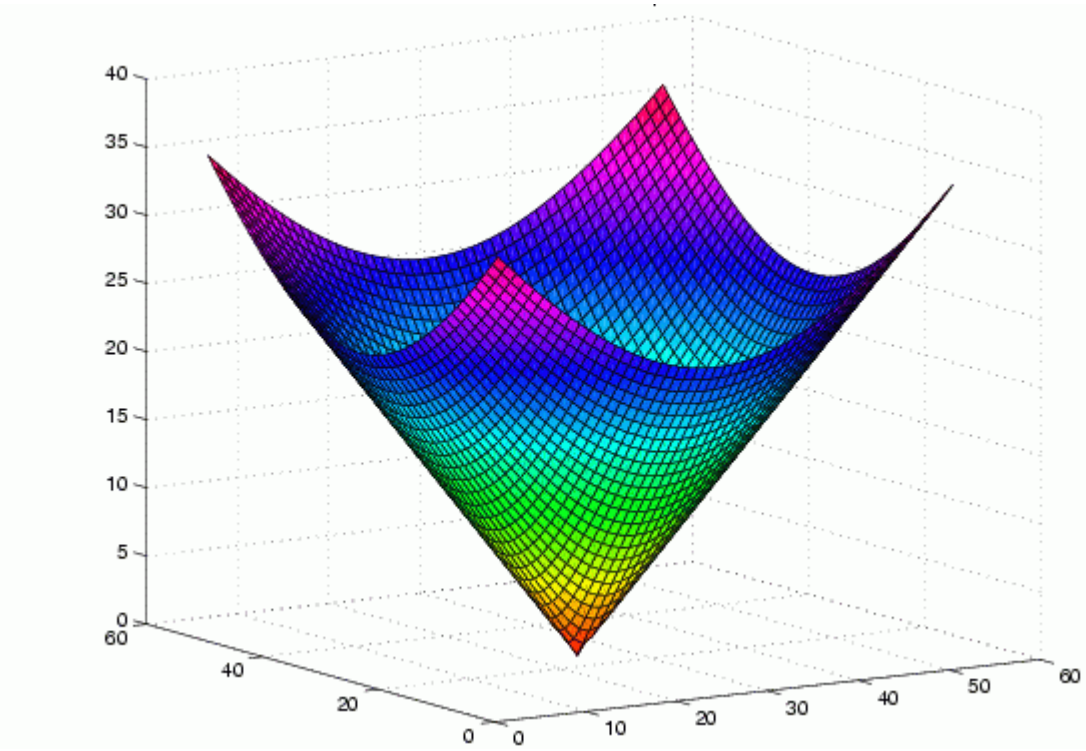
Трунов О.М., Волкова С.О.

ДОВІДНИК

Matlab7.0. Optimization functions

Дисципліна «Дослідження операцій»



Миколаїв – 2008

Аргументи функцій (Function Arguments)

Вікно інструментарію функцій оптимізації використовує такі аргументи:

- Вхідні аргументи: містить загальний опис вхідних аргументів, які використовуються функціями у вікні інструментарію.
- Вихідні аргументи: містить загальний опис вихідних аргументів, які використовуються функціями у вікні інструментарію

Якщо необхідно, специфічна інформація про аргументи міститься на сторінках з описом конкретних видів функцій.

Таблиця 1. Вхідні аргументи для оптимізаційних функцій

АРГУМЕНТ	ОПИС	ВИКОРИСТОВУЄТЬСЯ У ФУНКЦІЯХ
A, b	Матриця A та вектор b є, відповідно, коефіцієнтами лінійних обмежень-нерівностей. Відповідний вектор у правій частині: $A*x<=b$	***fgoalattain, fmincon, fminimax, fseminf, linprog, lsqlin, quadprog***
Aeq, beq	Матриця Aeq та вектор beq є, відповідно, коефіцієнтами лінійних обмежень-рівностей. Відповідний вектор у правій частині: $Aeq*x=beq$	***fgoalattain, fmincon, fminimax, fseminf, linprog, lsqlin, quadprog***
C, d	Матриця C та вектор d є, відповідно, коефіцієнтами визначеної або невизначеної системи лінійних рівнянь. Функція знаходитиме вектор правої частини (вектор рішень).	***lsqlin, lsqnonneg***
f	Вектор коефіцієнтів для лінійних членів в лінійному рівнянні $f'*x$ або квадратичному рівнянні $x'*H*x+f'*x$.	***linprog, quadprog***
fun	Позначення функції, яку слід оптимізувати. fun це оператор для M - файлу або функції.	***fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, lsqcurvefit, lsqnonlin***
$goal$	Вектор значень, яких може набувати цільова функція. Вектор має таку ж довжину, як і цільова функція.	***fgoalattain***
H	Матриця коефіцієнтів для квадратичних елементів в квадратному рівнянні $x'*H*x+f'*x$. Матриця H повинна бути симетричною.	***quadprog***
lb, ub	Вектор (або матриця) нижньої і верхньої межі. Аргумент зазвичай має той же розмір, що і x. Однак якщо lb має більше елементів, ніж x, задайте m, щоб тільки перші m елементів x були обмежені знизу; верхні межі в ub можна визначити у той же спосіб. Ви також можете визначити необмежені змінні, використовуючи $-Inf$ (для нижніх меж) або Inf (для верхніх	***fgoalattain, fmincon, fminimax, fseminf, linprog, lsqcurvefit, lsqlin, lsqnonlin, quadprog***

	меж). Наприклад, якщо $lb(i) = -Inf$, змінна $x(i)$ є обмеженою знизу.	
nonlcon	Функція, яка обчислює нелінійні обмеження-нерівності та обмеження-рівності.	***fgoalattain, fmincon, fminimax***
ntheta	Кількість напівнескінченних обмежень.	***fseminf***
options	Структура, яка визначає опції, що використовуються функціями оптимізації.	***В усіх функціях***
seminfcon	Функція, яка обчислює нелінійні обмеження-нерівності та обмеження-рівності, а також напівнескінченні обмеження. *seminfcon* – це назва *M* - файлу або *MEX* - файлу.	***fseminf***
weight	Вектор впливу для контролю досягнення відносної нижньої чи верхньої межі цільовою функцією.	***fgoalattain***
xdata, *ydata*	Вхідні дані *xdata* та шукані вихідні дані *ydata*, які треба підібрати для рівняння.	***lsqcurvefit***
*x*0	Початкова точка (скаляр, вектор або матриця). (Для *fzero* *x*0 може також бути двоелементним вектором, який представляє інтервал, для якого відомо, що він містить нуль).	***В усіх функціях, крім fminbnd***
*x*1, *x*2	Інтервал, в межах якого функція мінімізується.	***fminbnd***

Таблиця 2. Вихідні аргументи оптимізаційних функцій

АРГУМЕНТ	ОПИС	ВИКОРИСТОВУЄТЬСЯ У ФУНКЦІЯХ
attainfactor	Результуючий фактор в рішенні x.	***fgoalattain***
exitflag	Ціле число, яке вказує на причину припинення алгоритму оптимізації. Ви можете використовувати *exitflag* як інструмент програмування при створенні *M* - файлів, які виконують оптимізацію. Ви також можете повернути повідомлення, що міститиме пояснення з приводу того, чому припинена оптимізація, шляхом виклику функції оптимізації з вихідним аргументом *output*, потім відображатиметься *output.message*	***В усіх функціях***
fval	Значення цільової функції *fun* в рішенні x.	***fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, linprog, quadprog***
grad	Значення градієнта функції *fun* в рішенні x. Якщо *fun* не обчислює градієнт, *grad* є скінченне різницевим наближенням градієнта.	***fmincon, fminunc***
hessian	Значення визначника Гессе для *fun* в рішенні x. Для великомасштабних методів, якщо *fun* не обчислює визначник Гессе, *hessian* є	***fmincon, fminunc***

	скінченне різницевим наближенням визначника Гессе. Для середньомасштабних методів *hessian* є значенням квазіньютонівським наближенням до визначника Гессе в рішенні x.	
jacobian	Значення Якобі для функції *fun* в рішенні x. Якщо *fun* не обчислює значення Якобі, *jacobian* є скінченно-різницевим наближенням значення Якобі.	***lsqcurvefit, lsqnonlin, fsolve***
lambda	Це множники Лагранжа в рішенні x. *lambda* це структура, в якій кожне поле призначене для обмежень різного типу. Для *lsqnonneg lambda* є просто вектором, оскільки *lsqnonneg* оперує лише одним видом обмежень.	***fgoalattain, fmincon, fminimax, fseminf, linprog, lsqcurvefit, lsqlin, lsqnonlin, lsqnonneg, quadprog***
maxfval	$\max\{fun(x)\}$ в рішенні xfr.	***fminimax***
output	Це вихідна структура, що містить інформацію про результати оптимізації. Для отримання інформації щодо назви полів див. опис конкретних функцій.	***В усіх функціях.***
residual	Значення остачі в рішенні x.	***lsqcurvefit, lsqlin, lsqnonlin, lsqnonneg***
resnorm	Значення 2-norm, піднесеного до квадрату, для залишку в рішенні x.	***lsqcurvefit, lsqlin, lsqnonlin, lsqnonneg***
x	Рішення, знайдене функцією оптимізації. Якщо $exitflag > 0$, то x є рішенням; якщо ні (інакше), x є значенням в процедурі оптимізації, припиненій завчасно.	***В усіх функціях.***

Параметри оптимізації (Optimization Options)

Наступна таблиця описує поля в структурі *options*. Ви можете встановити значення цих полів, використовуючи функцію *optimset*. Стовпчик, названий L, M, B, відображає належність параметрів до великомасштабних, середньомасштабних методів чи до обох: L – тільки великомасштабні методи, M – тільки середньомасштабні методи, B – обидва – велико- і середньомасштабні методи.

Значення опцій за замовчуванням для параметрів відрізняються залежно від того, яка функція оптимізації Ви викликаєте як вхідний аргумент за допомогою *option*. Ви можете встановити значення за замовчуванням для будь-якої функції оптимізації шляхом введення *optimset* разом із назвою функції. Наприклад, *optimset fmincon* - повертає список опцій та значень за замовчуванням для *fmincon*.

Таблиця 3. Параметри оптимізації

НАЗВА ОПЦІЇ	ОПИС	L, M, B	ВИКОРИСТОВУЄТЬСЯ У ФУНКЦІЯХ
BranchStrategy	Стратегія *bintprog* використовує для вибору гілок змінних.	M	***bintprog***
DerivativeCheck	Порівнює запропоновані користувачу аналітичні похідні (градієнти або значення Якобі) для скінченних диференційних похідних.	B	***fgoalattain, fmincon, fminimax, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin***
Diagnostics	Надає діагностичні дані стосовно функції, яку треба мінімізувати або вирішити.	B	***Усі, крім fminbnd, fminsearch, fzero, lsqnonneg***
DiffMaxChange	Максимальна зміна в змінних для скінченного диференціювання.	M	***fgoalattain, fmincon, fminimax, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin***
DiffMinChange	Мінімальна зміна в змінних для скінченного диференціювання.	M	***fgoalattain, fmincon, fminimax, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin***
Display	Рівень відображення. *"off'* означає «немає виходу (вихідних даних)»; *'iter'* показує вихід на кожній ітерації; *'final'* показує лише кінцевий вихід (результат); *'notify'* показує вихід лише, якщо функція не збігається (незбіжна).	B	***В усіх. Див. опис конкретних функцій стосовно значень, які застосовуються.***
FunValCheck	Перевіряє, чи допустимі значення цільової функції та обмежень. *'on'* вказує на попередження, коли цільова функція чи обмеження повертають значення, що є *complex, NaN,* або *Inf*. Зверніть увагу: *FunValCheck* не повертає попередження для *Inf.*, використовуючи функції f*minbnd,*	B	***fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, lsqcurvefit, lsqnonlin***

	fminsearch, або *fzero*, які розраховують *Inf*. Наближено; '*off*' вказує на відсутність попередження.		
GoalsExactAchieve	Задає кількість задач, які потребує цільова функція, щоб зрівнятися з кінцевим результатом *goal*. Задачі повинні бути виділені в кілька перших елементів *F*.	M	***fgoalattain***
GradConstr	Визначені користувачем градієнти для нелінійних обмежень.	M	***fgoalattain, fmincon, fminimax***
GradObj	Визначені користувачем градієнти для цільових функцій.	B	***fgoalattain, fmincon, fminimax, fminunc, fseminf***
Hessian	Якщо '*on*', то функція використовує визначені користувачем визначники Гессе або інформацію про визначники Гессе (при використанні *HessMult*) для цільової функції. Якщо '*off* ', функція наближає визначник Гессе, використовуючи скінченну різницю.	L	***fmincon, fminunc***
HessMult	Визначена користувачем багатоцільова функція Гессе.	L	***fmincon, fminunc, quadprog***
HessPattern	Визначає закономірність Гессе для кінцевої різниці. Розмір матриці становить $n \times n$, де n - це кількість елементів $x0$ (початкової точки).	L	***fmincon, fminunc***
HessUpdate	Актуалізація за квазіньютонівською схемою.	M	***fminunc***
InitialHessMatrix	Початкова квазіньютонівська матриця.	M	***fminunc***
InitialHessType	Тип початкової квазіньютонівської матриці.	M	***fminunc***
Jacobian	Якщо '*on*', функція використовує визначені користувачем значення Якобі або інформацію про значення Якобі (при використанні *JacobMult*) для цільової функції. Якщо '*off* ', функція наближається до значення Якобі, використовуючи кінцеву різницю.	B	***fsolve, lsqcurvefit, lsqnonlin***
JacobMult	Визначена користувачем багатоцільова функція Якобі.	L	***fsolve, lsqcurvefit, lsqlin, lsqnonlin***
JacobPattern	Визначає закономірність Якобі	L	***fsolve, lsqcurvefit, lsqnonlin***

	для кінцевої різниці. Розмір матриці становить $m \times n$, де m - це кількість значень першого аргументу, який видається визначеною користувачем функцією *fun*, а n - це кількість елементів в $x0$ (початковій точці).		
LargeScale	Використовує великомасштабний алгоритм, якщо це можливо.	B	***fmincon, fminunc, fsolve, linprog, lsqcurvefit, lsqlin, lsqnonlin, quadprog***
LevenbergMarquardt	Обирає алгоритм Левенберга-Маркардта у методі Гауса-Ньютона. *'on'* визначає алгоритм Левенберга-Маркардта. *'off'* визначає алгоритм Гауса-Ньютона.	M	***lsqcurvefit, lsqnonlin***
LineSearchType	Вибір алгоритму лінійного пошуку.	M	***fsolve, lsqcurvefit, lsqnonlin***
MaxFunEvals	Максимальна кількість оцінок функції.	B	***fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin***
MaxIter	Максимальна кількість ітерацій.	B	***В усіх крім fzero та lsqnonneg***
MaxNodes	Максимальна кількість можливих рішень або точок з'єднання, які шукає функція двоїстого цілочисельного програмування *bintprog*.	M	***bintprog***
MaxPCGIter	Максимальна кількість ітерацій у методі попередніх спряжених градієнтів.	L	***fmincon, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog***
MaxRLPIter	Максимальна кількість ітерацій в методі лінійної програмної релаксації.	M	***bintprog***
MaxSQPIter	Максимальна кількість ітерацій в методі послідовного квадратичного програмування.	M	***fmincon***
MaxTime	Максимальний час в секундах, відведений на виконання алгоритму.	M	***bintprog***
MeritFunction	Використовує цільову установку/ *minimax* функції корисності (багатоцільової) – *fmincon* (одна ціль).	M	***fgoalattain, fminimax***
MinAbsMax	Кількість $F(x)$, які треба мінімізувати за найгірших часткових абсолютних значень.	M	***fminimax***

NodeDisplayInterval	Точка, що відображає інтервал для *bintprog*.	M	***bintprog***
NodeSearchStrategy	Шукає стратегію, яку використовує *bintprog*.	M	***bintprog***
NonlEqnAlgorithm	Визначає один з алгоритмів для рішення нелінійних рівнянь: *'dogleg'* – алгоритм *Trust-region dogleg* (за замовчуванням) *'lm'* – Левенберга-Маркадта, '*gn'* – Гауса-Ньютона	M	***fsolve***
OutputFcn	Визначає задану користувачем функцію, яку викликає функція оптимізації при кожній ітерації.	B	***fgoalattain, fmincon, fminimax, fminunc, fseminf, fsolvelsqcurvefit, lsqnonlin***
PrecondBandWidth	Вищий рівень пропускної здатності посилання для PCG (*Precondition Gradient*).	L	***fmincon, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog***
RelLineSrchBnd	Відносна границя довжини кроку лінійного пошуку.	M	***fgoalattain, fmincon, fminimax, fseminf***
RelLineSrchBndDuration	Кількість ітерацій, для яких границя визначена в *RelLineSrchBnd* повинна бути активною.	M	***fgoalattain, fmincon, fminimax, fseminf***
Simplex	Якщо *'on'*, функція використовує алгоритм симплекс.	M	***linprog***
TolCon	Лінія стійкості в порушенні обмеження.	B	***bintprog, fgoalattain, fmincon, fminimax, fseminf***
TolFun	Лінія стійкості в значеннях функції.	B	***bintprog, fgoalattain, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, linprog (LS only), lsqcurvefit, lsqlin (LS only), lsqnonlin, quadprog (LS only)***
TolPCG	Лінія стійкості в PCG-ітераціях.	L	***fmincon, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog***
TolRLPFun	Лінія стійкості значення функції в проблемі лінійної релаксації.	M	***bintprog***
TolX	Лінія стійкості для x.	B	***Всі функції, крім середньомасштабного алгоритму для linprog, lsqlin, and quadprog***
TolXInteger	Стійкість, в межах якої *bintprog* приймає значення змінних як ціле.	M	***bintprog***
TypicalX	Типові значення x. Довжина вектора дорівнює кількості елементів в $x0$.	B	***fgoalattain, fmincon, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog***

Опис ідентифікаторів функцій (Function-Specific Output Headings)

Цей розділ описує заголовки , що повторюються, значення яких є визначеними для функції оптимізації, які ми використовуємо.

**** bintprog;***
**** fminsearch;***
**** fzero and fminbnd;***
**** fminunc;***
****fsolve;***
**** fgoalattain, fmincon, fminimax, fseminf;***
**** linprog;***
**** lsqnonlin and lsqcurvefit.***

BINTPROG

Таблиця 4. Опис заголовків виводу для функції bintprog

Заголовки функцій	Значення
Explored nodes (точки, що досліджуються)	Сукупність значень, точок перетину, що досліджуються.
Obj of LP relaxation (функція мети для ЛП)	Зменшення величини функції мети лінійного програмування.
Obj of best integer point (найкраще цілочисельне значення)	Найкраще цілочисельне набуте значення функції мети. Це найвищий (найбільше) стрибок (значення) кінцевої величини функції мети.
Unexplored nodes (точки, що не досліджувались)	Число точок перетину, що були знайдені, але не досліджувались.
Best lower bound on obj (найнижча границя)	Величина функції мети лінійного програмування, яка зменшується, що дай найменше значення нижньої межі кінцевого значення цільової функції.
Relative gap between bounds (розмір відносного розриву між границями)	$\frac{100(b-a)}{\lvert b\rvert+1}$, де b – найкраще цілочисельне значення цільової функції, a – нижня границя значення цільової функції.

FMINSEARCH

Таблиця 5. Вихідні значення для функції fminsearch

Заголовки функції	Значення
min f(x)	Найменше значення функції в поточному симплексі.
Procedure	Процедура симплексу при поточній ітерації. Процедура включає initial, expand, reflect, shrink, contract inside, and contract outside (заголовок, розширення, відображення, стиснення, скорочення внутрішнє та скорочення зовнішнє).

FZERO, FMINBND

Таблиця 7. Опис вихідних даних функцій fzero та fminbnd

Заголовки функції	Значення
Procedure	Дії для ***fzero*:** • *initial (initial point)* – заголовок; • *search (search for an interval containing a zero)* – пошук інтервалу, що містить нуль; • *bisection (bisection search)* – пошук поділу навпіл; • *interpolation* – інтерполяція. Дії для ***fminbnd***: • *initial* – заголовок; • *golden (golden section search)* – пошук "золотої середини"; • *parabolic (parabolic interpolation)* – параболічна інтерполяція.

FSOLVE

Таблиця 8. Опис вихідних даних функції fsolve

Заголовки функції	Значення
First-order optimality (оптимізація першого рівня)	Поточне порушення умов оптимальності.
Trust-region radius (інтервал довіри)	Поточний довірчий інтервал.
Residual (залишок)	Залишок (остача) (сума квадратів) функції.
Directional derivative (похідна за напрямом)	Градієнт функції за похідною за напрямом.

FGOALATTAIN, FMINCON, FMINIMAX, FSEMINF

Таблиця 9. Опис вихідних даних функцій fmincon та fminimax

Заголовки функції	Значення
Max constraint (максимальне обмеження)	Порушення максимального обмеження.
Max{F,constraints} (Максимум (значення функції;обмеження)	Максимальне значення функції або порушення обмеження для *fminimax*.
First-order optimality (оптимізація першого рівня)	Поточне порушення умов оптимальності.
CG-iterations	Число спряжених ітерацій градієнта взятих з поточної (оптимізації) ітераці.ї
Trust-region radius (довірчі інтервали)	Поточний довірчий інтервал.
Residual (остача)	Залишок (сума квадратів) функції.
Attainment factor (досягнутий фактор)	Поточне значення γ в задачі досягнутої проблеми.
Directional derivative (похідні за напрямом)	Градієнт функції за похідною за напрямом.
Procedure (дії)	Hessian оновленні та QP підпроблеми.

