ME469A
Handout #7

Gianluca Iaccarino

Iterative Solvers for Linear Systems

A generic iterative method can be written as:

$$M\phi^{p+1} = N\phi^{p} + B$$

- At convergence $\phi^{p+1} = \phi^{p} = \phi$ which implies $A = M - N, \quad Q = B$
- The convergence increases if $M \approx A \rightarrow N \approx 0$
- We can use $M = diag(A)$ or $M = lower(A)$ which lead to systems that are easy to solve.
- An interesting case is $M = Tridiag(A)$

Iterative Solvers for Linear Systems

A generic iterative method can be written as:

$$M\phi^{p+1} = N\phi^{p} + B$$

- At convergence $\phi^{p+1} = \phi^{p} = \phi$ which implies $A = M - N, \quad Q = B$
- The convergence increases if $M \approx A \rightarrow N \approx 0$
- We can use $M = diag(A)$ or $M = lower(A)$ which lead to systems that are easy to solve.
- An interesting case is $M = Tridiag(A)$

Line-by-line solvers

Consider the 5-point discretization we introduced to solve the steady convection/diffusion problem in 2D

$$A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$$

It is possible to define an iterative system as:

$$A_W\phi_W^{p+1} + A_P\phi_P^{p+1} + A_E\phi_E^{p+1} = Q_P - A_S\phi_S^{p} - A_N\phi_N^{p}$$

- Each $W - E$ line is treated implicitly, the $S - N$ lines are explicit! It is obviously possible to treat the $S - N$ line implicitly.

Line-by-line solvers

Consider the 5-point discretization we introduced to solve the steady convection/diffusion problem in 2D

$$A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$$

It is possible to define an iterative system as:

$$A_W\phi_W^{p+1} + A_P\phi_P^{p+1} + A_E\phi_E^{p+1} = Q_P - A_S\phi_S^p - A_N\phi_N^p$$

- Each $W - E$ line is treated implicitly, the $S - N$ lines are explicit! It is obviously possible to treat the $S - N$ line implicitly.

Line-by-line solvers

Consider the 5-point discretization we introduced to solve the steady convection/diffusion problem in 2D

$$A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$$

It is possible to define an iterative system as:

$$A_W\phi_W^{p+1} + A_P\phi_P^{p+1} + A_E\phi_E^{p+1} = Q_P - A_S\phi_S^{p} - A_N\phi_N^{p}$$

- Each $W - E$ line is treated implicitly, the $S - N$ lines are explicit! It is obviously possible to treat the $S - N$ line implicitly.

Incomplete LU (ILU) Decomposition

- We mentioned that the *LU* decomposition leads to an excellent solver, although it requires high computational cost and does not take advantage of the matrix sparsity.

  Idea: We can use an approximate factorization of the matrix

  Assume $M = LU$ where L and U are constructed using the Gauss elimination steps but remain sparse!
- We simply ignore (i.e. set to zero) all the entries that are originally zero in A - *incomplete Cholesky Factorization* - not very effective
- We allow for an increase in the matrix filling - *Stone's methods, strong implicit procedure*

Incomplete LU (ILU) Decomposition

- We mentioned that the *LU* decomposition leads to an excellent solver, although it requires high computational cost and does not take advantage of the matrix sparsity.

  Idea: We can use an approximate factorization of the matrix

  Assume $M = LU$ where L and U are constructed using the Gauss elimination steps but remain sparse!
- We simply ignore (i.e. set to zero) all the entries that are originally zero in A - *incomplete Cholesky Factorization* - not very effective
- We allow for an increase in the matrix filling - *Stone's methods, strong implicit procedure*

Incomplete LU (ILU) Decomposition

- We mentioned that the *LU* decomposition leads to an excellent solver, although it requires high computational cost and does not take advantage of the matrix sparsity.

  Idea: We can use an approximate factorization of the matrix

  Assume $M = LU$ where L and U are constructed using the Gauss elimination steps but remain sparse!

- We simply ignore (i.e. set to zero) all the entries that are originally zero in A - *incomplete Cholesky Factorization* - not very effective
- We allow for an increase in the matrix filling - *Stone's methods, strong implicit procedure*

Incomplete LU (ILU) Decomposition

- We mentioned that the *LU* decomposition leads to an excellent solver, although it requires high computational cost and does not take advantage of the matrix sparsity.

  Idea: We can use an approximate factorization of the matrix

  Assume $M = LU$ where L and U are constructed using the Gauss elimination steps but remain sparse!
- We simply ignore (i.e. set to zero) all the entries that are originally zero in A - *incomplete Cholesky Factorization* - not very effective
- We allow for an increase in the matrix filling - *Stone's methods, strong implicit procedure*

Incomplete LU (ILU) Decomposition

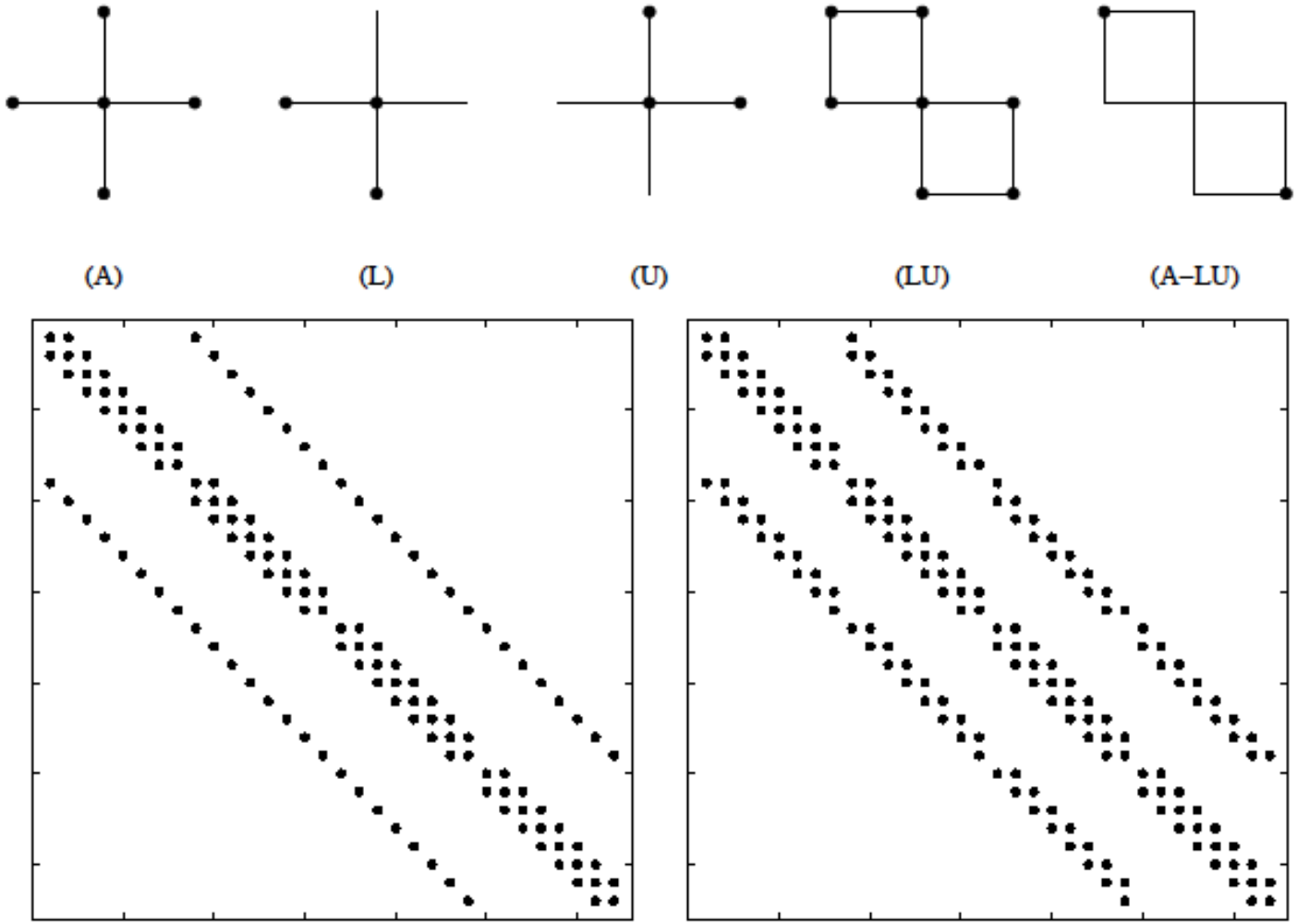
- We mentioned that the *LU* decomposition leads to an excellent solver, although it requires high computational cost and does not take advantage of the matrix sparsity.

  Idea: We can use an approximate factorization of the matrix

  Assume $M = LU$ where L and U are constructed using the Gauss elimination steps but remain sparse!
- We simply ignore (i.e. set to zero) all the entries that are originally zero in A - *incomplete Cholesky Factorization* - not very effective
- We allow for an increase in the matrix filling - *Stone's methods, strong implicit procedure*

LU decomposition

Consider a 5-point stencil (2D convection/diffusion equation):



Stone's Method

- Consider the effect of the matrix M on the vector ϕ:
  $(M\phi)_P = M_P\phi_P + M_S\phi_S + M_N\phi_N + M_E\phi_E + M_W\phi_W +$
  $M_{SE}\phi_{SE} + M_{NW}\phi_{NW}$
- N should contain the additional diagonals because
  $N = M - A = ILU - A$
- We also want $N\phi \approx 0$ which results in
  $(N\phi)_P = N_P\phi_P + N_S\phi_S + N_N\phi_N + N_E\phi_E + N_W\phi_W +$
  $M_{SE}\phi_{SE} + M_{NW}\phi_{NW} \approx 0$
- The elements of N should be such that
  $N_P\phi_P + \sum_{nb} N_{nb}\phi_{nb} \approx M_{SE}\phi_{SE} + M_{NW}\phi_{NW}$

Remark: $\sum_{nb} N_{nb}\phi_{nb}$ is a short hand notation for $N_S\phi_S + N_N\phi_N + N_E\phi_E + N_W\phi_W$

