la guspira de la bugia. L'índex 100 d'octanatge és assignat als hidrocarburs que tenen una facilitat de detonació igual que la de l'isooctà (que en té poca). L'índex 0 és assignat als que tenen la mateixa facilitat que l'heptà (que en té molta). La gasolina super d'automoció té un índex de 98.

La Fig. 1.1 mostra de forma esquemàtica com es duu a terme l'obtenció de l'*alquilat*. Bàsicament hi ha dos elements principals: un reactor (a l'esquerra) i un fraccionador (a la dreta). Tres són els productes d'entrada al reactor: l'*olefina*, l'*isobutà* i l'*àcid* que actua com a catalitzador. Un cop han reaccionat, una combinació d'olefina no reaccionada, isobutà no reaccionat (referenciats com $olefina_n$ i $isobuta_n$ a la Fig. 1.1) i *alquilat* ja produït passa al fraccionador. De l'àcid usat com a catalitzador part s'expulsa i part queda dins el reactor com àcid residual, però mai passa al fraccionador (considerem que no es recircula automàticament). En el fraccionador el producte d'entrada és escalfat i separat en les seves parts (segons el punt d'ebullició diferent de cada una d'elles). Llavors, de la combinació d'*olefina*, *isobutà* i *alquilat* entrat al fraccionador, l'isobutà i una part de l'olefina són tornats al reactor (i els anomenarem *olefina recirculada* i *isobutà recirculat*), mentre que l'*alquilat* barrejat amb l'altra part d'olefina que no s'ha pogut separar del tot (anomenada $olefina_f$ a la Fig. 1.1) s'obtenen com a producte final. Si l'*alquilat* final no tingués gens d'olefina diríem que és totalment pur.

1.2 Modelització del problema.

L'objectiu final de tot aquest procés serà la determinació de, per una banda les quantitats òptimes de tots els productes que intervenen a la reacció, i per altra, de les característiques (qualitat, mida etc.) tant del reactor com del fraccionador. I tot això intentant maximitzar els

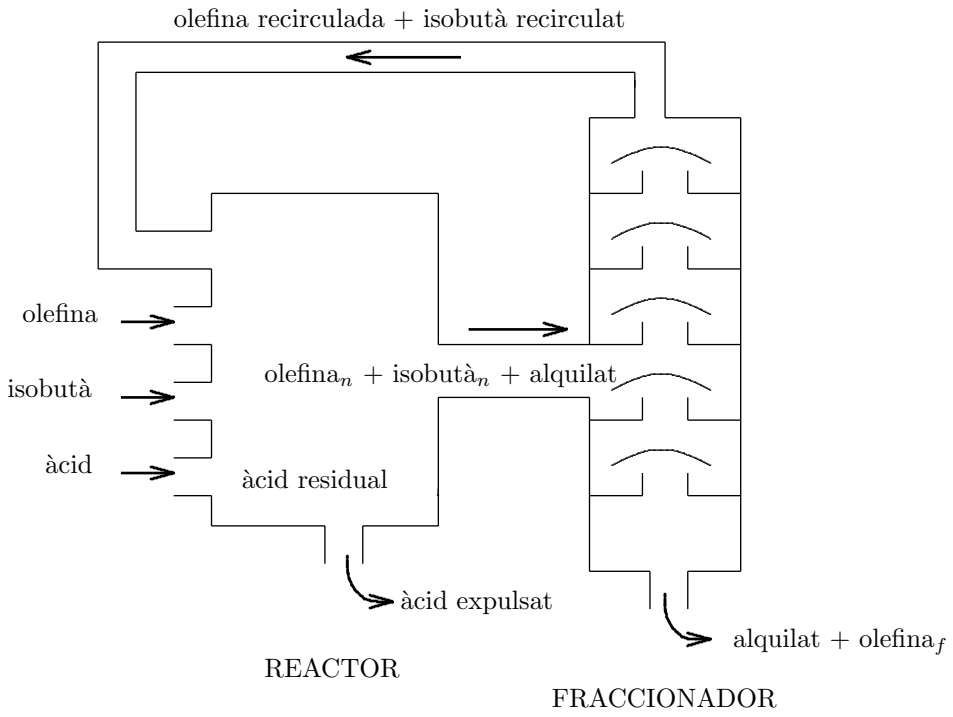

Figura 1.1. Esquema del procés d'alquilació.

beneficis nets obtinguts durant el procés.

Començarem primer modelitzant les equacions que ens representaran el procés pròpiament dit; aquestes seran les restriccions del nostre problema. En segon lloc descriurem l'equació que ens servirà de base per tal de formular la nostra funció objectiu a maximitzar. A més considerarem que a la Fig. 1.1 l'entrada d'olefina i isobutà, així com la seva recirculació, i la sortida d'alquilat i olefina del fraccionador es mesuren en nombre de barrils per dia, mentre que l'entrada d'àcid al sistema es mesura en nombre de tones per dia.

1.2.1 Restriccions.

Un dels valors que, com veurem més tard, intervé en l'expressió de diverses equacions és la relació o ratio que existeix entre les quantitats d'isobutà i olefina al reactor. Com s'observa a la Fig. 1.1, l'olefina total que hi ha dins el reactor és l'olefina d'entrada més l'olefina recirculada. Anàlogament, l'isobutà total és la suma de l'alimentació inicial d'isobutà més el recirculat. Per tant aquesta relació entre isobutà i olefina al reactor s'expressa com:

$$relació = \frac{isobutà + isobutà\_recirculat}{olefina + olefina\_recirculada} \tag{1.1}$$

Una altra restricció clara que cal tenir en compte és la de conservació de la massa a tot el procés. Si s'observa la Fig. 1.1 es veu que de tot el que entra al reactor (isobutà, isobutà recirculat, olefina, olefina recirculada i àcid) una part ja surt directament pel reactor (l'àcid), i la resta surt pel fraccionador. Del que surt del fraccionador, una part és la mateixa que entrava al reactor (olefina i isobutà recirculats), mentre que l'altra surt directament a l'exterior. Per

tant, a l'hora de tenir en compte un balanç de masses només hem d'incloure l'olefina i isobutà entrats (no els recirculats) i l'alquilat i olefina final (olefina$_f$) expulsats. De fet, en comptes de considerar un balanç de masses, l'equació descriurà un balanç de volums. En aquest sentit cal saber que hi ha una pèrdua de volum dels productes de sortida respecte els d'entrada. Concretament es perd un 22% del volum en l'alquilat produït (això té la seva lògica, donat que l'alquilat produït és més dens que l'olefina i isobutà a partir dels quals es forma). El balanç de volums pot ser llavors escrit com:

$$olefina + isobutà - 0.22 \cdot alquilat = alquilat + olefina_f \tag{1.2}$$

Una altra variable que intervé en posteriors equacions és la concentració de l'àcid dins el reactor (mesurat en tant per cent). Suposant que la concentració de l'àcid (en tant per cent) entrat inicialment al reactor es denota per $c_{àcid}$, l'equació resultant és:

$$\frac{c_{àcid\_reactor}}{100} = \frac{\frac{c_{àcid}}{100} \cdot 1000 \cdot àcid}{alquilat \cdot factor\_dilució + 1000 \cdot àcid} \tag{1.3}$$

on $c_{àcid\_reactor}$ denota la concentració d'àcid dins del reactor (que és el valor que estem determinant en aquesta equació). Com s'observa en aquesta equació hi ha dos tipus de concentracions ben diferents. En primer lloc, l'àcid entrat al reactor té una certa concentració $c_{àcid}$; però aquest àcid resulta diluït dins al reactor ja que es mescla amb altres corrents que no contenen àcid, i per tant la concentració d'àcid dins del reactor ($c_{àcid\_reactor}$) és diferent que la concentració de l'àcid inicial ($c_{àcid}$). S'observa que, a més de l'*àcid* i la *concentració d'àcid*, l'equació (1.3) és també funció de l'*alquilat* produït i d'un cert *factor de dilució*. Aquest *factor de dilució* serà presentat posteriorment. Així mateix cal observar que la variable *àcid* va multiplicada en tot moment per 1000. Això es fa per passar les dades a kilograms d'àcid per dia (recordem que l'alimentacio d'àcid al reactor es mesurava en tones per dia).

El següent concepte que cal introduir és el grau de conversió del reactor. Aquest valor ens indica com d'efectiva ha estat la reacció, i pot mesurar-se en funció de la proporció d'alquilat obtingut en sortir del reactor respecte el conjunt de productes que surten i van al fraccionador. L'equació que usarem per determinar aquest grau de conversió (denotat per *conversió*) és:

$$conversió = \frac{alquilat}{alquilat + olefina_n} \tag{1.4}$$

on $olefina_n$ denota l'olefina no reaccionada que passa al fraccionador tal i com mostra la Fig. 1.1. Cal avançar que com major sigui la conversió obtinguda, millor ha de ser la "qualitat" del reactor. Per tant el grau de conversió serà un dels factors que intervendran en el cost del reactor.

L'olefina no reaccionada (olefina$_n$) que surt del reactor, s'observa clarament com es separa al fraccionador en dos parts: una part torna al reactor (olefina recirculada) i una altra surt barrejada amb l'alquilat (olefina$_f$). Caldrà, doncs, afegir una restricció més que indiqui el balanç de masses anterior.

Pel que fa a l'isobutà no reaccionat (isobutà$_n$) tot ell torna al reactor com isobutà recirculat. És a dir

$$isobutà_n = isobutà\_recirculat \tag{1.5}$$

Aquesta darrera equació ens permet obviar una variable; a la formulació final del problema usarem de fet només l'isobutà recirculat.

De forma anàloga a com es va definir un factor de conversió pel reactor que ens indicava com

de completa havia estat la reacció, considerarem un grau de separació d'olefina del fraccionador (valor que sempre haurà de variar entre 0 i 1). A partir d'aquest valor podem relacionar directament l'olefina no reaccionada (olefina$_n$) amb l'olefina recirculada de forma que:

$$olefina\_recirculada = olefina_n \cdot separació \tag{1.6}$$

on *separació* denota el grau de separació de l'olefina al fraccionador. També com al cas del grau de conversió, com major sigui la separació dins el fraccionador, millor ha de ser la "qualitat" d'aquest. Per tant el grau de separació serà el factor que posteriorment s'usarà per determinar el cost del fraccionador.

Una altra variable que cal definir, i que serà usada posteriorment a la funció objectiu, fa referència a la puresa de l'alquilat final obtingut. Aquesta puresa es defineix simplement com la proporció d'alquilat en el producte final, i la seva expressió és:

$$puresa = \frac{alquilat}{alquilat + olefina_f} \tag{1.7}$$

Les restriccions que fins ara s'han presentat podrien catalogar-se com equacions "exactes", ja que representen propietats físiques del sistema (com ara definició de valors que denoten certes proporcions, equacions de balanç de volums ...). Per la seva banda les quatre equacions que encara queden no són exactes, i han estat ajustades a partir de dades de caràcter experimental. Aquestes quatre equacions empíriques són vàlides en un determinat rang de temperatures (que no ens afecten) i un determinat rang de concentració de l'àcid dins el reactor (definit a (1.3)), que posteriorment s'incorporarà a la formulació del problema com a fites superior i inferior de la variable $c_{àcid\_reactor}$.

La primera d'aquestes quatre equacions ajustades ens determina la quantitat d'alquilat final produït. Aquest valor s'ha trobat, mitjançant diverses observacions i posteriors ajustos, que depèn de la quantitat d'olefina que entra al reactor, d'una expressió quadràtica funció de la variable de *relació* definida a l'equació (1.1) i de la *conversió* del reactor definida a (1.4). L'equació que lliga tot això s'escriu com:

$$alquilat = (olefina + olefina\_recirculada) \cdot conversió \cdot (a_0 + a_1 \cdot relació + a_2 \cdot relació^2) \tag{1.8}$$

Els coeficients $a_i, i = 0, 1, 2$ han estat ajustats mitjançant un determinat procés de regressió efectuat prèviament, i en el qual no entrarem.

A la secció anterior hem comentat que una de les característiques importants que defineixen l'alquilat final és el seu octanatge (que ens indica la "qualitat" de l'alquilat obtingut). Aquest octanatge varia en funció de la variable de *relació* definida a (1.1) i de la concentració de l'àcid dins el reactor definida a (1.3), segons una expressió tal com:

$$octà = b_0 + b_1 \cdot relació + b_2 \cdot relació^2 + b_3 \cdot (c_{àcid\_reactor} + b_4) \tag{1.9}$$

on *octà* representa l'octanatge de l'alquilat i $b_i, i = 0, \ldots, 4$ són coeficients prèviament ajustats.

Un altre paràmetre important és el número F-4 de l'alquilat. Aquest és un paràmetre empíric utilitzat pels petrolers que permet deduir les condicions a les quals s'opera quan s'obté una certa "qualitat" de producte final. La "qualitat" del producte final ve donat pel seu octanatge, i per tant el número F-4 pot ajustar-se com a funció d'aquesta variable:

$$F4 = c_0 + c_1 \cdot octà \tag{1.10}$$

on c_o, c_1 són coeficients predeterminats.

A l'equació (1.3) per determinar la concentració d'àcid dins el reactor es va usar un valor anomenat "factor de dilució de l'àcid". El factor de dilució s'obté com una funció lineal del número F-4 detallat anteriorment. L'equació resultant és:

$$factor\_dilució = d_0 + d_1 \cdot F4 \tag{1.11}$$

on d_0, d_1 són coeficients predeterminats.

Fins aquí hem presentat totes les restriccions que intervenen al problema. Ara cal, però, detallar la funció objectiu a maximitzar. Això es farà a la següent secció.