FMINUNC

Таблиця 10. Опис вихідних даних функції fminunc

Заголовки функції	Значення
First-order optimality (оптимізація першого рівня)	Поточний градієнт рівний нескінченність.
CG-iterations	Число спряжених ітерацій градієнта взятих з поточної (оптимізації) ітерації.

LINPROG

Таблиця 11. Опис вихідних даних функції linprog

Заголовки функції	Значення
*Primal Infeas A*x-b*	Основне невиконання.
*Dual Infeas A'*y+z-w-f*	Подвійне невиконання.
*Duality Gap x'*z+s'*w*	Подвійний розрив (дивись Large-Scale Linear Programming між основною ціллю та двоїстою. S та w з'являються в рівнянні, де обмежений верхній стрибок.
Total Rel Error	Сумарна відносна помилка, описується в кінці підтеми Main Algorithm (Large-Scale Linear Programming).
*Objective f'*x*	Поточне цільове значення.

LSQNONLIN, LSQCURVEFIT

Таблиця 12. Опис вихідних даних функції lsqnonlin, lsqcurvefit

Заголовки функції	Значення
Residual (залишок)	Залишок (остача) (сума квадратів) функції.
First-order optimality(оптимізація першого рівня)	Поточне порушення умов оптимальності.
CG-iterations	Число спряжених ітерацій градієнта взятих з поточної (оптимізації) ітерації.
Directional derivative (похідні за напрямом)	Градієнт функції за похідною за напрямом.
Lambda (Лямбда)	λ_k величина визначається в Least-Squares Optimization. (Оптимізація найменших квадратів). Це значення використовується, коли користуються методом Levenberg-Marquardt та виключається, коли користуємось методо Гауса-Ньотона.

Функція Fminbnd

Функція fminbnd знаходить мінімум функції однієї змінної в межах заданого інтервалу.

Рівняння

$$\min_{X} f(X) \text{ при чому } X_1 < X < X_2,$$

де X, X_1 та X_2 скалярні величини, а $f(x)$ – це функція, яка повертає скаляр.

Синтаксис

- *x = fminbnd(fun, x1,x2)*
- *x = fminbnd(fun, x1,x2,options)*
- *[x,fval] = fminbnd(...)*
- *[x,fval,exitflag] = fminbnd(...)*
- *[x,fval,exitflag,output] = fminbnd(...)*

Опис

- *x = fminbnd(fun, x1,x2)* повертає значення X, який є локальним мінімумом функції, яка описується у *fun* на інтервалі $X_1 < X < X_2$. *fun* – ідентифікатор функції.
- *x = fminbnd(fun, x1,x2,options)* мінімізує функцію з опціями, які задані у структурі *options*. Ви можете визначити ці параметри, використовуючи функцію *optimset. fminbnd*.
- *[x, fval] = fminbnd(...)* повертає *fval*, тобто значення цільової функції.
- *[x, fval, exitflag] = fminbnd(...)* повертає *exitflag*, який описує вихідну умову функції *fminbnd:*
  - 1 – *fminbnd* зводиться до знаходження X за допомогою опції *options TolX*,
  - 0 – було досягнуто максимального числа оцінок функції або ітерацій,
  - -1 – алгоритм закінчився кінцевою функцією,
  - -2 – границі непослідовні ($X_1 > X_2$).
- *[x, fval, exitflag, output] = fminbnd(...)* повертає структуру *output*, яка містить інформацію про оптимізацію: *output.algorithm* – використовується алгоритм, *output.funcCount* – кількість оцінок функції, *output.iterations* – кількість ітерацій, *output.message* – вихідне повідомлення.

Аргументи

fun це функція, яка мінімізується. *fun* бере значення скалярного X і видає скалярний f, функція мети оцінюється в X. Функція *fun* може бути вказана як функція управління для функції файлу MathLab.

x = fminbnd(@myfun,x1,x2), де *myfun.m* – функція файлу MathLab такі як: *function f = myfun(x); f = ... % Compute function value at x,* або як ідентифікатор для будь-якої функції: *x = fminbnd(@(x) sin(x*x),x1,x2);*

Алгоритм

Алгоритм заснований на пошуку правила золотого перетну і параболічній інтерполяції. Якщо тільки ліва кінцева точка X_1 була дуже близька до правої кінцевої точки X_2, *fminbnd* ніколи не оцінює *fun* в кінцевих точках, тому *fun* потрібно буде визначити тільки для X в інтервалі $X_1 < X < X_2$.

Якщо мінімум фактично знаходиться в X_1 або X_2, *fminbnd* повертає внутрішню точку на відстань не більше, ніж $2 \cdot TolX$ від X_1 або X_2, де *TolX* – це закінчення допустимих значень X. Більше конкретної інформації про алгоритм можна знайти у [1] або [2].

Обмеження алгоритму

Функція, яку потрібно мінімізувати, повинна бути безперервна. *fminbnd* може знайти тільки місцеві рішення. *fminbnd* часто показує повільну збіжність, коли рішення знаходяться на границях інтервалу. *fminbnd* працює тільки з дійсними змінними.

Приклад

Мінімум функції синуса знаходиться наступним чином:

```
>> x = fminbnd(@sin,-pi/2,2*pi)

x =

    4.7124
```

Мінімум функції дорівнює:

```
>> y = sin(x)

y =

    -1
```

Для того, щоб знайти мінімум функції $f(x)=(x-3)^2-1$ на інтервалі (0, 5), потрібно записати файл MathLab:

```
function f = myfun(x)
f = (x-3)^2 - 1;
```

Після звернення до шаблону оптимізації отримуємо рішення.

```
>> x = fminbnd(@myfun,0,5)

x =

    3
```

Якщо *fun* містить параметри, ви можете використовувати будь-які функції. Припустимо, що ви хочете мінімізувати цільову функцію *myfun*, визначену наступною функцією файлу MathLab.

```
function f = myfun(x,a)
f = (x - a)^2;
```

Відзначте, що *myfun* має додатковий параметр, тому ви не можете передавати її безпосередньо до *fminbind*. Щоб оптимізувати її для специфічного значення, наприклад a = 1.5, потрібно: Призначте значення: *a = 1.5; % define parameter first*. Далі зверніться до *fminbnd* з одноаргументною функцією, і потім зверніться до *myfun* з двома аргументами:

```
>> a = 1.5;
>> x = fminbnd(@(x) myfun(x,a),0,1)

x =

    0.9999
```

Посилання

[1] Forsythe, G. E., M. A. Malcolm, and C. B. Moler, Computer Methods for Mathematical Computations, Prentice-Hall, 1976.

[2] Brent, Richard. P., Algorithms for Minimization without Derivatives, Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

Функція Lsqnonlin

Функція Lsqnonlin функція, що вирішує задачі нелінійної оптимізації методом найменших квадратів.

Синтаксис

- *x = lsqnonlin(fun,x0)*
- *x = lsqnonlin(fun,x0,lb,ub)*
- *x = lsqnonlin(fun,x0,lb,ub,options)*
- *[x,resnorm] = lsqnonlin(...)*
- *[x,resnorm,residual] = lsqnonlin(...)*
- *[x,resnorm,residual,exitflag] = lsqnonlin(...)*
- *[x,resnorm,residual,exitflag,output] = lsqnonlin(...)*
- *[x,resnorm,residual,exitflag,output,lambda] = lsqnonlin(...)*
- *[x,resnorm,residual,exitflag,output,lambda,jacobian] = lsqnonlin(...)*

Замість того, щоб обчислювати значення *f(x)* (сума квадратів), *lsqnonlin* вимагає, щоб призначена для користувача функція обчислила оцінену вектором функцію.

$$F(x) = \begin{bmatrix} f1(x) \\ f2(x) \\ f3(x) \end{bmatrix}$$

Тоді, у векторних компонентах, ви можете знов описати дану задачу оптимізації як:

$$\min \frac{1}{2}\|F(x)\|\frac{2}{2} = \frac{1}{2}\sum fi(x)^2,$$

де *x* - вектор, і *F(x)* - функція, яка повертає векторне значення.

Опис

- *x = lsqnonlin (fun,x0)* починається в точці x_0 і знаходить мінімум суми квадратів функцій, описаних на *fun*. *fun* повинна повернути вектор значень а не суми квадратів значень.
- *x = lsqnonlin (fun,x0,lb,ub)* визначає набір нижніх і верхніх меж на змінних задачі в *x*, таким чином, щоб рішення знаходилося завжди в діапазоні <= *x* <= *ub*.
- *x = lsqnonlin (fun,x0,lb,ub,options)* мінімізує з варіантами оптимізації, вказаними у варіантах структури. Використовуйте *optimset*, щоб встановити ці варіанти. Прохід порожніх матриці для *lb* і *ub*, якщо не існує обмежень.
- *[x, resnorm] = lsqnonlin (...)* повертає значення наданої квадратної форми 2-норми залишку в x: *sum(fun(x).^2)*
- *[x, resnorm, residual] = lsqnonlin (...)* повертає значення із залишкової *fun(x)* в рішення *x*.
- *[x, resnorm, residual, exitflag] = lsqnonlin (...)* повертає значення *exitflag*, яке описує умову виходу.
- *[x,resnorm,residual,exitflag,output] = lsqnonlin (...)* повертає висновок структури, який містить інформацію про оптимізацію.
- *[x,resnorm,residual,exitflag,output,lambda] = lsqnonlin(...)* повертає лямбду структури, поля якої містять множники Лагранжа при рішенні *x*.
- *[x,resnorm,residual,exitflag,output,lambda,jacobian] = lsqnonlin (...)* повертає матрицю Якобі (Jacobian) з *fun* при рішенні *x*.

Вхідні аргументи

Аргументи функції містять загальні описи аргументів, повернених до *lsqnonlin*. Цей розділ забезпечує функціонально-спеціалізовані деталі функції для *fun* і варіантів:

Fun - функція, сума квадратів якої мінімізована. *fun* - функція, яка приймає вектор x і повертає вектор *F*, об'єктивні функції, оцінені в *x*. Функціональна *fun* може бути визначений як функціональний ідентифікатор для функції *Mfile*.

x = lsqnonlin (@myfun,x0), де *myfun* - функція MATLAB типу функції *F = myfun (x)*

F =... % Обчисліть значення функції в x.

fun може також бути функціональним маркером для невідомої функції.

*x = lsqnonlin ((x) sin (x.*x), x0);*

Якщо *Jacobian* може бути також обчислений, і *Jacobian ' on'* встановлена як:

options = optimset('Jacobian','on'), тоді функція *fun* повинна повернутися, в другому параметрі висновку, значення *J*, матриця, в *x*. Зверніть увагу, що, перевіряючи значення *nargout* функція може уникнути обчислювати *J*, коли *fun* називають тільки з одним параметром виводу (у випадку, де алгоритм оптимізації потребує значення *F*, але не *J*).

функція [F,J] = myfun(x)

F = ... % Об'єктивні значення функції в x

if nargout > 1 % Два вихідні аргументи

J = ... % Jacobian функція оцінена в x

end

Якщо *fun* повертає вектор (матрицю) компонентів *m* і *x* має довжину *n*, де *n* - довжина x_0, потім *Jacobian J* є матрицею $m \times n$, де *J(i,j)* - частинна похідна *F(i)* по *x(j)*. (Зверніть увагу, що *Jacobian J* переміщує градієнт *F*.) *Options* – опції, що забезпечують функціонально-спеціалізовані деталі для значень варіантів.

Вихідні аргументи

Аргументи функції містять загальні описи параметрів, повернених *lsqnonlin*. Цей розділ забезпечує функціонально-спеціалізовані деталі для *exitflag*, лямбди і виводу:

Exitflag - ціле число, що ідентифікує причину закінчення алгоритму. Процедура складає список значень *exitflag* у відповідності аналізує закінчений алгоритм

1 Функція зводилася до рішення x.

2 Зміна в x була менше ніж вказаний допуск.

3 Зміна в залишку була менше ніж вказаний допуск.

4 Величина пошукового напряму була менше ніж вказаний допуск.

0 Число повторень перевищило варіанти. *MaxIter* або число оцінок функції перевищило варіанти. *FunEvals.*

-1 Алгоритм був закінчений вихідною функцією.

-2 Проблема не може бути вирішена: lb меж і ub несумісні.

-4 Пошук рядка не міг достатньо зменшувати залишок уздовж поточного пошукового напряму.

Lambda - структура, що містить множники Лагранжа при рішенні x.

Output - структура, що містить інформацію про оптимізацію, поля структури - ітерації.

Iterations - номер узятих ітерацій (число повторень).

funcCount - число алгоритму оцінок функції.

algorithm - алгоритм, що використовується.

cgiterations - номер PCG ітерацій (число повторень) (великомасштабний алгоритм).

stepsize - заключний розмір кроку (алгоритм середнього масштабу).

firstorderopt - міра *optimality* першого рівня (великомасштабний алгоритм тільки).

Приклад

Мінімізувати вираз:

$$\sum_{k=1}^{10}(2+2k-e^{kx1}-e^{kx2})^2$$, початок в точці x = [0.3, 0.4].

Оскільки *lsqnonlin* припускає, що сума квадратів явно не сформована в призначеній для користувача функції, функція передана до *lsqnonlin*, повинна натомість обчислити оцінену функцію вектора.

$F_k(x) = 2+2k-e^{kx1}-e^{kx2}$, *k=1, F* повинен містити компоненти *k*.

Спочатку, запишіть М-файл, щоб обчислити вектор к-компонента F.

```
function F = myfun(x)
k = 1:10;
F = 2 + 2*k-exp(k*x(1))-exp(k*x(2));
```

Потім, викличте шаблон оптимізації. Приблизно після 24 функціональних оцінок, цей приклад дає рішення *x* та *resnorm* (сума квадратів).

```
>> x0 = [0.3 0.4]                          % Starting guess
[x,resnorm] = lsqnonlin(@myfun,x0)     % Invoke optimizer

x0 =

    0.3000    0.4000

Optimization terminated: norm of the current step is less
 than OPTIONS.TolX.

x =

    0.2578    0.2578


resnorm =

  124.3622
```

Обмеження

Функція, яка мінімізована повинна бути безперервною. *lsqnonlin* може дати тільки місцеві рішення. *lsqnonlin* тільки обробляє реальні змінні. Коли *x* має комплексні змінні, змінні повинні бути розбиті на реальні і уявні частини.

Великомасштабний метод для *lsqnonlin* не вирішує *underdetermined* системи; це вимагає, щоб номер рівнянь (тобто, номер елементів *F*) був не менше номера змінних. В *underdetermined* випадку алгоритм середнього масштабу використовується. Якщо компоненти *x* не мають ніякої верхньої (або нижньої) межі, то *lsqnonlin* вважає за краще, що відповідні компоненти *ub* (або *lb*) встановлені в *inf* (або-*inf* для нижніх меж) як протилежна довільному але дуже великому позитивному (або негативно для нижніх меж) номеру.

Посилання

[1] Coleman, T.F. and Y. Li, "An Interior, Trust Region Approach for Nonlinear Minimization Subject to Bounds," SIAM Journal on Optimization, Vol. 6, pp. 418-445, 1996.

[2] Dennis, J.E., Jr., "Nonlinear Least-Squares," State of the Art in Numerical Analysis, ed. D. Jacobs, Academic Press, pp. 269-312, 1977.

Функція Lsqnonneg

Функція *lsqnonneg* призначена для вирішення задач лінійної оптимізації з невід'ємними обмеженнями методом найменших квадратів.

Синтаксис

- *x = lsqnonneg(C,d)*
- *x = lsqnonneg(C,d,x0)*
- *x = lsqnonneg(C,d,x0,options)*
- *[x,resnorm] = lsqnonneg(...)*
- *[x,resnorm,residual] = lsqnonneg(...)*
- *[x,resnorm,residual,exitflag] = lsqnonneg(...)*
- *[x,resnorm,residual,exitflag,output] = lsqnonneg(...)*
- *[x,resnorm,residual,exitflag,output,lambda] = lsqnonneg(...)*

Опис

- *x = lsqnonneg (c, d)* повертає вектор x, який мінімізує норму *(C*x-d)* при чому x> = 0. c і d повинні бути реальними.
- *x = lsqnonneg (c, d, x0)* використовує x0 як початкову точку якщо вісь x0> = 0; інакше використовується значення за замовчуванням. Задана за замовчуванням точка - початок координат (значення за замовчуванням використовується, коли *x0 == []* або коли забезпечено тільки два вхідні параметри).
- *x = lsqnonneg (c, d, x0, options)* мінімізують з параметрами оптимізації, вказаними у варіантах структури. Ви можете визначити ці параметри, використовуючи функцію optimset. *lsqnonneg* використовує поля структури варіантів:
  - ✓ *Display* Рівень відображення (Level of display). 'off' не відображає ніякого виводу; 'final' показує тільки кінцевий вивід; 'notify' (заданий за замовчуванням) показує вивід, тільки якщо функція не зводиться в одну точку.
  - ✓ *TolX* Закінчення припущення щодо x.
- *[x, resnorm] = lsqnonneg (...)* повертає значення заданої двовимірної квадратної матриці залишків : норма *(C*x-d)^2*.
- *[x, resnorm, residual] = lsqnonneg (...)* повертає залишок, *d-C*x*.
- *[x, resnorm, residual, exitflag] = lsqnonneg (...)* повертає значення *exitflag*, що описує умову виходу lsqnonneg:
  - ✓ > 0 Вказує що функція зводиться до рішення x.
  - ✓ 0 Вказує, що повторний рахунок був перевищений. Збільшення допуску (параметр TolX у варіантах) може привести до рішення.
- *[x,resnorm,residual,exitflag,output] = lsqnonneg (...)* повертає вивід структури, який містить інформацію про операцію:
  - ✓ *output.algorithm* використовують алгоритм
  - ✓ *output.iterations* номер узятих ітерацій
- *[x,resnorm,residual,exitflag,output,lambda] = lsqnonneg (...)* повертає подвійний вектор (множники Лагранжа) лямбда, де лямбда (i) <=0, коли x (i) складає (приблизно) 0, і лямбда (i) складає (приблизно) 0 коли x (i)> 0.

Алгоритм

lsqnonneg використовує алгоритм, описаний в [1]. Алгоритм починається з набору можливих векторів і обчислює зв'язану подвійну векторну лямбду. Тоді вибирається вектор, що відповідає максимальному значенню в лямбді. Це продовжується доки лямбда <= 0.

Приклад

Вводимо матриці:

```
>> C = [
    0.0372    0.2869
    0.6861    0.7071
    0.6233    0.6245
    0.6344    0.6170];
>> d = [
    0.8587
    0.1781
    0.0747
    0.8405];
```

Виконуємо операції над вхідними матрицями:

```
>> [C\d lsqnonneg(C,d)]

ans =

   -2.5627         0
    3.1108    0.6929
```

Виконуємо оптимізації за методом найменших квадратів:

```
>> [norm(C*(C\d)-d) norm(C*lsqnonneg(C,d)-d)]

ans =

    0.6674    0.9118
```

Посилання

[1] Lawson, C.L. and R.J. Hanson, Solving Least Squares Problems, Prentice-Hall, 1974, Chapter 23, p. 161.

[2] Gill, P.E., W. Murray, and M.H. Wright, Practical Optimization, London, Academic Press, 1981.

[3] Han, S.P., "A Globally Convergent Method for Nonlinear Programming," Vol. 22, Journal of Optimization Theory and Applications, p. 297, 1977.

[4] Powell, M.J.D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," Numerical Analysis, ed. G.A. Watson, Lecture Notes in Mathematics, Springer Verlag, Vol. 630, 1978.

[5] Powell, M.J.D., "The Convergence of Variable Metric Methods For Nonlinearly Constrained Optimization Calculations," Nonlinear Programming 3 (O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, eds.), Academic Press, 1978.

Функція Linprog

Функція linprog – вирішує задачі лінійного програмування наступної форми:

Рівняння

$$\min_X f^T x \text{ при цьому } A \cdot X \le b,$$
$$Aeq \cdot X = beq,$$
$$lb \le x \le ub$$

де *f, x, b, beq, lb* та *ub* – це вектори, *A* і *Aeq* – матриці.

Синтаксис

- *x = linprog(f,A,b)*
- *x = linprog(f,A,b,Aeq,beq)*
- *x = linprog(f,A,b,Aeq,beq,lb,ub)*
- *x = linprog(f,A,b,Aeq,beq,lb,ub,x0)*
- *x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)*
- *[x,fval] = linprog(...)*
- *[x,lambda,exitflag] = linprog(...)*
- *[x,lambda,exitflag,output] = linprog(...)*
- *[x,fval,exitflag,output,lambda] = linprog(...)*

Опис

- *x = linprog(f)* вирішує задачу лінійного програмування: $\min_X f' \cdot X$
- *x = linprog(f, A, b)* вирішує задачу лінійного програмування: $\min_X f' \cdot X$ при цьому $A \cdot X \le b$.
- *x = linprog(f, A, b, Aeq, beq)* вирішує попередню задачу з додатковими рівняннями обмежень: $Aeq \cdot X = beq$ Встановіть *A*=[] і *b*=[] якщо немає обмежень.
- *x = linprog(f, A, b, Aeq, beq, lb, ub*) визначає набір нижніх і верхніх меж для змінних x, таким чином, що рішення знаходиться завжди у проміжку: $lb \le X \le ub$. Встановіть *Aeq*=[] і *beq*=[] якщо немає обмежень.
- *x* = linprog*(f, A, b,Aeq, beq, x0)* встановлює початкові значення алгоритму в $X0$. Якщо $X0$ знаходиться у нездійсненному діапазоні, то функція *linprog* використовує початкові значення задані по замовченню.
- *x* = linprog*(f, A, b,Aeq, beq, x0, options)* мінімізує функцію з опціями, які задані у структурі *options*.
- *[x, fval]* = linprog*(...)* повертає *fval*, тобто значення цільової функції: $fval = f' \cdot X$
- *[x, lambda, exitflag]* = linprog*(...)* повертає *exitflag*, який описує вихідну умову функції *linprog*.
- *[x, lambda, exitflag, output]* = linprog*(...)* повертає структуру *output*, яка містить інформацію про оптимізацію.
- *[x,fval,exitflag,output,lambda] = linprog(...)* повертає структуру *lambda*, чиї поля містять множники Лагранжа в рішення x.