Stone's Method

- Consider the effect of the matrix M on the vector ϕ:
  $(M\phi)_P = M_P\phi_P + M_S\phi_S + M_N\phi_N + M_E\phi_E + M_W\phi_W +$
  $M_{SE}\phi_{SE} + M_{NW}\phi_{NW}$
- N should contain the additional diagonals because
  $N = M - A = ILU - A$
- We also want $N\phi \approx 0$ which results in
  $(N\phi)_P = N_P\phi_P + N_S\phi_S + N_N\phi_N + N_E\phi_E + N_W\phi_W +$
  $M_{SE}\phi_{SE} + M_{NW}\phi_{NW} \approx 0$
- The elements of N should be such that
  $N_P\phi_P + \sum_{nb} N_{nb}\phi_{nb} \approx M_{SE}\phi_{SE} + M_{NW}\phi_{NW}$

Remark: $\sum_{nb} N_{nb}\phi_{nb}$ is a short hand notation for $N_S\phi_S + N_N\phi_N + N_E\phi_E + N_W\phi_W$

Stone's Method

- Consider the effect of the matrix M on the vector ϕ:

  $(M\phi)_P = M_P\phi_P + M_S\phi_S + M_N\phi_N + M_E\phi_E + M_W\phi_W +$

  $M_{SE}\phi_{SE} + M_{NW}\phi_{NW}$
- N should contain the additional diagonals because $N = M - A = ILU - A$
- We also want $N\phi \approx 0$ which results in

  $(N\phi)_P = N_P\phi_P + N_S\phi_S + N_N\phi_N + N_E\phi_E + N_W\phi_W +$

  $M_{SE}\phi_{SE} + M_{NW}\phi_{NW} \approx 0$
- The elements of N should be such that

  $N_P\phi_P + \sum_{nb} N_{nb}\phi_{nb} \approx M_{SE}\phi_{SE} + M_{NW}\phi_{NW}$

Remark: $\sum_{nb} N_{nb}\phi_{nb}$ is a short hand notation for $N_S\phi_S + N_N\phi_N + N_E\phi_E + N_W\phi_W$

Stone's Method

- Consider the effect of the matrix M on the vector ϕ:

  $(M\phi)_P = M_P\phi_P + M_S\phi_S + M_N\phi_N + M_E\phi_E + M_W\phi_W +$

  $M_{SE}\phi_{SE} + M_{NW}\phi_{NW}$
- N should contain the additional diagonals because $N = M - A = ILU - A$
- We also want $N\phi \approx 0$ which results in

  $(N\phi)_P = N_P\phi_P + N_S\phi_S + N_N\phi_N + N_E\phi_E + N_W\phi_W +$

  $M_{SE}\phi_{SE} + M_{NW}\phi_{NW} \approx 0$
- The elements of N should be such that

  $N_P\phi_P + \sum_{nb} N_{nb}\phi_{nb} \approx M_{SE}\phi_{SE} + M_{NW}\phi_{NW}$

Remark: $\sum_{nb} N_{nb}\phi_{nb}$ is a short hand notation for $N_S\phi_S + N_N\phi_N + N_E\phi_E + N_W\phi_W$

Stone's Method

- Consider the effect of the matrix M on the vector ϕ:

  $(M\phi)_P = M_P\phi_P + M_S\phi_S + M_N\phi_N + M_E\phi_E + M_W\phi_W +$

  $M_{SE}\phi_{SE} + M_{NW}\phi_{NW}$
- N should contain the additional diagonals because $N = M - A = ILU - A$
- We also want $N\phi \approx 0$ which results in

  $(N\phi)_P = N_P\phi_P + N_S\phi_S + N_N\phi_N + N_E\phi_E + N_W\phi_W +$

  $M_{SE}\phi_{SE} + M_{NW}\phi_{NW} \approx 0$
- The elements of N should be such that

  $N_P\phi_P + \sum_{nb} N_{nb}\phi_{nb} \approx M_{SE}\phi_{SE} + M_{NW}\phi_{NW}$

Remark: $\sum_{nb} N_{nb}\phi_{nb}$ is a short hand notation for $N_S\phi_S + N_N\phi_N + N_E\phi_E + N_W\phi_W$

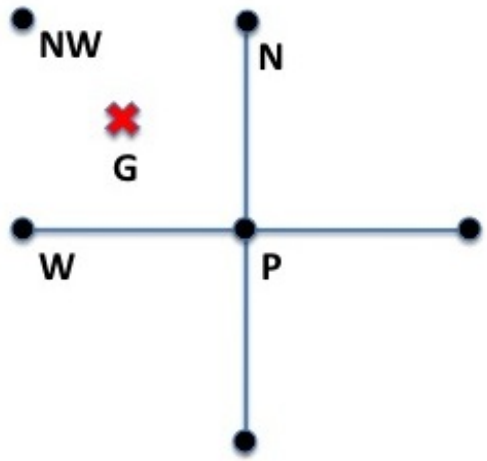
Stone's Method

- Stone showed that is possible to obtain a unique splitting (M and N) by choosing

  $N_P\phi_P + \sum_{nb} N_{nb}\phi_{nb} \approx M_{SE}\phi^*_{SE} + M_{NW}\phi^*_{NW}$

  where ϕ^* are interpolated values

- For example $\phi^*_{NW} \approx \alpha(\phi_W + \phi_N - \phi_P)$ is obtained equating $(\phi_W + \phi_N)/2$ and $(\phi^*_{NW} + \phi_P)/2$ which are two approximations of ϕ_G
- Note: $\alpha < 1$ for stability
- This is NOT a general algorithm



H. Stone, Solution of Implicit Approximations of Partial Differential Eqs., SIAM J. Num. An. 5(3), 1968

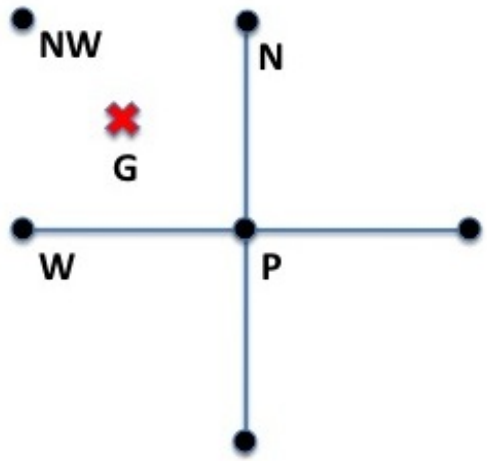
Stone's Method

- Stone showed that is possible to obtain a unique splitting (M and N) by choosing

  $N_P\phi_P + \sum_{nb} N_{nb}\phi_{nb} \approx M_{SE}\phi^*_{SE} + M_{NW}\phi^*_{NW}$

  where ϕ^* are interpolated values

- For example
  $\phi^*_{NW} \approx \alpha(\phi_W + \phi_N - \phi_P)$ is obtained equating $(\phi_W + \phi_N)/2$ and $(\phi^*_{NW} + \phi_P)/2$ which are two approximations of ϕ_G
- Note: $\alpha < 1$ for stability
- This is NOT a general algorithm



H. Stone, Solution of Implicit Approximations of Partial Differential Eqs., SIAM J. Num. An. 5(3), 1968

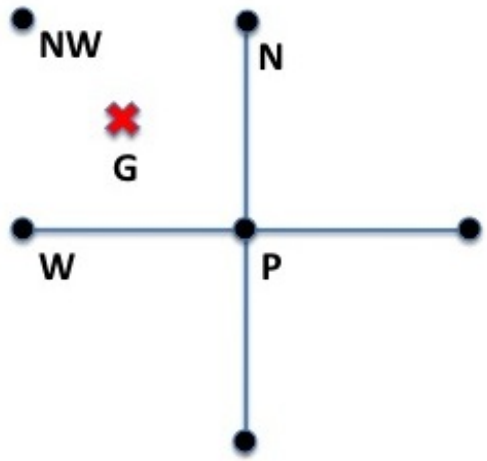
Stone's Method

- Stone showed that is possible to obtain a unique splitting (M and N) by choosing

  $N_P\phi_P + \sum_{nb} N_{nb}\phi_{nb} \approx M_{SE}\phi^*_{SE} + M_{NW}\phi^*_{NW}$

  where ϕ^* are interpolated values

- For example
  $\phi^*_{NW} \approx \alpha(\phi_W + \phi_N - \phi_P)$ is obtained equating $(\phi_W + \phi_N)/2$ and $(\phi^*_{NW} + \phi_P)/2$ which are two approximations of ϕ_G
- Note: $\alpha < 1$ for stability
- This is NOT a general algorithm



H. Stone, Solution of Implicit Approximations of Partial Differential Eqs., SIAM J. Num. An. 5(3), 1968

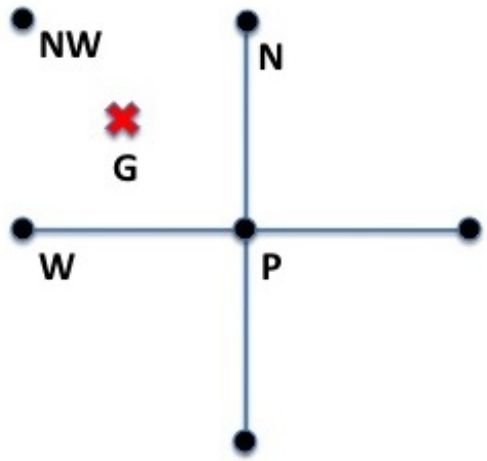
Stone's Method

- Stone showed that is possible to obtain a unique splitting (M and N) by choosing

  $N_P\phi_P + \sum_{nb} N_{nb}\phi_{nb} \approx M_{SE}\phi^*_{SE} + M_{NW}\phi^*_{NW}$

  where ϕ^* are interpolated values

- For example
  $\phi^*_{NW} \approx \alpha(\phi_W + \phi_N - \phi_P)$ is obtained equating $(\phi_W + \phi_N)/2$ and $(\phi^*_{NW} + \phi_P)/2$ which are two approximations of ϕ_G
- Note: $\alpha < 1$ for stability
- This is NOT a general algorithm



H. Stone, Solution of Implicit Approximations of Partial Differential Eqs., SIAM J. Num. An. 5(3), 1968

ILU(T) methods

The idea behind the Stone's method can be generalized to allow for an increase in filling for the *LU* factors.