1.2.2 Funció objectiu.

Qualsevol empresa química que realitzés el procés d'alquilació que aquí estem tractant, tindria com primer objectiu maximitzar els seus beneficis. Llavors la nostra funció objectiu a grans trets pot ser descrita com:

$$\max \quad valor\_alquilat\_produït - costos\_del\_procés$$

El valor de l'alquilat produït el denotarem com O_1 i serà descrit a continuació. La part de costos del procés serà desglossada en 7 parts, que denotarem com $O_i, i = 2, \ldots, 8$. Cal recordar que a la Fig. 1.1 l'entrada de productes al reactor i la sortida d'alquilat i olefina del fraccionador s'expressava en quantitats per dia (barrils per dia, tones per dia). Per tant la funció objectiu maximitzarà de fet els beneficis diaris del nostre procés, expressat en pts (llavors l'unitat de cada un dels termes O_i serà de pts./dia). Passem a descriure cada una de les $O_i, i = 1, \ldots, 8$ components de la funció objectiu.

• O_1: El valor de l'alquilat final depèn, en primer lloc, del nombre de barrils diaris que es produeixen (això sembla prou lògic). A part, però, també té més valor com major sigui el seu octanatge i la seva puresa (obtinguts a les equacions (1.9) i (1.7)). L'expressió final del valor diari del nostre alquilat és:

$$O_1 \equiv preu\_alquilat \cdot alquilat \cdot octà \cdot puresa \tag{1.12}$$

on *preu_alquilat* és un valor conegut a priori.

Les parts O_2, O_3 i O_4 que a continuació descriurem detallen els costos associats directament a l'alimentació de productes al reactor:

• O_2: L'olefina entrada al reactor té, òbviament, un cost per barril. El cost global de l'olefina entrada al reactor és simplement el nombre de barrils per dia multiplicat pel preu del barril. Això és:

$$O_2 \equiv preu\_olefina \cdot olefina \tag{1.13}$$

on $preu\_olefina$ és un valor conegut a priori.

• O_3: Anàlogament amb el cas anterior, el preu global de l'alimentació d'àcid és:

$$O_3 \equiv preu\_àcid \cdot àcid \tag{1.14}$$

on *preu_àcid* és un valor conegut a priori.

• O_4: Finalment tenim el preu total de l'isobutà, calculat com als casos anteriors com el producte del preu per barril pel nombre total de barrils:

$$O_4 \equiv preu\_isobutà \cdot isobutà \tag{1.15}$$

on *preu_isobutà* és un valor conegut a priori.

Les dos part següents O_5 i O_6 fan referència als costos associats a la recirculació d'isobutà i olefina:

• O_5: El fet de que l'isobutà sigui transportat de nou del fraccionador al reactor implica un cert cost (per exemple, podem considerar que és necessari usar bombes per realitzar el transport). Aquest cost, d'acord amb les lleis de la fluïdica, varia amb la quantitat d'isobutà recirculat elevat a un cert exponent fraccionari. L'expressió d'aquest cost de recirculació és, doncs:

$$O_5 \equiv e_0 \cdot isobutà\_recirculat^{e_1} \tag{1.16}$$

on els coeficients e_0 i e_1 han estat estimats prèviament.

• O_6: L'expressió del cost de la recirculació d'olefina és similar a la del cost d'isobutà (excepte en els coeficients usats). Per tant:

$$O_6 \equiv f_0 \cdot olefina\_recirculada^{f_1} \tag{1.17}$$

on els coeficients f_0 i f_1 han estat estimats prèviament.

Les dues darreres parts de la funció objectiu fan referència als costos associats al reactor i fraccionador:

• O_7: Pel que fa al reactor, diversos factors són els que intervenen a l'hora de determinar el seu cost. Un d'ells és el grau de *conversió* del reactor definit a l'equació (1.4). Això sembla prou lògic, ja que com més "bo" sigui el reactor usat més completa serà la reacció química. Per altra banda també sembla lògic que el cost del reactor s'incrementi amb la seva capacitat (volum de productes que permet emmagatzemar). En aquest volum s'ha de tenir en compte tot el que entra al reactor a la Fig. 1.1. A part, donat que els costos associats a la quantitat d'olefina, isobutà etc. usats, representen costos diaris, s'ha de tenir en compte que el cost associat amb el reactor també ha de ser un cost diari (no un cost absolut), tenint en compte el període de "vida" que pugui tenir (la seva amortització). Tot plegat, i tenint en compte tot els aspectes que aquí s'han dit, l'aspecte final de l'equació de cost diari del reactor ve donada per:

$$\begin{gathered} O_7 \equiv g_0 \cdot e^{g_1 \cdot conversió} \cdot volum^{g_2} \\ volum = olefina + isobutà + àcid + olefina\_recirculada + isobutà\_recirculat \end{gathered} \tag{1.18}$$

on els coeficients $g_i, i = 0, 1, 2$ són coneguts a priori (podem veure que en definir el volum estem sumant unitats que signifiquen barrils de producte —per ex., $olefina$— i d'altres que són tones de producte —per ex. *àcid*—; suposarem, però, que el coeficient g_2 ja es troba ajustat per tal de considerar aquest fet). Aquesta equació de tipus exponencial indica que el cost puja molt fortament amb l'increment de qualitat i de volum.

• O_8: Finalment cal tenir en compte el cost del fraccionador. Anàlogament al cas anterior aquest cost s'ha de considerar com un cost diari, per poder ser sumat amb la resta de valors O_i. De la mateixa forma que el cost del reactor incrementava segons el grau de conversió, el cost del fraccionador incrementarà també en funció del grau de separació —usat a l'equacio (1.6) per obtenir la quantitat d'olefina recirculada en funció de l'olefina no reaccionada. L'equació que ens donarà el cost del fraccionador serà:

$$O_8 \equiv h_0 \cdot separació^{h_1} \tag{1.19}$$

1.3 Formulació matemàtica del problema.

Un cop hem modelitzat el problema, tal i com s'ha vist a la secció anterior, només resta fer la seva formulació matemàtica final. El següent pas serà la resolució mitjançant algun paquet d'optimització.

Tot problema pot ser formulat en la seva forma estandard com:

$$\begin{array}{ll} \max\limits_{X}. & f(X) \\ \text{subj.} & g(X) = r \\ & l \leq X \leq u \end{array} \tag{1.20}$$

essent X el vector de variables, $f(X)$ la funció objectiu, $g(X)$ la funció de les restriccions, r el vector de termes independents i l i u els límits inferiors i superiors de les variables. Clarament la funció objectiu és una funció $f : \mathbb{R}^n \rightarrow \mathbb{R}$, essent n el nombre de variables del problema, mentre que la funció de les restriccions és una funció $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ essent m el nombre de restriccions del problema.

Al nostre problema la funció objectiu i restriccions han estat modelitzades a la secció anterior. L'únic que cal és escriure-les conjuntament de forma que donin lloc a un problema d'optimització. Tanmateix, cal tenir present un parell de coses. En primer lloc, tal i com s'ha vist en la secció anterior, les quatre equacions (1.8–1.11) han estat ajustades i expressen relacions "aproximades". Per tant no les considerarem com restriccions d'igualtat del tipus $g_i(X) = r_i$, sinó que permetrem un cert marge de variació donat que l'ajust pot ser més o menys bo; llavors les considerarem restriccions de la forma $r_i - \varepsilon_i \leq g_i(X) \leq r_i + \varepsilon_i$ essent ε_i una tolerància predeterminada. Associarem els valors de ε_1, ε_2, ε_3 i ε_4 a les equacions (1.8), (1.9), (1.10) i (1.11) respectivament. Així, per exemple, l'equació (1.10) podria escriure's com:

$$c_0 - \varepsilon_3 \leq F4 - c_1 \cdot \textit{octà} \leq c_0 + \varepsilon_3$$

A part d'això considerarem limitades només algunes variables del problema. Aquestes són concretament: l'alquilat, la concentració d'àcid dins el reactor, el nombre d'octans de l'alquilat, la relació entre l'isobutà i l'olefina al reactor, el factor de dilució de l'àcid, el número F-4 de l'alquilat, el grau de conversió del reactor i el grau de separació del fraccionador. A la següent secció s'explica com obtenir tots aquest límits. La resta de variables estaran afitades inferiorment per 0 segons la seva unitat o magnitud (per exemple, la variable *àcid* representa tones d'àcid, i per tant no pot prendre valors negatius; el mateix succeeix amb l'$olefina$, $olefina_n$, etc.).

1.4 Dades del problema.

Al llarg de l'explicació que s'ha fet durant la modelització i formulació, han anat apareixent diversos coeficients que cal conèixer per tal de tenir completament determinat el problema i poder realitzar la seva optimització. A continuació es llisten els valors que haureu d'usar per solucionar el problema (si voleu els podeu variar, sempre i quan aquesta variació sigui molt lleugera; d'altra banda podeu obtenir problemes d'infactibilitat o estabilitat)

- $a_0 = 1.12, a_1 = 0.13167, a_2 = -0.0067$, i $\varepsilon_1 = 0.0010$ a l'equació (1.8).
- $b_0 = 86.35, b_1 = 1.098, b_2 = -0.038, b_3 = 0.325, b_4 = -89.0$ i $\varepsilon_2 = 0.0015$ a l'equació (1.9).

- $c_0 = -133.0, c_1 = 3.0$, i $\varepsilon_3 = 0.0020$ a l'equació (1.10).
- $d_0 = 16.226, d_1 = -0.1005$, i $\varepsilon_4 = 0.0010$ a l'equació (1.11).
- $c_{àcid} = 98\%$ a l'equació (1.3),
- El preu de l'alquilat per barril i octà segons puresa a usar a l'equació (1.12) és de 17.50 pts. El preu del barril d'olefina a usar a l'equació (1.13) és de 705.6 pts. El preu de l'àcid per tona a usar a l'equació (1.14) és de 3090.5 pts. El preu del barril d'isobutà (equació (1.15)) és de 470.4 pts.
- $e_0 = 2.1, e_1 = 1.43$ a l'equació (1.16) de cost de recirculació d'isobutà. Per la seva banda, $f_0 = 2.94, f_1 = 1.35$ a l'equació (1.17) de cost de recirculació d'olefina.
- $g_0 = 3036.0, g_1 = 3.12, g_2 = 0.34$ a l'equació (1.18) de cost del reactor. Anàlogament, $h_0 = 350000.0, h_1 = 10.67$ a l'equació (1.19) de cost del fraccionador.
- El nombre de barrils per dia d'alquilat produït es troba entre 2000 i 4000.
- La concentració d'àcid dins el reactor es troba entre el 90 i 93 %.
- El nombre d'octans de l'alquilat ha d'estar entre 90 i 95.
- La relació isobutà/olefina al reactor ha d'estar entre 3.0 i 12.0.
- El factor de dilució ha d'estar entre 1.2 i 4.0.
- El número F-4 de l'alquilat es troba entre 145.0 i 162.0.
- La conversió del reactor s'ha de trobar entre 0.0 i 1.0.
- La separació del fraccionador s'ha de trobar entre 0.6 i 1.0.

1.5 Detalls importants a tenir en compte!.

Com en la majoria de branques de la ciència, en el camp de l'optimització una cosa és la teoria i l'altra és la pràctica. Amb això volem dir que el model abans introduït és, en principi, coherent i la seva solució no hauria de donar cap problema. De fet això no és així. Per això és recomanable que, a banda dels límits reals sobre algunes variables que ja han estat comentats anteriorment, afegiu dos límits més totalment ficticis. Concretament es tracta d'imposar que les variables $isobutà\_recirculat$ i $olefina\_recirculada$ tinguin un límit inferior diferent de 0 (es pot considerar, per exemple, una fita inferior de 0.1). Això és per evitar que, en imposar simplement una fita inferior de 0, per problemes de precissió, es tingui en algun moment del procés d'optimització un valor de, per exemple, $-0.1 \cdot 10^{-16}$, que es pot considerar com 0, però té l'inconvenient de ser un valor negatiu (amb el qual el paquet Lingo —o qualsevol altre paquet— tindria seriosos problemes, i fins i tot podria abortar l'execució).

Una altra cosa que cal que tingueu present és que les restriccions d'aquest problema no determinen un conjunt convex. Això provoca que pot haver diversos òptims locals, i llavors els mètodes d'optimització "habituals", com el que usa Lingo, poden no detectar l'òptim global del nostre problema. Aquest és un greu problema que apareix en modelitzar problemes amb conjunt de solucions no convex i/o funció objectiu no convexa. Això vol dir que la solució obtinguda per Lingo pot variar segons el problema sigui formulat d'una o altra manera. Per aquest problema de dimensió petita, però, aquestes variacions (si existeixen) poden ser no significatives.

1.6 Presentació de la pràctica.

L'objectiu de la pràctica és que formuleu el procés descrit a les seccions anteriors i el solucioneu mitjançant l'ús del paquet Lingo. Trobareu una breu descripció sobre aquest paquet a l'annex I.

L'informe ha de tenir l'estructura general recomanada a la "Introducció" de la col·lecció de pràctiques. Entre la informació específica que ha de contenir l'informe, ha d'aparèixer

- La formulació detallada de la funció objectiu, restriccions i límits de les variables, tot indicant quines variables heu considerat.
- Un llistat del fitxer amb la modelització del problema en el llenguatge de Lingo.
- Un llistat del resultat que Lingo us ha proporcionat, indicant el valor de funció objectiu que heu obtingut.
- La representació gràfica de l'òptim proporcionat per Lingo en una figura similar a la Fig. 1.1, indicant el valor numèric de només les variables que allà apareixen.
- Qualsevol observació o comentari que considereu escaient, així com qualsevol problema que us hagi aparegut durant la realització de la pràctica.
- Una justificació de per què creieu que Lingo tindria problemes en el cas que alguna de les dues variables $isobutà\_recirculat$ i $olefina\_recirculada$ (les que tenen afegit un límit fictici) arribés a tenir un valor negatiu.

PRÀCTICA 2. Optimització d'un sistema turbo-generador (4.5 punts).

Les tècniques d'optimització són sovint usades per al disseny i determinació del règim de funcionament de sistemes de vapor a l'indústria química. En aquest cas concret mirarem de determinar els valors dels corrents de vapor i fluxos de potència que minimitzen els costos d'un sistema turbo-generador.