Вхідні аргументи

Таблиця 13. Опис вхідних параметрів для функції linprog

f	**Вектор, який містить коефіцієнти лінійної цільової функції**
A	Матриця, яка містить коефіцієнти обмежень лінійних нерівностей $A \cdot X \le b$
b	Вектор, що відповідає правій частині нерівності лінійних обмежень
Aeq	Матриця, яка містить коефіцієнти обмежень лінійних рівнянь $Aeq \cdot X = beq$

beq	Вектор, що містить обмеження лінійних рівнянь
$X0$	Стартові значення для алгоритму
options	Структура, яка містить опції для алгоритму

Вихідні аргументи

Таблиця 14. Деталі аргументів exitflag, lambda та output

exitflag	Ціле число, що ідентифікує причину, за якої алгоритм закінчився. Наступні списки значення exitflag і відповідні причини: 1 – Функція збігається до розв'язку X. 0 – Кількість ітерацій перевищила максимальне значення. -2 – Задача не має розв'язків. -3 – Задача не має границь. -4 – Значенням NAN збивається протягом виконання алгоритму. -5 – Як першочергові, так і похідні задачі неможливо виконати. -7 – Напрям пошуку занадто малий. Подальший розв'язок неможливий.
lambda	Структура, яка містить інформацію про множники Лагранжа. Поля структури: lower lb – нижня межа upper ub – верхня межа ineqlin – лінійні нерівності eqlin – лінійні рівності
output	Структура, яка містить інформацію про оптимізацію. Поля структури: Algorithm – алгоритм, який був використаний, Cgiterations – кількість сполучених ітерацій градієнта (тільки для Large-Scale алгоритмів). Iterations – кількість ітерацій, Message – причина закінчення алгоритму

Опції

Деякі опції використовуються для всіх алгоритмів, інші доречні тільки при використанні Large-Scale алгоритмів. Ви можете використовувати функцію *optimset*, щоб встановити або змінити значення полів в структурі опцій.

Large-Scale і Medium-Scale

Ці опції використовують як для Medium-Scale, так і для Large-Scale алгоритмів:

- *Diagnostics* – Показує діагностичну інформацію про функцію, яка мінімізується.
- *Display* – якщо стоїть «off», результати не показуються на екрані; при «iter» результати показуються після кожної ітерації (працює тільки з Large-Scale алгоритмами); «final» (встановлено за замовченням) показує тільки кінцеві результати.
- *MaxIter* – Максимальна кількість дозволених ітерацій.
- *Simplex* – Використовують тільки для Medium-Scale алгоритмів. Якщо «on», то *linprog* використовує симплекс-алгоритм. Симплекс-алгоритм використовує вбудовану початкову точку, нехтуючи значенням X0. За замовченням стоїть «off».
- *TolFun* - Використовують тільки для Large-Scale алгоритмів. Допускає закінчення Толерантність закінчення на function значенні.

Алгоритм

Large-Scale алгоритм. Large-Scale метод заснований на LIPSOL (Linear Interior Point Solver), який походить від алгоритму predictor-corrector Mehrotra [2], метод одночасного вирішення прямої та двоїстої задачі у внутрішній точці. Кількість попередніх кроків виводиться до початку ітерації.

Оптимізація Medium-Scale. Linprog використовує такий же метод проекції, що і в алгоритмі *quadprog*. *linprog* є методом активного ряду і тому є варіантом відомого симплекс-методу для лінійного програмування [1]. Алгоритм знаходить початкове допустиме вирішуючи іншу задачу лінійного програмування. Альтернативно, ви можете використовувати симплекс-алгоритм описаний в *Simplex Algorithm*, ввіши опцію: *options = optimset('LargeScale', 'off', 'Simplex', 'on').*

Приклад

Мінімізувати функцію:

$$F = 8X_1 - 5X_2 - 10X_3$$

Обмеження:

$$2X_1 - 3X_2 + X_3 \le 45$$
$$4X_1 + 5X_2 + 4X_3 \le 51$$
$$2X_2 \le 25$$
$$0 \le X_1, 0 \le X_2, 0 \le X_3$$

1. Спочатку необхідно задати цільову функцію:

```
>> f = [8; -5; -10]

f =

     8
    -5
   -10
```

2. Потім ввести обмеження:

```
>> A =  [2 -3  1
4  5  4
0  2  0];
>> b = [45; 51; 25];
>> lb = zeros(3,1);
```

3. Потім зверніться до шаблону лінійного програмування:

```
>> [x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb)
Optimization terminated.

x =

    0.0000
    0.0000
   12.7500


fval =

 -127.5000


exitflag =

     1
```

```
output =

      iterations: 5
       algorithm: 'large-scale: interior point'
    cgiterations: 0
         message: 'Optimization terminated.'


lambda =

    ineqlin: [3x1 double]
      eqlin: [0x1 double]
      upper: [3x1 double]
      lower: [3x1 double]
```

4. Вводячи *lambda.ineqlin, і lambda.lower* отримуємо:

```
>> lambda.ineqlin

ans =

    0.0000
    2.5000
    0.0000

>> lambda.lower

ans =

   18.0000
    7.5000
    0.0000
```

Посилання

[1] Dantzig, G.B., A. Orden, and P. Wolfe, "Generalized Simplex Method for Minimizing a Linear from Under Linear Inequality Constraints," Pacific Journal Math., Vol. 5, pp. 183-195.

[2] Mehrotra, S., "On the Implementation of a Primal-Dual Interior Point Method," SIAM Journal on Optimization, Vol. 2, pp. 575-601, 1992.

[3] Zhang, Y., "Solving Large-Scale Linear Programs by Interior-Point Methods Under the MATLAB Environment," Technical Report TR96-01, Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD, July 1995.

Функція Fseminf

Функція *fseminf* призначена для пошуку мінімального значення багатоваріантної нелінійної функції обмеженої зліва або обмеженої справа, тобто знаходження мінімального значення частково обмеженої нелінійної функції багатьох змінних.

Рівняння

$$\min_x \quad f(x) \qquad subject\ \ to \qquad c(x) \le 0$$

$$ceq(x) = 0$$

$$A \cdot x \le b$$

$$Aeq \cdot x = beq$$

$$lb \le x \le ub$$

$$K_1(x, w_1) \le 0$$

$$K_2(x, w_2) \le 0$$

$$...$$

$$K_n(x, w_n) \le 0$$

де *x, b, beq, lb, та ub* – вектори, *A та Aeq* – матриці, *c(x), ceq(x) та* $K_i(x, w_i)$ – функції, що перетворюють вектор, а *f(x)* – функція, що перетворює скалярні величини. *f(x), c(x), ceq(x)* можуть бути нелінійними функціями.

Вектори (чи матриці) $K_i(x, w_i) \le 0$ є неперервними функціями як *x* так і додаткового набору змінних $w_1, w_2, w_3 \ldots$. Змінні – це вектори з найбільшою довжиною рівною двом.

Синтаксис

- *x= fseminf(fun,x0,ntheta,seminfcon)*
- *x = fseminf(fun,x0,ntheta,seminfcon,A,b)*
- *x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq)*
- *x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq,lb,ub)*
- *x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq,lb,ub,options)*
- *[x,fval] = fseminf(...)*
- *[x,fval,exitflag] = fseminf(...)*
- *[x,fval,exitflag,output] = fseminf(...)*
- *[x,fval,exitflag,output,lambda] = fseminf(...)*

Опис

Fseminf знаходить мінімум скалярної обмеженої з обох боків функції багатьох змінних, починаючи з початкової оцінки. Мета – мінімізувати функцію *f(x)* таким чином, щоб обмеження збереглися для всіх можливих значень. Тому що неможливо обрахувати всі можливі значення, повинна бути обрана область, для того, щоб обраховувати відповідно обраний набір значень.

x = fseminf(fun,x0,ntheta,seminfcon) – починається в x_0 та знаходить мінімальне значення функції обмеженої з обох сторін *fun* обмеженої обмеженням ntheta, що визначені в *seminfcon*.

x = fseminf(fun,x0,ntheta,seminfcon,A,b) – також намагається задовольнити нелінійні нерівності *A*x <= b*.

x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq) – мінімізує предмет лінійних рівнянь *Aeq*x = beq*. Встановлює *A=[] and b=[]*, якщо ріняння-обмеження відсутні.

x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq,lb,ub) – визначає нижні та верхні границі для змінних в *x*, так щоб рішення завжди знаходилось в діапазоні *lb <= x <= ub*.

x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq,lb,ub,options) – мінімізує з опціями мінімізації, визначеними в опціях структури. Використовується optimset, щоб встановити ці опції.

[x,fval] = fseminf(...) – повертає значення цільової функції для *x*.

[x,fval,exitflag] = fseminf(...) – повертає значення *exitflag*, що описує умови виходу.

[x,fval,exitflag,output] = fseminf(...) – повертає структурний вихід, що містить інформацію про оптимізацію

[x,fval,exitflag,output,lambda] = fseminf(...) – повертає значення лямбда, області яких утворюють множники Лагранжа в розв'юзку *x*.

Вхідні дані

Параметри функції мають загальний опис параметрів, що присутні в *fseminf***.** Цей розділ описує функціонально-специфічні деталі для *fun, ntheta, options та seminfcon:*

Fun **–** функція, що буде мінімізована, *fun* **–** це функція, що включає вектор *x* і повертає в скаляр *f*, цільова функція оцінюється в *x*. функція *fun* може бути визначена як назва функціїдля трактування *M-file* функції.

x = fseminf(@myfun,x0,ntheta,seminfcon), де myfun – це функція MATLAB, наприклад,

function f = myfun(x)

f = . % Compute function value at x – розраховує значення функції в x,

*fun***,** також може бути назвою функції, що визначає неявно задану функцію *fun = @(x)sin(x''*x);*

якщо градієнт fun може бути обрахований, і опція *GradObj* – "on" то маємо наступне

options = optimset('GradObj','on'),

тоді функція fun повинна бути повернена в другому параметрі виводу, значення градієнта *g*, а – вектор при *x*. Відмітимо, що перевіряючи значення функції в *nargout* функція може не розрахувати *g*, коли *fun* має лише один параметр виводу (у випадку, коли алгоритм оптимізації потребує лише однієї величини *f*, але не *g*).

function [f,g] = myfun(x)

f = ... % Compute the function value at x – розраховує значення функції в x,

if nargout > 1 % fun called with 2 output arguments – fun має два параметри виводу

g = ... %Compute the gradient evaluated at x – розраховує значення градієнта оцінене в x

end

Градієнт – часткові похідні функції в точці *x*, таким чином і-тий компонент, *g* – частинна похідна функції відносно і-компонента *x*.

ntheta **–** число обмежень з обох боків.

Options **–** забезпечують функціонально-специфічні деталі для значень опцій.

Seminfcon **–** функція, яка розраховує вектор нелінійних нерівностей обмежень *c*, нелінійних рівнянь обмежень *ceq*, а *ntheta* обмеження з обох боків (вектора чи матриці) *K1, K2,..., Kntheta* оцінює по інтервалу *S*, відносно *x*. Функція *seminfcon* може використовуватись як функціональний заголовок.

x = fseminf(@myfun,x0,ntheta,@myinfcon), де *myfun* – це функція MATLAB, наприклад

function [c,ceq,K1,K2,...,Kntheta,S] = myinfcon(x,S) початковий відбір інтервалу

if isnan(S(1,1)),

S = ...% S has ntheta rows and 2 columns (матриця має 2 колонки та ntheta- стовпчиків)

end

w1 = ...% Compute sample set- рахує набір установок

w2 = ...% Compute sample set - рахує набір установок

wntheta = ... % Compute sample set - рахує набір установок

K1 = ... % 1st semi-infinite constraint at x and w – перше обмеження x та w

K2 = ... % 2nd semi-infinite constraint at x and w – друге обмеження x та w

Kntheta = ...% Last semi-infinite constraint at x and w – останнє обмеження x та w

c = ... % Compute nonlinear inequalities at x – розраховує нелінійні нерівності для x

ceq = ... % Compute the nonlinear equalities at x – розраховує рівняння обмеження для x

S – рекомендований інтервал для здійснення вибірки.

Вектори або матриці $K_1, K_2, ..., K_{ntheta}$ мають обмеження з обох боків для оціненого набору значень для незалежних змінних $w_1, w_2, ... w_{ntheta}$, відповідно. Матриця *S*, що містить два стовпчики містить рекомендований інтервал здійснення вибірки для значень $w_1, w_2, ... w_{ntheta}$, який використовується для оцінки $K_1, K_2, ..., K_{ntheta}$. Таким чином, і-ий рядок матриці *S* містить рекомендований інтервал вибірки для оцінки K_i.. Коли K_i – вектор, використовується *S(i,1)* (другий рядок всі компоненти якого нулі). Коли K_i – матриця, для вибірки строчок в K_i використовується *S(i,2), S(i,1)* використовується для вибору стовпців для K_i. На перші ітерації *S* рівний *NaN*, для того, щоб деякий початковий інтервал для здійснення вибірки був визначений *seminfcon*.

Параметри виводу

Параметри функції включають загальний опис параметрів повернутих через *fseminf*.

Exitflag – ціле число, яким закінчується алгоритм. Наступний перелік величин *exitflag* ат відповідні причини закінчення алгоритму.

1 – функція сходиться до рішення *x*.

4 – величина керування пошуку, що була менше ніж визначене допустиме відхилення і обмеження були менші ніж опції TolCon.

5 - – величина для керування пошуку, що була менше ніж визначене допустиме відхилення і обмеження були менші ніж опції TolCon.

0 – кількість ітерацій перевищило дії. MaxIter чи число функціональних оцінок перевищило опції FunEvals.

-1 – алгоритм був закінчений функцією виводу.

-2 – не було знайдено жодного виконаного пункту.

Lambda - містить множники Лагранжа при рішенні x (окремий тип обмеження). Поля структури:

- *Lower* - нижня границя.
- *Upper* – верхня границя.
- *Ineqlin* – лінійна нерівність.
- *Eqlin* – лінійне рівняння.
- *Ineqnonlin* – нелінійна нерівність.
- *Eqnonlin* – нелінійне рівняння.
- *Output* – структура, що містить інформацію про оптимізацію.

Поля структури:

- *Iterations* – число проведених ітерацій.
- *FuncCount* – кількість функціональних оцінок.
- *Algorithm*-використаний алгоритм.
- *Stepsize*-кінцевий розмір кроку.

Дії (Опції)

Опції оптимізації, що використовуються в *fseminf*. Можемо використовувати *optimset*, для того щоб встановити або змінити значення цих полів в опціях структури опцій.

DerivativeCheck – порівнює визначені користувачем похідні (градієнти) скінченно продиференційованою похідною.

Diagnostics – відображає інформацію діагностики функції, що мінімізується чи вирішується.

DiffMaxChange – максимальні зміни змінних при остаточно визначених градієнтах.

DiffMinChange – мінімальні зміни змінних при визначених градієнтах.

Display – рівні демонстрації. *'off'* – показ відсутній. *'iter'* – показ вихідних даних при кожній ітерації, *'notify'* – показ вихідних данних тільки якщо функція не прямує до границі, *'final'* – (значення по замовчуванню) відображає лише кінцевий вивід.

FunValCheck – перевіряє чи дійсні значення функції та обмежень. *'on'* показує попередження, коли цільова функція чи обмеження повертають значення, що є комплексними (складними), *Inf, or NaN. 'off'* (значення по замовчуванню) не відображають ніякого попередження.

GradObj – градієнт для цільової функції визначений користувачем.

MaxFunEvals – максимальне значення допустимих функціональних оцінок.

MaxIter – максимальна кількість допустимих ітерацій.

OutputFcn – визначає визначену користувачем функцію, що є функцією оптимізації в кожній ітерації .

RelLineSrchBnd – відносні обмеження (дійсні невід'ємні скалярні величини) на лінії пошуку крок довжиною такою, що задовольняє загальне зміщення *x*, $|\Delta x(i) \le |relLineSrchBnd \times \max(|x(i)|, t|ypicalx(i)|)$.

Ця опція забезпечує контроль над величиною зміщень в *x* для випадків, в яких рішення приймає кроки, що вважаються занадто великими.

RelLineSrchBndDuration – кількість ітерацій, для яких границі визначені в *RelLineSrchBnd* повинні бути активними. (за замовчуванням значення 1).

TolCon- границя кінцевого відхилення при порушенні обмеження.

TolFun – закінчення кінцевого відхилення від допустимих значень на функціональному значенні.

TolX – результат відхилення на *x*.

Примітки

Підпрограма оптимізації *fseminf* може змінювати (довірчий) інтервал здійснення вибірки *S*, встановити в *seminfcon*, протягом обчислення, тому що інші значення окрім довірчого інтервалу можуть бути більш підходящими для ефективності чи надійності. Також кінцеве значення w_i, за якою розрахований $K_i(x,w_i)$ допускають зміни протягом оптимізації, за умов, що це не призводить до суттєвих змін в кількості локальних мінімумів в $K_i(x,w_i)$.

Приклад 1

Необхідно мінімізувати функцію:

$$f(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_3 - 0.5)^2,$$

при обмеженнях:

$$K_1(x, w_1) = \sin(w_1 x_1)\cos(w_1 x_2) - \frac{1}{1000}(w_1 - 50)^2 - \sin(w_1 x_3) - x_3 \le 1,$$

$$K_2(x, w_2) = \sin(w_2 x_2)\cos(w_2 x_1) - \frac{1}{1000}(w_2 - 50)^2 - \sin(w_2 x_3) - x_3 \le 1.$$

причому $1 \le w_1 \le 100, 1 \le w_2 \le 100.$

Перетворюємо нерівності-обмеження:

$$K_1(x, w_1) = \sin(w_1 x_1)\cos(w_1 x_2) - \frac{1}{1000}(w_1 - 50)^2 - \sin(w_1 x_3) - x_3 - 1 \le 0,$$

$$K_2(x, w_2) = \sin(w_2 x_2)\cos(w_2 x_1) - \frac{1}{1000}(w_2 - 50)^2 - \sin(w_2 x_3) - x_3 - 1 \le 0.$$

1. Створюємо *M-file* з функцією мети:

```
function f = myfun(x,s)
% Objective function
f = sum((x-0.5).^2);
```

2. Створюємо *M-file* з відповідними функціями та інтервалом оптимізації:

```
function [c,ceq,K1,K2,s] = mycon(X,s)
% Initial sampling interval
if isnan(s(1,1)),
   s = [0.2 0; 0.2 0];
end
% Sample set
w1 = 1:s(1,1):100;
w2 = 1:s(2,1):100;

% Semi-infinite constraints
K1 = sin(w1*X(1)).*cos(w1*X(2)) - 1/1000*(w1-50).^2 -...
       sin(w1*X(3))-X(3)-1;
K2 = sin(w2*X(2)).*cos(w2*X(1)) - 1/1000*(w2-50).^2 -...
       sin(w2*X(3))-X(3)-1;

% No finite nonlinear constraints
c = []; ceq=[];

% Plot a graph of semi-infinite constraints
plot(w1,K1,'-',w2,K2,':'),title('Semi-infinite constraints')
drawnow
```

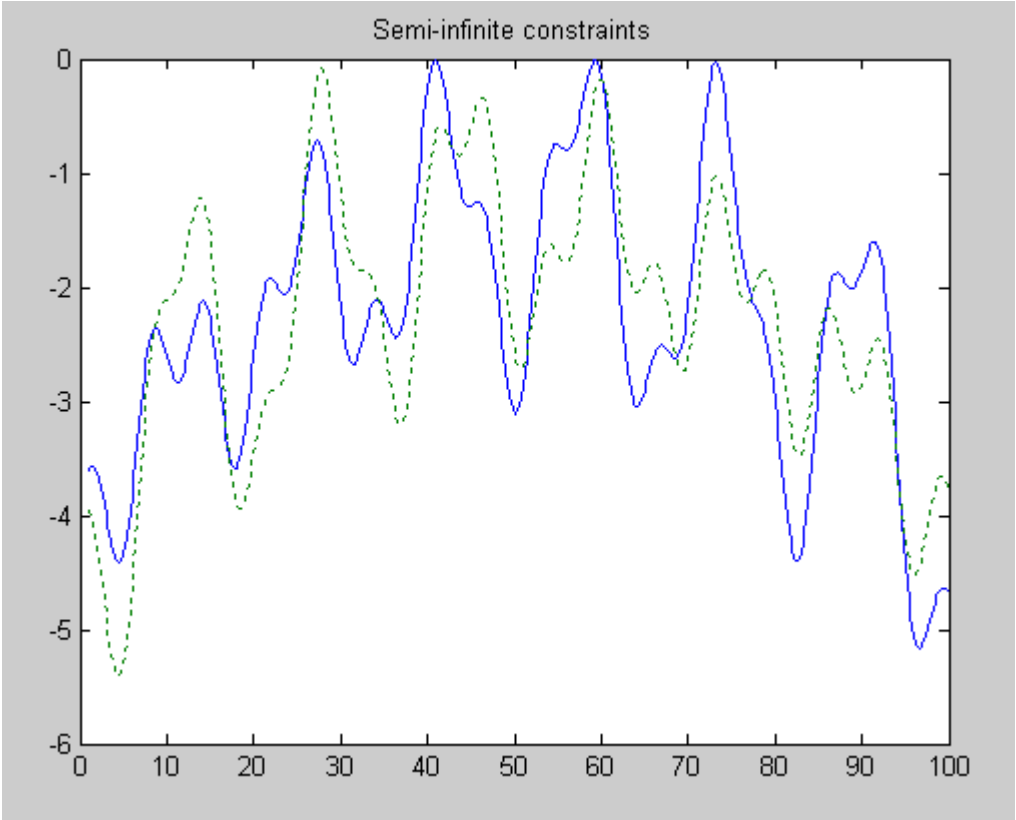
3. Виконуємо оптимізацію:

```
>> x0 = [0.5; 0.2; 0.3];      % Starting guess
[x,fval] = fseminf(@myfun,x0,2,@mycon)
Optimization terminated: magnitude of search direction less than 2*options.TolX
 and maximum constraint violation is less than options.TolCon.
No active inequalities.

x =

    0.6673
    0.3013
    0.4023


fval =

    0.0770
```

4. Знайдемо інші параметри оптимізації:

```
>> [c,ceq,K1,K2] = mycon(x,NaN);  % Use initial sampling interval
max(K1)

ans =

   -0.0017

>> max(K2)

ans =

   -0.0845
```

5. В результаті оптимізації отримали наступний графік:



Приклад 2

Необхідно мінімізувати функцію:

$$f(x) = (x_1 - 0.2)^2 + (x_2 - 0.2)^2 + (x_3 - 0.2)^2,$$

при обмеженнях:

$$K_1(x,w) = \sin(w_1 x_1)\cos(w_2 x_2) - \frac{1}{1000}(w_1 - 50)^2 - \sin(w_1 x_3) - x_3 + \ldots$$

$$\sin(w_2 x_2)\cos(w_1 x_1) - \frac{1}{1000}(w_1 - 50)^2 - \sin(w_1 x_3) - x_3 \le 1.5$$

причому $1 \le w_1 \le 100, 1 \le w_2 \le 100$,

в початковій точці $x = [0.25, 0.25, 0.25]$.