- Cholesky's and Stone's methods are examples of fill/drop by position
- ILU(T) methods are typically based on fill/drop by *numerical* size: T is the size threshold.

Extensive literature, for example see Y. Saad

ILU(T) methods

The idea behind the Stone's method can be generalized to allow for an increase in filling for the *LU* factors.

- Cholesky's and Stone's methods are examples of fill/drop by position
- ILU(T) methods are typically based on fill/drop by *numerical* size: T is the size threshold.

Extensive literature, for example see Y. Saad

ILU(T) methods

The idea behind the Stone's method can be generalized to allow for an increase in filling for the *LU* factors.

- Cholesky's and Stone's methods are examples of fill/drop by position
- ILU(T) methods are typically based on fill/drop by *numerical* size: T is the size threshold.

Extensive literature, for example see Y. Saad

ILU(T) methods

The idea behind the Stone's method can be generalized to allow for an increase in filling for the *LU* factors.

- Cholesky's and Stone's methods are examples of fill/drop by position
- ILU(T) methods are typically based on fill/drop by *numerical* size: T is the size threshold.

Extensive literature, for example see Y. Saad

Splitting Methods

- Originally developed for the time-dependent heat condition equation

$$\frac{\partial \phi}{\partial t} = \Gamma \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \right)$$

- Consider a FD discretization scheme (Crank-Nicholson)

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{\Gamma}{2} \left[\left(\frac{\delta^2 \phi}{\delta x^2} \right)_i^{n+1} + \left(\frac{\delta^2 \phi}{\delta y^2} \right)_i^{n+1} + \left(\frac{\delta^2 \phi}{\delta x^2} \right)_i^{n} + \left(\frac{\delta^2 \phi}{\delta y^2} \right)_i^{n} \right]$$

- This leads to a penta-diagonal matrix

Splitting Methods

- Originally developed for the time-dependent heat condition equation

  $$\frac{\partial \phi}{\partial t} = \Gamma \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \right)$$

- Consider a FD discretization scheme (Crank-Nicholson)

  $$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{\Gamma}{2} \left[\left(\frac{\delta^2 \phi}{\delta x^2} \right)_i^{n+1} + \left(\frac{\delta^2 \phi}{\delta y^2} \right)_i^{n+1} + \left(\frac{\delta^2 \phi}{\delta x^2} \right)_i^{n} + \left(\frac{\delta^2 \phi}{\delta y^2} \right)_i^{n} \right]$$

- This leads to a penta-diagonal matrix

Splitting Methods

- Originally developed for the time-dependent heat condition equation

  $$\frac{\partial \phi}{\partial t} = \Gamma \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \right)$$

- Consider a FD discretization scheme (Crank-Nicholson)

  $$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} = \frac{\Gamma}{2} \left[\left(\frac{\delta^2 \phi}{\delta x^2} \right)_i^{n+1} + \left(\frac{\delta^2 \phi}{\delta y^2} \right)_i^{n+1} + \left(\frac{\delta^2 \phi}{\delta x^2} \right)_i^{n} + \left(\frac{\delta^2 \phi}{\delta y^2} \right)_i^{n} \right]$$

- This leads to a penta-diagonal matrix

Splitting Methods

- It is possible to manipulate the discretized equations to obtain

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}=$$
$$\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}-\frac{(\Gamma\Delta t)^2}{4}\frac{\delta^2}{\delta x^2}\left[\frac{\delta^2(\phi^{n+1}-\phi^n)}{\delta y^2}\right]$$

- The last contribution can be rewritten noting that:

$$\phi^{n+1}-\phi^n\approx\Delta t\frac{\partial\phi}{\partial t}$$

- The last term is $O(\Delta t)^3$ and therefore can be ignored! (the approximation of the time derivative is second order)

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}\approx$$
$$\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}$$

Splitting Methods

- It is possible to manipulate the discretized equations to obtain

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}=$$
$$\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}-\frac{(\Gamma\Delta t)^2}{4}\frac{\delta^2}{\delta x^2}\left[\frac{\delta^2(\phi^{n+1}-\phi^n)}{\delta y^2}\right]$$

- The last contribution can be rewritten noting that:

$$\phi^{n+1}-\phi^n\approx\Delta t\frac{\partial\phi}{\partial t}$$

- The last term is $O(\Delta t)^3$ and therefore can be ignored! (the approximation of the time derivative is second order)

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}\approx$$
$$\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}$$

Splitting Methods

- It is possible to manipulate the discretized equations to obtain

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}=$$
$$\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}-\frac{(\Gamma\Delta t)^2}{4}\frac{\delta^2}{\delta x^2}\left[\frac{\delta^2(\phi^{n+1}-\phi^n)}{\delta y^2}\right]$$

- The last contribution can be rewritten noting that:

$$\phi^{n+1}-\phi^n\approx\Delta t\frac{\partial\phi}{\partial t}$$

- The last term is $O(\Delta t)^3$ and therefore can be ignored! (the approximation of the time derivative is second order)

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}\approx$$
$$\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}$$

Splitting Methods

- The discretized system is

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}=$$
$$\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}$$

- This is equivalent to a two-step solution:

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\phi^{*}=\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}$$
$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}=\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\phi^{*}$$

- Each system requires a tridiagonal solve: Alternating Direction Implicit, ADI

Splitting Methods

- The discretized system is

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}=$$
$$\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}$$

- This is equivalent to a two-step solution:

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\phi^{*}=\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}$$
$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}=\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\phi^{*}$$

- Each system requires a tridiagonal solve: Alternating Direction Implicit, ADI

Splitting Methods

- The discretized system is

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}=$$
$$\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}$$

- This is equivalent to a two-step solution:

$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\phi^{*}=\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n}$$
$$\left(1-\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta y^2}\right)\phi^{n+1}=\left(1+\frac{\Gamma\Delta t}{2}\frac{\delta^2}{\delta x^2}\right)\phi^{*}$$

- Each system requires a tridiagonal solve: Alternating Direction Implicit, ADI

Generalized Splitting Methods

- The splitting of the operators used in the ADI is based on the fact that the discrete operators ($\delta^2/\delta x^2$ and $\delta^2/\delta y^2$) are decoupled!
- This decoupling is typical of structured grids but does not apply in general
- It is possible to generate the splitting in a general fashion:

  $A = L + U$ (additive decomposition, as opposed to multiplicative $A = LU$).
- The solution procedure is then:

  $(I - L\Delta t)\phi^* = (I + U\Delta t)\phi^n$

  $(I - U\Delta t)\phi^{n+1} = (I + L\Delta t)\phi^*$

Generalized Splitting Methods

- The splitting of the operators used in the ADI is based on the fact that the discrete operators ($\delta^2/\delta x^2$ and $\delta^2/\delta y^2$) are decoupled!
- This decoupling is typical of structured grids but does not apply in general
- It is possible to generate the splitting in a general fashion:

  $A = L + U$ (additive decomposition, as opposed to multiplicative $A = LU$).
- The solution procedure is then:

  $(I - L\Delta t)\phi^* = (I + U\Delta t)\phi^n$

  $(I - U\Delta t)\phi^{n+1} = (I + L\Delta t)\phi^*$

Generalized Splitting Methods

- The splitting of the operators used in the ADI is based on the fact that the discrete operators ($\delta^2/\delta x^2$ and $\delta^2/\delta y^2$) are decoupled!
- This decoupling is typical of structured grids but does not apply in general
- It is possible to generate the splitting in a general fashion:

  $A = L + U$ (additive decomposition, as opposed to multiplicative $A = LU$).
- The solution procedure is then:

  $(I - L\Delta t)\phi^* = (I + U\Delta t)\phi^n$

  $(I - U\Delta t)\phi^{n+1} = (I + L\Delta t)\phi^*$

Generalized Splitting Methods

- The splitting of the operators used in the ADI is based on the fact that the discrete operators ($\delta^2/\delta x^2$ and $\delta^2/\delta y^2$) are decoupled!
- This decoupling is typical of structured grids but does not apply in general
- It is possible to generate the splitting in a general fashion:

  $A = L + U$ (additive decomposition, as opposed to multiplicative $A = LU$).
- The solution procedure is then:

  $(I - L\Delta t)\phi^* = (I + U\Delta t)\phi^n$

  $(I - U\Delta t)\phi^{n+1} = (I + L\Delta t)\phi^*$

General Linear System Solvers

- Many of the methods summarized before use directly the ordering of the unknowns and therefore are limited to structured grids
- More general approaches can be derived by transforming into a minimization problem
- Of particular interest in CFD are descent methods

General Linear System Solvers

- Many of the methods summarized before use directly the ordering of the unknowns and therefore are limited to structured grids
- More general approaches can be derived by transforming into a minimization problem
- Of particular interest in CFD are descent methods

General Linear System Solvers

- Many of the methods summarized before use directly the ordering of the unknowns and therefore are limited to structured grids
- More general approaches can be derived by transforming into a minimization problem
- Of particular interest in CFD are descent methods