2.1 Presentació del problema.

La Fig. 2.1 mostra l'esquema del sistema de vapor i turbo-generació elèctrica que serà considerat. L'aigua és escalfada a la caldera, usant un determinat combustible. El vapor generat s'incorpora a un corrent de vapor d'alta pressió (635 psig) de V_a lb_m/h . Amb aquest vapor d'alta pressió s'alimenten dues turbines (denotades per Turb. 1 i Turb. 2 a l'esquema). El flux d'entrada a cada turbina és de I_1 i I_2 lb_m/h, i la potència elèctrica generada és de P_1 i P_2 kw, respectivament. Si el flux de vapor del corrent d'alta pressió excedeix la capacitat de les turbines, aquest vapor en excès pot ser incorporat directament a dos corrents de vapor a mitja (195 psig) i baixa (52 psig) pressió (de V_m i V_b lb_m/h), a través de les vàlvules indicades a la Fig. 2.1 (la vàlvula$_{am}$ per passar d'alta a mitja pressió, i la vàlvula$_{mb}$ per passar de mitja a baixa pressió). El vapor que a través de les vàlvules passa d'alta a mitja pressió és de V_{am} lb_m/h, mentre que el que passa de mitja a baixa és de V_{mb} lb_m/h.

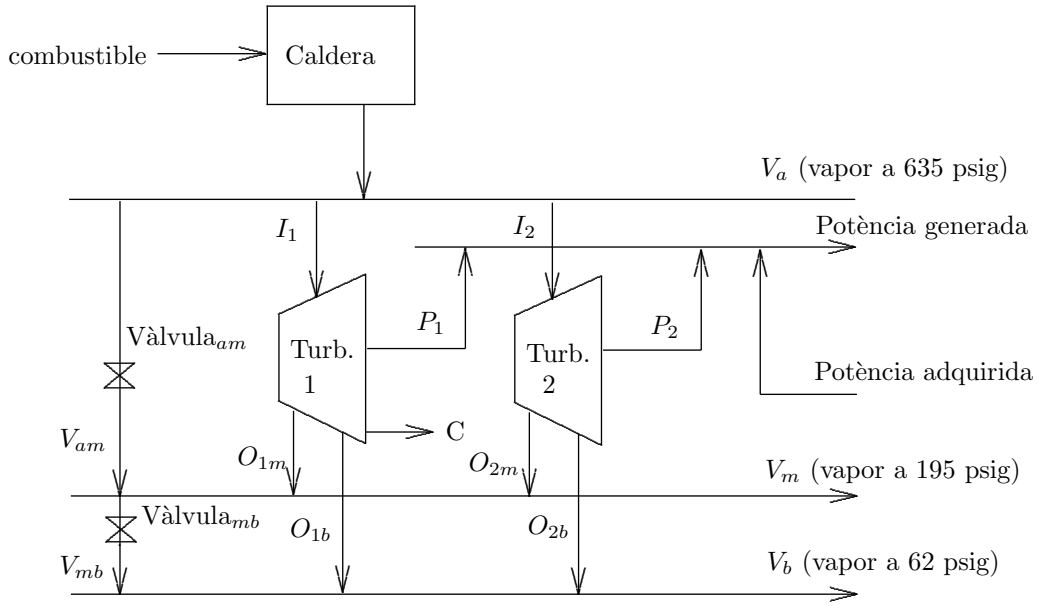

Figura 2.1. Esquema del sistema turbo-generador.

La turbina 1 és de doble extracció, i treballa produint dos corrents intermedis a mitja i baixa pressió de O_{1m} i O_{1b} lb_m/h respectivament, i amb una etapa final on s'obtenen per condensació C lb_m/h d'aigua, usats per realimentar la caldera. La turbina 2, per la seva banda, és d'extracció simple, amb només un corrent intermedi de mitja pressió de O_{2m} lb_m/h, i un corrent de sortida de baixa pressió de O_{2b} lb_m/h. Els corrents O_{1m} i O_{2m} de sortida de cada turbina s'incorporen al corrent de mitja pressió principal V_m. De forma anàloga, O_{1b} i O_{2b} s'afegeixen al corrent de baixa pressió V_b. Clarament, la primera turbina és més eficient que la segona degut a la doble transformació d'energia mecànica en elèctrica que fa en passar el vapor de 635 a 195 psig en el primer corrent intermedi, i de 195 a 62 psig en el segon corrent intermedi (la segona turbina només efectua la primera d'aquestes transformacions, ja que no incorpora la fase final de condensació). La Taula 2.1 indica l'energia (en Btu's) que correspon a cada lb_m de vapor segons estigui a alta, mitja o baixa pressió, i en el cas d'estar condensat (aigua d'alimentació de la caldera).

Taula 2.1. Energia dels corrents de vapor i vapor condensat.

Corrent	Pressió	Energia per lb_m
Alta pressió	635 psig	E_a = 1360 Btu/lb_m
Mitja pressió	195 psig	E_m =1268 Btu/lb_m
Mitja pressió	195 psig	E_b =1251 Btu/lb_m
Vapor condensat	—	E_0 =193 Btu/lb_m

Les característiques tècniques d'ambdues turbines són mostrades a la Taula 2.2. Cada turbina té una potència màxima ($\overline{P_i}$ $i = 1, 2$) i mínima ($\underline{P_i}$ $i = 1, 2$) de generació. La potència mínima implica que, o bé aquella turbina no genera res, o bé si comença a generar ho farà a partir d'aquest valor mínim (no hi ha, doncs, continuïtat). El flux màxim d'entrada $\overline{I_i}$ $i = 1, 2$ fa referència a les capacitats de I_1 i I_2. El flux màxim final ($\overline{C}$, $\overline{O_{2b}}$) denota les capacitats de la fase final de cada turbina, això és, del condensat C per a la turbina 1, i del vapor a baixa pressió O_{2b} per a la turbina 2. Finalment, el flux màxim intern $\overline{T}$ només té sentit per a la turbina 1, i denota la quantitat màxima de vapor entrat que pot quedar dins la turbina, per ser processat en etapes posteriors, un cop ha sortit el corrent a mitja pressió O_{1m}.

Taula 2.2. Característiques tècniques de les turbines.

	Turbina 1	Turbina 2
Potència màxima ($\overline{P_i}$)	6250 kw	9000 kw
Potència mínima ($\underline{P_i}$)	2500 kw	3000 kw
Flux màxim d'entrada ($\overline{I_i}$)	192000 lb_m/h	244000 lb_m/h
Flux màxim final ($\overline{C}$, $\overline{O_{2b}}$)	($\overline{C}$) 62000 lb_m/h	($\overline{O_{2b}}$) 142000 lb_m/h
Flux màxim intern ($\overline{T}$)	132000 lb_m/h	—

Els requeriments del sistema són tant a nivell de potència elèctrica generada, com de vapor a mitja i baixa pressió (V_m i V_b). La Taula 2.3 mostra la demanda per cada concepte. En alguns casos, tota la potència generada per les dues turbines pot ser insuficient per cobrir les necessitats energètiques requerides. En aquest cas cal adquirir potència elèctrica a un productor extern.

Aquest productor, però, i de forma anàloga al que ens succeia amb la generació de les turbines, té una potència mínima de subministrament, denotada per $\underline{P_e}$ a la Taula 2.4, la qual presenta les dades sobre la potència externa. Això implica que, si ens subministra energia externa, com a mínim ens proporcionarà $\underline{P_e}$ kwh. Això implica que, de tota la potència externa adquirida, una part pot no ser necessària i no serà consumida. La Taula 2.4 mostra el cost, en pts, del kwh extern adquirit, en funció de si ha estat o no consumit ($C_{P_{ec}}$, $C_{P_{en}}$).

Taula 2.3. Demanda de vapor i potència.

	Demanda
Vapor a 195 psig	$D_{V_m} = 275000$ lb_m/h
Vapor a 62 psig	$D_{V_b} = 100000$ lb_m/h
Potència generada	$D_P = 20000$ kw

Taula 2.4. Dades de la potència externa.

Cost potència externa consumida	$C_{P_{ec}} = 5$ pts/kwh en promig
Cost potència externa no consumida	$C_{P_{en}} = 1.5$ pts/kwh
Potència externa mínima	$\underline{P_e} = 12000$ kw

Per tenir completament determinat el problema només ens cal conèixer les dades sobre el cost del combustible amb que funciona la caldera (C_f), i l'eficiència d'aquesta (F). Aquestes són presentades a la Taula 2.5. Amb aquestes dades, i usant els valors d'energia del vapor E_a i E_0 de la Taula 2.1, directament podem calcular el que costa passar una lb_m d'aigua a vapor a 635 psig (cost denotat per C_{V_a}):

$$C_{V_a} = \frac{C_f}{F}(E_a - E_0) = \frac{201 \cdot 10^{-6} \text{ pts/Btu}}{0.75} \ (1360 \text{ Btu/lb}_\text{m} - 193 \text{ Btu/lb}_\text{m}) = 0.313 \text{pts/lb}_\text{m}$$

Taula 2.5. Dades de la caldera.

Cost combustible	$C_f = 201 \cdot 10^{-6}$ pts/Btu
Eficiència de la caldera	$F = 0.75$
Cost producció vapor a 635 psig	$C_{V_a} = 0.313$ pts/lb_m

2.2 Modelització del problema.

Un cop detallat el funcionament i les dades particulars del sitema cal realitzar la seva modelització per tal de determinar els corrents òptims de vapor, la generació de cada turbina, i l'energia que serà adquirida a l'exterior. En primer lloc ens centrarem en la funció objectiu, passant tot seguit a parlar de les restriccions.

2.2.1 Funció objectiu.

La funció objectiu que hem de minimitzar és en aquest cas lineal, i únicament ha de contemplar la suma dels:

- Costos de producció del corrent de vapor a alta pressió: $C_{V_a} V_a$ pts/h.
- Costos d'adquisició de la potència externa consumida: $C_{P_{ec}} P_{ec}$ pts/h, on P_{ec} denota la potència externa consumida.
- Costos d'adquisició de la potència externa no consumida: $C_{P_{en}} P_{en}$ pts/h, on P_{en} denota la potència externa no consumida.

Amb aquesta funció objectiu minimitzarem els costos d'operació del sistema per hora.

2.2.2 Restriccions.

Les restriccions del problema poden ser agrupades en sis grups diferents, els quals són detallats als següents apartats. Pel que fa a les variables, tot i que no s'explicitarà, queda clar que totes elles hauran de ser valors positius, ja que no són més que fluxos de vapor, o potències generades i adquirides.

2.2.2.1 Turbina 1.

Les restriccions que modelitzen el funcionament de la turbina 1 són:

- Límit sobre el flux màxim d'entrada I_1.
- Límit sobre el condensat produït C.
- Límit sobre el flux màxim intern que quedarà dins la turbina després d'haver expulsat el corrent a mitja pressió O_{1m}:
$$I_1 - O_{1m} \leq \overline{T}$$
- Límits sobre la potència generada per la turbina. Cal tenir present, però, que la turbina o bé no genera res, o si ho fa com a mínim ha de produir $\underline{P_1}$ kw. Per modelitzar aquesta situació podem definir una variable binària $y_{t_1} \in \{0, 1\}$, de forma que quan val 0 la turbina no genera res, i quan val 1 sí té generació. En aquest cas, els límits de potència poden ser escrits com:
$$y_{t_1} \underline{P_1} \leq P_1 \leq y_{t_1} \overline{P_1}$$

2.2.2.2 Turbina 2.

Les restriccions que determinen el comportament de la turbina 2 són:

- Límit sobre el flux màxim d'entrada I_2.
- Límit sobre el flux màxim del corrent de sortida de baixa pressió $\overline{O_{2b}}$.
- Límits sobre la potència generada per la turbina 2. Com al cas de la turbina 1, podem definir una variable binària $y_{t_2} \in \{0, 1\}$ que ens permeti definir la restricció com:
$$y_{t_2} \underline{P_2} \leq P_2 \leq y_{t_2} \overline{P_2}$$

2.2.2.3 Potència externa adquirida.

Només ens cal la següent restricció per modelar la potència externa (formada, tant per la consumida com per la no consumida):

- Límit sobre la potència externa adquirida. De forma similar al que succeïa amb la generació de les turbines, si comprem energia externa ens veiem obligats a adquirir una quantitat mínima. Per modelar aquest comportament podem definir la variabla binària $y_{P_e} \in \{0, 1\}$ que valdrà 0 si no comprem potència externa, o 1 si en comprem. Aleshores la restricció que tenim és:
$$P_{ec} + P_{en} \geq y_{P_e} \underline{P_e}$$
Amb això, però, no n'hi ha prou, ja que podria passar que per $y_{P_e} = 0$ obtinguem valors de potència externa entre 0 i $\underline{P_e}$, cosa que no volem que passi. Aleshores cal imposar un

límit superior sobre el valor de $P_{ec} + P_{en}$, de forma anàloga a com es va procedir amb la generació de les turbines. En aquest cas, però, el límit superior no existeix. Podem usar, però, el valor D_P (potència demandada), ja que de fet implícitament actua com a fita superior (no té sentit comprar més potència externa que la quantitat total demandada pel sistema). Per tant, caldria afegir també:

$$P_{ec} + P_{en} \leq y_{P_e} D_P$$

2.2.2.4 Balanç de fluxos.

Les equacions de balanç de fluxos ens permeten modelitzar les relacions que hi ha entre els diferents fluxos i corrents de vapor del sistema, segons l'esquema de la Fig. 2.1. Les unitats de tots els fluxos són $\mathrm{lb_m/h}$. Aquestes restriccions són:

- Balanç de fluxos general sobre el vapor d'entrada a alta pressió, i el de sortida a mitja i baixa pressió i el condensat obtingut:

$$V_a = V_m + V_b + C$$

- Balanç de fluxos a la turbina 1: el vapor d'entrada ha de ser igual a tot el vapor i condensat de sortida.
- Balanç de fluxos a la turbina 2: de nou, el vapor d'entrada ha de ser igual a tot el vapor de sortida.
- Balanç de fluxos al corrent d'alta pressió: el que entra a aquest corrent (V_a) ha de ser igual al que surt d'aquest corrent (I_1, I_2 i V_{am}).

$$V_a = I_1 + I_2 + V_{am}$$

- Balanç de fluxos al corrent de mitja pressió. Com abans, tot el que entra a aquest corrent ha de ser igual a tot el que surt (V_m).
- Balanç de fluxos al corrent de baixa pressió, on tot el que entra al corrent ha de ser igual a V_b.