1. Створюємо *M-file* з функцією мети:

```
function f = myfun(x,s)
% Objective function
f = sum((x-0.2).^2);
```

2. Створюємо *M-file* з відповідними функціями та інтервалом оптимізації:

```
function [c,ceq,K1,s] = mycon(X,s)
% Initial sampling interval
if isnan(s(1,1)),
   s = [2 2];
end
% Sampling set
w1x = 1:s(1,1):100;
w1y = 1:s(1,2):100;
[wx,wy] = meshgrid(w1x,w1y);

% Semi-infinite constraint
K1 = sin(wx*X(1)).*cos(wx*X(2))-1/1000*(wx-50).^2 -...
       sin(wx*X(3))-X(3)+sin(wy*X(2)).*cos(wx*X(1))-...
       1/1000*(wy-50).^2-sin(wy*X(3))-X(3)-1.5;

% No finite nonlinear constraints
c = []; ceq=[];

% Mesh plot
m = surf(wx,wy,K1,'edgecolor','none','facecolor','interp');
camlight headlight
title('Semi-infinite constraint')
drawnow
```

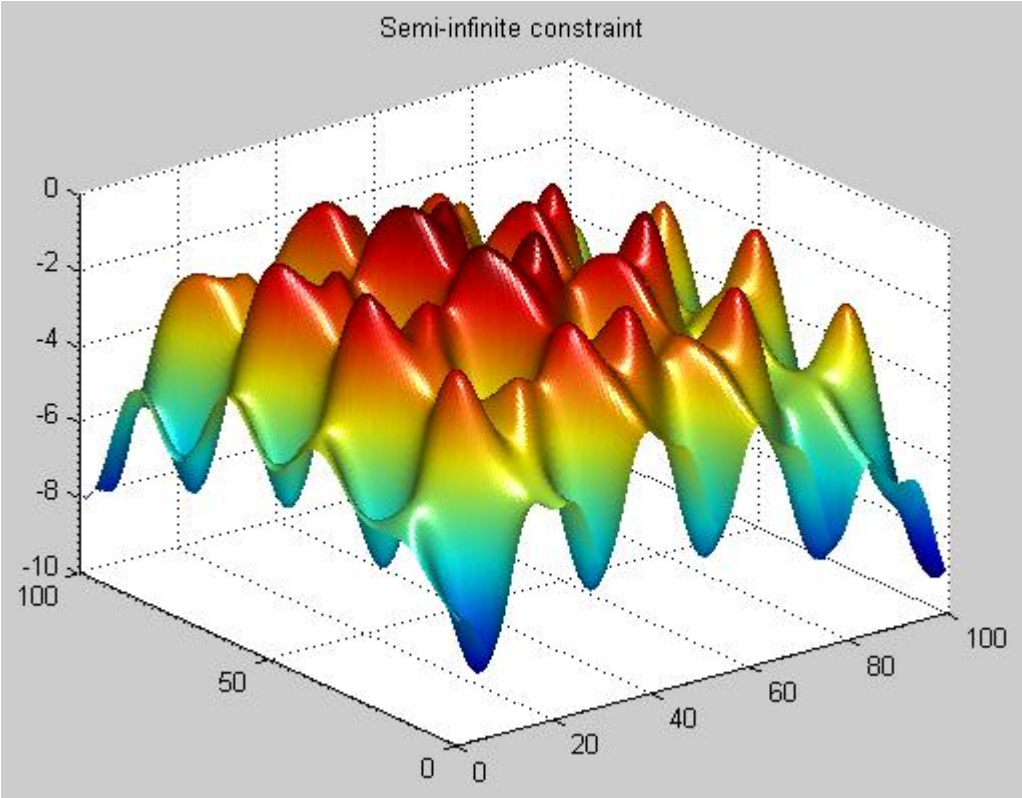
3. Виконуємо оптимізацію:

```
>> x0 = [0.25, 0.25, 0.25];     % Starting guess
[x,fval] = fseminf(@myfun,x0,1,@mycon)
Optimization terminated: magnitude of directional derivative in search
 direction less than 2*options.TolFun and maximum constraint violation
  is less than options.TolCon.
No active inequalities.

x =

    0.2979    0.2469    0.2610


fval =

    0.0155
```

4. Знайдемо інші параметри оптимізації:

```
>> [c,ceq,K1] = mycon(x,[0.5,0.5]);  % Sampling interval 0.5
max(max(K1))

ans =

   -0.0582
```

5. В результаті оптимізації отримали наступний графік:



Алгоритм

Fseminf використовує квадратичні та кубічні методи інтерполяції для оцінки пікових значень в часткових обмеженнях. Пікові значення використовуються, для того щоб сформувати ряд обмежень, які визначаються методом SQP, як в функції *fmincon*. Коли кількість обмежень змінюється, множники Лагранжа перерозподіляються до нового набору обмежень.

Розрахунок вибірки довірчого інтервалу використовує різницю між інтерпольованими піковими значеннями та піковими значеннями, представленими в наборі даних оцінюючи чи потребує функція взяття більших чи менших пунктів. Функція також оцінює ефективність інтерполяції, екстраполюючи криву та порівнюючи її з іншими значеннями на кривій. Рекомендований інтервал здійснення вибірки зменшується, коли пікові значення близькі до границь обмежень та нуля.

Обмеження

Функція, що мінімізується, обмеження та часткові обмеження, повинні бути неперервними функціями *x* та *w*. *Fseminf* може давати лише локальні рішення. Коли проблема не розв'язана, *fseminf* намагається мінімізувати максимальне значення обмеження.

Функція Lsqlin

Вирішує лінійні задачі оптимізації з обмеженнями методом найменших квадратів.

Рівняння

$$\min_x \frac{1}{2}\|Cx - D\|_2^2 \text{ при обмеженнях } \begin{array}{l} A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub \end{array}$$

де C, A, Aeq - матриці і D, b, beq, lb та ub - вектори.

Синтаксис

$x = lsglin(C, d, A, b)$

$x = lsglin(C, d, A, b, Aeq, beq)$

$x = lsglin(C, d, A, b, Aeq, beq, lb, ub)$

$x = lsglin(C, d, A, b, Aeq, beq, lb, ub, x0)$

$x = lsglin(C,d,A,b,Aeq,beq,lb,ub,x0,options)$

$[x, resnorm] = lsglin(...)$

$[x, resnorm, residual] = lsglin(...)$

$[x, resnorm, residual, exitflag\ ] = lsglin(...)$

$[x, resnorm, residual, exitflag\ , output] = lsglin(...)$

$[x, resnorm, residual, exitflag\ , output, lambda] = lsglin(...)$

Опис

- $x = lsglin(C, d, A, b)$ вирішує методом найменших квадратів лінійну систему $C * x = d$, яка обмежена нерівністю $A * x \le b$, де C - матриця $m \times n$.
- $x = lsglin(C, d, A, b, Aeq, beq)$ вирішує попередню задачу, яка додатково повинна задовольняти рівняння обмеження $Aeq \cdot x = beq$. Встановіть A=[] і b=[], якщо немає обмежень-нерівностей.
- $x = lsglin(C, d, A, b, Aeq, beq, lb, ub)$ визначає ряд найменших і найбільших значень, які можуть приймати коефіцієнти x, отже оптимальний розв'язок знаходиться на проміжку $lb \le x \le ub$. Встановіть Aeq=[] і beq=[], якщо немає обмежень-рівностей.
- $x = lsglin(C, d, A, b, Aeq, beq, lb, ub, x0)$ встановлює початкові значення $x0$. Встановіть lb=[] і ub=[], якщо немає граничних значень для результату.
- $x = lsglin(C,d,A,b,Aeq,beq,lb,ub,x0,options)$ мінімізує з варіантами оптимізації, які визначені у варіантах структури. Використовуйте *optimset* для встановлення цих варіантів.
- $[x, resnorm] = lsglin(...)$ надає значення двокомпонентного вектору квадратів залишків $(C * x - d)$^2.
- $[x, resnorm, residual] = lsglin(...)$ повертає результат залишку $C * x - d$.
- $[x, resnorm, residual, exitflag\ ] = lsglin(...)$ надає значення *exitflag*, що описує вихідні умови.
- $[x, resnorm, residual, exitflag\ , output] = lsglin(...)$ надає структуру вихідних даних, що містить інформацію про оптимізацію.

– $[x, resnorm, residual, exitflag, output, lambda] = lsglin(...)$ надає структуру $lambda$, поля якої містять значення множників Лагранжа при значеннях коефіцієнтів x.

Вхідні аргументи

Таблиця 15. Загальний опис аргументів для функції lsqlin

A, b	Матриця A і вектор b відповідно матриця коефіцієнтів лінійних нерівностей обмежень і вектор вільних членів $A \cdot x \le b$
Aeq, beq	Матриця Aeq і вектор beq відповідно матриця коефіцієнтів рівнянь обмежень і вектор вільних членів $Aeq \cdot x = beq$
C, d	Матриця C і вектор d відповідно матриця коефіцієнтів перевизначеної або невизначеної лінійної системи і вектор вільних членів, які повинні бути розв'язані
lb, ub	Вектори нижніх і верхніх границь. Ці аргументи повинні бути того ж розміру, що й x. Але, якщо lb містить менше елементів ніж x, скажімо m, то тільки перші m елементів вектора x обмежені знизу; верхні границі можуть бути визначені таким самим чином. Ви також можете встановити відсутність границь, використовуючи $-Inf$ для нижніх границь і Inf для верхніх.
$x0$	Початкове значення (може бути числом, вектором або матрицею).

Вихідні аргументи

Тут визначені такі аргументи функції, *як exitflag, lambda* и *output.*

Exitflag - це ціле число, що визначає причину завершення алгоритму. Нижче надано перелік значень *exitflag* і відповідних причин завершення алгоритму.

Таблиця 16. Перелік значень exitflag

1	Функція збігається до рішення x
3	Зміна залишку була меншою за визначену точність обчислення (ε)
0	Кількість ітерацій перевищила значення, зазначене в options (MaxIter).
-2	Проблема не має розв'язку.
-4	Погано задані умови гальмують подальшу оптимізацію
-7	Значимість напрямку пошуку є надто малою. Подальші дії не будуть успішними.

Lambda - структура містить множники Лагранжа для результату вектора x (визначена за типовими обмеженнями).

Таблиця 17. Поля структури lambda

lower	Нижня границя lb
upper	Верхнє границя ub
ineqlin	Лінійна нерівність
eqlin	Лінійне рівняння

Output - структура містить інформацію про оптимізацію.

Таблиця 18. Поля структури output

iterations	Число здійснених повторень
algorithm	Використаний алгоритм
cgiterations	Число PCG (попередній оцінок методом зведеного градієнту) повторень
firstorderopt	Міра оптимальності першого порядку (тільки large-scale algorithm)

Options (варіанти оптимізації)

Таблиця 19. Функції, що використовуються для обох алгоритмів оптимізації

Diagnostics	Показує інформацію про функцію, що буде мінімізована
Display	Рівень представлення інформації. '*off*' – вихідні данні не показуються, '*final*' – показуються тільки кінцеві вихідні данні
MaxIter	Максимальне число дозволених повторень
TypicalX	Типові значення x

Таблиця 20. Функції, що використовуються тільки в Large-scale алгоритмі

JacobMult	Це задана ідентифікатор для матриці Якобі складної функції. Для великомасштабних структурованих задач, ця функція розраховує матрицю Якобі у вигляді показників $J*Y, J'*Y$ або $J'*(J*Y)$, фактично не формуючи Y. Функція має форму: $W = jmfun(Jinfo, Y, flag, p1, p2,...)$, де $Jinfo$ і додаткові параметри $p1, p2$ містять матриці, що використовуються для розрахунку $J*Y, J'*Y$ або $J'*(J*Y)$. Перший аргумент $Jinfo$ повинен бути таким самим, як другий аргумент, який розраховується цільовою функцією fun. Наприклад, $[F, \text{Jinfo}] = \text{fun(x)}$. Y - це матриця, яка має таку кількість рядків, як і розмірність задачі. Flag визначає, який показник розраховувати: Якщо $flag = 0$, тоді $W = J*Y$, якщо $flag > 0$, то $W = J'*Y$, якщо $flag < 0$, то $W = J'(J*Y)$, у будь-якому випадку Y не сформована явно. Відзначимо, що 'Jacobian' повинен бути включеним ('on') для того, щоб $Jinfo$ був переданий від функції fun до $jmfun$.
MaxPCGIter	Максимальне число повторень PCG (методом зведеного градієнта)
PrecondBand Width	верхня пропускна спроможність попередніх оцінок для PCG. За замовчуванням, застосовуються діагональні попередні оцінки (верхня пропускна спроможність дорівнює 0). Для деяких проблем, збільшення пропускної спроможності зменшує число PCG повторень.
TolFun	Найбільша допустима похибка значення функції .
TolPCG	Найбільша допустима похибка завершення на повторенні PCG.

Приклад

Знайти розв'язок методом найменших квадратів перевизначеної системи $C*x = d$ з границями $lb \le x \le ub$.

1. Вводимо матриці:

```
C =

    0.9501    0.7620    0.6153    0.4057
    0.2311    0.4564    0.7919    0.9354
    0.6068    0.0185    0.9218    0.9169
    0.4859    0.8214    0.7382    0.4102
    0.8912    0.4447    0.1762    0.8936

d =

    0.0578
    0.3528
    0.8131
    0.0098
    0.1388

A =

    0.2027    0.2721    0.7467    0.4659
    0.1987    0.1988    0.4450    0.4186
    0.6037    0.0152    0.9318    0.8462

b =

    0.5251
    0.2026
    0.6721
```

2. Задаємо значення лівої та правої границі оптимізації:

```
lb = -0.1*ones(4,1);
ub = 2*ones(4,1);
```

3. Виконуємо лінійну оптимізацію:

```
[x,resnorm,residual,exitflag,output,lambda] = lsqlin(C,d,A,b,[ ],[ ],lb,ub);
```

4. Отримуємо результати оптимізації:

```
x =

   -0.1000
   -0.1000
    0.2152
    0.3502

>> lambda.ineqlin

ans =

         0
    0.2392
         0
```

```
>> lambda.lower

ans =

    0.0409
    0.2784
         0
         0
```

```
>> lambda.upper

ans =

     0
     0
     0
     0
```

Елементи, відмінні від нуля векторів в полях лямбди вказують на активні обмеження у розв’язку. У цьому випадку друге обмеження нерівності (в lambda.ineqlin) і перша нижня і друга нижня границі є активними – тобто розв’язок знаходиться на кінці обмеження. Для проблем без обмежень використовуйте ділення матриці. Наприклад, $x = A\backslash b$.

Оскільки розв'язувана задача завжди опукла, *lsqlin* знайде глобальне, хоча не обов'язково унікальне, рішення. Числові результати більш ймовірні, якщо Ви визначаєте рівності явно, використовуючи Aeq і beq, замість неявно заданих lb, ub. Якщо $x0$ не можливо задати явно, *lsqlin* вибере нове явно задане відправне значення.

Якщо компоненти x не мають верхніх (нижніх) границь, то *lsqcurvefit* бажає, щоб відповідні елементи *ub* (або *lb*) були встановлені в *inf* (або –*inf* для нижніх границь) замість встановлення довільних дуже великих додатніх (або від’ємних для нижньої границі) чисел.

Алгоритм

Large-Scale Optimization

Коли задача має тільки верхні й нижні границі, наприклад, не визначені лінійні нерівності або рівності, а матриця C має рядків не менше, ніж стовпчиків, алгоритм за замовчуванням є large-scale method. Цей алгоритм методу довірчого інтервалу, що базується на внутрішньо рефлективному методі Ньютона [1]. Кожне повторення надає приблизне розв’язання великої лінійної системи, використовуючи метод зведеного градієнта (PCG).

Medium-Scale Optimization

Lsqlin, у випадку, коли варіант *LargeScale* не включений (встановлено *'off'* у *optimset*) або коли присутні лінійні обмеження у вигляді нерівностей або рівностей, базується на *quadprog*, яка використовує активний метод набору [2]. Він знаходить початкове можливе рішення першим обчислення задача лінійного програмування.

Посилання

[1] Coleman, T.F. and Y. Li, "A Reflective Newton Method for Minimizing a Quadratic Function Subject to Bounds on Some of the Variables," SIAM Journal on Optimization, Vol. 6, Number 4, pp. 1040-1058, 1996.

[2] Gill, P.E., W. Murray, and M.H. Wright, Practical Optimization, Academic Press, London, UK, 1981.

Функція Lsqcurvefit

Функція *lsqcurvefit* вирішує нелінійні задачі підгонки кривої (відповідності даним) методом найменших квадратів. Таким чином, при наявних вхідних даних $xdata$ і вихідних $ydata$, отриманих дослідним шляхом, знаходяться коефіцієнти x, які найбільше відповідають рівнянню

Рівняння

$$\min_x \frac{1}{2}\|F(x, xdata) - ydata\|_2^2 = \frac{1}{2}\sum_{i=1}^{m}(F(x, xdata_i) - ydata_i)^2$$

де $xdata$ и $ydata$ - вектори довжиною m і $F(x, xdata)$ - векторна функція.

Мета цієї функції – забезпечити вигляд, створений спеціально для задач відповідності даних.

Синтаксис

$x = lsqcurvefit(fun, x0, xdata, ydata)$

$x = lsqcurvefit(fun, x0, xdata, ydata, lb, ub)$

$x = lsqcurvefit(fun, x0, xdata, ydata, lb, ub, options)$

$[x, resnorm] = lsqcurvefit(...)$

$[x, resnorm, residual] = lsqcurvefit(...)$

$[x, resnorm, residual, exitflag] = lsqcurvefit(...)$

$[x, resnorm, residual, exitflag, output] = lsqcurvefit(...)$

$[x, resnorm, residual, exitflag, output, lambda] = lsqcurvefit(...)$

$[x, resnorm, residual, exitflag, output, lambda, jacobian] = lsqcurvefit(...)$

Опис

Функція *lsqcurvefit* вирішує нелінійні проблеми відповідності даних. *Lsqcurvefit* вимагає, визначену користувачем функцію для того, щоб обчислити векторну функцію $F(x, xdata)$. Розмір вектора отриманої функції повинен співпадати з розміром векторів $xdata$ і $ydata$.

- $x = lsqcurvefit(fun, x0, xdata, ydata)$ починається з $x0$ і знаходить коефіцієнти x (методом найменших квадратів), завдяки яким нелінійна функція $fun(x, xdata)$ найбільше підходить до даних $ydata$. Вектор (матриця) $ydata$ повинен бути такого ж розміру, як і вектор (матриця) F, подана у вигляді fun.
- $x = lsqcurvefit(fun, x0, xdata, ydata, lb, ub)$ визначає ряд найменших і найбільших значень, які можуть приймати коефіцієнти x, отже оптимальний розв'язок знаходиться на проміжку $lb \le x \le ub$.
- $x = lsqcurvefit(fun, x0, xdata, ydata, lb, ub, options)$ мінімізує з варіантами оптимізації, які визначені у структурі опцій. Використовуйте *optimset* для встановлення цих варіантів. Встановлюйте порожні матриці для lb і ub, якщо граничні значення для коефіцієнтів відсутні.
- $[x, resnorm] = lsqcurvefit(...)$ розраховує двокомпонентний вектор квадратів залишків при значеннях коефіцієнтів x: $sum((fun(x, xdata) - ydata)\text{^}2)$.
- $[x, resnorm, residual] = lsqcurvefit(...)$ розраховує залишок різниці $fun(x, xdata) - ydata$ при значеннях коефіцієнтів x.