Geometrical Example

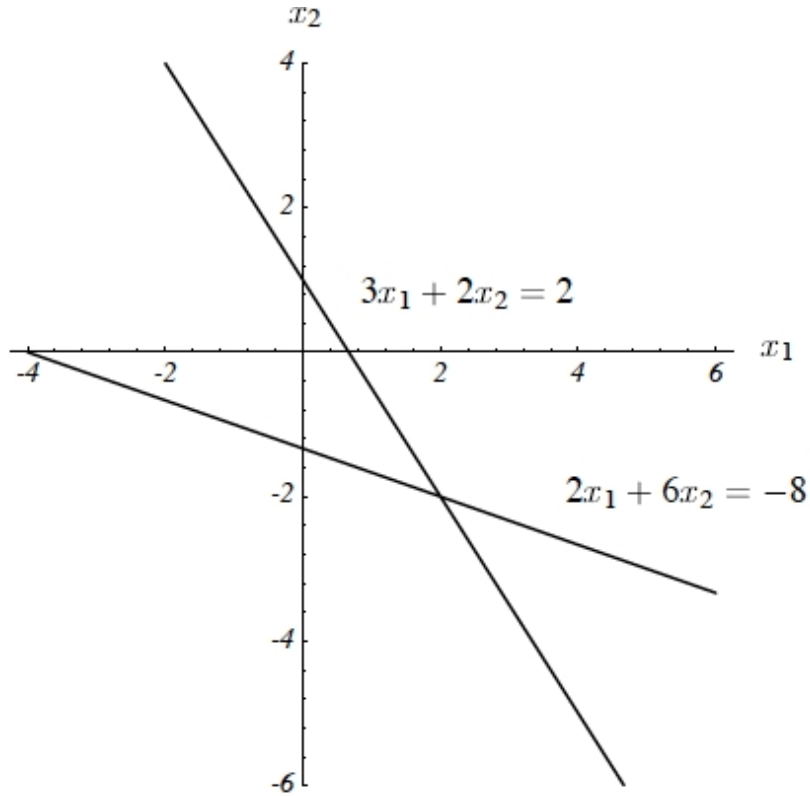
Consider a two-equation system

$$3x_1 + 2x_2 = 2$$
$$2x_1 + 6x_2 = -8$$

$$Ax = b$$

$$A = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}, b = \begin{pmatrix} 2 \\ -8 \end{pmatrix}$$

The solution is $x = [2, -2]^T$



J. R. Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain

Geometrical Example

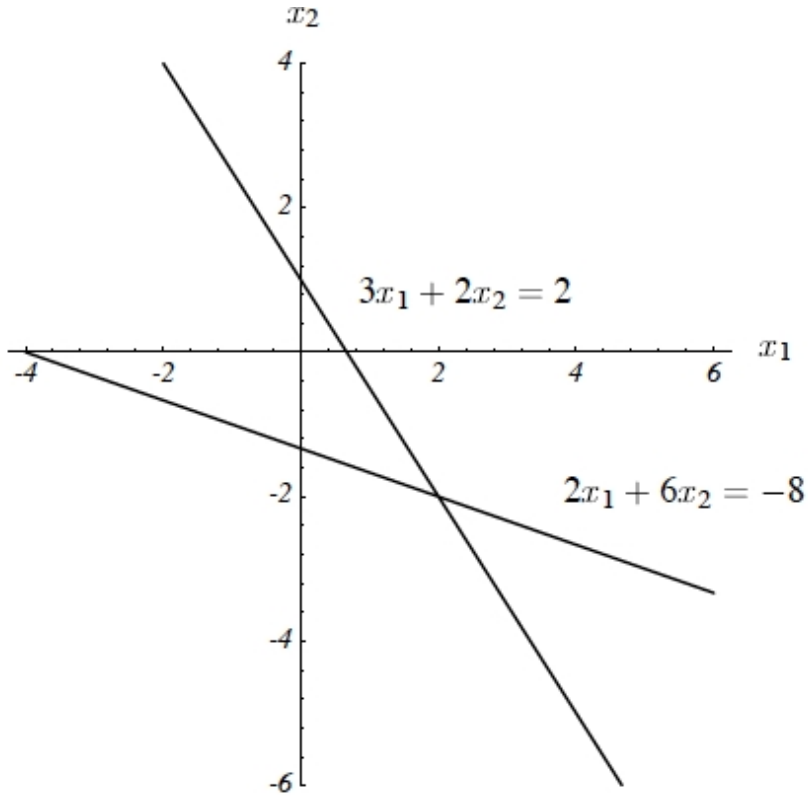
Consider a two-equation system

$$3x_1 + 2x_2 = 2$$
$$2x_1 + 6x_2 = -8$$

$$Ax = b$$

$$A = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}, b = \begin{pmatrix} 2 \\ -8 \end{pmatrix}$$

The solution is $x = [2, -2]^T$



J. R. Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain

Geometrical Example

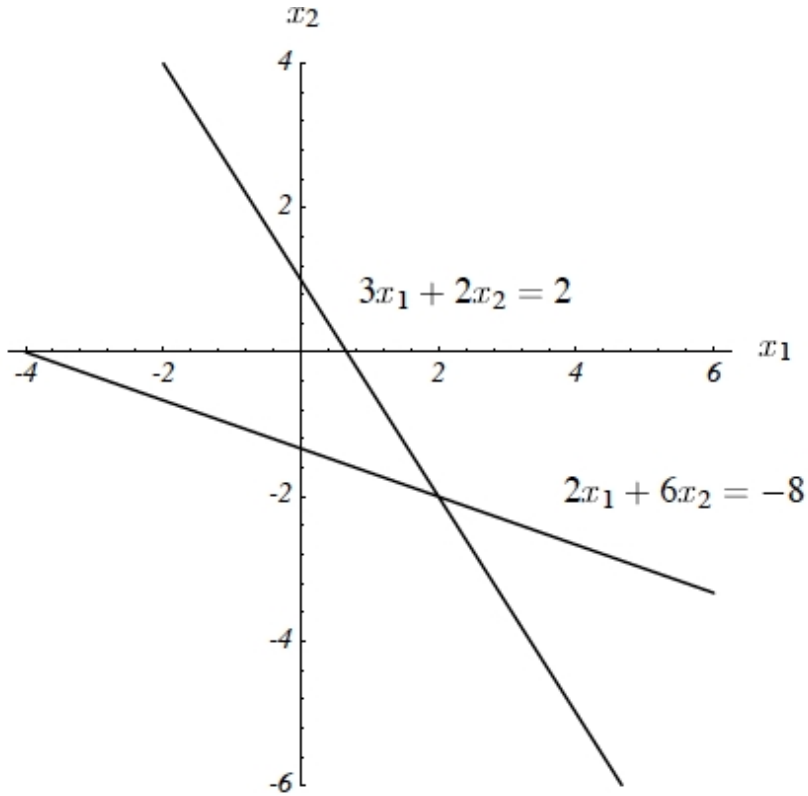
Consider a two-equation system

$$3x_1 + 2x_2 = 2$$
$$2x_1 + 6x_2 = -8$$

$$Ax = b$$

$$A = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}, b = \begin{pmatrix} 2 \\ -8 \end{pmatrix}$$

The solution is $x = [2, -2]^T$



J. R. Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain

Geometrical Example

- Consider an iterative method: $Mx^{p+1} = Nx^p + b$ with $A = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}, b = \begin{pmatrix} 2 \\ -8 \end{pmatrix}$
- Jacobi's iteration is based on $M = Diag(A) = \begin{pmatrix} 3 & 0 \\ 0 & 6 \end{pmatrix}, N = M - A = \begin{pmatrix} 0 & -2 \\ -2 & 0 \end{pmatrix}$
- More explicitly $x^{p+1} = (M^{-1}N)x^p + M^{-1}b = \begin{pmatrix} 0 & -2/3 \\ -1/3 & 0 \end{pmatrix} x^p + \begin{pmatrix} 2/3 \\ -4/3 \end{pmatrix}$ given that $M^{-1} = -\begin{pmatrix} 1/3 & 0 \\ 0 & 1/6 \end{pmatrix}$
- Convergence rate is dictated by the spectral radius of $(M^{-1}N)$, and the eigenvalues are $\lambda_{MN} = [-\sqrt{2}/3, \sqrt{2}/3]^T$

Geometrical Example

- Consider an iterative method: $Mx^{p+1} = Nx^p + b$ with $A = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}, b = \begin{pmatrix} 2 \\ -8 \end{pmatrix}$
- Jacobi's iteration is based on $M = Diag(A) = \begin{pmatrix} 3 & 0 \\ 0 & 6 \end{pmatrix}, N = M - A = \begin{pmatrix} 0 & -2 \\ -2 & 0 \end{pmatrix}$
- More explicitly $x^{p+1} = (M^{-1}N)x^p + M^{-1}b = \begin{pmatrix} 0 & -2/3 \\ -1/3 & 0 \end{pmatrix} x^p + \begin{pmatrix} 2/3 \\ -4/3 \end{pmatrix}$ given that $M^{-1} = -\begin{pmatrix} 1/3 & 0 \\ 0 & 1/6 \end{pmatrix}$
- Convergence rate is dictated by the spectral radius of $(M^{-1}N)$, and the eigenvalues are $\lambda_{MN} = [-\sqrt{2}/3, \sqrt{2}/3]^T$

Geometrical Example

- Consider an iterative method: $Mx^{p+1} = Nx^p + b$ with $A = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}, b = \begin{pmatrix} 2 \\ -8 \end{pmatrix}$
- Jacobi's iteration is based on $M = Diag(A) = \begin{pmatrix} 3 & 0 \\ 0 & 6 \end{pmatrix}, N = M - A = \begin{pmatrix} 0 & -2 \\ -2 & 0 \end{pmatrix}$
- More explicitly $x^{p+1} = (M^{-1}N)x^p + M^{-1}b = \begin{pmatrix} 0 & -2/3 \\ -1/3 & 0 \end{pmatrix} x^p + \begin{pmatrix} 2/3 \\ -4/3 \end{pmatrix}$ given that $M^{-1} = -\begin{pmatrix} 1/3 & 0 \\ 0 & 1/6 \end{pmatrix}$
- Convergence rate is dictated by the spectral radius of $(M^{-1}N)$, and the eigenvalues are $\lambda_{MN} = [-\sqrt{2}/3, \sqrt{2}/3]^T$