2.2.2.5 Balanç de potències.

Les equacions de balanç de potències ens permetran lligar les potències P_1 i P_2 generades amb els diferents fluxos de vapor d'entrada i sortida de cada turbina. Per obtenir la potència d'un determinat corrent de vapor (expressat en $\mathrm{lb_m/h}$), només cal multiplicar el seu valor per algun dels termes E_a, E_m, E_b o E_0 presentats a la taula Fig. 2.1 (expressats en $\mathrm{Btu/lb_m}$), en funció de la seva pressió. La potència així obtinguda queda expressada en Btu/h. Les potències P_1 i P_2 vénen, però, expressades en kw. Per poder transformar-les a Btu/h cal saber que un Btu equival a 1055 Joules, i ara directament tenim

$$1 \text{ kw} = 1000 \text{ w} = \frac{1000 \text{ Joules}}{1 \text{ s}} \times \frac{3600 \text{ s}}{1 \text{ h}} \times \frac{1 \text{ Btu}}{1055 \text{ Joules}} = 3412.3 \text{ Btu/s}$$

Ara ja podem expressar les nostres equacions de balanç de potències per a cada turbina com:

- Balanç de potències per a la turbina 1:

$$E_a I_1 = E_m O_{1m} + E_b O_{1b} + E_0 C + 3412.3 P_1$$

- Balanç de potències per a la turbina 2 (es fa de forma anàloga a l'anterior).

2.2.2.6 Demanda del sistema.

Finalment només ens cal indicar quant de vapor i de potència ha de produir el sistema

que estem considerant.

- Demanda de vapor a mitja potència obtingut. Aquest valor V_m no ha de ser inferior a la seva demanda D_{V_m}.
- Demanda de vapor a baixa potència generat.

$$V_b \geq D_{V_b}$$

- Demanda de potència. La potència generada i l'externa consumida no ha de ser inferior a la potència demandada:

$$P_1 + P_2 + P_{ec} \geq D_P$$

Un cop hem modelitzat el problema només resta fer la seva formulació matemàtica final. El següent pas serà la resolució mitjançant algun paquet d'optimització (en aquest cas amb el paquet Lingo).

2.3 Presentació de la pràctica.

L'objectiu de la pràctica és que formuleu el problema descrit a les seccions anteriors i el solucioneu mitjançant l'ús del paquet Lingo. Trobareu una breu descripció sobre aquest paquet a l'annex I.

L'informe ha de tenir l'estructura general recomanada a la "Introducció" de la col·lecció de pràctiques. Entre la informació específica que ha de contenir l'informe, ha d'aparèixer

- La formulació detallada de la funció objectiu, restriccions i límits de les variables, tot indicant quines són les variables del vostre problema.
- Un llistat del fitxer amb la modelització del problema en el llenguatge de Lingo.
- Un llistat del resultat que Lingo us ha proporcionat, indicant el valor de funció objectiu que heu obtingut.
- La representació gràfica de l'òptim proporcionat per Lingo en una figura similar a la Fig. 2.1, indicant el valor numèric de les variables que allà apareixen.
- La resposta a la següent qüestió:

  Considereu dues noves situacions on les demandes del sistema han variat. A la primera situació les noves demandes de vapor i potència són: $D_{V_m} = 27500$, $D_{V_b} = 10000$, $D_P = 20000$. A la segona situació tenim: $D_{V_m} = 27500$, $D_{V_b} = 10000$, $D_P = 10000$. Solucioneu el problema anterior per a cada una de les noves situacions, indicant les turbines que s'han activat i si s'ha comprat o no potència externa. Trobeu lògics els resultats? Raoneu-ho.
- Qualsevol observació o comentari que considereu escaient, així com qualsevol problema que us hagi aparegut durant la realització de la pràctica.

PRÀCTICA 3. Obtenció dels pesos d'una xarxa neuronal (5 punts).

Les xarxes neuronals són un tipus de funcions, o eines matemàtiques, que durant els darrers anys han gaudit d'una forta acceptació per solucionar una gran diversitat de problemes. Dues de les raons que expliquen aquesta acceptació han estat, per una banda, la seva facilitat d'ús, i en segon lloc, el fet que permeten solucionar, de forma acceptable, certs tipus de problemes on altres tècniques, o bé requeririen una gran quantitat de temps, o bé no són capaces de trobar una solució el suficientment acurada. Sempre que es vol usar una xarxa neuronal, el primer que cal fer és ajustar una sèrie de paràmetres (anomenats pesos) que determinaran la seva forma de funcionament. L'objectiu de la pràctica serà, doncs, el de trobar aquests paràmetres (calibració de la xarxa) usant tècniques d'optimització. En primer lloc es descriurà de forma simple què és una xarxa neuronal, i a continuació s'indicarà el problema d'optimització que caldrà solucinar, amb l'ajut del paquet Lingo.

3.1 Funcionament d'una xarxa neuronal.

Una xarxa neuronal no és més que una funció matemàtica $F : \mathbb{R}^{n_e} \rightarrow \mathbb{R}^{n_s}$ que rep un vector d'entrada $x_e \in \mathbb{R}^{n_e}$ i genera un vector de sortida $x_s \in \mathbb{R}^{n_s}$. El vector d'entrada correspon a una sèrie de valors coneguts per nosaltres. La xarxa neuronal és avaluada amb aquests valors, i ens proporciona un determinat resultat (que pot estar compost per un o varis valors, en funció de la dimensió n_s dels vectors de sortida x_s). Per entendre com funciona una xarxa neuronal, considerarem una de molt simple, tal i com es mostra a la Fig. 3.1. La xarxa neuronal es troba formada per un conjunt de nodes (anomenats "neurones" en l'argot de les xarxes neuronals), els quals es transmeten informació en el sentit indicat pels arcs. Aquests nodes es disposen per capes. Al cas concret mostrat a la Fig. 3.1, tenim un total de set nodes disposats en tres capes. A la xarxa que estem considerant, la capa inferior (formada per quatre nodes) rep quatre valors d'entrada (els $x_i,\ i = 1, ..., 4$), mentre que la capa de sortida, formada només per una neurona, ens proporciona el resultat de la xarxa. En aquest cas, doncs, tenim que $n_e = 4$ i $n_s = 1$.

El funcionament de la xarxa és el següent. Cada node i rep un "input" I_i, i el transforma en un "output" O_i. Aquesta transformació es realitza mitjançant una determinada funció $f(x)$. Entre les més usades tenim la funció tangent hiperbòlica $tgh(x) = (e^x - e^{-x})/(e^x + e^{-x})$, i la funció sigmoidal

$$f(x) = \frac{1}{1 + e^{-x}}$$

la qual té un aspecte com el que es mostra a la Fig. 3.2. Podem observar com la funció sigmoidal sempre torna un valor entre 0 i 1 (la tangent hiperbòlica torna valors entre -1 i 1). En aquesta pràctica usarem la funció sigmoidal. Per tant, els valors O_i de sortida d'una neurona seran calculats de forma $O_i = f(I_i)$, on f serà la funció sigmoidal. L'"output" O_7 de la darrera neurona ens proporciona el resultat de la xarxa neuronal.

Ara cal veure com calcular els "inputs" I_i de cada neurona i. Aquests es calcularan sumant els "outputs" de les neurones j que envien informació a la neurona i, multiplicant-los abans pels valors $w_{j,i}$, que són mostrats a la Fig. 3.1. Aquests valors $w_{j,i}$ s'anomenen "pesos",

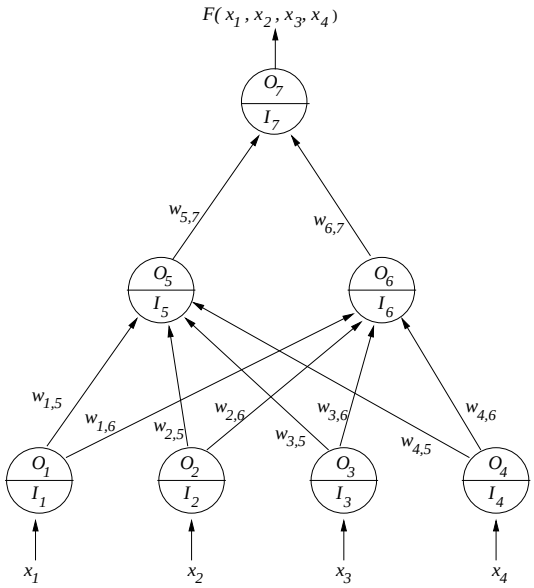

Figura 3.1. Esquema d'una xarxa neuronal simple.

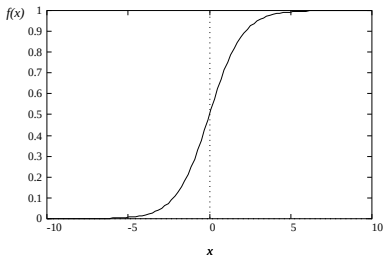

Figura 3.2. Gràfica de la funció sigmoidal $f(x) = 1/(1+e^{-x})$.

i serveixen per ponderar la informació que arriba a cada neurona. Per exemple, al node 5 de la Fig. 3.1, l'"input" I_5 que li arriba es calcularia com:

$$I_5 = w_{1,5}O_1 + w_{2,5}O_2 + w_{3,5}O_3 + w_{4,5}O_4$$

Anàlogament es faria per a la resta de nodes. Procedint d'aquesta forma, proporcionant uns valors x_1, x_2, x_3 i x_4 determinats a la xarxa neuronal, el resultat que obtindríem seria:

$$\begin{aligned} F(x_1,x_2,x_3,x_4) = O_7 = f(I_7) = f(w_{5,7}O_5 + w_{6,7}O_6) = f(w_{5,7}f(I_5) + w_{6,7}f(I_6)) = \\ = f(w_{5,7}f(\sum_{i=1}^{4} w_{i,5}O_i) + w_{6,7}f(\sum_{i=1}^{4} w_{i,6}O_i)) = \\ = f(w_{5,7}f(\sum_{i=1}^{4} w_{i,5}f(x_i)) + w_{6,7}f(\sum_{i=1}^{4} w_{i,6}f(x_i))) \end{aligned} \tag{3.1}$$

essent $f()$ la funció sigmoidal, com abans hem indicat. En aquest cas tan senzill hem pogut

escriure de forma detallada la funció que representa la xarxa neuronal mitjançant l'equació (3.1). Si tinguéssim una quantitat més elevada de nodes i capes, això ja no seria possible, pels múltiples nivells de recurrència que apareixerien. Tanmateix, per realitzar la pràctica n'hi ha prou amb aquesta xarxa neuronal tan simple.

Abans de continuar, val la pena indicar que, a les xarxes neuronals usades realment a la pràctica, a part dels pesos w mostrats a la Fig. 3.1, cada node té un pes addicional que sempre es suma directament a l'"input" de la xarxa (és a dir, sense multiplicar-lo per cap "output" d'un node d'una capa inferior). Aquests pesos especials s'anomenen "bias" a l'argot de les xarxes neuronals. A la pràctica, per tal de reduir el nombre de variables del problema plantejat, no considerarem aquests pesos addicionals.

En funció del vist fins ara, podem veure com el comportament de la xarxa neuronal ve regit precisament pels valors dels pesos w. Depenen dels valors concrets de w, la xarxa produirà uns o altres resultats. Ara queda la qüestió de: com obtenir els pesos w escaients?. Doncs calen dues coses: en primer lloc, unes dades per tal de poder fer "aprendre" a la xarxa el comportament que ha de tenir (veurem això més clar una mica més endavant), i en segon lloc, un mètode (numèric) per poder fer aquest "aprenentatge" (el concepte d'"aprenentatge" també forma part de l'argot de les xarxes neuronals). Un cop hem calibrat la xarxa (hem ajustat els valors w), aquesta ja està llesta per, a partir d'un vector x_e de dades d'entrada, donar una determinada resposta.

3.2 Obtenció dels pesos w.

Tal i com abans s'ha indicat, ens queda veure com determinar els pesos w. Hi ha diverses formes. En aquest treball, però, usarem una basada en un problema de mínims quadrats no lineals (problema de minimització sense restriccions), on les variables a optimitzar seran els pesos w. Per obtenir aquests pesos cal disposar de p vectors de dades d'entrada $x_e \in \mathbb{R}^{n_e}$, i p vectors de dades de sortida $x_s \in \mathbb{R}^{n_s}$, que han de correspondre amb els valors associats a les dades d'entrada. La idea és que s'ajustin els pesos w de la xarxa de forma que $F(x_{e_i}; w) \approx x_{s_i},\ i = 1, ..., p$, on $F(x; w)$ representa la resposta de la xarxa quan la seva entrada és el vector x, i té uns pesos w. Podríem dir que els p vectors $x_{e_i}, x_{s_i},\ i = 1, ..., p$ són una mostra per a que la xarxa "aprengui" quina resposta ha de donar en funció d'una determinada entrada.

En funció del dit abans, una bona forma d'ajustar els w serà plantejar un problema de mínims quadrats no lineals, on es minimitzi la distància entre $F(x_{e_i}; w)$ i x_{s_i}. El problema que plantejaríem seria:

$$\min_w \sum_{i=1}^{p} ||F(x_{e_i}; w) - x_{s_i}||^2 \tag{3.2}$$

En el cas més simple, on la sortida de la xarxa té un únic valor, tenim que $n_s = 1$ (aquesta és la situació que considerarem a la pràctica), i el problema a solucionar en aquest cas pot ser rescrit com:

$$\min_w \sum_{i=1}^{p} (F(x_{e_i}; w) - x_{s_i})^2 \tag{3.3}$$

3.3 Camps d'aplicació de les xarxes neuronals.

Les xarxes neuronals són usades en diferents camps. Un d'ells és el de la previsió. En

aquest cas, les dades d'entrada podrien correspondre, a n dades d'una determinada sèrie temporal (com, per exemple, les hores de sol mensuals d'un país, temperatura d'un determinat compost a cada hora, nombre de neixements anuals d'una regió, consum diari d'energia elèctrica —en Kwh— d'una determinada ciutat, etc.), i el resultat de la xarxa neuronal seria una estimació sobre la dada corresponent al següent interval.