- $[x, resnorm, residual, exitflag\ ] = lsqcurvefit(...)$ надає значення $exitflag$, що описує вихідні умови.
- $[x, resnorm, residual, exitflag\ , output] = lsqcurvefit(...)$ надає структуру вихідних даних, що містить інформацію про оптимізацію.
- $[x, resnorm, residual, exitflag\ , output, lambda] = lsqcurvefit(...)$ надає структуру $lambda$, поля якої містять значення множників Лагранжа при значеннях коефіцієнтів x.
- $[x, resnorm, residual, exitflag\ , output, lambda, jacobian] = lsqcurvefit(...)$ надає матрицю Якобі функції fun при значеннях коефіцієнтів x.

Вхідні аргументи

Далі більш детально будуть описані такі вхідні аргументи функції, як fun та $options$.

fun - це функція, що оптимізується. Це функція, що бере вектор x та надає значення вектору F, цільові функції оцінені в x. Функція fun може бути визначена ідентифікатором для M-file function:

$x = lsqcurvefit(@myfun, x0, xdata, ydata)$,

де $@myfun$ - це функція MATLAB

$functionF\ = myfun(x, xdata)$

$F = ...$

Функція fun може бути також функцією ідентифікатором для анонімної функції.

$f\ = @(x, xdata)x(1) * xdata\text{^}2 + x(2) * \sin(xdata),...$

$'x','xdata';$

$x = lsqcyrvefit(f, x0, xdata, ydata);$

Зауважимо, що функція видає в результаті $fun(x, xdata)$, а не квадрат суми відхилень $sum((fun(x, xdata) - ydata)\text{^}2)$.

Якщо матриця Якобі може бути обчислена і ця опція „включена”, то встановлюємо її так:

$options = optimset('Jacobian','on')$

Тоді функція fun видає у результаті у другому вихідному аргументі значення матриці Якобі J для коефіцієнтів x. Відзначте, що перевіряючи значення *nargout* (функція, що визначає число вхідних аргументів), функція може не обчислювати J, якщо fun має лише 1 вихідний аргумент (у випадку, коли оптимізаційний алгоритм потребує лише значення F без J).

function [F,J] = myfun(x, xdata)

F = ... *Функція мети від* x.

if nargout >1 *Два вихідні аргументи.*

J = ... *Матриця Якобі (від* x*).*

Якщо fun видає вектор (матрицю) із m компонентів і x має довжину n, де n - це довжина $x0$, тоді матриця Якобі J - це матриця $m \times n$, у якої $J(i, j)$ - це частинна похідна $F(i)$ від $x(i, j)$ (відзначимо, що матриця Якобі J - це значення градієнта функції $F(i)$).

Вихідні аргументи

Тут визначені такі аргументи функції, *як exitflag, lambda* и *output. Exitflag* - це ціле число, що визначає причину завершення алгоритму.

Таблиця 21. Перелік значень exitflag і відповідних причин завершення алгоритму

1	Функція збігається до рішення x
2	Зміна x була меншою за визначену точність обчислення (ε)
3	Зміна залишку була меншою за визначену точність обчислення (ε)
4	Значимість напрямку пошуку є меншою за визначену точність обчислення (ε)
0	Кількість ітерацій перевищила значення, зазначене в options (MaxIter).
-1	Алгоритм був закінчений вихідною функцією.
-2	Проблема не має розв'язку: ряди верхніх і нижніх границь суперечливі
-4	Подальша оптимізація не можлива

Lambda - структура містить множники Лагранжа для результату вектора x (виділеного типом обмеження).

Таблиця 22. Поля структури lambda

lower	Нижня границя lb
upper	Верхня границя ub

Output - структура містить інформацію про оптимізацію.

Таблиця 23. Поля структури output

iterations	Число здійснених повторень
funcCount	Число оцінок функції
algorithm	Використаний алгоритм
cgiterations	Число PCG повторень (тільки large-scale algorithm)
stepsize	Розмір кінцевого кроку (тільки medium-scale algorithm)
firstorderopt	Міра оптимальності першого порядку (тільки large-scale algorithm)

Відзначте, що сума квадратів не повинна бути сформована явно. Замість цього ваша функція повинна повернути вектор значень функції.

Options (варіанти оптимізації)

Деякі варіанти оптимізації застосовують для усіх алгоритмів, деякі - тільки при використанні *large-scale algorithm*, деякі – тільки *medium-scale algorithm*. Ви можете використовувати *optimset* (створює або редагує структуру опцій) для встановлення або зміни значень цих полів у варіантах структури *options.*

The *Large-Scale* опція надає перевагу алгоритму, який буде використовуватися. Це тільки перевага, бо певні умови повинні бути виконані, щоб застосовувався *large-scale algorithm* або *medium-scale algorithm*. Для великомасштабного алгоритму нелінійна система рівнянь не може бути невизначеною: число рівнянь (елементів F, функцією, що повертає значення fun) повинно бути не менше, ніж довжина вектора x. Крім того, тільки великомасштабний алгоритм працює з активними обмеженнями.

LargeScale - використовуйте large-scale algorithm, якщо можливо, коли встановлено 'on'. Алгоритм середнього масштабу використовується, коли встановлено 'off'.

Таблиця 24. Функції, що використовуються для обох алгоритмів оптимізації

DerivativeCheck	Порівнює похідні, що надані користувачем (матриця Якобі) з остаточно розрахованими похідними
Diagnostics	Показує інформацію про функцію, що буде мінімізована
DiffMaxChange	Максимальна зміна змінних при остаточному диференціюванні
DiffMinChange	Мінімальна зміна змінних при остаточному диференціюванні
Display	Рівень представлення інформації. '*off*' – вихідні данні не показуються, '*final*' – показуються тільки кінцеві вихідні данні
Jacobian	'*on*' – *Lsqcurvefit* використовує задану користувачем матрицю Якобі (визначену в *fun*) або інформацію про Якобі (якщо використовується *JacobMult*) для цільової функції. Якщо 'off' – приблизно знаходить матрицю Якобі через остаточне диференціювання.
MaxFunEvals	Максимальне число дозволених оцінок функції
MaxIter	Максимальне число дозволених повторень
OutputFcn	Визначить вид функції, який приймає функція оптимізації на кожному кроці ітерації
TolFun	Найбільша допустима похибка значення функції
TolX	Найбільша допустима похибка коефіцієнта x
TypicalX	Типові значення x

Таблиця 25. Функції, що використовуються тільки для великомасштабних алгоритмів

JacobMult	Це ідентифікатор для матриці Якобі складної функції. Для великомасштабних структурованих задача, ця функція розраховує матрицю Якобі у вигляді показників $J*Y, J'*Y$ або $J'*(J*Y)$, фактично не формуючи Y. Функція має форму: $W = jmfun(Jinfo, Y, flag, p1, p2,...)$, де *Jinfo* і додаткові параметри $p1, p2$ містять матриці, що використовуються для розрахунку $J*Y, J'*Y$ або $J'*(J*Y)$. Перший аргумент *Jinfo* повинен бути таким самим, як другий аргумент, який розраховується цільовою функцією *fun*. Наприклад, $[F, \text{Jinfo}] = \text{fun(x)}$. Y - це матриця, яка має таку кількість рядків таку, як і розмірність задачі. Flag визначає, який показник розраховувати: Якщо $flag = 0$, тоді $W = J*Y$, якщо $flag > 0$, то $W = J'*Y$, якщо $flag < 0$, то $W = J'(J*Y)$, У будь-якому випадку Y не сформована явно. Відзначимо, що 'Jacobian' повинен бути включеним ('on') для того, щоб *Jinfo* був переданий від функції *fun* до *jmfun*.
JacobPattern	Визначає розрідженість матриці Якобі для кінцевого диференціювання. Якщо не зручно обчислити матрицю Якобі через *fun*, то *lsqcurvefit* може отримати приблизне J через остаточні відхилення, забезпечивши структуру J, тобто знаходження ненулів, що використовуються як значення для JacobPattern. У найгіршому разі, якщо структура невідома, Ви можете встановити JacobPattern як щільну матрицю, і повне наближення кінцевого диференціювання обчислюватиметься при кожному повторенні (це не відбудеться, якщо JacobPattern не встановлений). Це може бути дуже коштовним для великих проблем, тому варто прикласти зусилля для визначення

	структури розрідженості.
MaxPCGIter	Максимальне число повторень попередніх обчислень PCG (методом зведеного градієнта)
PrecondBandWidth	Верхня межа попередніх умов для PCG. За замовчуванням, діагональні попередні умови використовуються. Для деяких проблем збільшення пропускної спроможності зменшує число ітерації PCG.
TolPCG	Найбільша допустима похибка завершення на повторенні PCG.

Приклад

Припустимо, що маючи вектори даних xdata і ydata, ви хочете знайти коефіцієнти x, які б найкраще задовольняли рівняння:

$$ydata(i) = x(1)\cdot xdata(i)^2 + x(2)\cdot \sin(xdata(i)) + x(3)\cdot xdata(i)^3$$

Тоді ви хочете мінімізувати:

$$\min_x \frac{1}{2}\left\|F(x, xdata) - ydata\right\|_2^2,$$

де $xdata$ и $ydata$ - вектори довжиною m і $F(x, xdata)$ - така векторна функція:

*F(x,xdata) = x(1)*xdata.^2 + x(2)*sin(xdata) + x(3)*xdata.^3*

Початкові значення дорівнюють *x0 = [10, 10, 10]*

1. По-перше, пишемо M-file для того, щоб повернути значення функції *F*

```
function F = myfun(x,xdata)
F = x(1)*xdata.^2 + x(2)*sin(xdata) + x(3)*xdata.^3;
```

2. В результаті оптимізації отримуємо:

```
>> % Assume you determined xdata and ydata experimentally
xdata = [3.6 7.7 9.3 4.1 8.6 2.8 1.3 7.9 10.0 5.4];
ydata = [16.5 150.6 263.1 24.7 208.5 9.9 2.7 163.9 325.0 54.3];
x0 = [10, 10, 10]                    % Starting guess
[x,resnorm] = lsqcurvefit(@myfun,x0,xdata,ydata)

x0 =

    10    10    10

Optimization terminated: relative function value
 changing by less than OPTIONS.TolFun.

x =

    0.2269    0.3385    0.3021


resnorm =

    6.2950
```

Алгоритм

Large-Scale Optimization. За замовчуванням *lsqcurvefit* вибирає large-scale algorithm. Цей алгоритм методу довірчого інтервалу, що базується на внутрішньо рефлективному методі Ньютона [1, 2]. Кожне повторення надає приблизне розв'язання великої лінійної системи, використовуючи метод зведеного градієнта (PCG).

Medium-Scale Optimization. *Lsqcurvefit* , у випадку, коли варіант *LargeScale* не включений (встановлено *'off'* у *optimset*), використовується метод лінійного пошуку

Levenberg-Marquardt [4,5,6]. Альтернативно може бути вибраний метод Гауса-Ньютона (лінійний пошук) [3].

Ви можете вибрати алгоритм, встановлюючи варіант *LevenbergMarquardt* на *'on'* за допомогою *optimset*. Встановлення варіанту LevenbergMarquardt на 'off'' (LargeScale на 'off') вибирає метод Гауса-Ньютона, який в загальному випадку швидший, якщо залишок $\|F(x)\|_2^2$ є малим. За замовчуванням лінійний алгоритм пошуку, наприклад, варіант *LineSearchType* встановлений на *'quadcubic',* який є гарантованим об'єднанням методів квадратичний і кубічний поліноміальної інтерполяції і екстраполяції.

Ви можете вибрати безпечний кубічний поліноміальний метод, встановлюючи *LineSearchType* на *'cubicpoly'*. Звичайно цей метод вимагає меншу кількість оцінок функції, але більше оцінок градієнта. Таким чином, якщо градієнти відомі й можуть бути нескладно обчислені, кубічний поліноміальний метод лінійного пошуку кращий.

Обмеження

Функція, що буде мінімізована, повинна бути неперервною. *Lsqcurvefit* повинна надавати одне локальне рішення. *Lsqcurvefit* працює тільки з дійсними змінними (визначена користувачем функція повинна надавати тільки дійсні значення). Якщо x має комплексні змінні, то вони повинні бути поділені на дійсну і уявну частини.

The *large-scale* метод для *lsqcurvefit* не вирішує невизначені системи. Він вимагає, щоб число рівнянь, тобто вимірів задачі F, було не менше за число змінних. У випадку невизначених систем, навпаки, застосовують *medium-scale algorithm*.

Розрахунок попередніх умов, використаний у такій частині *large-scale method*, як метод зведеного градієнта формує *JTJ* (де *J* – матриця Якобі) до розрахунку попередніх умов, тому ряд *J* з більшість ненулями, результати якого знаходяться в *JTJ*, може призвести до складного процесу вирішення великих проблем.

Якщо компоненти x не мають верхніх (нижніх) границь, то *lsqcurvefit* віддає перевагу, коли відповідні компоненти *ub* (або *lb*) визначені як *inf* (або *–inf* для нижніх границь), а не як довільні дуже великі додатні (або від'ємні для нижніх границь) числа.

Medium-Scale Optimization не працює з активними обмеженнями. Через те, що *large-scale algorithm* не розв'язує невизначені системи, а medium-scale не працює з активними обмеженнями, ці задачи не можуть бути розв'язані за допомогою фінкції *lsqcurvefit.*

Посилання

[1] Coleman, T.F. and Y. Li, "An Interior, Trust Region Approach for Nonlinear Minimization Subject to Bounds," SIAM Journal on Optimization, Vol. 6, pp. 418-445, 1996.

[2] Coleman, T.F. and Y. Li, "On the Convergence of Reflective Newton Methods for Large-Scale Nonlinear Minimization Subject to Bounds," Mathematical Programming, Vol. 67, Number 2, pp. 189-224, 1994.

[3] Dennis, J. E. Jr., "Nonlinear Least-Squares," State of the Art in Numerical Analysis, ed. D. Jacobs, Academic Press, pp. 269-312, 1977.

[4] Levenberg, K., "A Method for the Solution of Certain Problems in Least-Squares," Quarterly Applied Math. 2, pp. 164-168, 1944.

[5] Marquardt, D., "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," SIAM Journal Applied Math., Vol. 11, pp. 431-441, 1963.

[6] More, J. J., "The Levenberg-Marquardt Algorithm: Implementation and Theory," Numerical Analysis, ed. G. A. Watson, Lecture Notes in Mathematics 630, Springer Verlag, pp. 105-116, 1977.

Функція Fmincon

Знаходить мінімум обмеженої нелінійної функції багатьох змінних.

Рівняння

$$\min_x f(x),$$

при цьому

$$\begin{aligned} &c(x) \le 0 \\ &ceq(x) = 0 \\ &A * x \le b \\ &Aeq * x = beq \\ &lb \le x \le ub \end{aligned},$$

де *x, b, beq, ul, lb* вектори, *A, Aeq* – матриці, *c(x), ceq(x)* – функції, що повертають значення векторів, *f(x)* – функція, що повертає скалярне значення. *f(x), ceq(x), c(x)* можуть бути нелінійними функціями.

Синтаксис

- *x = fmincon(fun,x0,A,b)*
- *x = fmincon(fun,x0,A,b,Aeq,beq)*
- *x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)*
- *x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)*
- *x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)*
- *[x,fval] = fmincon(...)*
- *[x,fval,exitflag] = fmincon(...)*
- *[x,fval,exitflag,output] = fmincon(...)*
- *[x,fval,exitflag,output,lambda] = fmincon(...)*
- *[x,fval,exitflag,output,lambda,grad] = fmincon(...)*
- *[x,fval,exitflag,output,lambda,grad,hessian] = fmincon(...)*

Опис

- *fmincon* використовують, щоб знайти скалярний мінімум функції декількох змінних, яким задано початкові значення. Це є нелінійним програмуванням або оптимізацією нелінійних обмежень.
- *x = fmincon(fun,x0,A,b)* – починаючи з *x0*, знаходимо мінімум функції, причому задовольняються лінійні нерівності *A*x <= b*. *x0* може бути скаляром, вектором, матрицею.
- *x = fmincon(fun,x0,A,b,Aeq,beq)* – мінімізує функцію, при цьому виконуються не тільки нерівності, але й рівності типу *Aeq*x = beq*. Якщо нерівностей немає, необхідно встановити *A = [], b = []*.
- *x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)* – визначає верхні та нижні границі області змінних *x,* при цьому розв'язок завжди знаходиться в області зміни *lb <= x <= ub*. *Aeq = [], beq = [],* якщо не існує рівностей. *lb (i) = -Inf*, якщо *x(i)* не обмежений знизу, *ub (i) = Inf*, якщо *x(i)* необмежених зверху.
- *x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)* – мінімізує нелінійні нерівності *c(x)* або рівності *ceq(x),* що визначені в *nonlcon*. При цьому оптимізація визначає *c (x) <= 0* та *ceq (x) = 0. lb = [], ub = [],* якщо не існує границь.
- *x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)* – мінімізує, використовуючи опції оптимізації, що були заздалегідь визначені. Використовуйте *optimset*, щоб

установити ці опції. Множина *nonlcon = [],* якщо немає нелінійних обмежень рівностей та нерівностей.

- *[x,fval] = fmincon(...)* – визначає значення цільової функції.
- *[x,fval,exitflag] = fmincon(...)* – визначає значення *exitflag*, що описує вихідні умови *fmincon***.**
- *[x,fval,exitflag,output] = fmincon(...)* – виводить інформацію про оптимізацію.
- *[x,fval,exitflag,output,lambda] = fmincon(...)* – визначає лямбду структури, що містить множники Лагранжа для знаходження *x*.
- *[x,fval,exitflag,output,lambda,grad] = fmincon(...)* – знаходить значення градієнта функції.
- *[x,fval,exitflag,output,lambda,grad,hessian] = fmincon(...)* - надає значення *Hessian* (матриця Гессе) при розв'язку.

Вхідні аргументи

Функціональні параметри містять загальний опис параметрів, що передаються в *fmincon*. Цей розділ параметрів визначає функціонально-спеціалізовані особливості *fun, nonlcon, options.*

Функція, що буде мінімізована. Функція, що приймає вектор *x* та визначає скаляр *f*. Цільова функція. Може бути визначена як функціональний дескриптор для M-file функції

x = fmincon(@myfun,x0,A,b)

де *myfun* це функція MATLAB

function f = myfun(x)

f = ... % Compute function value at x

fun може бути також функціональним описом будь-якої функції

x = fmincon(@(x)norm(x)^2,x0,A,b)

Якщо може бути визначений градієнт (при цьому опція *GradObj* «on»

options = optimset('GradObj','on')), тоді функція має бути повернена у другому параметрі виводу, значення градієнта *g* при векторі *x*.

Зверніть увагу, що при перевірці *nargout* є можливість, що градієнт не буде розраховано, якщо цільова функція описана лише одним вихідним аргументом. при цьому алгоритм оптимізації потребує лише значення функції, а не градієнта.

```
function [f,g] = myfun(x)
f = ... % Compute the function value at x
if nargout > 1 % fun called with two output arguments
 g = ... % Compute the gradient evaluated at x
end
```

Градієнт складається з часткових похідних функції в точці *x*. Таким чином і компонент градієнта є частковою похідною функції за і компонентом *x*.

Якщо матриця *Hessian* може бути розрахована, опція *Hessian* «on», тоді функція *fun* має визначити значення цієї симетричної матриці *H* при *x* в третьому вихідному аргументі. Зверніть увагу, що при перевірці значення *nargout* Ви можете уникнути розрахунку *H*, якщо функція описана лише одним або двома вихідними аргументами.

```
function [f,g,H] = myfun(x)
f = ... % Compute the objective function value at x
if nargout > 1 % fun called with two output arguments
 g = ... % Gradient of the function evaluated at x
 if nargout > 2
 H = ... % Hessian evaluated at x
 end
end
```

Матриця *Hessian* – друга матриця частинних похідних *f* в точці *x*. Таким чином компоненти матриці є другими частковими похідними функції відносно компонентів *x*. *Hessian* за визначенням симетрична матриця.

Nonlcon - функція, що обчислює нелінійні обмеження нерівності *c (x) <= 0* та рівності *ceq (x) = 0*. приймає вектор *x* та повертає два вектори *c* та *ceq*. Вектор c містить нерівності оцінені в *x*, а вектор *ceq* містить рівності оцінені в *x*. Функція *nonlcon* може бути визначена як функціональний дескриптор:

x = fmincon(@myfun,x0,A,b,Aeq,beq,lb,ub,@mycon)

де *mycon* це функція MATLAB

```
function [c,ceq] = mycon(x)
c = ... % Compute nonlinear inequalities at x.
ceq = ... % Compute nonlinear equalities at x.
```

Якщо градієнт обмежень може бут розраховано (*options = optimset('GradConstr','on')*), то функція *nonlcon* має повернути значення *GC*, градієнт *c(x),* та *GCeq*, градієнт *ceq(x),* як третій та четвертий аргументи виходу. Градієнти можуть бути не розраховані, якщо описано, що *nonlcon* має подати лише два вихідні аргументи (*c* та *ceq*).

```
function [c,ceq,GC,GCeq] = mycon(x)
c = ... % Nonlinear inequalities at x
ceq = ... % Nonlinear equalities at x
if nargout > 2 % nonlcon called with 4 outputs
 GC = ... % Gradients of the inequalities
 GCeq = ... % Gradients of the equalities
end
```

Вихідні аргументи

Загальний опис функціональних параметрів для функції *fmincon*.