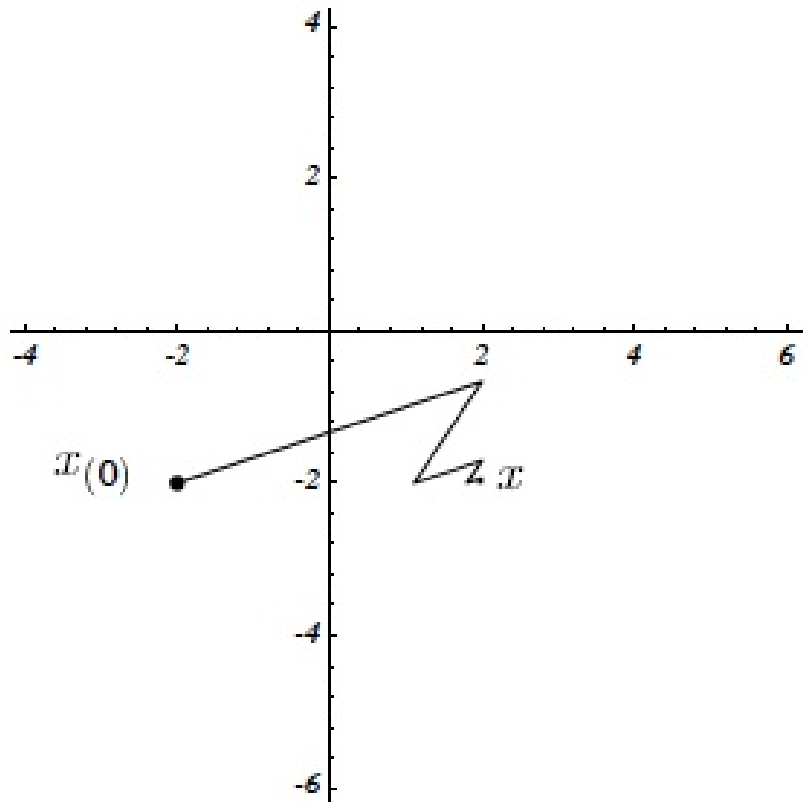
Geometrical Example

- Consider an iterative method: $Mx^{p+1} = Nx^p + b$ with $A = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}, b = \begin{pmatrix} 2 \\ -8 \end{pmatrix}$
- Jacobi's iteration is based on $M = Diag(A) = \begin{pmatrix} 3 & 0 \\ 0 & 6 \end{pmatrix}, N = M - A = \begin{pmatrix} 0 & -2 \\ -2 & 0 \end{pmatrix}$
- More explicitly $x^{p+1} = (M^{-1}N)x^p + M^{-1}b = \begin{pmatrix} 0 & -2/3 \\ -1/3 & 0 \end{pmatrix} x^p + \begin{pmatrix} 2/3 \\ -4/3 \end{pmatrix}$ given that $M^{-1} = -\begin{pmatrix} 1/3 & 0 \\ 0 & 1/6 \end{pmatrix}$
- Convergence rate is dictated by the spectral radius of $(M^{-1}N)$, and the eigenvalues are $\lambda_{MN} = [-\sqrt{2}/3, \sqrt{2}/3]^T$

Geometrical Example - Jacobi

Recall that the error evolution is $\epsilon^{p+1} = (M^{-1}N)\epsilon^p$

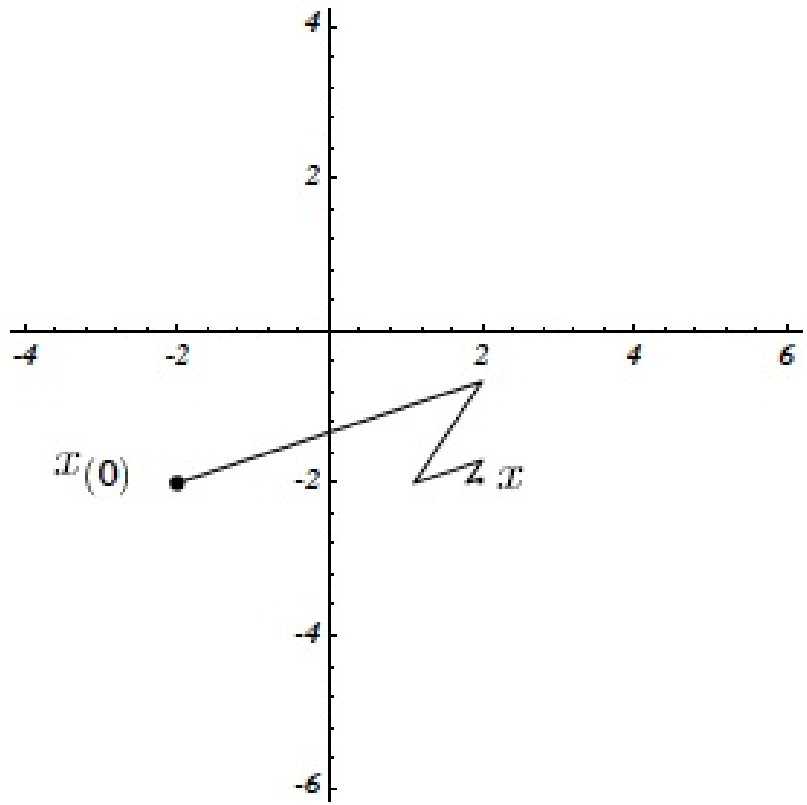
- Consider an initial guess $x^0 = [-2, -2]^T$ which results in $\epsilon^0 = [0, 2]^T$
- Given that $M^{-1}N = \begin{pmatrix} 0 & -2/3 \\ -1/3 & 0 \end{pmatrix}$ we obtain $\epsilon^1 = [-4/3, 0]^T$
- The Jacobi iteration proceeds in a zig-zag way. Can we do better?



Geometrical Example - Jacobi

Recall that the error evolution is $\epsilon^{p+1} = (M^{-1}N)\epsilon^p$

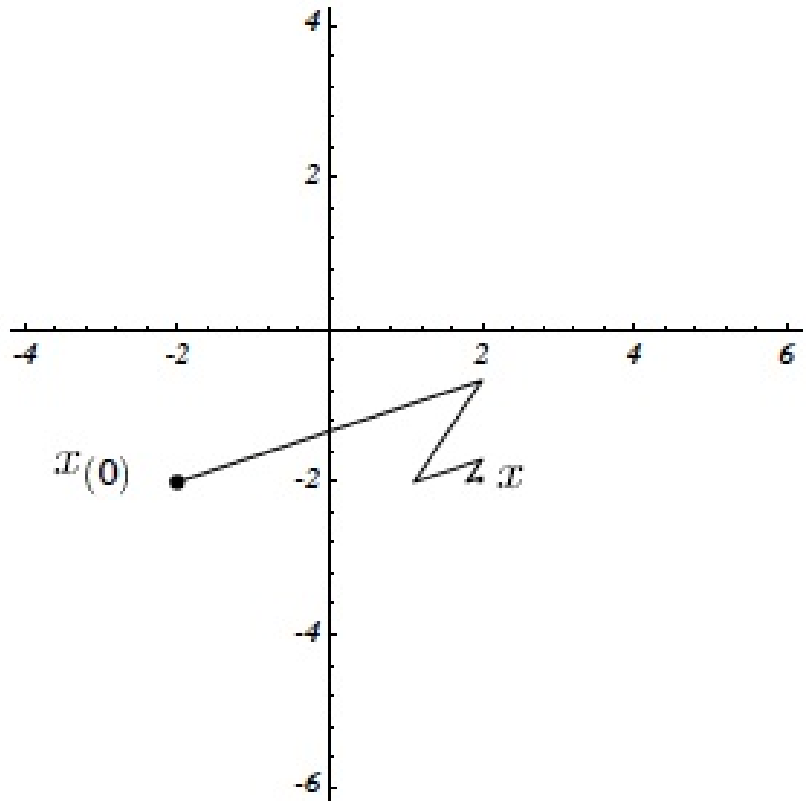
- Consider an initial guess $x^0 = [-2, -2]^T$ which results in $\epsilon^0 = [0, 2]^T$
- Given that $M^{-1}N = \begin{pmatrix} 0 & -2/3 \\ -1/3 & 0 \end{pmatrix}$ we obtain $\epsilon^1 = [-4/3, 0]^T$
- The Jacobi iteration proceeds in a zig-zag way. Can we do better?



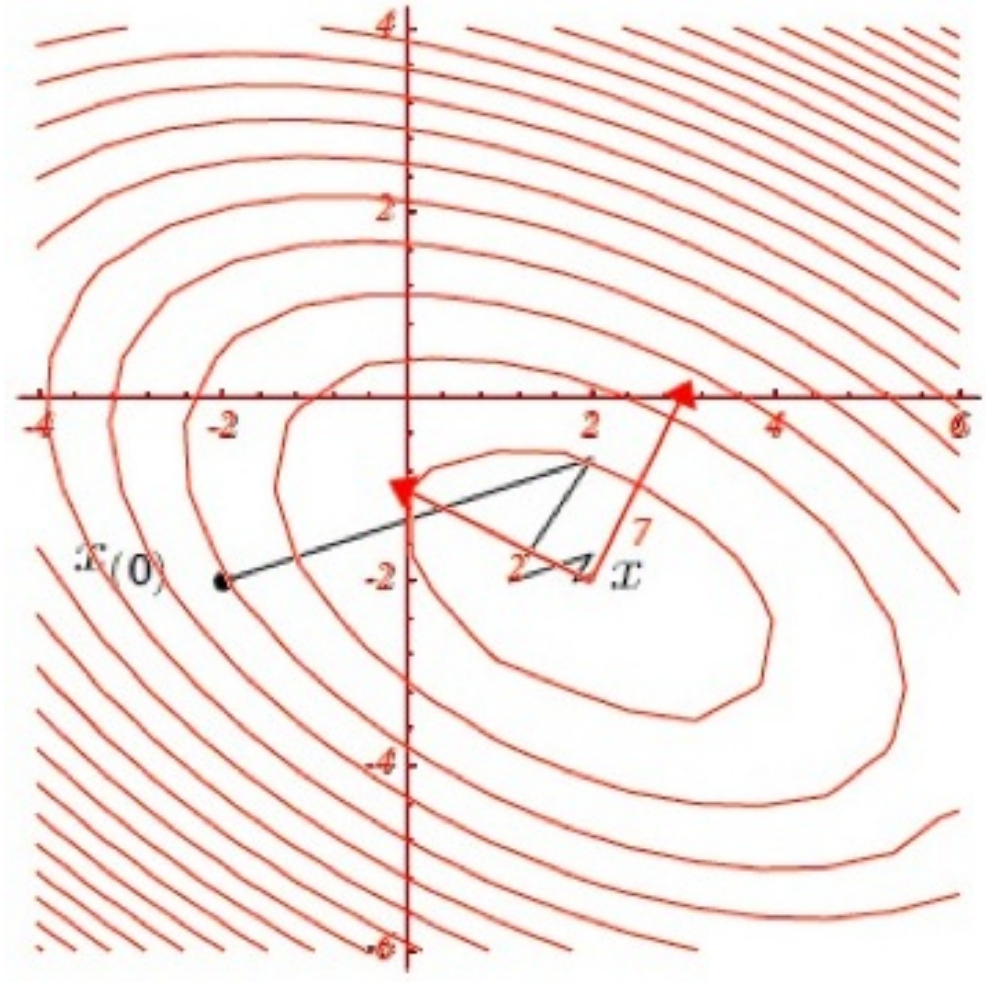
Geometrical Example - Jacobi

Recall that the error evolution is $\epsilon^{p+1} = (M^{-1}N)\epsilon^p$

- Consider an initial guess $x^0 = [-2, -2]^T$ which results in $\epsilon^0 = [0, 2]^T$
- Given that $M^{-1}N = \begin{pmatrix} 0 & -2/3 \\ -1/3 & 0 \end{pmatrix}$ we obtain $\epsilon^1 = [-4/3, 0]^T$
- The Jacobi iteration proceeds in a zig-zag way. Can we do better?