Un altre dels camps d'aplicació és del reconeixement de formes i patrons. Per exemple, considerem que podem classificar una sèrie d'objectes (o situacions) en funció de les seves característiques, i disposem d'un determinat nombre de classes. Llavors, donat un objecte, el que faríem és mesurar les seves característiques, i aquestes mesures serien el vector d'entrada a la xarxa neuronal. La sortida de la xarxa hauria de ser un valor que ens digués a quina classe pertany l'objecte, en funció de les dades d'entrada. Per posar un exemple més concret, podem pensar que les mesures corresponen a certs valors de proves clíniques fetes a un pacient, i la sortida de la xarxa (alimentada amb les dades anteriors) ens indicaria quin tipus de malaltia té el pacient en funció de les dades (hem "classificat" el pacient, doncs). Un dels avantatges (o desavantatges, segons com es miri) de les xarxes neuronals, quan són usades per classificar, és que no requereixen d'un coneixement previ sobre les dades mesurades, o les classes existents. És a dir, al cas de la classificació d'un pacient segons les dades clíniques, això vol dir que la xarxa neuronal no sap si la primera mesura correspon a la pressió sanguínia del pacient, o a la seva freqüència cardíaca, per exemple. Això fa que sigui còmode usar xarxes neuronals: només cal donar-li unes dades per fer l'aprenentatge, sense pensar en què volen dir aquestes dades; la xarxa no requereix d'aquesta informació addicional. Per altra banda, té l'inconvenient de que es difícil poder explicar la classificació que ha fet en funció de les dades entrades, el qual seria molt útil en alguns casos (com, per exemple, al cas anterior de les malalties, on podríem adonar-nos de certes patologies). En aquesta pràctica, tal i com veurem tot seguit, treballarem amb un cas molt simple de reconeixement de patrons.

3.4 Problema considerat a la pràctica.

Com abans s'ha indicat, a la pràctica considerarem un cas molt simple de reconeixement de patrons. El vector x_e de dades d'entrada tindrà quatre components $x_i,\ i = 1, ..., 4$. La sortida serà un únic valor (és a dir, $n_s = 1$), que podem anomenar y (és a dir, $x_s = y \in \mathbb{R}$. Normalment, quan s'usa una xarxa neuronal, no es coneix la relació entre les dades d'entrada i la de sortida (si es conegués, no ens caldria usar la xarxa). En aquest cas, però, si que coneixerem la relació entre l'entrada i la sortida, permetent, doncs, comprovar a posteriori si la xarxa ha fet un bon ajust dels pesos w, tot provant el seu funcionament amb vectors d'entrada qualsevols. Aquesta relació entre l'entrada i la sortida serà:

$$y = \begin{cases} 1 & \text{si } \sum_{i=1}^{4} x_i > 2 \\ 0 & \text{si } \sum_{i=1}^{4} x_i \leq 2 \end{cases} \tag{3.4}$$

És a dir, si els quatre valors d'entrada sumen més de 2 la sortida serà un 1, i serà un 0 altrament. Cal indicar que tots els valors $x_i,\ i = 1, ..., 4$ d'entrada a la xarxa han d'estar entre 0 i 1. Aquest és un requeriment de caràcter pràctic de les xarxes neuronals, per poder garantir el seu bon funcionament.

Per poder realitzar l'"aprenentatge" de la xarxa neuronal, tal i com abans s'ha comentat, cal disposar d'un total de p vectors d'entrada amb la seva corresponent sortida. Per tal de generar aquests valors teniu a la vostra disposició un generador automàtic de dades segons la

relació (3.4). Aquest es troba a l'estació de treball tanit.etse.urv.es dels Recursos Informàtics de l'E.T.S.E., a la qual tots vosaltres teniu accés. Per obtenir les dades del vostre problema cal que, des del vostre directori, executeu l'ordre:

```
~jcastro/bin/genxndat fitxer p llavor
```

El primer dels tres paràmetres, `fitxer`, serà el nom del fitxer (del vostre directori) on les dades seran escrites. El segon paràmetre, `p`, representa el nombre de dades d'entrada i sortida que seran escrites al fitxer. En teoria, com major sigui p, més bo serà l'aprenentage. Podeu usar un valor de $p = 50$, el qual no generarà un conjunt de dades molt gran, però sí suficient per garantir un aprenentatge acceptable. El darrer paràmetre és una llavor (entera) per generar les dades de forma aleatòria. Podeu introduir, per exemple, el número del vostre DNI. Així, si feu `~jcastro/bin/genxndat dades.dat 6 654321`, obtindríeu un fitxer `dades.dat` similar a:

```
0.793 0.027 0.877 0.811   1.0
0.449 0.830 0.353 0.267   0.0
0.640 0.312 0.655 0.890   1.0
0.626 0.680 0.439 0.534   1.0
0.184 0.247 0.848 0.823   1.0
0.730 0.213 0.660 0.318   0.0
```

Cada fila correspon a unes dades d'entrada/sortida determinades (tenim $p = 6$). Dins de cada fila, les quatre primeres columnes corresponen als valors d'entrada $x_i,\ i = 1, ..., 4$, i la darrera columna indica el valor de sortida y. Es pot observar com les dades preserven la relació indicada a (3.4).

La xarxa que considerarem serà com la de la Fig. 3.1: una xarxa de tres capes i set nodes (quatre a la primera capa, dos a la segona i un node a la darrera capa). El problema de minimització que plantejarem serà el presentat a (3.3), usant l'expressió de la xarxa neuronal indicada a (3.1). Les variables del problema de mínims quadrats seran els pesos w. Observant la Fig. 3.1, podem veure com el nombre de variables a optimitzar és de 10. El problema de minimització sense restriccions resultant haurà de ser solucionat amb el paquet Lingo. Trobareu una breu descripció sobre aquest paquet a l'annex I.

Un cop realitzat l'"aprenentatge" de la xarxa neuronal amb l'ajut del paquet Lingo, convé que verifiqueu el bon funcionament de la mateixa introudint-li vectors de dades aleatoris, i comprovant que el resultat proporcionat per la xarxa està d'acord amb la regla (3.4). Podeu veure que, si hi ha alguns vectors d'entrada on la solució proporcionada no és correcta, aquest corresponen a casos on $\sum_{i=1}^{4} x_i \approx 2$.

3.5 Detalls importants a tenir en compte!.

Hi ha dues coses que cal tenir presents en intentar solucionar aquest problema amb el paquet Lingo. La primera d'elles és que els pesos w no tenen cap restricció sobre el seu signe (tant poden ser negatius com positius), i per tant són variables lliures. És convenient indicar-li a Lingo aquesta circumstància, per evitar que, per defecte, les consideri del tipus $w \geq 0$.

El segon punt a tenir en compte fa referència al punt inicial d'iteració. Lingo (com la majoria de paquets d'optimització) realitza un procés iteratiu, i a cada iteració troba un nou punt w^k. El punt inicial w^0, si no s'indica el contrari, és inicialitzat pel paquet a un valor determinat. El problema plantejat en aquesta pràctica (que ve donat per l'expressió (3.3)), és força no lineal, i té molts mínims locals (això és degut a la funció sigmoidal $f(x)$ usada). Si deixem que Lingo inicialitzi el punt w^0, fàcilment podem anar a parar a un mínim local

on l'"aprenentatge" de la xarxa sigui nul. Per evitar aquest inconvenient, sovint s'acostuma a inicialitzar el punt inicial amb valors aleatoris per evitar caure sempre dins el mateix mínim local poc satisfactori. Al nostre cas concret, podeu mirar d'inicialitzar el punt inicial de forma que $w^0_{i,j} = 1\ \forall_{i,j}$, excepte la component $w_{6,7}$, la qual serà inicialitzada com $w^0_{6,7} = -1$. Si aquest punt inicial no proporcionés un "bon" mínim local, mireu d'introduir d'altres de forma aleatòria. Podeu observar si el mínim local obtingut és un "bon mínim" amb el valor final de la funció objectiu: aquest ha de ser proper a 0 si l'"aprenentatge" ha estat satisfactori (un valor de 0 significa que hi ha pocs errors a l'ajust per mínims quadrats). Per indicar el punt inicial d'iteració a Lingo podem usar les ordres

```
INIT:
   variables a inicialitzar
ENDINIT
```

3.6 Presentació de la pràctica.

L'informe que haureu de presentar ha de tenir l'estructura general recomanada a la "Introducció" de la col·lecció de pràctiques. Entre la informació específica que ha de contenir l'informe, ha d'aparèixer:

- Un llistat del fitxer amb la modelització del problema en el llenguatge de Lingo.
- Un llistat del resultat que Lingo us ha proporcionat, indicant el valor de funció objectiu i pesos w que heu obtingut.
- Els resultats que heu obtingut avaluant la xarxa amb uns quants vectors d'entrada aleatoris, comprovant el bon (o mal) funcionament de la xarxa per classificar les entrades.
- Qualsevol observació o comentari que considereu escaient, així com qualsevol problema que us hagi aparegut durant la realització de la pràctica.

PRÀCTICA 4. Programació de l'algorisme del símplex (6 punts).

L'objectiu de la pràctica és realitzar la programació de l'algorisme del símplex. Aquesta programació es farà en un llenguatge de molt alt nivell que permet operar directament amb matrius i vectors, simplificant enormement la tasca a realitzar. Teniu a la vostra disposició el paquet Octave, especialment adequat per realitzar aquest tipus d'operacions (entre moltes altres coses). Trobareu una breu descripció sobre aquest paquet a l'annex II. De totes formes, si algú volgués fer la programació en qualsevol altre llenguatge només ha de posar-se en contacte amb el professor responsable de l'assignatura i comunicar-li quin llenguatge pensa utilitzar.

Més concretament, s'haurà d'elaborar un programa per solucionar problemes de programació lineal en forma estàndard:

$$\begin{aligned} \min \quad & c^T x \\ \text{subj. a} \quad & \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

No cal, aleshores, que considereu fites superiors a les variables ni restriccions de desigualtat. El programa ha de donar el punt solució, o bé indicar si el problema és infactible o il·limitat. Les dades que rebrà seran el vector de costos c, la matriu A de restriccions, i els termes independents b. A més, podem considerar, per simplificar la cerca d'una solució inicial factible, que el vector b sempre serà proporcionat de forma que sigui no negatiu ($b_i \geq 0$).

La programació és totalment lliure. Tot i així, es podria pensar en tenir una rutina que rebés uns costos $\tilde{c}$, una matriu de restriccions $\tilde{A}$ i els termes independents b (tal i com abans s'ha dit) i un vector x_0 inicial que fos factible. Aquesta rutina hauria de ser capaç de solucionar el problema $\{\min \tilde{c}^T x \text{ subj. a } \tilde{A}x = b, \ x \geq 0\}$ començant a iterar a partir del punt factible x_0. Recordant la forma de treball de l'algorisme del símplex, els passos que hauria de tenir aquesta rutina serien:

Donada la base factible inicial B, la matriu $A = [B\ N]$, i els vectors b, $c^T = [c_B^T\ c_N^T]$:
Calcular ρ i x_B: $\rho = c_N^T - c_B^T B^{-1} N$, $x_B = B^{-1} b$
mentre alguna component $\rho_q < 0$ fer
  Calcular $T_q = B^{-1} N_q$
  si $T_q \leq 0$ llavors
    Stop: problema il·limitat
  fi_si
  Trobar l'índex p de la variable bàsica tal que $T_{q_p} = \min\{x_{B_i}/T_{q_i} \ \ \forall T_{q_i} > 0\}$
  Actualitzar B i N: $B := B - \{B_p\} + \{N_q\}$, $N := N + \{B_p\} - \{N_q\}$
  Calcular ρ i x_B amb la nova base: $\rho = c_N^T - c_B^T B^{-1} N$, $x_B = B^{-1} b$
fi_mentre

Si tenim la rutina anterior feta, ara podem solucionar el problema original cridant-la dues vegades. La primera vegada seria cridada per trobar un punt inicial factible del problema original (fase I). En aquest cas caldria ampliar el problema amb variables artificials. La matriu $\tilde{A}$ seria la matriu A ampliada amb les columnes de les variables artificials, els costos $\tilde{c}$ serien

de 0 per a les variables originals i de 1 per a les variables artificials, i el punt inicial x_0 el podríem obtenir de forma immediata (les components de les variables originals iguals a 0 i la component de la i-èssima variable artificial igual a b_i). En acabar, tornaríem a cridar a la rutina anterior (fase II), ara fent que $\tilde{c} = c$ i $\tilde{A} = A$, i donant com a punt inicial el punt òptim assolit en la primera crida. Si alguna de les variables artificials va quedar a la fase I dins la base com a variable degenerada, la base que tenim en iniciar la fase II no tindrà el nombre suficient de variables (no seria per tant una base). En aquest cas caldria ampliar-la amb les variables-columnes necessàries fins tenir una base.

L'informe que heu de presentar haurà d'incloure un breu esment sobre el disseny del programa, el codi desenvolupat (en el llenguatge d'Octave o del paquet que hàgiu escollit) totalment comentat (afegiu tantes línies de comentari com sigui necessari), un parell d'exemples on es mostri el funcionament del programa, i aquelles observacions o problemes trobats durant la realització del programa que considereu que val la pena esmentar. També convé que esmenteu qualsevol ampliació o modificació incorporada a la pràctica (si s'escau).

PRÀCTICA 5. Programació de l'algorisme de l'escalat afí primal (4.5 punts).