Таблиця 26. Спеціалізовані особливості exitflag, lambda, output

exitflag	Ціле число, що пояснює, чому завершилося виконання алгоритму. У списку вказані можливі причини	
	1	Умови оптимальності задоволені
	2	Зміна x менше вказаного допуску
	3	Зміна цільової функції менше вказаного допуску
	4	Значимість напрямку пошуку менше вказаного допуску та порушення обмежень менші за options.TolCon.
	5	Величина косої похідної менша за вказаний допуск та порушення обмежень менші за options.TolCon.
	0	Кількість ітерацій перевищує options.MaxIter або число функціональних оцінок перевищило options.FunEvals
	-1	Алгоритм було завершено функцією виходу
	-2	Не було знайдено розв'язку
grad	Градієнт x	
hessian	Hessian x	
lambda	Структура, що містить множники Лагранжа для знаходженя x	
	lower	Нижня межа lb
	upper	Верхня межа ub
	ineqlin	Лінійні нерівності
	eqlin	Лінійні рівності
	ineqnonlin	Нелінійні нерівності
	eqnonlin	Нелінійні рівності

output	Структура, що містить інформацію про оптимізацію	
	iterations	Кількість ітерацій
	funcCount	Число розрахунків фкнкціонального алгоритму
	algorithm	Алгоритм, що був використаний
	cgitrrations	Число ітерацій PCG (large-scale algorithm)
	stepsize	Кінцевий розмір (рівень) кроку (medium-scale algorithm)
	firstorderopt	Міра оптимальності першого рівня
		Для зв'язаних обмежених задач large-scale оптимальність першого рівня – норма безкінечності v*g, де v визначено в обмеженнях, g – градієнт. Для задач large-scale з лінійними рівняннями оптимальність першого рівня – норма безкінечності спроектованого градієнта (тобто, градієнт, спроектований на nullspace of Aeq).

Hessian (матриця Гессе)

fmincon розраховує вихідний аргумент *Hessian* таким чином:

- Використовуючи *medium-scale algorithm*, функція розраховує наближення квазіНьютона до *Hessian* функції Лагранжа в x.
- При використанні *large-scale algorithm* застосовуються функції:
  - ✓ *options.Hessian*, застосування її розраховує *Hessian* для *x*
  - ✓ Скінченно-розрізнене наближення до *Hessian* в *x*, якщо використовувати лише градієнт. *large-scale algorithm* не застосовує нелінійні обмеження. *Hessian* функції Лагранжа те ж, що й *Hessian* цільової функції.

Опції

Опції оптимізації, що використовуються в *fmincon*. Деякі опції застосовуються до всіх алгоритмів, деякі доцільно використовувати в *large-scale algorithm*, інші – лише в *medium-scale algorithm*. Можна використовувати *optimset*, щоб установити або змінювати значення областей опцій.

Опція *LargeScale* визначає переваги використання алгоритму. Це лише переваги, так як необхідно мати певні умови для використання *large-scale algorithm*. Для *fmincon* Ви маєте надати градієнт, в іншому випадку необхідно використовувати *medium-scale algorithm*. Використання *large-scale algorithm* можливе, якщо встановлено в опціях «on». Використовуйте *medium-scale algorithm* якщо в опціях встановлено «off».

Таблиця 27. Функції, що використовуються для medium-scale algorithm та large-scale

DerivativeCheck	Порівняння похідних (градієнтів цільової функції та обмежень) з похідними кінцевого розрахунку різниць
Diagnostics	Відображення діагностичної інформації про функцію, що буде мінімізована
DiffMaxChange	Максимальна зміна в змінних для кінцевого розрахунку різниць
DiffMinChange	Мінімальна зміна в змінних для кінцевого розрахунку різниць
Display	Рівні відображення. «*off*» не відображає вихідних даних, «*iter*» відображає результати кожної ітерації, «*notify*» виводить дані, якщо тільки функція не збігається, «*final*» - відображення лише кінцевого результату.
FunValCheck	Перевіряє, чи допустимі значення цільової функції. «*on*» - виводить попередження, якщо цільова функція дає значення, яке є *complex, Inf,* або *NaN. «off»* не виводить попереджень.'

GradObj	Градієнт цільової функції, що визначений користувачем. Ви маєте розрахувати градієнт для використання large-scale algorithm. Ця умова є додатковою для методу medium-scale.
MaxFunEvals	Максимальне число функціональних оцінок (розрахунків)
MaxIter	Максимальна кількість допустимих ітерацій
OutputFcn	Визначає, що функція оптимізації проходить кожну ітерацію
TolFun	Допуск завершення процесу для функціонального значення
TolCon	Допуск завершення процесу для порушення обмежень
TolX	Допуск завершення на *x*
TypicalX	Типічні значення *x*

Таблиця 28. Опції, що використовуються лише методом large-scale

Hessian	Якщо «on», то fmincon використовує *Hessian*, що визначений користувачем, або інформацію про *Hessian* (використовуючи *HessMult*) для цільової функції. Якщо «off» то *Hessian* апроксимізується з використанням кінцевих різниць.
HessMult	Функціональний дескриптор для *Hessian,* помножує функцію. Для задач *large-scale* ця функція розраховує добуток матриці *Hessian H*Y,* при цьому фактично не формується *H*. Функція має форму *W = hmfun(Hinfo,Y,p1,p2,...)* Де *Hinfo* та додаткові параметри *p1,p2,...* містять матриці, що використовуються для розрахунку *H*Y.* Перший аргумент має бути таким самим, як третій аргумент , що визначений цільовою функцією, наприклад *[f,g,Hinfo] = fun(x)* *Y* це матриця, що має таку ж кількість рядків, скільки невідомих в задачі. *W = H*Y*, хоча *H* не сформований явно. *fminunc* використовує *Hinfo,* щоб розрахувати попередню форму. Додаткові параметри *p1, p2*…можуть бути будь-якими додатковими параметрами, що необхідні *hmfun.* Зверніть увагу, *Hessian* має бути встановлено на “on” , щоб перейти від *fun* до *hmfun.*
HessPattern	Зразок розрідженості *Hessian* для скінченного розрахунку різниць. Якщо не зручно розраховувати розріджену матрицю *Hessian H* в *fun, large-scale* метод в *fmicon* може наблизити *H* через розрідженні кінцеві різниці (градієнта), при цьому забезпечується структура розрідженості *H* – через функцію *HessPattern*. В найгіршому випадку, якщо структура невідома, можна установити *HessPattern* як щільну матрицю, а повні кінцеві наближені будуть розраховані при кожній ітерації. Це досить трудомісткий спосіб для великих задач, тому доцільно визначити структуру розрідженості.
MaxPCGIter	Максимальне число PCG ітерацій
PrecondBandWidth	Верхня пропускна здатність попереднього формування для PCG. За замовчуванням, діагональне попереднє створення використовується (верхня пропускна спроможність 0). Для деяких задач збільшення пропускної спроможності призводить до зменшення кількості PCG ітерацій.
TolPCG	Допуск завершення для ітерацій PCG

Таблиця 29. Опції, що використовуються лише методом medium-scale

MaxSQPIter	Максимальна кількість ітерацій SQP
RelLinesrchBnd	Відносне число (дійсне невід'ємне скалярне значення) на лінії, що відшукує розмір кроку, який би повністю при зміщенні в x задовольняв $\lvert\Delta x(i)\rvert \le \text{relLineSrchBnd} \times max(\lvert x(i)\rvert, \lvert typicalx(i)\rvert)$. Ця опція забезпечує управління величиною зміщень в x для випадків, в яких кроки, що здійснюються, вважаються дуже великими.
RelLineSrchBndDuration	Число ітерацій, для яких границі, визначені в *RelLinesrchBnd* мають бути активними (значення за замовчкуванням 1).

Приклад

Знайти значення x, що мінімізує функцію $f(x) = -x_1 x_2 x_3$, починаючи зі значень $x = [10;10;10]$

При цьому обмеження становлять $0 \le x_1 + 2x_2 + 2x_3 \le 72$

1. По-перше, запишемо *M-file*, який надає функції скалярне значення при *x*.

```
function f = myfun(x)
f = -x(1) * x(2) * x(3);
```

2. Потім перепишемо обмеження як нерівності до констант.

$$-x_1 - 2x_2 - 2x_3 \le 0$$
$$x_1 + 2x_2 + 2x_3 \le 72$$

Так як обидва обмеження лінійні, сформуємо з них матрицю нерівностей $A*x \le b$, де

$$A = \begin{bmatrix} -1 & -2 & -2 \\ 1 & 2 & 2 \end{bmatrix} \text{ та } b = \begin{bmatrix} 0 \\ 72 \end{bmatrix}.$$

3. Далі визначаємо початкову точку та визиваємо підпрограму оптимізації

```
x0 = [10; 10; 10];    % Starting guess at the solution
[x,fval] = fmincon(@myfun,x0,A,b)
```

4. Після 66 функціональних оцінок (розрахунків) маємо рішення:

```
x =
    24.0000
    12.0000
    12.0000
```

При цьому значення функції

```
fval =
   -3.4560e+03
```

Оптимізація large-scale

Для того, щоб використовувати метод *large-scale*, необхідно:

- Забезпечити розрахунок градієнта в *fun*.
- Встановити *GradObj* на “on” в опціях.
- Визначити, що використовується лише один, але не два, з наведених обмежень:
  1. нижні та верхні границі;
  2. лінійні рівняння-обмеження (матриця *Aeq* не може мати більше рядків ніж колонок), *Aeq* звичайно розріджена.

Ви можете використовувати нерівності-обмеження з *large-scale algorithm*. Якщо наведені умови не зустрічаються, то необхідно використовувати *medium-scale algorithm*.

Функція *fmincon* виводить попередження, якщо не надано градієнт та опція *LargeScale* не встановлена на «off». *fmincon* дозволяє *g(x)* бути приблизним градієнтом, але використовувати цю опцію не рекомендується. Числова поведінка більшості методів оптимізації більш стійка, якщо використовується істинний градієнт.

Метод *large-scale* найбільш ефективний в *fmincon*, якщо матриця других похідних, тобто матриця *Hessian,* також розрахована. Проте оцінка (розрахунок) істинної матриці *H* не вимагається. Наприклад, якщо ви можете визначити структуру розрідженості *H* (з використанням опції *HessPattern*), то *fmincon* розраховує кінцеве наближення до *H(x).*

Якщо *x0* не визначено строго, функція *fmincon* визначає нову строго визначену відправну точку. Якщо компоненти *x* не мають верхніх (нижніх) меж, тоді *fmincon* визначає, що відповідні компоненти *ub* (або *lb)* мають бути встановлені як *Inf* (або *–Inf*). Це є альтернативою дуже великому додатному або дуже малому від'ємному числу.

fmincon знищує лінійно-залежні рядки в *Aeq,* проте цей процес включає матричні розклади, що повторюються, на множники і може бут коштовним, якщо є багато залежностей. Кожна ітерація включає розріджене рішення найменших квадратів з матрицею $\overline{A}_{eq} = A_{eq}^{T} R^{-T}$, де *RT* – це фактор *Cholesky* попереднього формування. Існує потенційний конфлікт між вибором ефективного попереднього формувача та мінімізацією в *Aeq.*

Оптимізація medium-scale

Можна отримати кращі числові результати, якщо визначати обмеження явно (наприклад *Aeq, beq*), ніж коли використовують неявне визначення обмежень (*lb, ub*). Якщо рівняння-обмеження задані та залежні рівняння знайдені та знищені в квадратичній підзадачі, то буде виведено '*dependent*' під *Procedures* (коли Ви робите запит на вихідні дані, при цьому *Display option* встановлена на *'iter'*). Залежні рівності знищуються лише втому випадку, коли вони не суперечливі. Якщо система рівностей не суперечлива, то підзадача може бути вирішена і '*infeasible'* виводиться під заголовком *Procedures.*

Алгоритм

Large-scale algorithm це метод значимості довіри півпростору, що оснований на внутрішньо-рефлексивному методі Ньютона. Кожна ітерація включає наближений розв'язок великої лінійної системи, з використанням методу PCG.

Medium-scale algorithm для *fmincon* використовує послідовне квадратичне програмування *(SQP).* В цьому методі функція вирішує підзадачі квадратичного програмування *(QP)* при кожній ітерації. Оцінка *Hessian* функції Лагранжа оновлена при кожній ітерації за допомогою використання формули *BFGS.* Пошук лінії виконується з використанням оціночної функції. Підзадача *QP* розв'язується з використанням активної стратегії множин.

Обмеження

Функція fmincon обробляє лише дійсні змінні. Цільова функція та обмеження мають бути неперервними. *fmincon* може надати лише локальні розв'язки. Якщо задача не має розв'язку, *fmincon* намагається мінімізувати максимальне значення обмеження. Цільова функція та обмеження мають бути дійсно оцінені, тобто вони не можуть надати комплексні значення.

Посилання

[1] Coleman, T.F. and Y. Li, "An Interior, Trust Region Approach for Nonlinear Minimization Subject to Bounds," SIAM Journal on Optimization, Vol. 6, pp. 418-445, 1996.

[2] Coleman, T.F. and Y. Li, "On the Convergence of Reflective Newton Methods for Large-Scale Nonlinear Minimization Subject to Bounds," Mathematical Programming, Vol. 67, Number 2, pp. 189-224, 1994.

Функція Quadprog

Вирішує задачі оптимізації для квадратичного програмування.

Рівняння

$$\min_x \frac{1}{2} x^T H_x + f^t x,$$

при цьому

$$A * x \le b$$
$$Aeq * x = beq,$$
$$lb \le x \le ub$$

де *H, A* і *Aeq* матриці, *a f, b, beq, lb, ub, x* – вектори.

Синтаксис

- *x = quadprog(H,f,A,b)*
- *x = quadprog(H,f,A,b,Aeq,beq)*
- *x = quadprog(H,f,A,b,Aeq,beq,lb,ub)*
- *x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)*
- *x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)*
- *[x,fval] = quadprog(...)*
- *[x,fval,exitflag] = quadprog(...)*
- *[x,fval,exitflag,output] = quadprog(...)*
- *[x,fval,exitflag,output,lambda] = quadprog(...)*

Опис

- *x = quadprog(H,f,A,b)* - знаходить вектор *x*, що мінімізує функцію квадратичного програмування, при цьому виконується обмеження $A*x \le b$.
- *x = quadprog(H,f,A,b,Aeq,beq)* - вирішує задачу квадратичного програмування, при цьому задовольняється додаткова умова у вигляді рівняння *Aeq*x = beq*.
- *x = quadprog(H,f,A,b,Aeq,beq,lb,ub)* – визначає верхнє та нижнє значення перемінної *x* так, щоб рішення знаходилося в діапазоні *lb <= x <= ub*.
- *x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)* - надає початкові значення *x0.*
- *x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options*) – мінімізує.
- *[x,fval] = quadprog(...)* – надає значення цільової функції *x.*
- *[x,fval,exitflag] = quadprog(...)* – надає значення *exitflag,* що описує вихідні умови *quadprog*.
- *[x,fval,exitflag,output] = quadprog(...)* – виводить інформацію про оптимізацію.
- *[x,fval,exitflag,output,lambda] = quadprog(...)* – надає лямбду у випадку, коли задача містить множники Лагранжа.

Вихідні дані

Функціональні параметри містять загальний опис параметрів , що повертаються *quadrpog*.

Таблиця 29. Функціонально-спеціалізовані деталі для exitflag, lambda та output

exitflag	Ціле число, що пояснює, чому виконання алгоритму завершилося. Подано список значень та відповідних причин закінчення алгоритму.	
	1	Функція збігається до розв'язку *x*
	3	Зміна в цільовій функції була меншою, за вказану межу
	4	Локальна мінімізуючи зміна була знайдена

	0	Кількість ітерацій перевищила встановлену норму
	-2	Задача не має рішення
	-3	Задача не має обмежень
	-4	Направлення пошуку не є направленням зменшення. Ніякого успіху при пошуку рішення не буде
	-7	Важливість напрямку пошуку стала незначною. Не можна досягти успіху.
lambda	Структура, що містить множники Лагранжа при розв'язку *x*	
	lower	Нижня границя *lb*
	upper	Верхня границя *ub*
	ineqlin	Лінійні нерівності
	eqlin	Лінійні рівності
output	Структура, що містить інформацію про оптимізацію	
	iterations	Кількість ітерацій
	algorithm	Алгоритм, що використовувався
	cgiterations	Кількість ітерацій PCG, (large-scale algorithm)
	firstorderopt	Міра оптимальності першого рівня (large-scale algorithm) Для large-scale задач з обмеженнями оптимальність першого рівня – норма безкінечності *v.*g*, де *v* визначено в полі обмежень, а *g* – градієнт. Для задач large-scale з лінійними рівняннями оптимальність першого рівня – 2 норми різниці, що пиведені визначеним обчисленням спряженого градієнта.

Опції

Використовуйте *optimset*, щоб установити або змінити значення цих опцій. Деякі опцій можуть бути використані для всіх алгоритмів. Деякі застосовуються лише для *large-scale algorithm*, деякі доцільні лише для *medium-scale algorithm*.

Large-scale algorithm

Використовуйте *large-scale algorithm*, якщо це можливо, встановлено «on». Використовуйте *medium-scale algorithm*, якщо встановлено «off». «on» - лише преференція.

Якщо задача має тільки верхні та нижні границі, тобто ніякі лінійні нерівності чи рівності не визначені – алгоритм значення за замовчуванням - *large-scale algorithm*. Або, якщо задача, що дана в *quadprog*, має лише лінійні рівності, тобто ніякі верхні та нижні границі та нерівності не визначені та число рівностей не більше за *x,* тоді алгоритм значень за замовчуванням - *large-scale algorithm*.

В інших випадках використовується *medium-scale algorithm.*

Таблиця 30. Опції, що використовуються двома алгоритмами

Diagnostics	Відображення діагностичної інформації про функцію, яка буде мінімізована
Display	Рівень виводу діагностики. «off» не показує кінцевого результату, «final» показує лише кінцевий результат.
MaxIter	Максимальна кількість дозволених ітерацій
TypicalIX	Типічні значення *x*

Таблиця 31. Опції, що використовуються лише для large-scale algorithm

HessMult	Множить функцію. Для структурованих задач *large-scale* ця функція знаходить добуток матриці *Hessia H*Y*, причому *H* фактично не визначається. Функція має форму *W = hmfun (Hinfo, Y)*

	Де *Hinfo* та можливі додаткові параметри містять матриці, що використовувалися для розрахунку *H*Y*.
MaxPCGIter	Максимальне число ітерацій.
PrecondBandWidth	Верхня пропускна спроможність попереднього формування для PCG. За замовчуванням діагональне попереднє формування використовується (верхня пропускна спроможність 0). Для деяких задач збільшення пропускної можливості приводить до зменшення кількості ітерацій.
TolFun	Використовується як критерій виходу для задач з простими верхніми та нижніми межами (*lb, ub*).
TolPCG	Допуск завершення на ітерації PCG. Використовується як критерій виходу для задач з обмеженнями-рівностями.
TolX	Допуск завершення для *x*.

Приклад

Знайти значення *x*, що мінімізує функцію:

$$f(x)=\frac{1}{2}x_1^2+x_2^2-x_1x_2-2x_1-6x_2,$$

при цьому:

$$\begin{aligned}&x_1+x_2\le 2\\&-x_1+2x_2\le 2\\&2x_1+x_2\le 3\\&x_1\ge 0, x_2\ge 0\end{aligned}$$

1. По-перше, функція може бути записана у матричному вигляді:

$$f(x)=\frac{1}{2}x^T Hx+f^T x,$$

де

$$H=\begin{bmatrix}1 & -1\\ -1 & 2\end{bmatrix},\ f=\begin{bmatrix}-2\\ -6\end{bmatrix},\ x=\begin{bmatrix}x_1\\ x_2\end{bmatrix}.$$

2. Введіть коефіцієнти цих матриць:

```
H = [1 -1; -1 2]
f = [-2; -6]
A = [1 1; -1 2; 2 1]
b = [2; 2; 3]
lb = zeros(2,1)
```

3. Далі введіть формулу для вирішення задачі квадратичного програмування:

```
[x,fval,exitflag,output,lambda] = quadprog(H,f,A,b,[],[],lb)
```

4. Це дозволить отримати таке рішення:

```
x =                          fval =          exitflag =

    0.6667
    1.3333                     -8.2222            1
```

```
output =

       iterations: 3
        algorithm: 'medium-scale: active-set'
    firstorderopt: []
     cgiterations: []
          message: 'Optimization terminated.'


lambda =

      lower: [2x1 double]
      upper: [2x1 double]
      eqlin: [0x1 double]
    ineqlin: [3x1 double]
```

Ненульові елементи вектора лямбда вказують на існування активних обмежень розв'язку. В цьому випадку перші та другі обмеження нерівностей – активні обмеження (тобто рішення знаходиться в межах їх обмежень). Для такої задачі всі нижні обмеження неактивні.

Найкращі числові значення можна отримати, якщо визначати рівності явно (тобто *Aeq, beq*), на відміну від неявного визначення (з використанням *lb, ub).*

Якщо компоненти вектора *x* не мають верхніх (або нижніх) меж, то *quadprog* визначає відповідні компоненти *ub (lb*) як *Inf* (на противагу дуже великому додатному або від'ємному (у випадку нижньої межі) числу.

Алгоритм

Large-scale algorithm – метод довірчої області підпростору, заснований на методі Ньютона. Кожна ітерація підносить до степеня приблизний розв'язок великої лінійної системи, використовуючи метод визначених спряжених градієнтів (PCG).

Quadprog використовує активний набір, який є також методом проекції. Він знаходить необхідний розв'язок при першому вирішення задачі лінійного програмування.

Посилання

[1] Coleman, T.F. and Y. Li, "A Reflective Newton Method for Minimizing a Quadratic Function Subject to Bounds on some of the Variables," SIAM Journal on Optimization, Vol. 6, Number 4, pp. 1040-1058, 1996.