Geometrical Example - Jacobi



Build a terrain that we can navigate towards the solution

Minimization framework

- Given $Ax = b$ consider the quadratic form $f(x) = x^T Ax/2 - b^T x + c$
- The gradient of $f(x)$ is $f'(x) = A^T x/2 + Ax/2 - b$
- If A is symmetric then $f'(x) = Ax - b$
- if A is positive definite, than $f(x)$ is a paraboloid and $f'(x) = 0$ is a minimum!
- Instead of solving $Ax = b$ we can minimize the quadratic form $f(x)$
- Original problem was linear, this is non-linear!
- The *terrain* shown before are the isolines of $f(x)$

Minimization framework

- Given $Ax = b$ consider the quadratic form $f(x) = x^T Ax/2 - b^T x + c$
- The gradient of $f(x)$ is $f'(x) = A^T x/2 + Ax/2 - b$
- If A is symmetric then $f'(x) = Ax - b$
- if A is positive definite, than $f(x)$ is a paraboloid and $f'(x) = 0$ is a minimum!
- Instead of solving $Ax = b$ we can minimize the quadratic form $f(x)$
- Original problem was linear, this is non-linear!
- The *terrain* shown before are the isolines of $f(x)$

Minimization framework

- Given $Ax = b$ consider the quadratic form $f(x) = x^T Ax/2 - b^T x + c$
- The gradient of $f(x)$ is $f'(x) = A^T x/2 + Ax/2 - b$
- If A is symmetric then $f'(x) = Ax - b$
- if A is positive definite, than $f(x)$ is a paraboloid and $f'(x) = 0$ is a minimum!
- Instead of solving $Ax = b$ we can minimize the quadratic form $f(x)$
- Original problem was linear, this is non-linear!
- The *terrain* shown before are the isolines of $f(x)$

Minimization framework

- Given $Ax = b$ consider the quadratic form $f(x) = x^T Ax/2 - b^T x + c$
- The gradient of $f(x)$ is $f'(x) = A^T x/2 + Ax/2 - b$
- If A is symmetric then $f'(x) = Ax - b$
- if A is positive definite, than $f(x)$ is a paraboloid and $f'(x) = 0$ is a minimum!
- Instead of solving $Ax = b$ we can minimize the quadratic form $f(x)$
- Original problem was linear, this is non-linear!
- The *terrain* shown before are the isolines of $f(x)$

Minimization framework

- Given $Ax = b$ consider the quadratic form $f(x) = x^T Ax/2 - b^T x + c$
- The gradient of $f(x)$ is $f'(x) = A^T x/2 + Ax/2 - b$
- If A is symmetric then $f'(x) = Ax - b$
- if A is positive definite, than $f(x)$ is a paraboloid and $f'(x) = 0$ is a minimum!
- Instead of solving $Ax = b$ we can minimize the quadratic form $f(x)$
- Original problem was linear, this is non-linear!
- The *terrain* shown before are the isolines of $f(x)$

Minimization framework

- Given $Ax = b$ consider the quadratic form $f(x) = x^T Ax/2 - b^T x + c$
- The gradient of $f(x)$ is $f'(x) = A^T x/2 + Ax/2 - b$
- If A is symmetric then $f'(x) = Ax - b$
- if A is positive definite, than $f(x)$ is a paraboloid and $f'(x) = 0$ is a minimum!
- Instead of solving $Ax = b$ we can minimize the quadratic form $f(x)$
- Original problem was linear, this is non-linear!
- The *terrain* shown before are the isolines of $f(x)$

Minimization framework

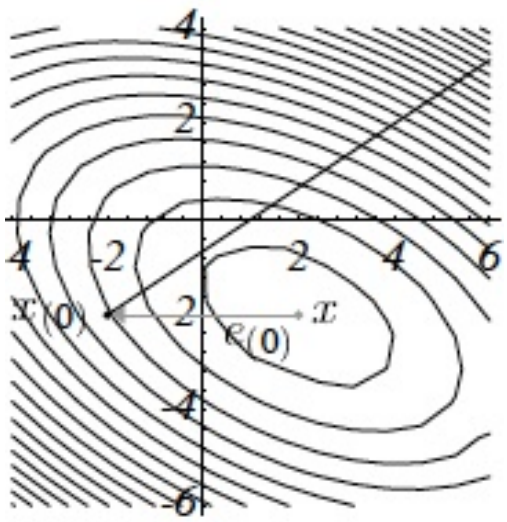
- Given $Ax = b$ consider the quadratic form $f(x) = x^T Ax/2 - b^T x + c$
- The gradient of $f(x)$ is $f'(x) = A^T x/2 + Ax/2 - b$
- If A is symmetric then $f'(x) = Ax - b$
- if A is positive definite, than $f(x)$ is a paraboloid and $f'(x) = 0$ is a minimum!
- Instead of solving $Ax = b$ we can minimize the quadratic form $f(x)$
- Original problem was linear, this is non-linear!
- The *terrain* shown before are the isolines of $f(x)$

Steepest Descent

- We have a *terrain* and an objective: get to the lowest point!
- The simplest idea is to select the direction in which $f(x)$ decreases more rapidly

  this is obviously the direction opposite to $f'(x)$
- Recall the residual: $b - Ax^p = \rho^p$ and the fact that $f'(x) = Ax - b$

- A steepest descent iterate will be $x^{p+1} = x^p + \alpha\rho_p$
- How do we stop?
- How do we define α?

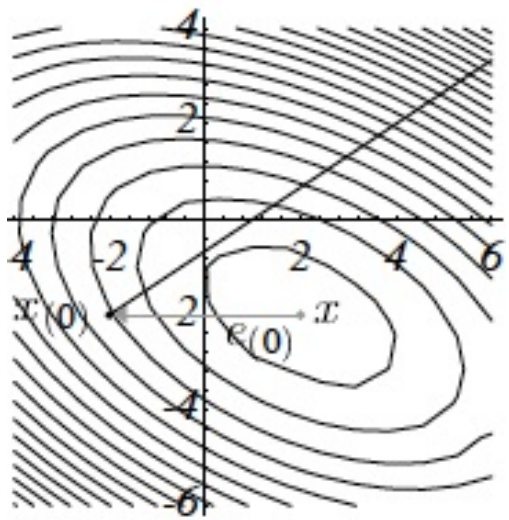


Steepest Descent

- We have a *terrain* and an objective: get to the lowest point!
- The simplest idea is to select the direction in which $f(x)$ decreases more rapidly

  this is obviously the direction opposite to $f'(x)$
- Recall the residual: $b - Ax^p = \rho^p$ and the fact that $f'(x) = Ax - b$

- A steepest descent iterate will be $x^{p+1} = x^p + \alpha\rho_p$
- How do we stop?
- How do we define α?

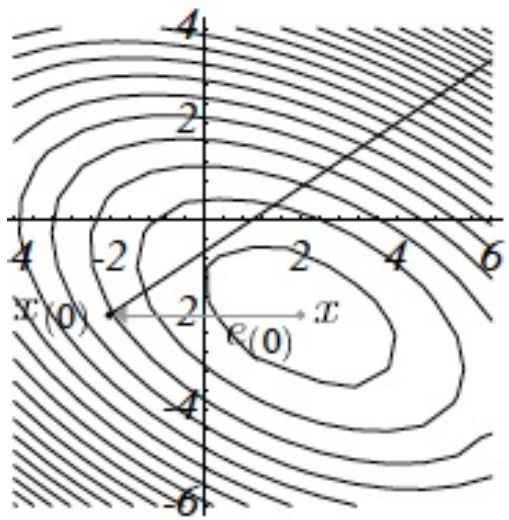


Steepest Descent

- We have a *terrain* and an objective: get to the lowest point!
- The simplest idea is to select the direction in which $f(x)$ decreases more rapidly

  this is obviously the direction opposite to $f'(x)$
- Recall the residual: $b - Ax^p = \rho^p$ and the fact that $f'(x) = Ax - b$

- A steepest descent iterate will be $x^{p+1} = x^p + \alpha\rho_p$
- How do we stop?
- How do we define α?

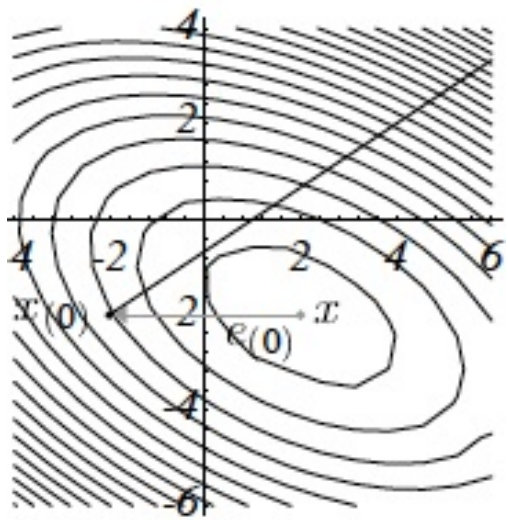


Steepest Descent

- We have a *terrain* and an objective: get to the lowest point!
- The simplest idea is to select the direction in which $f(x)$ decreases more rapidly

  this is obviously the direction opposite to $f'(x)$
- Recall the residual: $b - Ax^p = \rho^p$ and the fact that $f'(x) = Ax - b$

- A steepest descent iterate will be $x^{p+1} = x^p + \alpha\rho_p$
- How do we stop?
- How do we define α?