L'objectiu de la pràctica és realitzar la programació de l'algorisme de l'escalat afí primal tal i com s'ha presentat a les classes de teoria. A l'igual que a la pràctica 4 podeu usar el paquet Octave, especialment adequat per realitzar operacions matricials directament. Si voleu, però, podeu usar qualsevol altre llenguatge (consultant-ho prèviament amb el professor responsable de l'assignatura).

El programa haurà de solucionar problemes de programació lineal en forma estàndard:

$$\begin{aligned} \min \quad & c^T x \\ \text{subj. a} \quad & \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

No cal, aleshores, que considereu fites superiors a les variables ni restriccions de desigualtat. Només ens preocuparem d'aquells problemes que són factibles. En aquest cas el programa haurà de proporcionar la solució òptima. Si el problema és infactible o il·limitat no ens interessa el comportament de l'algorisme.

Les dades que rebrà el programa seran el vector de costos c, la matriu A de restriccions, i els termes independents b. La matriu A haurà de ser ampliada amb una nova variable per tal de poder usar el mètode Big-M que ens permeti obtenir directament un punt inicial factible i interior.

Recordeu que l'algorisme de l'escalat afí primal consta, a grans trets, dels següents passos:

Inicialitzacions: $x_0 > 0$, $Ax_0 = b$, $i = 0$
mentre no es satisfan els criteris d'aturada fer
    $D = \text{diag}(x_{i_1}, , ..., x_{i_n})$
    Trobar y_i: $(AD^2A)^T y_i = AD^2c$
    Trobar z_i: $z_i = c - A^T y_i$
    Trobar dx_i: $dx_i = -D^2 z_i$
    Trobar α: $\alpha = \min\{-x_{i_j}/dx_{i_j} \quad \forall dx_{i_j} < 0\}$
    Trobar x_{i+1}: $x_{i+1} = x_i + \rho\alpha dx_i$ on $\rho \in [0.95, 0.9995]$
    $i := i + 1$
fi_mentre

L'informe que heu de presentar haurà d'incloure un breu esment sobre el disseny del programa, el codi desenvolupat (en el llenguatge d'Octave o del paquet que hàgiu escollit) totalment comentat (afegiu tantes línies de comentari com sigui necessari), un parell d'exemples on es mostri el funcionament del programa, i aquelles observacions o problemes trobats durant la realització del programa que considereu que val la pena esmentar (especialment comenteu tot allò referent sobre el paràmetre M escollit al mètode Big-M). També seria interessant que, usant algun dels dos exemples presentats per mostrar el funcionament del programa, comparéssiu el comportament de l'algorisme quan la matriu D no es defineix com $D = \text{diag}(x_{i_1}, , ..., x_{i_n})$, i s'usa

simplement $D =$ (matriu identitat), amb el qual no s'està realitzant cap tipus d'escalat. Esmenteu també qualsevol ampliació o modificació que hàgiu incorporat a la pràctica (si s'escau).

PRÀCTICA 6. Formulació d'un problema amb el format MPS (3.5 punts).

El format MPS (Mathematical Programming System) és un format d'introducció de problemes de programació lineal, que gairebé la majoria de paquets d'optimització accepten. Aquest format va ser pensat ja fa alguns anys, quan el Fortran era pràcticament l'únic llenguatge usat en el desenvolupament de paquets d'optimització. Això fa que actualment s'hagi quedat una mica "antiquat". Tot i així, el seu ús està encara força arrelat, i és força probable que algú que hagi de treballar amb paquets d'optimització tard o d'hora hagi de formular un problema usant la sintaxi del format MPS. És per això que val la pena saber què és i conèixer mínimament la seva sintaxi. L'objectiu de la pràctica consistirà, doncs, en solucionar un petit problema de programació lineal mitjançant el paquet lp_solve(veure annex III), tot codificant el problema en qüestió en format MPS. En primer lloc detallarem com és el format MPS, i finalment indicarem com serà el problema que heu de solucionar.

6.1 El format MPS.

Aquest format s'usa per especificar els noms de les restriccions i variables, indicar com intervé cada variable dins cada restricció, i definir els termes independents de les restriccions i els límits de les variables. Tal i com hem dit, el MPS és un format estandard d'especificació de problemes usat per diversos paquets d'optimització.

El format d'entrada no és lliure i cada paraula clau ha d'estar entre unes columnes determinades al fitxer. L'aspecte general de l'apartat MPS és el següent (el símbol ␣ representa una separació entre paraules sempre i quan no sigui un espai en blanc —no podem usar espais en blanc per definir noms de variables o restriccions—, i el números al principi de tot representen l'encolumnat del fitxer MPS) :

```
         1         2         3         4         5         6         7
1234567890123456789012345678901234567890123456789012345678901234567890

NAME          nom␣problema

ROWS
 ww restricció

COLUMNS
    variable  restri␣1  coeficient␣1   restri␣2  coeficient␣2

RHS
    nom␣indp  restri␣␣  terme␣independent

RANGES
    nom␣rang  restri␣␣  valor␣del␣rang

BOUNDS
```

```
 zz nom␣boun  variable  valor␣del␣límit

ENDDATA
```

Al MPS anterior en majúscula apareixen les paraules clau i en minúscula les dades que varien d'un problema a l'altre (i subratllats hi ha el nombre de caràcters màxim que pot ocupar cada nom). Els camps que s'han marcat com `ww, nom␣indp, nom␣rang, zz` i `nom␣boun` indiquen el tipus de restricció (`ww`), el nom donat al conjunt de termes independents (`nom␣indp`) , el nom donat al conjunt de rangs (`nom␣rang`) , el tipus de límit de la variable (`zz`) i el nom donat al conjunt de límits (`nom␣boun`). Descriurem a continuació cadascuna de les seccions del MPS.

6.1.1 Secció NAME.

S'usa per donar un nom al problema que s'està codificant. El format que hem d'utilitzar és:

```
          1         2
12345678901234567890123
NAME.........problema
```

on `problema` és el nom que donem al problema i els punts indiquen espais en blanc.

6.1.2 Secció ROWS.

Declara el nom i tipus de les restriccions (i de la part lineal de la funció objectiu si n'hi ha). El format que hem d'utilitzar és:

```
          1
1234567890123
ROWS
 .ww.restricc
```

on els punts indiquen espais en blanc, `restricc` és el nom de la restricció i `ww` ens indica el tipus de restricció que pot ser:

$$= \begin{cases} E & : \; = \\ G & : \; \geq \\ L & : \; \leq \\ N & : \; \text{funció objectiu o restricció lliure} \end{cases}$$

6.1.3 Secció COLUMNS.

Aquesta secció del MPS serveix per:

1. Declarar els noms de les variables.
2. Donar valors als coeficients amb els que intervenen les variables dins de cada restricció lineal.

El format d'escriptura d'aquesta secció és:

```
          1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890123456789012
COLUMNS

....variable..restri␣1..coeficient␣1...restri␣2..coeficient␣2
```

on els punts indiquen espais en blanc, `variable` és el nom de la variable que estem tractant, `restri␣1` i `restri␣2` són noms de restriccions on intervé la variable en qüestió, i `coeficient␣1` i `coeficient␣2` indiquen els valors amb que la variable que tractem afecta a cada restricció. Cal tenir en compte que en aquest apartat han d'aparèixer els noms de totes les variables lineals, fins i tot si no tenen cap coeficient associat. També cal afegir que fins que no hem acabat de definir tots els coeficients per una variable, no podem passar a la següent.

6.1.4 Secció RHS.

Declara els termes independents de totes les restriccions. Poden anar en qualsevol ordre. El format d'escriptura és:

```
          1         2         3
12345678901234567890123456789012345678901234567890123456789012345678901234
RHS

....nom␣indp..restri␣i..terme␣indept
```

on els punts indiquen espais en blanc, `nom␣indp` indica el nom que donem al conjunt de termes RHS, `restri␣i` indica el nom de la restricció que tractem i `terme␣indept` representa el valor del terme independent. El nom `nom␣indp` és arbitrari però ha de ser el mateix per a totes les components d'un mateix vector de termes independents, i només serveix per a donar nom a aquest vector.

6.1.5 Secció RANGES.

S'usa per definir restriccions del tipus $l \leq f_i \leq u$. El format d'escriptura d'aquesta secció és:

```
          1         2         3
12345678901234567890123456789012345678901234567890123456789012345678901234
RANGES

....nom␣rang..restri␣i..terme␣ranges
```

on els punts indiquen espais en blanc, `nom␣rang` indica el nom que donem al conjunt de rangs, `restri␣i` indica el nom de la restricció que tractem i `terme␣ranges` representa el valor del rang. El nom `nom␣rang` té la mateixa funció que el nom `nom␣indp`. Si a l'apartat RHS s'ha definit $F(x) \leq u$ i volem tenir $l \leq F(x) \leq u$ el valor del rang ha der ser $rang = u - l$.

6.1.6 Secció BOUNDS.

Declara les fites de les variables. El seu format és:

```
          1         2         3
12345678901234567890123456789012345678901234567890123456789012345678901234
```

```
BOUNDS
.zz.nom␣boun..variable..terme␣bounds
```

on els punts indiquen espais en blanc, `nom␣boun` indica el nom que donem al conjunt de fites, `variable` indica el nom de la variable que tractem i `terme␣bounds` representa el valor de la fita. El nom `nom␣boun` té la mateixa funció que als dos apartats anteriors. El camp `zz` ens indica el tipus de fita i pot prendre els valors:

$$= \begin{cases} LO & : \leq \\ UP & : \geq \\ FR & : \text{variable lliure} \\ FX & : = \end{cases}$$

En la majoria de paquets, si no s'indica cap fita per a una variable aquesta es considera del tipus $x_i \geq 0$.

6.1.7 Exemple de codificació d'un problema en format MPS.

Sigui el problema:

$$\begin{array}{lccccc} \min. & x_1 & +x_2 & +x_3 & & \\ \text{subj. a} & x_1 & +x_2 & & = 2 \\ & x_1 & & +x_3 & \geq 0 \\ & 2x_1 & +4x_2 & -x_3 & \leq 10 \\ & x_1 \geq 0, & x_2 \geq 0, & 0 \leq x_3 \leq 1 & & \end{array}$$

El format MPS associat a aquest problema seria:

```
        1         2         3         4         5         6         7
12345678901234567890123456789012345678901234567890123456789012345678901234567890
NAME          PROBLEM1
ROWS
 N  FOBJ
 E  REST1
 G  REST2
 L  REST3
COLUMNS
    X1        FOBJ      1.0
    X1        REST1     1.0
    X1        REST2     1.0
    X1        REST3     2.0
    X2        FOBJ      1.0
    X2        REST1     1.0
    X2        REST3     4.0
    X3        FOBJ      1.0
    X3        REST2     1.0
    X3        REST3     -1.0
RHS
    TERMINDE  REST1     2.0
```

```
    TERMINDE  REST2     0.0
    TERMINDE  REST3     10.0
BOUNDS
 LO LIMSIMP   X1        0.
 LO LIMSIMP   X2        0.
 LO LIMSIMP   X3        0.
 UP LIMSIMP   X3        1.
ENDDATA
```

6.2 Problema a solucionar.

El problema a solucionar serà lliure, i haureu de crear un qualsevol. Aquest problema, però, haurà de tenir un mínim de 10 variables i 5 restriccions. La fita inferior de totes les variables serà 0, mentre que la fita superior variarà. Entre les restriccions hi haurà d'haver de $=$, $\geq$, $\leq$, i com a mínim una amb rangs (desigultats del tipus $l \leq a^T x \leq u$). Les restriccions no poden ser límits simples a les variables (és a dir, no han de ser restriccions del tipus $x_i \geq l$ o $x_i \leq l$). El problema haurà de ser factible.

El problema l'haureu de solucionar amb el paquet lp_solve. En aquest cas només cal executar aquest paquet indicant-li el fitxer de dades en format MPS. Trobareu una molt breu explicació d'aquest paquet a l'annex III.

6.3 Presentació de la pràctica.

Es recomana que l'informe a presentar tingui l'estructura general exposada a la "Introducció" de la col·lecció de pràctiques. Per aquesta pràctica en particular haureu d'adjuntar la formulació del problema que heu creat (formulat com a problema de programació lineal), el fitxer en format MPS que modelitza aquest problema, i la solució obtinguda amb el paquet lp_solve. Podeu adjuntar qualsevol comentari que considereu rellevant.

ANNEX I. Breu introducció al paquet Lingo.

Lingo és un paquet comercial de modelització de problemes. Això vol dir que amb ell es poden formular, en un llenguatge més o menys senzill, una gran varietat de problemes de programació matemàtica (tant lineal, com no lineal, com entera), i obtenir directament la solució del mateix. Internament el que fa Lingo quan li entrem un model és analitzar-lo, decidir quin tipus de problema és (lineal, no lineal, etc.) i cridar a una rutina interna d'optimització específica per aquest tipus de problemes. La versió que teniu disponible de Lingo corre sota Windows 3.xx i Windows95, i es troba instal·lada als PC's de l'aula informàtica.

Aquí només esmentarem algunes de les característiques del llenguatge de modelització de Lingo. Per tenir més detalls es pot consultar el manual de referència del paquet (adreceu-vos al vostre professor), o bé es pot donar una ullada al "help" (força clar) que té incorporat.

La forma de treballar amb Lingo es pot resumir breument de la següent manera. Quan entreu a Lingo us apareixerà una finestra on podreu editar el vostre model. Aquest convé que el salveu en un fitxer. Això ho podeu fer amb l'opció "Save" del menú "File" de la barra superior (a través d'aquesta barra tenim accés a totes les possibilitats que Lingo ens ofereix). Un cop introduït el model, podem mirar de solucionar-lo amb l'opcio "Solve" del menú "Lingo". Si el model que heu entrat es correcte, Lingo mirarà de solucionar-lo. Si hi ha algun error de sintaxi, us ho dirà. Si tot ha anat bé, una nova finestra us mostrarà la solució obtinguda. A l'igual que abans amb el model, podeu salvar aquesta solució en un altre fitxer (per tal de poder incorporar-la a l'informe). Adoneu-vos que, en la solució, Lingo ens mostra el valor òptim de cada variable, i el valor de les restriccions en aquest punt. La primera restricció acostuma a ser el valor de la funció objectiu.