[2] Gill, P. E. and W. Murray, and M.H. Wright, Practical Optimization, Academic Press, London, UK, 1981.

Функція Fminunc

Функція *fminunc* знаходить мінімум функції багатьох змінних без обмежень.

Рівняння

$$\min_{x} f(x),$$

де x - вектор, f (x) - функція, що повертає скаляр.

Синтаксис

x = fminunc(fun,x0)
x = fminunc(fun,x0,options)
[x,fval] = fminunc(...)
[x,fval,exitflag] = fminunc(...)
[x,fval,exitflag,output] = fminunc(...)
[x,fval,exitflag,output,grad] = fminunc(...)
[x,fval,exitflag,output,grad,hessian] = fminunc(...)

Опис

fminunc знаходить мінімум скалярної функції декількох змінних, починаючи з початкової оцінки. Загалом це відноситься до нелінійної оптимізації без обмежень.

x = fminunc(fun,x0) починається в точці x0 і прагне знайти локальний мінімум x функції, описаної в *fun*. x0 може бути скаляром, вектором або матрицею.

x = fminunc(fun,x0,options) мінімізує за допомогою опцій оптимізації, визначеними у структурі *options*. Використовуйте *optimset*, щоб встановити ці опції.

[x, fval] = fminunc (...) повертає в *fval* значення функції мети *fun* у рішенні x.

[x,fval,exitflag] = fminunc(...) повертає значення *exitflag*, який описує умову виходу.

[x,fval,exitflag,output] = fminunc(...) повертає структуру *output* , яка містить інформацію про оптимізацію.

[x,fval,exitflag,output,grad] = fminunc(...) повертає в *grad* значення градієнта функції *fun* у рішенні x.

[x,fval,exitflag,output,grad,hessian] = fminunc(...) повертає в *hessian* значення *Hessian* функції мети *fun* у рішенні x.

Вхідні аргументи

Аргументи функції містять загальні описи аргументів fminunc. Ця секція забезпечує визначені для функції деталі для *fun* і *options*:

fun - функція, яка буде мінімізована. *fun* – це функція, яка приймає вектор *x* і повертає скаляр *f*, функція мети, оцінена в x. Функція *fun* може бути визначена як ідентифікатор функції (*function handle*) для функції M-file:

x = fminunc(@myfun,x0)

де *myfun* - функція MATLAB типу *function f = myfun(x)*

f =... % Обчислює значення функції в x

fun може також бути ідентифікатор функції (function handle) для анонімної функції:

x = fminunc(@(x)norm(x)^2,x0);

Якщо градієнт функції *fun* може також бути обчислений, і функція *GradObj* 'включена' (‘on’), як встановлено:

options = optimset('GradObj','on'),

тоді функція *fun* повинна повернутися, в другий вихідний аргумент, градієнт значення g, векторі, в x. Відзначте, що, перевіряючи значення *nargout* функція може уникнути обчислення g, коли *fun* викликається з тільки з одним вихідним аргументом (у випадку, де алгоритм оптимізації потребує тільки значення *f*, але не *g*).

```
function [f,g] = myfun(x)
f = ... % Обчислює значення функції в x
if nargout > 1 % функція fun з двома вихідними аргументами
 g = ... % Обчислює градієнт, оцінений в x
end
```

Градієнт – це частинні похідні $\partial f / \partial x$ f в точці x. Таким чином, *i*-тий компонент *g* – це частинна похідна f відносно i-го компоненту x.

Якщо матриця Гессе (*Hessian matrix*) може також бути обчислена, і опція Hessian 'включена', тобто, *options = optimset ('Hessian','on')*, то функція fun повинна повернути значення *Hessian* H, симетричну матрицю, в x в третьому вихідному аргументі. Зауважте, що, перевіряючи значення *nargout* Ви можете уникнути обчислення H, коли *fun* має тільки один або два вихідні аргументи (у випадку, коли алгоритм оптимізації потребує тільки значення f та g, але не H).

```
function [f,g,H] = myfun(x)
f = ... % Обчислює значення функції мети в x
if nargout > 1 % fun called with two output arguments
 g = ... % Градієнт функції, оцінений в x
 if nargout > 2
 H = ... % Hessian оцінена в x
 end
end
```

Матриця Гессе - друга матриця частинних похідних *f* в точці *x*. Таким чином, *(i,j)*-ий компонент *H* - друга часткова похідна f відносно xi та xj, $\partial^2 f / \partial x_i \partial x_j$. Матриця Гессе - за визначенням симетрична матриця.

Вихідні аргументи

Аргументи функції містять загальні описи аргументів, повернених *fminunc*. Ця секція забезпечує визначені для функції деталі для *exitflag* і *output*:

Таблиця 32. Аргументи функції для fminunc

exitflag	Ціле число, що ідентифікує причину, за якою алгоритм закінчився. Наступне виводить список значень *exitflag* і відповідних причин, за якими алгоритм закінчився.	
	1	Величина градієнта менша, ніж вказана допустима.
	2	Зміна x була менша, ніж вказана допустима.
	3	Зміна значення функції мети була менша, ніж вказана допустима.
	0	Кількість ітерацій перевищило *options. MaxIter* або кількість оцінок функції перевищила *options. FunEvals*.
	-1	Алгоритм був закінчений вихідною функцією.
	-2	Пошук лінії не може знайти прийнятний точку по поточному напряму пошуку.
grad	*Градієнт в x*	
hessian	*Hessian в x*	
output	Структура, що містить інформацію про оптимізацію. Області структури –це:	
	iterations	Кількість зроблених ітерацій

	funcCount	Кількість оцінок функції
	algorithm	Алгоритм, що використовувався
	cgiterations	Кількість ітерацій PCG (тільки великомасштабний алгоритм)
	stepsize	Завершальний крок (алгоритм тільки середнього масштабу)

Hessian

fminunc обчислює вихідний аргумент *hessian* таким чином:

❖ Використовуючи алгоритм середнього масштабу, функція обчислює кінцево-різницеве наближення до *Hessian* при використанні *x*.

- Градієнт *grad*, якщо забезпечується його визначення.
- Функція мети *fun* якщо не забезпечується визначення градієнта.

❖ Використовуючи великомасштабний алгоритм, функція використовує:

- *options.Hessian* обчислювати Hessian в *x*.
- кінцево-різницеве наближення до *Hessian* в *x*.

Options

fminunc використовує ці варіанти оптимізації. Деякі опції застосовуються до всіх алгоритмів, деякі доречно застосовувати тільки тоді, коли Ви використовуєте великомасштабний алгоритм, а інші - тоді, коли Ви використовуєте алгоритм середнього масштабу. Ви можете використовувати *optimset*, щоб встановити або змінити значення цих полів структури *options*.

Опція *LargeScale* надає перевагу, який алгоритм використовувати. Це - тільки перевага, тому що певні умови повинні бути виконані, щоб використовувати великомасштабний алгоритм. Для *fminunc*, Ви повинні забезпечити градієнт (в іншому випадку використовуйте алгоритм середнього масштабу

LargeScale - Використовуйте *LargeScale* алгоритм, якщо встановлено в 'on'. Використовуйте алгоритм середнього масштабу, коли встановлено в 'off'.

LargeScale алгоритм та алгоритм Medium-Scale

Ці опції використовуються, як великомасштабним, так і середньомасштабним алгоритмом:

Таблиця 33. Опції, які використовуються для обох алгоритмів

DerivativeCheck	Порівняєте похідні, що визначаються користувачем (градієнт), з кінцево-різницевими похідними.
Diagnostics	Покажіть діагностичну інформацію про функцію, яка буде мінімізована.
DiffMaxChange	Максимальна зміна змінних для кінцевої різниці.
DiffMinChange	Мінімальна зміна змінних для кінцевої різниці.
Display	Рівень 'off' не показує ніякого випуску; 'iter' показує випуск при кожній ітерації; 'notify' показує випуск, тільки якщо функція не сходиться; 'final' (default) показує тільки кінцевий випуск.
FunValCheck	Перевірте, чи є значення функції мети дійсними . 'on' показує попередження, коли функція мети повертає значення, яке є складним або *NaN. 'off' (за замовчуванням)* не показує ніякого попередження.
GradObj	Градієнт для функції мети, яку Ви визначаєте.
MaxFunEvals	Максимальна кількість дозволених оцінок функції.
MaxIter	Максимальна кількість дозволених ітерацій.
OutputFcn	Визначає визначену користувачем функцію, яку функція оптимізації викликає кожної ітерації.
TolFun	Завершення допустимого відхилення значення функції.
TolX	Завершення допустимого відхилення x.

TypicalX	Типові значення x.

LargeScale алгоритм

Таблиця 34. Опції, які використовуються тільки LargeScale алгоритмом

Hessian	Якщо *'on'* ('включено'), *fminunc* використовує визначену користувачем *Hessian* (що визначено в *fun*), або інформацію. *Hessian* (використовуючи *Hessmult*), для функції мети. Якщо *'off'* ('вимкнено'), *fminunc* наближає *the Hessian*, використовуючи кінцеві відмінності.
HessMult	Ідентифікатор функції для збільшеної функції Hessian. Для великомасштабних структурованих проблем, ця функція обчислює добуток матриці Гессе *H*Y*, фактично не формуючи *H*. Функція має форму: *W = hmfun(Hinfo,Y,p1,p2,...)* де *Hinfo* та можливо додаткові параметри p1, *p2*... містять матриці, що використовуються для обчислення *H*Y*. Перший аргумент повинен бути таким самим, як третій аргумент, повернений функцією мети fun, наприклад: *[f,g,Hinfo] = fun(x)* Y - матриця, яка має таку кількість рядків, скільки сторін (аспектів) має проблема. *W = H*Y,* хоча H не сформований явно. *fminunc* використовує *Hinfo*, щоб обчислити попередню умову. Додаткові параметри p1, p2... можуть бути будь-якими додатковими параметрами, потрібними *hmfun*. Відзначте, що *'Hessian'* повинна бути встановлена *'on'* для *Hinfo* для того, щоб пройти від *fun* до *hmfun*.
HessPattern	Небагаточисельний зразок *Hessian* для кінцевої різниці. Якщо незручно обчислити розріджену матрицю Гессе *H* в *fun*, великомасштабний метод в *fminunc* може наблизитися до *H* через розріджені кінцеві відмінності (градієнта), забезпечив структуру розрідженості *H* - тобто, місцеположення ненолей - підставляються як значення для *Hesspattern*. У гіршому разі, якщо структура невідома, Ви можете змусити *Hesspattern* бути щільною матрицею, і повне наближення кінцевої відмінності обчислюється при кожній ітерації (це - *default*). Це може коштувати дуже дорого для великих проблем, тому зазвичай варто витратити зусилля, щоб визначити структуру розсіяності.
MaxPCGIter	Максимальне число ітерацій *PCG* (*preconditioned conjugate gradient*)
PrecondBandWidth	Верхня смуга пропускання попередньої умови для *PCG*. За умовчанням, використовується діагональне перетворення умов (верхня смуга пропускання 0). Для деяких проблем, збільшуючи смугу пропускання, зменшуємо число ітерацій *PCG*.
TolPCG	Завершення допустимого відхилення ітерацій *PCG*.

Алгоритм Medium-Scale

Таблиця 35. Опції, які використовуються тільки MediumScale алгоритмом

HessUpdate	Метод для того, щоб вибирати напрям пошуку в алгоритмі Квазі-Ньютона (*Quasi-Newton algorithm*). Вибори такі: *'bfgs', 'dfp', 'steepdesc'*
InitialHessMatrix	Початкова матриця квазі-Ньютона (*quasi-Newton matrix*). Ця опція доступна, тільки якщо Ви встановлюєте *InitialHessType* в *'user-supplied'* ('забезпечений користувачем'). В цьому разі, Ви можете встановити *InitialHessMatrix* в одне з наступного: • скаляр - початкова матриця, яка є скаляром • вектор - початкова матриця - діагональна матриця із записами вектора на діагоналі.

InitialHessType	Початковий тип матриці квазі-Ньютона. Варіанти такі: • *'identity'* ('ідентичність') • *'scaled-identity'* ('ідентичність масштабу') • *'user-supplied'* ('поставляється користувачем')

Приклад

Мінімізуйте функцію: $f(x) = 3x^2 + 2x_1x_2 + x_2^2$

1. Щоб використати M-file, створіть файл myfun.m.

```
function f = myfun(x)
f = 3*x(1)^2 + 2*x(1)*x(2) + x(2)^2;     % Cost function
```

2. Потім викличте *fminunc*, щоб знайти мінімум myfun біля [1,1].

```
>> x0 = [1,1];
[x,fval] = fminunc(@myfun,x0)
```

3. Після декількох повторень, повернено рішення, *x*, і *fval* значення функції в *x*.

```
x =

  1.0e-006 *

    0.2541   -0.2029


fval =

  1.3173e-013
```

4. Щоб мінімізувати цю функцію з градієнтом, змініть M-file myfun.m, таким чином, щоб градієнт був другим вихідним аргументом.

```
function [f,g] = myfun(x)
f = 3*x(1)^2 + 2*x(1)*x(2) + x(2)^2;     % Cost function
if nargout > 1
   g(1) = 6*x(1)+2*x(2);
   g(2) = 2*x(1)+2*x(2);
end

>> options = optimset('GradObj','on');
x0 = [1,1];
[x,fval] = fminunc(@myfun,x0,options)
```

5. Після декількох повторень повернено рішення, *x*, і *fval*, значення функції в x.

```
x =

  1.0e-015 *

    0.2220   -0.2220


fval =

  9.8608e-032
```

6. Для того, щоб мінімізувати функцію $f(x) = \sin(x) + 3$, використовуємо функцію:

```
>> f = @(x)sin(x)+3;
x = fminunc(f,4)
    x =

    4.7124
```

fminunc не є привілейованим вибором для того, щоб вирішити проблеми, які є сумами квадратів, тобто, такої форми:

$$\min_x(f(x)) = f_1(x)^2 + f_2(x)^2 + f_3(x)^2 + ... + f_m(x)^2$$

Замість цього використовуйте функцію *lsqnonlin*, яка була оптимізована для проблем цієї форми. Щоб використовувати великомасштабний метод, Ви повинні забезпечити градієнт в fun (і встановити опцію *Gradobj 'on'*, використовуючи *optimset*). Попередження дається, якщо ніякого градієнта не має, а опція *LargeScale* не 'вимкнена'.

Алгоритми

LargeScale оптимізація. За умовчанням *fminunc* обирає великомасштабний алгоритм, якщо користувач встановлює градієнт в *fun* (і опція *Gradobj* встановлена *'on'*, використовуючи *optimset*). Кожна ітерація залучає приблизне вирішення великої лінійної системи, використовуючи метод передобумовлених зв'язаних градієнтів (*PCG*).

Оптимізація Medium-Scale. *fminunc*, коли опція *LargeScale* встановлено 'вимкнено' за допомогою optimset, використовує BFGS метод Квазі-Ньютона із змішаною квадратною та кубічною процедурою пошуку лінії. Ви можете вибрати найкрутіший метод спуску, встановлюючи *HessUpdate* в *'steepdesc'* (а опцію *LargeScale* 'вимкнено'), хоча цього не рекомендують.

Обмеження

Функція, яка буде мінімізована, повинна бути безперервною. *fminunc* може дати тільки локальні рішення. *fminunc* тільки мінімізує по реальних числах, тобто, x повинен складатися тільки з реальних чисел, і *f(x)* повинен повернути тільки реальні числа. Коли x має складні змінні, вони повинні бути роздроблені в реальні і уявні частини.

Щоб використовувати великомасштабний алгоритм, користувач повинен обчислити градієнт в fun (*GradObj* повинен бути встановлений *'on'* в опціях).

Посилання

[1] Broyden, C.G., "The Convergence of a Class of Double-Rank Minimization Algorithms," Journal Inst. Math. Applic., Vol. 6, pp. 76-90, 1970.

[2] Coleman, T.F. and Y. Li, "An Interior, Trust Region Approach for Nonlinear Minimization Subject to Bounds," SIAM Journal on Optimization, Vol. 6, pp. 418-445, 1996.

[3] Coleman, T.F. and Y. Li, "On the Convergence of Reflective Newton Methods for Large-Scale Nonlinear Minimization Subject to Bounds," Mathematical Programming, Vol. 67, Number 2, pp. 189-224, 1994.

[4] Davidon, W.C., "Variable Metric Method for Minimization," A.E.C. Research and Development Report, ANL-5990, 1959.

[5] Fletcher, R.,"A New Approach to Variable Metric Algorithms," Computer Journal, Vol. 13, pp. 317-322, 1970.

[6] Fletcher, R., "Practical Methods of Optimization," Vol. 1, Unconstrained Optimization, John Wiley and Sons, 1980.

[7] Fletcher, R. and M.J.D. Powell, "A Rapidly Convergent Descent Method for Minimization," Computer Journal, Vol. 6, pp. 163-168, 1963.

[8] Goldfarb, D., "A Family of Variable Metric Updates Derived by Variational Means," Mathematics of Computing, Vol. 24, pp. 23-26, 1970.

Функція Fminsearch

Мінімізує функцію декількох змінних.

Синтаксис

- *x = fminsearch(fun,x0)*
- *x = fminsearch(fun,x0,options)*
- *[x,fval] = fminsearch(...)*
- *[x,fval,exitflag] = fminsearch(...)*
- *[x,fval,exitflag,output] = fminsearch(...)*

Опис

- *fminsearch* знаходить мінімум скалярної функції декількох змінних, починаючи з початкової оцінки. Загалом це відноситься до нелінійної оптимізації без обмежень.
- *x = fminsearch(fun,x0)* починається в точці x0 і знаходить місцевий мінімум x функції, що описана в *fun*. *x0* може бути скаляром, вектором, або матрицею. *Fun* - це ідентифікатор функції.
- *x = fminsearch(fun,x0,options)* мінімізує з параметрами оптимізації, визначеними у структурі *options*. Ви можете визначити ці параметри, використовуючи функцію *optimset*. *fminsearch* використовує такі поля структури *options*:

Таблиця 36. Опції, які використовуються для optimset. fminsearch

Display	Рівень показу. 'off' не показує ніякого випуску; 'iter' показує випуск при кожній ітерації; 'final' показує тільки кінцевий випуск; 'notify' (default) показує випуск, тільки якщо функція не сходиться.
FunValCheck	Перевіряє, чи дійсні значення функції мети. 'on' показує попередження, коли функція мети повертає значення, яке є складним, Inf or NaN. 'off' (the default) не показує ніякого попередження.
MaxFunEvals	Максимальна кількість дозволених оцінок функції.
MaxIter	Максимальна кількість дозволених ітерацій
OutputFcn	Визначена користувачем функція, яку функція оптимізації викликає при кожній ітерації.
TolFun	Завершення допустимого відхилення значень функції.
TolX	Завершення допустимого відхилення x.

- *[x,fval] = fminsearch(...)* повертає в fval значення функції мети *fun* у рішенні x.
- *[x,fval,exitflag] = fminsearch(...)* повертає значення exitflag, яке описує вихідну умову *fminsearch*:

Таблиця 36. Вихідні значення функції fminsearch

1	fminsearch сходиться дорішення x.
0	Максимальна кількість зроблених оцінок або ітерацій функції.
-1	Алгоритм був закінчений вихідною функцією.

- *[x,fval,exitflag,output] = fminsearch(...)* повертає структуру випуску, яка містить інформацію про оптимізацію:

Таблиця 36. Вихідні відомості про результати оптимізації fminsearch

output.algorithm	Алгоритм, що використовується
output.funcCount	Кількість оцінок функції
output.iterations	Кількість ітерацій
output.message	Вихід з повідомлення

Аргументи

fun - функція, яка буде мінімізована. Вона приймає вхідний x і повертає скаляр f, функція мети, оцінена в x. Функція *fun* може бути визначена як опис функції для функції M-file.

x = fminsearch(@myfun, x0)

де myfun - функція M-file, типу function f = myfun(x)

f = ... % Compute function value at x або як опис функції для анонімної функції, типу

x = fminsearch(@(x)sin(x^2), x0).

Приклад

Класичний випробувальний приклад для багатовимірної мінімізації - бананова функція (функція Розенброка) - Rosenbrock banana function:

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Мінімум знаходиться в точці (1,1) і має значення 0. Традиційна відправна точка - (-1.2,1). Анонімна функція, показана тут визначає функцію і повертає ідентифікатор функції, що названа бананом:

```
>> banana = @(x)100*(x(2)-x(1)^2)^2+(1-x(1))^2;
>> [x,fval] = fminsearch(banana,[-1.2, 1])

x =

    1.0000    1.0000


fval =

  8.1777e-010
```

Це вказує на те, що мінімальне значення було знайдене принаймні до чотирьох десяткових місць зі значенням біля нуля.

Алгоритм

fminsearch використовує симплексний метод пошуку [1]. Це - прямий метод пошуку, який не використовує числові або аналітичні градієнти. Якщо n - довжина x, симплекс в n-розширеному місці характеризується n+1 відмінними векторами, які є його вершинами. У двох місцях, симплекс - трикутник; у трьох місцях, це - піраміда. У кожному кроці пошуку, генерується нова точка в або біля поточного симплексу. Значення функції в новій точці порівнюється зі значеннями функції у вершинах симплексу і, зазвичай, одна з вершин замінюється новою точкою, даючи новий симплекс. Цей крок повторюється, поки діаметр симплексу не менше ніж вказана допустимість.

Обмеження

fminsearch може часто поводитися з неоднорідністю, особливо якщо це не відбувається біля рішення. fminsearch може дати тільки локальні рішення. *fminsearch* тільки мінімізує по реальних числах, тобто, x повинен складатися тільки з реальних чисел і $f(x)$ повинна повернути тільки реальні числа. Коли x має складні змінні, вони повинні бути роздроблені на реальні і уявні частини.