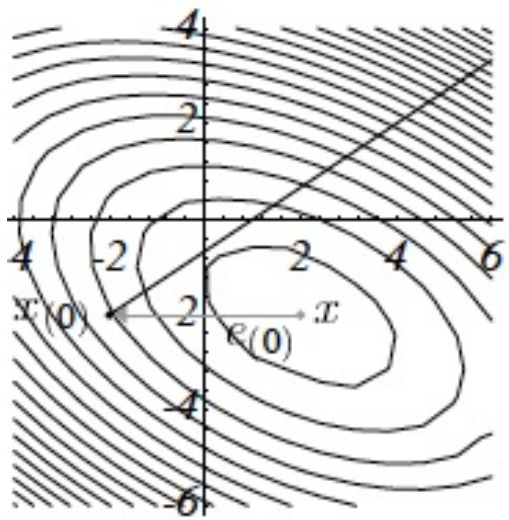


Steepest Descent

- We have a *terrain* and an objective: get to the lowest point!
- The simplest idea is to select the direction in which $f(x)$ decreases more rapidly

  this is obviously the direction opposite to $f'(x)$
- Recall the residual: $b - Ax^p = \rho^p$ and the fact that $f'(x) = Ax - b$

- A steepest descent iterate will be $x^{p+1} = x^p + \alpha\rho_p$
- How do we stop?
- How do we define α?

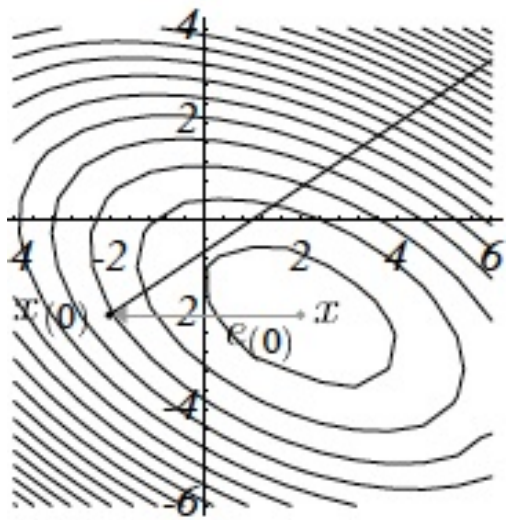


Steepest Descent

- We have a *terrain* and an objective: get to the lowest point!
- The simplest idea is to select the direction in which $f(x)$ decreases more rapidly

  this is obviously the direction opposite to $f'(x)$
- Recall the residual: $b - Ax^p = \rho^p$ and the fact that $f'(x) = Ax - b$

- A steepest descent iterate will be $x^{p+1} = x^p + \alpha\rho_p$
- How do we stop?
- How do we define α?

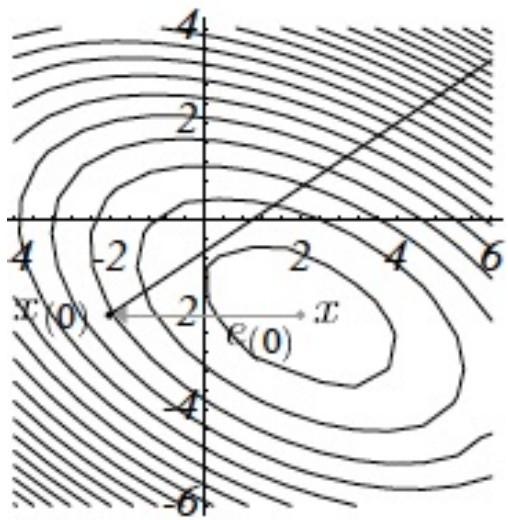


Steepest Descent

- We have a *terrain* and an objective: get to the lowest point!
- The simplest idea is to select the direction in which $f(x)$ decreases more rapidly

  this is obviously the direction opposite to $f'(x)$
- Recall the residual: $b - Ax^p = \rho^p$ and the fact that $f'(x) = Ax - b$

- A steepest descent iterate will be $x^{p+1} = x^p + \alpha\rho_p$
- How do we stop?
- How do we define α?