Ara ens falta el més important, que és saber com formular un problema amb el llenguatge de Lingo. Aquí només mostrarem el seu funcionament a través d'uns pocs exemples (fer una descripció detallada implicaria reproduir el manual de Lingo, cosa que no tindria sentit). Aquells que necessiteu o tingueu interès en saber més coses, consulteu el manual de referència o el "help" interactiu de Lingo. Passem a formular aquests exemples amb el llenguatge de Lingo.

Exemple 1.

Volem solucionar el següent problema de programació lineal:

$$\begin{aligned} \min \quad & x_1 + x_2 + x_3 \\ \text{subj. a} \quad & \\ & 2x_1 - 3x_2 \geq 3 \\ & -3x_1 + 3x_2 \leq 5 \\ & 1 \leq x_1 \leq 3 \quad x_2 \geq 0 \quad x_3 \text{ lliure} \end{aligned}$$

Aquest problema el podríem formular de la següent manera usant Lingo:

```
!tot el que comença amb admiració es un comentari;
```

```
!totes les sentències han d'acabar amb un punt i coma, comentaris inclosos;

!això es la funció objectiu;
!si ocupés més d'una línia no caldria;
!afegir punt i coma a cada una d'elles;
min= x1+x2+x3;

!ara escrivim les restriccions;
2*x1-3*x2 >= 3;
-3*x1+3*x2 <= 5;

!i finalment les fites a les variables;
!per defecte totes les variables tenen una fita inferior igual a 0;
!la funció @bnd ens permet definir límits superiors i inferiors;
@bnd(1,x1,3);
!la funció @free ens permet definir variables lliures;
@free(x3);
```

Exemple 2.

Volem solucionar el següent problema de programació no lineal:

$$\begin{aligned} \max \quad & e^{x_1} + \ln x_2 + x_3^2 \\ \text{subj. a} \quad & \\ & \frac{2x_1}{3x_2} \geq 3 \\ & x_1/x_2 \leq 5 \\ & x_1 \geq 3 \end{aligned}$$

Això podria ser formulat com:

```
!funció objectiu;
!aquí es mostren les funcions @exp i @log.  Lingo té moltes;
!més funcions matemàtiques;
max= @exp(x1)+@log(x2)+x3^2;
!restriccions;
(2*x1)/(3*x2) >= 3;
x1/x2 <= 5;
!fites a les variables;
@bnd(3,x1,1.0e+20);
!tambe podríem haver fet x1>=3.0, però en aquest cas consideraria;
```

```
!aixo com una nova restriccio, i no com un límit de variables;
```

Exemple 3.

Als exemples anteriors hem usat Lingo més com un paquet per solucionar problemes que per modelitzar-los. De fet als exemples anteriors no hem explotat prou les possibilitats de Lingo. Veurem ara com modelitzar un problema de programació entera, concretament el problema de la motxil·la. En aquest problema, disposem d'una sèrie d'objectes que volem col·locar dins una motxil·la. Tenim 5 objectes, i sabem que afegir l'objecte i ens suposa un benefici de g_i unitats, i ocupa un volum de v_i litres. El volum total de la motxil·la és de V litres. Hem de formular el problema de forma que carreguem la motxil·la maximitzant el nostre benefici. La formulació matemàtica d'aquest problema és:

$$\begin{aligned} \max \quad & \sum_{i=1}^{5} g_i x_i \\ \text{subj. a} \quad & \\ & \sum_{i=1}^{5} v_i x_i \leq V \\ & x_i \in \{0,1\} \; i = 1, ..., 5 \end{aligned}$$

Usant Lingo podríem fer:

```
!amb la sentència ‘‘model’’ indiquem que creem un nou model;
!acabem el model amb la darrera sentència ‘‘end’’;
!model:  no acaba amb punt i coma;
model:
!la sentència ‘‘sets’’ serveix per crear conjunts de dades;
!en aquest cas creem un conjunt de 5 objectes;
!cada objecte tindrà 3 atributs:  volum, benefici i x;
!(x és la variable a optimitzar que valdrà 1 o 0);
!sets:  no acaba amb punt i coma;
  sets:
        objectes/1..5/:x,volum,benefici;
  endsets

!la sentència data serveix per assignar valors a variables;
!assignarem per als 5 objectes el benefici i volum que ocupa, i el;
!volum total de la motxil·la;
```

```
!data:  no acaba amb punt i coma;
  data:
        V= 54;
        benefici= 10 11 9 13 12;
        volum= 17 18 15 21 20;
  enddata

!funcio objectiu;
!aqui indiquem amb la funció @sum que sumi el producte de x per benefici;
!per a tots els objectes;
  max= @sum(objectes:  x*benefici);

!restriccions;
!aqui indiquem que la suma per a tots els objectes de x pel volum ha de;
!ser menor que el volum total de la motxil·la;
   @sum(objectes:x*volum) <= V;

!límits de les variables;
!finalment indiquem amb la funcio @bin que totes les variables x de totes;
!els objectes són binàries (només poden prendre els valors 0 o 1);
!la funció @for ens permet tractar tots els elements del conjunt de dades;
!objectes;
   @for(objectes:@bin(x));
end
```

Exemple 4.

Acabarem modelitzant amb el llenguatge de Lingo el problema de la dieta. Disposem de 3 aliments A, B i C per confeccionar una dieta, cadascun dels quals té una certa proporció de dos nutrient N1 i N2. Aquestes proporcions ens vénen donades segons la taula següent:

	N1	N2
A	35%	10%
B	45%	20%
C	10%	30%

Sabem que cada kg. dels aliments A, B i C costa 100, 110 i 85 pts. respectivament. Volem obtenir quines quantitats d'aquests aliments ens calen per obtenir la dieta més barata possible que aporti més de 40 i 50 kg dels nutrients N1 i N2. La formulació d'aquest problema quedaria com:

$$\begin{aligned} \min \quad & 100x_A + 110x_B + 85x_C \\ \text{subj. a} \quad & \\ & 0.35x_A + 0.45x_B + 0.10x_C \geq 40 \\ & 0.10x_A + 0.20x_B + 0.30x_C \geq 50 \\ & x_A \geq 0 \;\; x_B \geq 0 \;\; x_C \geq 0 \end{aligned}$$

Utilitzant el llenguatge de modelització de Lingo podríem haver fet:

```
model:
   sets:
!  definim el conjunt de 3 productes A,B,C amb atributs preu i quantitat;
!  de producte (la variable a optimitzar);
        productes /A,B,C/:  quant_p,preu;
!  definim el conjunt d'ingredients, amb la quantitat requerida;
        ingredients /N1,N2/:  quant_i;
!  definim la matriu que dona el percentatge d'ingredient per producte;
!  les matrius es defineixen com un producte cartesià de conjunts;
        composicio(productes,ingredients):  percent;
   endsets
!  assignem valors;
   data:
        preu= 100 110 85;
        quant_i= 40 50;
        percent= 0.35 0.10
                 0.45 0.20
                 0.10 0.30;
   enddata
!  funcio objectiu;
!  suma de la quantitat pel preu per a tots els productes;
  min= @sum(productes:  quant_p*preu);
!  restriccions;
!  una restriccio per a cada ingredient, amb terme;
!  de la dreta igual a quant_i(i);
   @for(ingredients(i):
!  i el terme de l'esquerra de cada restricció suma per a tots;
!  els aliments el producte de la seva quantitat pel percentatge;
!  de l'ingredient considerat;
        @sum(productes(j):  percent(j,i)*quant_p(j) ) >= quant_i(i));
end
```

ANNEX II. Breu introducció al paquet Octave.

Existeixen diversos paquets que permeten operar amb estructures matemàtiques com vectors, matrius etc. directament. Entre els més coneguts es troben Mathematica, Maple i Matlab. Aquest darrer és especialment escaient per les seves possibilitats numèriques (Mathematica i Maple estan més orientats cap al càlcul simbòlic). Malauradament tots aquests són paquets comercials. Existeix, però, un paquet anomenat Octave, que és una rèplica de Matlab (encara que menys sofisticat), i és de lliure distribució (tot i que només hi ha versions disponibles per a sistemes Unix). Aquest paquet es troba instal·lat a l'estació de treball tanit.etse.urv.es dels Recursos Informàtics de l'E.T.S.E., a la qual tots vosaltres teniu accés.

Octave va ser desenvolupat originalment a la University of Texas (USA), precisament per a l'ensenyament d'Enginyeria Química, amb l'esperit de proporcionar una eina fàcil d'usar pels alumnes i que permetés fer ràpidament operacions que en Fortran o un altre llenguate similar implicaria molt temps de programació i detecció d'errors. D'aquesta forma és possible centrar-se en el què s'està fet i en els resultats obtinguts, i no en com s'ha de fer. Té l'avantatge de que tot el que es faci amb Octave després pot ser usat amb Matlab, amb petites modificacions (es va dissenyar de forma que fossin el màxim de compatibles).

En aquesta introducció no es pretén fer una descripció exhaustiva del paquet (seria inviable, donat la gran quantitat de possibilitats que ofereix). Tan sols mostrarem com fer algunes operacions matricials, i com dissenyar petits programes.

2.1 Com entrar, sortir, i obtenir informació.

Un cop heu entrat a la màquina tanit.etse.urv.es, podeu entrar dins Octave fent:

```
octave
```

Ara ja estem dins l'entorn d'Octave i podem introduir les comandes necessàries. Octave té un "help" bastant extens. Si feu

```
octave:1> help
```

podreu veure les diferents funcions i sentències que permet. Si voleu veure informació específica d'una sentència o funció, com ara de la funció "abs" (valor absolut) només cal que feu:

```
octave:2> help abs
```

A part del "help", Octave porta incorporat un manual. Per consultar-lo cal que entreu

```
octave:3> help -i
```

Podeu bellugar-vos per la pàgina actual del manual prement <Ctrl>-n (amunt), <Ctrl>-p (avall), <Ctrl>-f (dreta) i <Ctrl>-b (esquerra). Els temes de cada pàgina es troben iniciats amb un *. Si us situeu a sobre d'un d'aquests temes i premeu el <Return> anireu al punt del manual on es detalla. Per tornar a la pàgina anterior podeu fer-ho prement una "u" (a la primera línea de tota pàgina del manual es detalla les tecles que podeu prèmer i el resultat que s'obté). Si no voleu viatjar tant pel manual, sinó anar directament al punt d'aquest on es parla d'algun tema/funció/sentència concrets, per exemple si voleu anar directament al punt del manual on es parla de la funció "abs", podeu fer:

```
octave:4> help -i abs
```

Això us situarà a un punt del manual on surt la paraula "abs". Si voleu veure d'altres punts

podeu buscar una paraula dins el manual, prement <Ctrl>-s i introduint la paraula a buscar; cada cop que premeu <Ctrl>-s buscarà la següent aparició d'aquesta paraula dins el manual, i acabareu amb la tecla <Escape>. Quan esteu dins del manual podeu tornar a l'estat normal de Octave prement la lletra "q". Si no trobeu alguna informació concreta amb el manual ni el "help", podeu adreçar-vos al professor responsable de l'assignatura. Per sortir d'Octave només cal introduir l'ordre `quit` des de la seva línea de comandes.

2.2 Com definir vectors i matrius. Operacions simples amb vectors i matrius.

Ara veurem només com treballar amb vectors, matrius i com fer petits programes amb Octave. A part del que aquí es comentarà, és instructiu llegir-se dins el manual interactiu el tema de "Simple Examples" (a la primera pàgina del manual, accediu al tema "Introduction", i des d'aquí entreu al tema "Simple Examples"). Començarem definint una matriu. Només cal entrar, per exemple

```
octave:5> M= [3 1 2; 0 5 1; 3 1 6]
```

i Octave respondrà dient que

```
M =
  3  1  2
  0  5  1
  3  1  6
```

Ara M representarà la matriu anterior. Si volem veure la matriu M^T (transposada), només cal fer:

```
octave:6> M'
```

i Octave respondrà

```
ans =
  3  0  3
  1  5  1
  2  1  6
```

Aquí "ans=" significa que això és la resposta ("answer") de la nostra ordre. La matriu original M, però, no s'ha modificat. Si volem que M sigui igual a la seva transposada hauríem d'haver entrat:

```
octave:7> M= M'
```

Ara crearem un vector b:

```
octave:8> b= [1 2 3]
```

Per defecte, els vectors els crea per files. Si ara volem que multipliqui la matriu M (que ara és la transposada de l'original) pel vector b, només cal fer

```
octave:9> m*b
```

Això provoca un error, però, ja que el vector el considera per files i ens dirà una cosa com ara:

```
error:  nonconformant matrices (op1 is 3x3, op2 is 1x3)
error:  evaluating binary operator `*' near line 18, column 2
```

Per fer el producte bé hauríem de multiplicar M per b^T (transposat de b):

```
octave:10> m*b'
```

Per tenir b com a vector-columna i evitar anar-lo transposant farem:

```
octave:11> b= b'
```

2.3 Operacions més avançades amb vectors i matrius.

Suposem que ara volem solucionar el sistema d'equacions $Mx = b$. Això ho faríem com:

```
octave:12> x= inv(M)*b
```

i ens donaria

```
x =
  -0.16667
   0.33333
   0.50000
```

En aquest cas estem usant la funció inv() que dóna la inversa d'una matriu. Si volem veure l'aspecte de la inversa, només cal fer:

```
octave:13> inv(M)
```

Si volem crear una matriu $M2$ igual a la nostra matriu M però ampliada amb 3 columnes que corresponguin a la matriu identitat (això pot ser útil per trobar un punt factible al mètode del símplex) podem fer:

```
octave:14> M2= [M eye(3)]
```

i obtindríem:

```
M2 =
  3  0  3  1  0  0
  1  5  1  0  1  0
  2  1  6  0  0  1
```

Aquí hem usat la funcio "eye" que serveix per crear una matriu identitat (en aquest cas de 3 files i 3 columnes). Hi ha d'altres funcions interessants com "ones" (crea matrius/vectors amb uns) i "zeros" (crea matrius/vectors amb zeros). Si voleu obtenir més informació sobre les operacions amb matrius podeu consultar el tema "Linear Algebra" de la primera pàgina del manual interactiu de Octave.