Посилання

[1] Lagarias, J.C., J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," SIAM Journal of Optimization, Vol. 9 Number 1, pp. 112-147, 1998.

Функція Fgoalattain

Вирішите багатоцільову проблему досягнення мети

Рівняння

$$\min_{x,y} F(x) - w*y <= goal,$$
$$c(x)<=b,$$
$$ceq(x)=0,$$
$$Aeq*x=beq,$$
$$lb<=x<=ub,$$

де x, вага, мета, b, beq, lb, і ub - вектори, A, і Aeq - matrices, і c (x), ceq (x), і F (x) - функції ті вектори повернення. F (x), c (x), і ceq (x) може бути нелінійними функціями.

Синтаксис

- *x = fgoalattain (fun,x0,goal,weight)*
- *x = fgoalattain (fun,x0,goal,weight,A,b)*
- *x = fgoalattain (fun,x0,goal,weight,A,b,Aeq,beq)*
- *x = fgoalattain (fun,x0,goal,weight,A,b,Aeq,beq,lb,ub)*
- *x = fgoalattain (fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon)*
- *x = fgoalattain (fun,x0,goal,weight,A,b,Aeq,beq,lb,ub, onlcon,... options)*
- *[x, fval] = fgoalattain (...)*
- *[x, fval, attainfactor] = fgoalattain (...)*
- *[x, fval, attainfactor, exitflag] = fgoalattain (...)*
- *[x, fval, attainfactor, exitflag, output] = fgoalattain (...)*
- *[x,fval,attainfactor,exitflag,output,lambda] = fgoalattain (...)*

Опис

fgoalattain *вирішує проблему досягнення мети, що є одним формулюванням щоб мінімізувати багатоцільову задачу оптимізації.*

- *x = fgoalattain (fun,x0,goal,weight) забезпечує досягнення мети для функції, яка має назву fun, , вона називається goal, змінюючи x, починаючи з x0, з ваги, яка визначена як weight.*
- *x = fgoalattain (fun,x0,goal,weight,A,b) вирішує проблему досягнення мети, підлеглу лінійним нерівностям A*x <= b.*
- *x = fgoalattain (fun,x0,goal,weight,A,b,Aeq,beq) вирішує проблему досягнення мети, підпорядковану лінійним рівностям Aeq*x = beq також. Встановіть A= [] і b = [], якщо ніякі нерівності не існують.*
- *x = fgoalattain (fun,x0,goal,weight,A,b,Aeq,beq,lb,ub) визначає ряд більше нижніх і верхніх границь для змінних задачі в x, так, щоб рішення перебувало завжди у діапазоні lb <= x <= ub.*
- *x = fgoalattain (fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon) піддає проблему досягнення мети нелінійним нерівностям c (x), або нелінійним обмеженням рівності ceq (x) визначеної в nonlcon. fgoalattain оптимізує так що c (x) <= 0 і ceq (x) = 0. Встановіть lb = [] і/або ub = [], якщо ніякі границі не існують.*
- *x = fgoalattain (fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon... options) мінімізує з варіантами оптимізації, які визначені у варіантах структури. Використовуйте optimset, щоб установити ці варіанти.*
- *[x, fval] = fgoalattain (...) повертає значення функції мети, обчисленої fun при знаходженні рішення x.*
- *[x, fval, attainfactor] = fgoalattain (...) повертає фактор досягнення при рішенні x.*
- *[x, fval, attainfactor, exitflag] = fgoalattain (...) повертає значення exitflag, що описується вихідна умова fgoalattain.*

- *[x, fval, attainfactor, exitflag, output] = fgoalattain (...) повертає структуру output (випуску), що містить інформацію про оптимізацію.*
- *[x,fval,attainfactor,exitflag,output,lambda] = fgoalattain (...) повертає структуру лямбди, області(поля) яке містить множники Лагранжа Lagrange при рішенні x.*

Вхідні Аргументи

Аргументи Функції містять загальні описи аргументів, які передаються у fgoalattain. Ця секція забезпечує певні для функції специфічні деталі для fun, goal, nonlcon, options, and weight.

Fun - функція, що буде мінімізована. *Fun* - функція, що приймає вектор *x* і повертає вектор *F*, функції мети, оцінені в *x*. Функція fun може бути визначена як основна функція для функції M -file.

x = fgoalattain (@myfun,x0,goal,weight)
де myfun - функція MATLAB, типу
функція F = myfun (x)
F =... % Обчислюють значення функції в x.
fun може також бути основною для будь-якої функції.
*x = fgoalattain (@(x)sin(x.*x),x0,goal,weight);*

Щоб зробити функцію мети, майже наближену до значення мети, (тобто, не більше ніж, не менше ніж) не використовують *optimset*, щоб установити *GoalsExactAchieve* у числове значення, наближене до значення мети. Ці об'єктивності повинні бути розділені на перші елементи вектора *F*, який повернутаєтьсяфункцією fun .

Якщо градієнт функції мети може також бути обчислений, і *GradObj* функція є 'on', , як встановлено *options = optimset('GradObj','on'),* тоді функція *fun* повинна повернутися, до другого *output*(вихідного) аргументу, градієнт зі значенням *G*, матриця, в *x*. Відзначте, що, перевіряючи значення *nargout* функція може уникнути обчислення *G*, коли *fun* називають *only one output argument* (у випадку, коли алгоритм оптимізації має потребу в значенні).

```
function [F,G] = myfun(x)
F = ... % Compute the function values at x
if nargout > 1 % Two output arguments
 G = ... % Gradients evaluated at x
end
```

Градієнт складається із частинних похідних *dF/dx* кожного з *F* у пункті *x*. Якщо *F* - вектор довжиною *m*, а x має довжину *n*, де *n* - довжина *x0*, тоді градієнт *G of F(x)* є *n-на-m* матриця, де *G(i,j)* - частинна похідна *F(j)* відносно *x(i)* (тобто , j^{th} стовпчик *G* - градієнт j^{th} функції мети *F (j)*).

Goal - вектор значень, яких намагаються досягти. Вектор – має ту саму довжину, що і номер objectives *F*, що повертається *fun*. *fgoalattain* намагається мінімізувати значення у векторі *F*, щоб досягти потрібного значення, яке задане за допомогою *goal*.

Nonlcon - функція, що обчислює нелінійні обмеження нерівності *c(x) <= 0* і нелінійні обмеження рівності *ceq(x) = 0*. Функція *nonlcon* приймає вектор *x* і повертає два вектори *c* і *ceq*. Вектор *c* містить нелінійні нерівності, оцінені в *x*, і *ceq* містить нелінійні рівності, оцінені в *x*. Функція nonlcon може бути визначена як керуюча функція.

x = fgoalattain(@myfun,x0,goal,weight,A,b,Aeq,beq,... lb,ub,@mycon)

де *mycon* - функція MATLAB, типу:

```
function [c,ceq] = mycon(x)
c = ... % compute nonlinear inequalities at x.
ceq = ... % compute nonlinear equalities at x.
```

Якщо градієнти обмежень можуть також бути обчислені, і функція *GradConstr* is 'on', як встановлено *options = optimset('GradConstr','on'),* тоді функція nonlcon повинна також повернутися, до третього і четвертого вихідний аргументів, градієнт *c(x)*, і *Ceq*, градієнт *ceq(x)*. Відзначте, що, перевіряючи значення *nargout* функція може уникнути обчислювань

GC та *GCeq*, коли *nonlcon* називають - тільки із двома вихідними аргументами (у випадку, де алгоритм оптимізації має потребу тільки в значеннях *c* і *ceq*, а не в *GC* та *GCeq*).

```
function [c,ceq,GC,GCeq] = mycon(x)
c = ... % Nonlinear inequalities at x
ceq = ... % Nonlinear equalities at x
if nargout > 2 % Nonlcon called with 4 outputs
 GC = ... % Gradients of the inequalities
 GCeq = ... % Gradients of the equalities
end
```

Якщо nonlcon повертає вектор c з *m* компонентами та *x* має довжину *n*, де *n* - довжина *x0*, то градієнт *GC* з *c(x)* – є *n-на-m* матриця, де *GC(i, j)* - частинна похідна *c(j)* відносно *x(i)* (тобто, j^{th} стовпчик *GC* - градієнт j^{th} з обмеженими нерівностями *c(j)*). Аналогічно, якщо *ceq* має *p* компонентів, градієнт, *GCeq ceq (x)* є *n-на-p* матрицею, де *GCeq(i, j)* - частинна похідна *ceq(j)* відносно *x(i)* (тобто, j^{th} стовпчик *GCeq* - градієнт j^{th} з обмеженням рівності *ceq(j)*).

Options - функції забезпечують певні для функції деталі за їх значеннями.

Weight - вектор ваг,потрібен для того щоб управляти передосягненням цілей в *fgoalattain*. Коли значення *goal* (мети) всі відмінними від нуля, то щоб гарантувати той самий відсоток нижче- або вище активних цілей, встановіть функцію ваги в abs (goal). (Активні цілі - це набір цілей, які є бар'єрами до подальшого вдосконалення цілей при рішенні.)

Відзначте, що установка компонента вектора ваги до нуля змусить відповідне обмеження мети розглянути як тверде обмеження (hard constraint) обмеження, а не як обмеження мети. Кращий метод, для встановлення твердого обмеження повинен використовувати вхідний аргумент nonlcon.

Коли у функції ваги weight є позитивна, *fgoalattain* приділяє більшу увагу значенню мети. Щоб зробити функцію мети майже наближеною до значення мети, використовуйте функцію *GoalsExactAchieve* і ставте ту мету як перший елемент вектора, оберненого за допомогою *fun*.

Вихідні Аргументи

Аргументи Функції містять загальні описи аргументів, повернених *fgoalattain*. Ця секція забезпечує певні для функції деталі для *attainfactor*, *exitflag*, *lambda*, and *output*:

attainfactor

Кількість понад - або нижче досягнення цілей. Якщо *attainfactor* негативний, мета буде передосягнута(); якщо *attainfactor* позитивний нижче досягнутих.

exitflag

Ціле число, що ідентифікує причину закінчення алгоритму. Наступні списки значень *exitflag* і відповідних причин закінчення алгоритму.

1 Функція сходилася до рішень x.

4 Величина напрямку пошуку менше ніж зазначене порушення похибки й обмеження менше ніж options (функції). TolCon

5 Величина спрямованої похідної менше ніж зазначене порушення похибки й обмеження менше ніж options (функції).

0 Число повторень перевищило options. Maxiter або число(номер) оцінок функції перевищили options . FunEvals

-1 Алгоритм був закінчений функцією output.

-2 Ніякий здійсненний пункт не був знайдений.

lambda (лямбда)

Структура, що містить множники ЛагранжаLagrange при рішенні x (відділений типом обмеження). Області(поля) структури:

Lower(нижче)	Lower bounds lb
Upper (верхній)	Upper bounds ub
Ineqlin(Лінійні нерівності)	Linear inequalities

Eqlin (Лінійні рівності) Linear equalities
Ineqnonlin(Нелінійні нерівності) Nonlinear inequalities
Eqnonlin(Нелінійні рівності) Nonlinear equalities
Структура, що містить інформацію про оптимізацію. Області(поля) структури :
iterations (повторення) Число взятих повторень
funcCount Число алгоритму оцінок функції
algorithm(алгоритм) Використаний алгоритм

Функції (Options)

Функції оптимізації, використані *fgoalattain*. Ви можете використовувати optimset, щоб установити або змінити значення цих областей (полів) у функції структурі функцій.

DerivativeCheck

Зрівняєте похідні, що поставляються користувачем, (градієнти мети або обмежень) до кінцевих диференційованих похідних.

Діагностика(Diagnostics)

Показує діагностичну інформацію про функцію, що буде мінімізована або вирішена.

DiffMaxChange

Максимальна зміна у змінних для градієнтів кінцевого розходження.

DiffMinChange

Мінімальна зміна у змінних для градієнтів кінцевого розходження.

Показ(Display)

Рівень показу. 'off' displays – нема випуску, результату (output); 'iter' displays показує випуск при кожному повторенні; 'notify' (зареєструйте) випуск, тільки якщо функція не сходиться; 'final'(заключний) показує тільки заключний результат.

FunValCheck

Перевіряє, чи дійсні функція мети й значення обмежень. 'on' displays (показує) попередження, коли функція мети або обмеження повертають значення, що є *complex, Inf, або NaN*. 'off' displays без попередження.

GoalsExactAchieve

Визначає число цілей, для яких це потрібно для об'єктивної *fun* зрівнятися меті мети. Такі мети повинні бути розділені в перші небагато елементів *F*.

GradConstr

Градієнт для обмежень, визначений певним користувачем.

GradObj

Градієнт для функції мети, визначений користувачем.

MaxFunEvals

Максимальне число дозволених оцінок функції.

Maxiter

Максимальне число дозволених повторень.

MeritFunction

Використання мети досягнення/минимакс використовує функцію якщо встановлено в *'multiobj'*. Використовуйте функцію *fmincon* якщо встановлено в *'singleobj '*.

OutputFcn

Визначає певну функцію використання, якщо функція оптимізації заходить у кожне повторення.

RelLineSrchBnd

Relative bound (реальне ненегативне скалярне значення) на довжині кроку пошуку лінії такий, що задовольняє повний зсув в *x*. Ця опція забезпечує контроль(керування) над величиною зсувів в *x* для випадків, у яких *solver* уживає заходи, які розглядаються як завеликі.

RelLineSrchBndDuration

Число повторень, для яких *bound*, *specified* в *RelLineSrchBnd* та повинно бути активним.

TolCon

Завершення на порушенні обмеження.

TolFun

Похибка завершення на значенні функції.

TolX

Завершення на *x*.

TypicalX

Типові значення *x*. Довжина вектора дорівнює числу елементів в x0, відправній точці.

Приклад

Розглянемо лыныйну систему рівнянь:

$$\begin{aligned} x &= (A + BKC)x + Bu \\ y &= Cx \end{aligned}.$$

Вводимо вхідні матриці та встановлюємо вхідні параметри оптимізації:

$$A = \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & -2 & 10 \\ 0 & 1 & -2 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ -2 & 2 \\ 0 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Визначаємо значення, яке ми повинні отриати:

$$goal = [-5 \quad -3 \quad -1].$$

1. Вводимо вхідні матриці:

```
A =  [ -0.5  0  0;  0  -2  10;  0  1  -2 ];

B =  [ 1  0;  -2  2;  0  1 ];

C =  [ 1  0  0;  0  0  1 ];
```

2. Визначаємо ціль:

```
goal = [-5, -3, -1]

goal =

    -5    -3    -1
```

```
weight = abs(goal)

weight =

     5     3     1
```

3. Встановлюємо початкове значення:

```
x0 = [ -1 -1; -1 -1];
```

4. Вказуємо ліву та праву границю оптимізації:

```
lb = repmat(-4,size(x0))
ub = repmat(4,size(x0))
```

```
ub =

     4     4
     4     4
```

```
lb =

    -4    -4
    -4    -4
```

5. Розвязок системи рівнянь:

```
options = optimset('Display','iter');
```

```
eigfun = @(x) sort(eig(A+B*x*C))

eigfun =

    @(x) sort(eig(A+B*x*C))
```

6. Детальні відомості про процес оптимізації:

```
[x,fval,attainfactor,exitflag,output,lambda] = ...
        fgoalattain(eigfun,x0,goal,weight,[],[],[],[],lb,ub,[],options);


                    Attainment                  Directional
 Iter   F-count       factor      Step-size     derivative    Procedure
    0       6        1.88521
    1      12          1.061            1            1.03
    2      18         0.4211            1          -0.679
    3      24       -0.06352            1          -0.523    Hessian modified
    4      30        -0.1571            1          -0.053    Hessian modified twice
    5      36        -0.3489            1          -0.133
    6      42        -0.3643            1        -0.00768    Hessian modified
    7      48        -0.3645            1      -4.25e-005    Hessian modified
    8      54        -0.3674            1        -0.00303    Hessian modified twice
    9      60        -0.3806            1         -0.0213    Hessian modified
   10      66        -0.3862            1         0.00266
   11      72        -0.3863            1      -2.73e-005    Hessian modified twice
   12      78        -0.3863            1       -1.6e-013    Hessian modified twice
Optimization terminated: magnitude of search direction less than 2*options.TolX
 and maximum constraint violation is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
    1                                 1
    2                                 2
    4
```

7. Розв’язок системи:

```
x =

   -4.0000   -0.2564
   -4.0000   -4.0000
```

8. Інші параметри оптимізації:

```
eigfun(x)

ans =

   -6.9313
   -4.1588
   -1.4099
```

```
attainfactor

attainfactor =

   -0.3863
```

Фактор досягнення вказує, що кожна із цілей була зверхдосягнута принаймні на 38.63 % за початковими цілям проекту. Активні обмеження, у цьому випадку обмеження 1 і 2, є цілями, які є бар'єрами(перешкодами) до подальшого вдосконалення й для якого відсоток від зверх досягнутого точно виконаний. Три з нижче зв'язаних обмежень також активні.

У попередньому проекті, optimizer пробує зробити *objectives* менше ніж goals. Для проблеми гіршого випадку, де *objectives* повинні бути так близько до цілей наскільки це можливо, використовуйте optimset, щоб установити опцію *GoalsExactAchieve* у число *objectives*, для яких це потрібно.

9. Наступний крок оптимізації:

```
[x,fval,attainfactor,exitflag,output,lambda] = ...
    fgoalattain(eigfun,x0,goal,weight,[],[],[],[],lb,ub,[],options);
```

```
                    Attainment                  Directional
 Iter   F-count       factor      Step-size     derivative    Procedure
    0       6        1.88521
    1      12          1.061            1           1.03
    2      18         0.4211            1          -0.679
    3      24         0.1437            1           -0.2    Hessian modified
    4      30        0.06407            1          -0.126    Hessian modified
    5      36       0.005702            1         -0.0268    Hessian modified
    6      42     9.634e-006            1      -1.07e-017    Hessian modified
    7      48     4.773e-011            1       3.15e-019    Hessian modified
Optimization terminated: magnitude of directional derivative in search
 direction less than 2*options.TolFun and maximum constraint violation
  is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
                                       1
                                       2
                                       3
                                       4
                                       5
                                       6
```

10. Результати оптимізації:

```
eigfun(x)      % These values are also held in output fval


ans =

   -5.0000
   -3.0000
   -1.0000

attainfactor


attainfactor =

  2.9018e-022
```

Алгоритм

Багатоцільова оптимізація стосується мінімізації ряду цілей одночасно. Одне формулювання для цієї проблеми, і здійснений в *fgoalattain*, є проблемою досягнення мети *Gembicki* [3]. Це спричиняє побудову ряду значень мети для функцій мети.

У цьому виконанні, слабка змінна використовується як фіктивний аргумент, щоб мінімізувати вектор цілей *F(x)* одночасно; *goal* – це ряд значень, яких мета досягає. Взагалі, до оптимізації, не відомо, чи досягнуть мети цілей (при досягненні) або будуть мінімізовані менше ніж *goals (overattainment)*. Вектор *weighting*, управляє відносним *underattainment* або зверхдосягненням цілей.

fgoalattain використовує послідовне квадратне програмування (SQP) метод. Модифікації зроблені до пошуку лінії та *Hessian*. Пошук лінії закінчений, коли будь-яка функція заслуги показує вдосконалення. Змінена *Hessian*, що використовує у своїх інтересах спеціальну структуру проблеми, також використовується. Встановлюючи опцію MeritFunction в 'singleobj' з *options = optimset(options,'MeritFunction','singleobj')* використовує функцію мети та *Hessian*, використовувану в *fmincon*. *Attainfactor* містить значення при рішенні. Негативне значення указує на зверхдосягненням цілей. *Обмеження* objectives повинні бути безперервними. *fgoalattain* міг би дати тільки місцеві рішення.

Посилання

[1] Brayton, R.K., S.W. Director, G.D. Hachtel, and L.Vidigal, "A New Algorithm for Statistical Circuit Design Based on Quasi-Newton Methods and Function Splitting," IEEE Transactions on Circuits and Systems, Vol. CAS-26, pp 784-794, Sept. 1979.

[2] Fleming, P.J. and A.P. Pashkevich, Computer Aided Control System Design Using a Multi-Objective Optimisation Approach, Control 1985 Conference, Cambridge, UK, pp. 174-179.

[3] Gembicki, F.W., "Vector Optimization for Control with Performance and Parameter Sensitivity Indices," Ph.D. Dissertation, Case Western Reserve Univ., Cleveland, OH, 1974.

[4] Grace, A.C.W., "Computer-Aided Control System Design Using Optimization Techniques," Ph.D. Thesis, University of Wales, Bangor, Gwynedd, UK, 1989.

[5] Han, S.P., "A Globally Convergent Method For Nonlinear Programming," Journal of Optimization Theory and Applications, Vol. 22, p. 297, 1977.

[6] Powell, M.J.D., "A Fast Algorithm for Nonlinear Constrained Optimization Calculations," Numerical Analysis, ed. G.A. Watson, Lecture Notes in Mathematics, Vol. 630, Springer Verlag, 1978.

ЗМІСТ

Аргументи функцій (Function Arguments) 2

Параметри оптимізації (Optimization Options) 5

Опис ідентифікаторів функцій (Function-Specific Output Headings) 9

Функція Fminbnd 12

Функція Lsqnonlin 14

Функція Lsqnonneg 17

Функція Linprog 19

Функція Fseminf 23

Функція Lsqlin 31

Функція Lsqcurvefit 35

Функція Fmincon 41

Функція Quadprog 48

Функція Fminunc 52

Функція Fminsearch 58

Функція Fgoalattain 60