Considerarem ara l'exercici de, donats uns vectors x i dx, trobar el valor

$$\alpha = \min\{-x_i/dx_i \ \ \forall dx_i < 0\}$$

(mínim de $-x/dx$ tenint en compte només aquelles components que verifiquin $dx_i < 0$). Aquesta operació ens pot ser d'utilitat si hem de calcular passes màximes donat un punt actual i una direcció de moviment, tot intentant preservar el límit d'algunes components. Suposem que definim els vectors x i dx com:

```
octave:15> x= [1 2 3], dx=[0 1 -1]
```

(podem definir-los en la mateixa línia si els separem per comes). Podem calcular un vector que doni el quocient component a component fent:

```
octave:16> x ./ dx
```

Aquí l'operand "./" indica que el quocient s'ha de fer component a component. En general, afegir un "." davant d'algun operand fa que aquest operand afecti component a component (tant per vectors com matrius). El resultat de la operació anterior és:

```
ans =
   Inf    2   -3
```

Per saber quines components de dx són < 0 podem fer:

```
octave:17> dx < 0
```

i obtenim

```
ans =
  0  0  1
```

És a dir, es crea un vector amb 0 a la posició i si $dx_i \not< 0$ i amb 1 altrament.

Ara veurem com seleccionar només una sèrie de components d'un vector. Si volem seleccionar del nostre vector x una sèrie de components podem fer:

```
octave:18> x([1 1 0])
```

obtenint:

```
ans =
  1  2
```

Hem obtingut els valors de les dues primeres components. Això li hem indicat amb el vector de 1's i 0's [1 1 0] (un 1 vol dir que seleccioni aquella component, un 0 que no). Si volem seleccionar la 1a i 3a components, hauríem fet

```
octave:19> x([1 0 1])
```

Una cosa important que cal fer perquè funcioni bé Octave quan seleccionem totes les components és afegir sempre que entrem a Octave el següent:

```
octave:20> prefer_zero_one_indexing= "true"
```

O bé ho podeu afegir en el fitxer ".octaverc" del vostre directori arrel (Octave busca allà certes coses que ha de fer quan comença una sessió).

Continuant amb el tema de la selecció, si volem seleccionar les columnes 1 i 5 de la nostra matriu $M2$, per exemple, podríem fer indistintament:

```
octave:21> M2(:,[1,5])
```

o bé

```
octave:22> M2(:,[1 0 0 0 1 0])
```

En ambdós casos el resultat seria

```
ans=
  3  0
  1  1
  2  0
```

Això ens pot ser útil en programar l'algorisme del símplex (podem seleccionar bases, per exemple). Si pel contrari volem seleccionar les files 1 i 3 de $M2$ faríem:

```
octave:23> M2([1,3],:)
```

o bé

```
octave:24> M2([1 0 1 ],:)
```

Si voleu saber més sobre com seleccionar components de vectors, o files i columnes de matrius, consulteu al manual interactiu de Octave l'apartat de "Expressions", i, dins d'aquest, el tema de "Index Expressions".

Ara només ens cal saber que la funció "min" ("max") dóna el valor mínim (màxim) d'un vector. Llavors, combinant tot el que hem dit abans, podem calcular el valor α anterior directament fent:

```
octave:25> alpha= min(-x(dx<0) ./ dx(dx<0))
```

En aquest cas veiem com és d'útil que la comparació $dx < 0$ (o $dx > 0$, $dx >= 0$, $dx <= 0$, $dx == 0$, que són alguns dels altres operadors relacionals) ens proporcioni un vector d'uns i zeros.

2.4 Sentències i funcions. Creació de programes.

Octave té les sentències típiques de qualsevol llenguatge de programació: condicionals (if) i sentències iteratives (while i for). Podeu obtenir informació sobre aquestes consultant el manual interactiu (`help -i if`, `help -i while` i `help -i for`). Concretament, les sentències "while" (fer fins que es verifiqui una condició) i "for" (fer durant un cert nombre d'iteracions) ens poden ser d'utilitat per als nostres algorismes que poden tenir una estructura com ara:

```
octave:26> while (no_punt_optim)
     >#  aqui posem el cos del nostre programa
     >endwhile
```

o bé

```
octave:27> for i = 1:maxim_iteracions
     >#  aqui posem el cos del nostre programa
     >endfor
```

(en el llenguatge d'Octave tot el que segueix al caràcter "#" —o a "%"— és ignorat fins al final de la línea; usarem aquests caràcters, doncs, per posar comentaris). El segon cas (ús del for) ens pot ser útil si volem fixar un nombre màxim d'iteracions. En el primer cas (ús del while) no pararem fins que `no_punt_optim` sigui fals. Per exemple, si estem programant el mètode del símplex, la condició `no_punt_optim` podem definir-la com:

```
octave:28> no_punt_optim= any(costos_reduits < 0)
```

La funció "any" serveix per determinar si alguna component del vector de costos reduïts (`costos_reduits`), que prèviament hem calculat, es menor que 0. El resultat s'assigna a `no_punt_optim`.

Anem a fer un exemple de com crear un programa o funció nova amb Octave. Suposem que amb l'editor escrivim un fitxer que anomenarem "simplex.m" (amb Octave cada funció o rutina s'ha d'escriure en un fitxer, i s'acostuma a posar l'extensio ".m"). El contingut de "simplex.m" pot ser:

```
function x= simplex(A,b,c)

# us: simplex(A,b,c)
#     A es una matriu (m x n)
#     b es un vector columna de dimensio m (b>=0)
#     c es un vector columna de dimensio n
#
# Troba la solucio al problema de programacio lineal
#         min     c'x
#         subj.a  Ax=b
#                 x>=0

# comprovem que les dimensions son correctes
[m,n]= size(A);
[mc,nc]= size(c);
[mb,nb]= size(b);
#     comprovem si c i b son vectors columna
```

```
if ((nc != 1) || (nb != 1))
        disp("Error: c i b han de ser vectors");
        return;
endif
#     comprovem si les dimensions de b i c concorden amb la matriu A
if ((mc != n) || (mb != m))
        disp("Error: dimensions de c, b i A inconsistents");
        return;
endif
# si continuem per aqui les dimensions son correctes
endfunction
```

Observant al fitxer anterior, hem vist com una funció es defineix amb "function valor = nom_funció (llista d'arguments)". Al nostre cas el valor que retorna la funció és el vector x òptim, el nom de la funció serà "símplex", i els arguments són la matriu A de restriccions, i els vectors b i c. La funció s'acaba amb la paraula "endfunction". A continuació de la capçalera de la funció hi ha una explicació comentada de com usar la funció. Proveu, des de dins d'Octave ara, a fer:

```
octave:29> help simplex
```

Octave detecta la funció al fitxer que heu definit i mostra la petita explicació d'ús de la rutina. Per cridar la funció només cal crear un vector columna de costos qualsevols, per exemple:

```
octave:30> c= [1 ; -1 ; 3; 5 ; 0 ;1]
```

i fer ara

```
octave:31> simplex(M2,b,c)
```

(recordeu que la matriu $M2$ i el vector b ja els teníem creats d'abans). El que fa el programa ara es verificar que les mides de la matriu i vectors són consistents (proveu a executar-lo amb mides inconsistents, i veureu els missatges que dóna). La funció "size" retorna el nombre de files i columnes d'una matriu (feu "`help size`" per obtenir més informació), i permet guardar els valors a dos variables. Fixeu-vos que s'ha afegit un punt i coma (;) al final de cada sentència. Això ho podíem haver fet també abans en qualsevol ordre que hem donat a Octave. Afegir el ";" serveix perquè l'ordre s'executi sense mostrar el resultat per pantalla (això quedaria lleig en un programa). Proveu a treure el punt i coma de "[m,n]= size(A)", per exemple, i torneu a fer `simplex(M2,b,c)`. Veureu com ara es mostren per pantalla els valors de "m" i "n".

Algunes coses interessants d'aquests petit programa (que no fa res, només comprova mides) és l'ús dels condicionals "if" (que ja hem comentat), i dels operadors "!=" (que significa "no igual", el "!" davant d'algun operador de comparació provoca la seva negació) i "`||`" (que vol dir "o"). El "i" es denota amb "&&". La sintaxi dels operadors relacionals és igual que en el llenguatge C. Si voleu consultar més detalls podeu fer `help -i logical`.

Una altra cosa a destacar es la instrucció per escriure coses per pantalla: "disp". Dins podeu escriure el missatge que vulgueu. Podeu obtenir més informació sobre la instrucció "disp" fent `help -i disp`. Per escriure coses de forma més elaborada podeu usar també la instrucció "printf", més potent i no tan simple d'usar (és idèntica a la instrucció del mateix nom del llenguatge C de programació). Podeu obtenir més informació fent `help -i printf`. Cal observar també l'ús de la sentència "return". Això provoca que la rutina actual acabi i retorni.

Un cop detectat que les dimensions són correctes podríem cridar a una rutina auxiliar que faria la minimització donat un punt factible inicial x_0 factible. Aquesta rutina auxiliar la

podem escriure en un fitxer "simplex_fact.m", per exemple, i la seva capçalera seria:

```
function x= simplex_fact(A,b,c,x0)
 ...aquí cal escriure el cos de la funció...
endfunction
```

Aquesta rutina podria ser cridada dos cops dins la rutina "simplex". Aquestes crides serien tal com ara:

```
# calculem punt factible inicial
x=simplex_fact([A eye(m)], b, c2=[zeros(n,1);ones(m,1)], [zeros(n,1);b]);
# calculem punt optim usant com punt factible inicial anterior el
# trobat a la crida (caldria comprovar abans, pero, que c2'*x es igual a 0).
x= simplex_fact(A, b, c, x);
```

A la primera crida amplíem el nostre problema amb variables artificials. La nova matriu de restriccions ampliada és "[A eye(m)]" (perquè hem considerat que $b \geq 0$; si volem considerar vectors b qualsevols cal modificar l'expressió "[A eye(m)]"). El vector de costos nou és "c2=[zeros(n,1);ones(m,1)]", amb un cost de 1 per les variables artificials i 0 per la resta (aquí "zeros(n,1)" crea un vector de n components iguals a 0, i "ones(m,1)" un vector m-dimensional de 1's). I el punt solució factible inicial és "[zeros(n,1);b]" (assignem el vector "b" a les variables artificials i 0 a la resta). A la segona crida prendríem el punt òptim obtingut prèviament i el consideraríem com punt inicial factible del problema original. Tal i com s'indica a la línea de comentari, caldria, però, comprovar que el punt obtingut a la primera crida és un punt factible pel problema original (hem de veure si totes les variables artificials tenen un valor de 0).

Si voleu obtenir més informació sobre com definir funcions i rutines podeu consultar el tema "Functions and Scripts" de la primera pàgina del manual interactiu de Octave.

ANNEX III. Breu introducció al paquet lp_solve.

El paquet lp_solve ha estat desenvolupat per Michel Berkelaar (Eindhoven University of Technology, adreça de correu electrònic michel@es.ele.tue.nl) i pot ser obtingut lliurement (només per usos acadèmics) connectant-se al servidor ftp.es.ele.tue.nl, dins el directori pub/lp_solve. Hi ha versions tant per a sistemes Unix com MS-DOS. La descripció aquí presentada fa referència a la versió per Unix. Concretament aquesta versió es troba instal·lada a l'estació de treball tanit.etse.urv.es dels Recursos Informàtics de l'E.T.S.E., a la qual tots vosaltres teniu accés.

El paquet lp_solve permet solucionar tant problemes de programació lineal com de programació entera. Per aquest darrer tipus de problemes usa la tècnica de "branch and bound". El problema estàndard que pot solucionar pot ser escrit, doncs, com:

$$\begin{aligned} &\min_x \quad c^T x \\ &\text{subj. a} \\ &\qquad Ax \sim b \\ &\qquad 0 \leq x \leq u \\ &\qquad x \in I \text{ enteres} \end{aligned}$$

on $\sim$ representa un vector les components del qual poden ser $=$, $\geq$ o $\leq$, u són les fites superiors de les variables, i I és el conjunt de les variables que són enteres (si $I = \emptyset$ tenim un problema de programació lineal). Adonem-nos que lp_solve no pot tractar fites inferiors diferents de 0. Si volem afegir aquest tipus de fites hem de realitzar un canvi de variables.

La forma d'usar el paquet és tan simple com crear un fitxer de dades amb la descripció del problema en format MPS (per exemple, creem el fitxer "problema.mps") i a continuació indiquem a lp_solve que l'executi fent:

```
lp_solve -mps < problema.mps
```

En afegir l'opció "-mps" li estem indicant que les dades es troben en format MPS (en Unix les opcions d'execució d'un programa s'indiquen amb "-"). Amb "< problema.mps" li estem dient que llegeixi el problema del fitxer "problema.mps". lp_solve solucionarà el problema i ens mostrarà el resultat per pantalla (funció objectiu i valors òptims de les variables). Si volem que la sortida l'escrigui en un fitxer anomenat, per exemple, "problema.sol", farem:

```
lp_solve -mps < problema.mps > problema.sol
```

lp_solve accepta els problemes en un format propi molt més fàcil de ser llegit per les persones. En aquest cas no cal afegir l'opció "mps". Podeu transformar el vostre problema en format MPS a aquest format propi de lp_solve fent

```
mps2eq < problema.mps >problema.lp
```

Ara el nou fitxer "problema.lp" té la descripció del vostre problema en el format propi de lp_solve. A més, lp_solve té opcions addicionals. Entre aquestes destacarem només l'opció "-v" ("verbose mode"). Aquesta opció fa que lp_solve tregui informació addicional per pantalla.

Si voleu obtenir informació i una descripció completa de les opcions del paquet, podeu usar la comanda "man" de Unix (ajuda de Unix) fent:

```
man lp_solve
```