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

$df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

$= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

$\rho_1^T\rho_0 = 0$

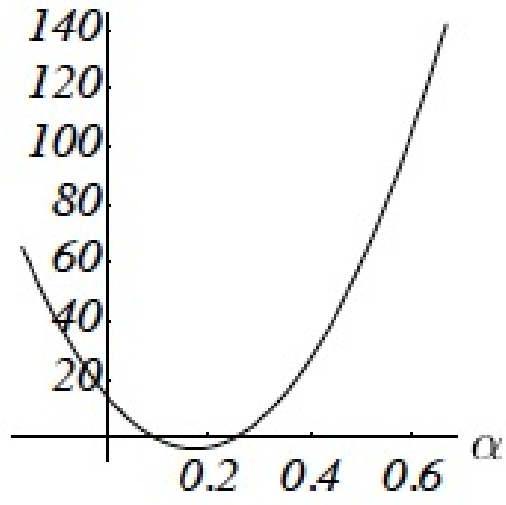
$(b - Ax^1)^T\rho_o = 0$

$\left[(b - A(x^0 + \alpha\rho_0)\right]^T\rho_0 = 0$

$(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

$\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

$$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

$df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

$= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

$\rho_1^T\rho_0 = 0$

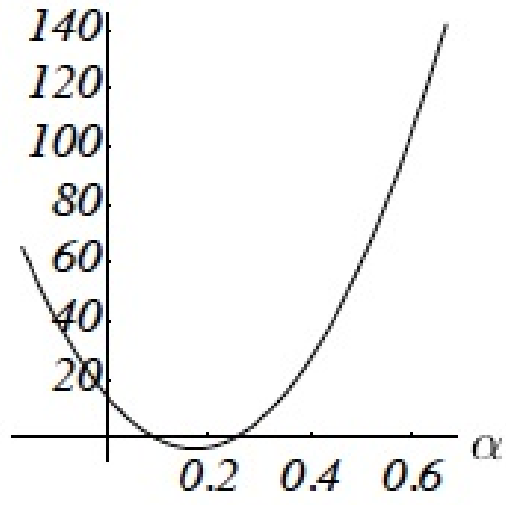
$(b - Ax^1)^T\rho_o = 0$

$\left[(b - A(x^0 + \alpha\rho_0)\right]^T\rho_0 = 0$

$(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

$\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

$$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

$df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

$= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

$\rho_1^T\rho_0 = 0$

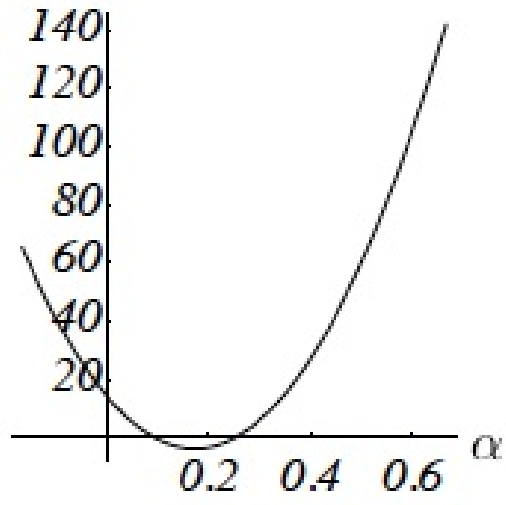
$(b - Ax^1)^T\rho_o = 0$

$\left[(b - A(x^0 + \alpha\rho_0)\right]^T\rho_0 = 0$

$(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

$\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

$$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

$df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

$= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

$\rho_1^T\rho_0 = 0$

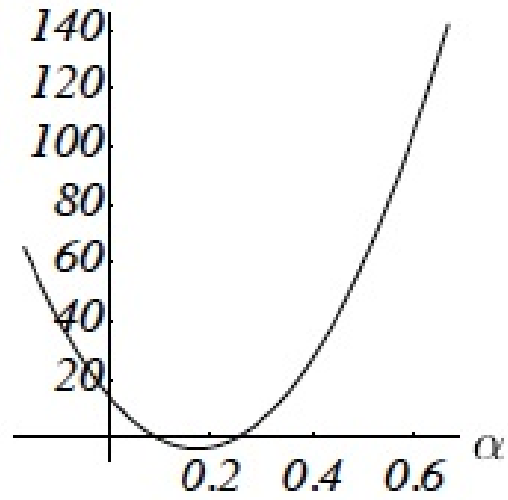
$(b - Ax^1)^T\rho_o = 0$

$[(b - A(x^0 + \alpha\rho_0)]^T\rho_0 = 0$

$(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

$\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

$$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

$df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

$= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

$\rho_1^T\rho_0 = 0$

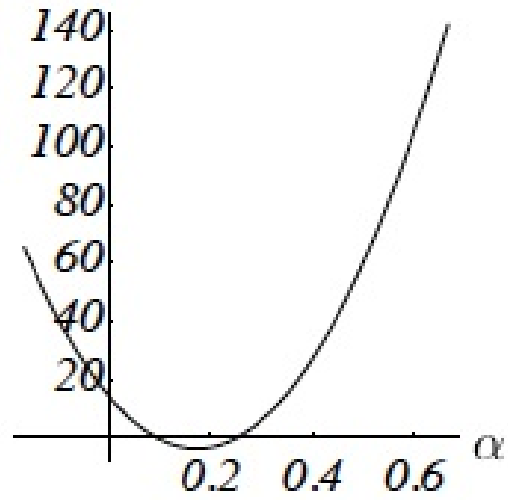
$(b - Ax^1)^T\rho_o = 0$

$[(b - A(x^0 + \alpha\rho_0)]^T\rho_0 = 0$

$(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

$\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

$$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

  $df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

  $= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

  $\rho_1^T\rho_0 = 0$

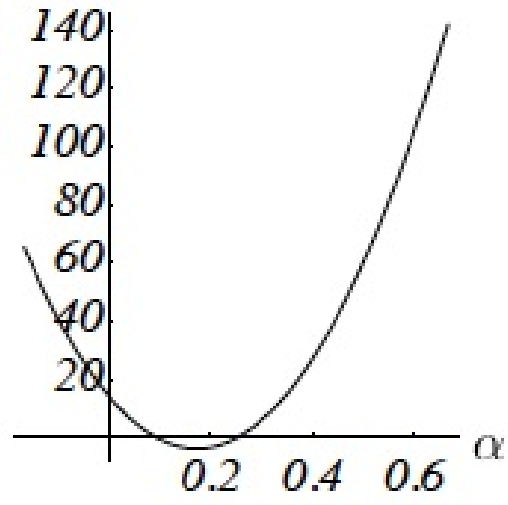
  $(b - Ax^1)^T\rho_o = 0$

  $\left[(b - A(x^0 + \alpha\rho_0)\right]^T\rho_0 = 0$

  $(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

  $\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

  $$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

$df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

$= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

$\rho_1^T\rho_0 = 0$

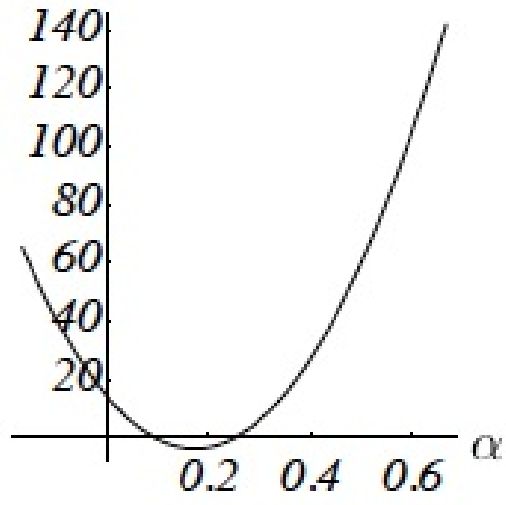
$(b - Ax^1)^T\rho_o = 0$

$\left[(b - A(x^0 + \alpha\rho_0)\right]^T\rho_0 = 0$

$(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

$\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

$$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

$df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

$= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

$\rho_1^T\rho_0 = 0$

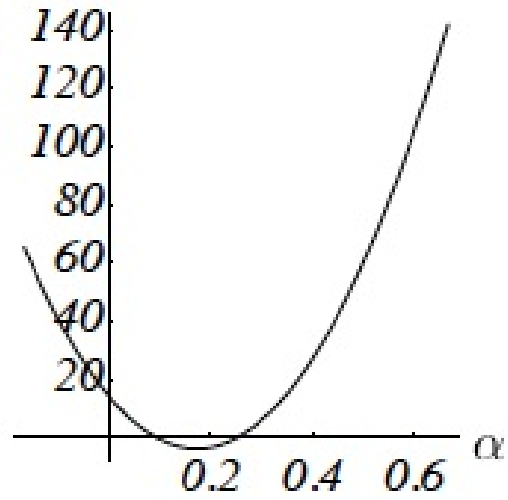
$(b - Ax^1)^T\rho_o = 0$

$[(b - A(x^0 + \alpha\rho_0)]^T\rho_0 = 0$

$(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

$\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

$$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

$df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

$= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

$\rho_1^T\rho_0 = 0$

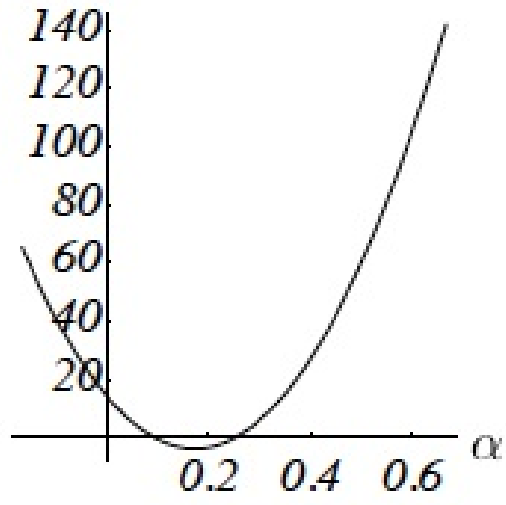
$(b - Ax^1)^T\rho_o = 0$

$\left[(b - A(x^0 + \alpha\rho_0)\right]^T\rho_0 = 0$

$(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

$\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

$$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

$df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

$= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

$\rho_1^T\rho_0 = 0$

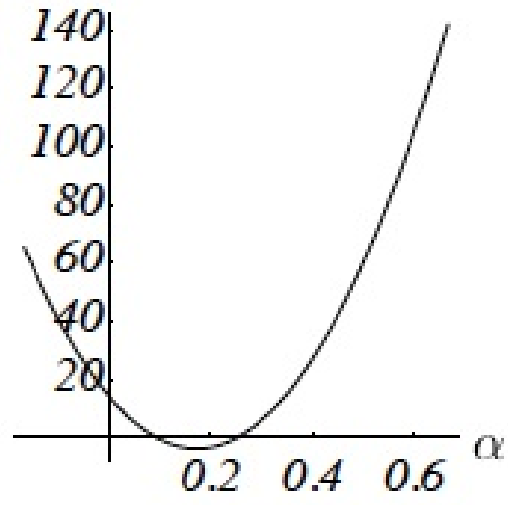
$(b - Ax^1)^T\rho_o = 0$

$\left[(b - A(x^0 + \alpha\rho_0)\right]^T\rho_0 = 0$

$(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

$\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

$$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

$df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

$= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

$\rho_1^T\rho_0 = 0$

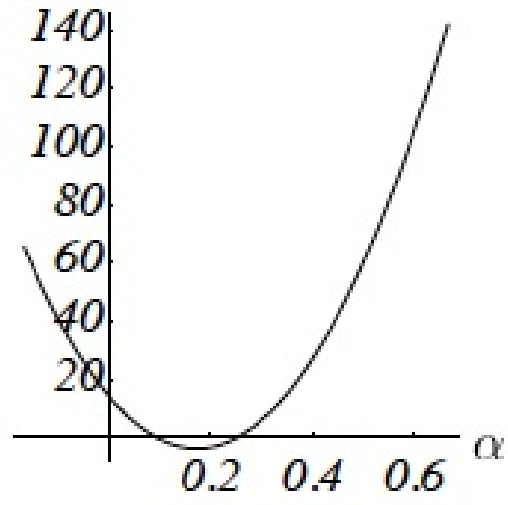
$(b - Ax^1)^T\rho_o = 0$

$\left[(b - A(x^0 + \alpha\rho_0)\right]^T\rho_0 = 0$

$(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

$\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

$$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$



Steepest Descent

- Iterate is $x^{p+1} = x^p + \alpha\rho_p$
- Select α such that $f(x)$ reaches its minimum along the chosen direction

- minimize $df(x^1)/d\alpha$

$df(x^1)/d\alpha = df(x^1)/dx \quad dx^1/d\alpha =$

$= f'(x^1)^T\rho_0 = -\rho_1^T\rho_0 = 0$

$\rho_1^T\rho_0 = 0$

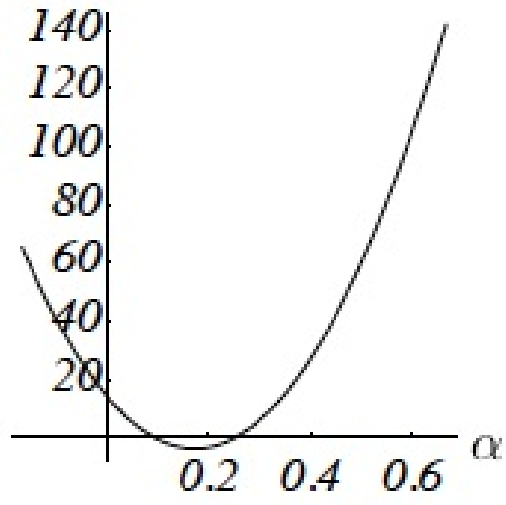
$(b - Ax^1)^T\rho_o = 0$

$\left[(b - A(x^0 + \alpha\rho_0)\right]^T\rho_0 = 0$

$(b - Ax^0)^T\rho_0 = \alpha(A\rho_0)^T\rho_0$

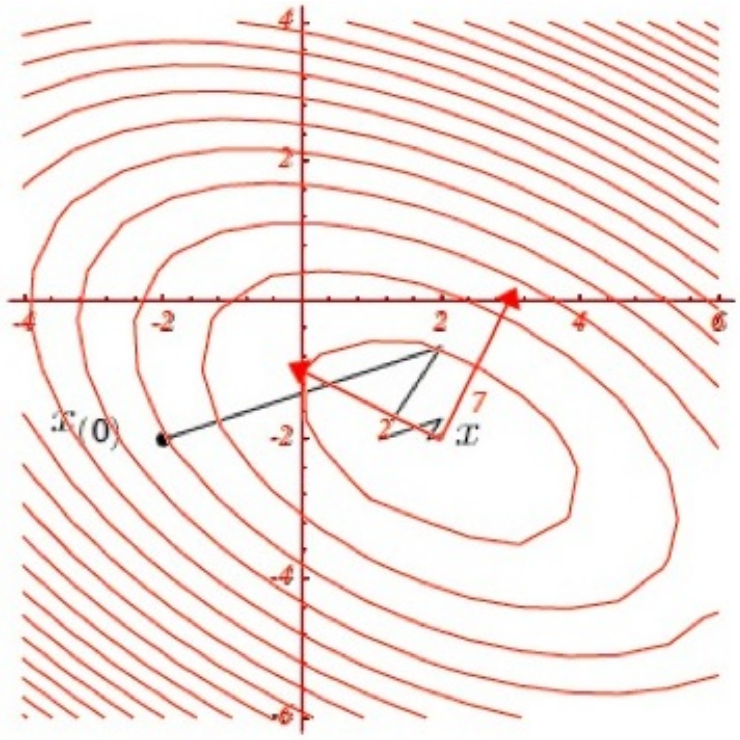
$\rho_0^T\rho_0 = \alpha\rho_0^T A\rho_0$

$$\alpha = \frac{\rho_0^T\rho_0}{\rho_0^T A\rho_0}$$

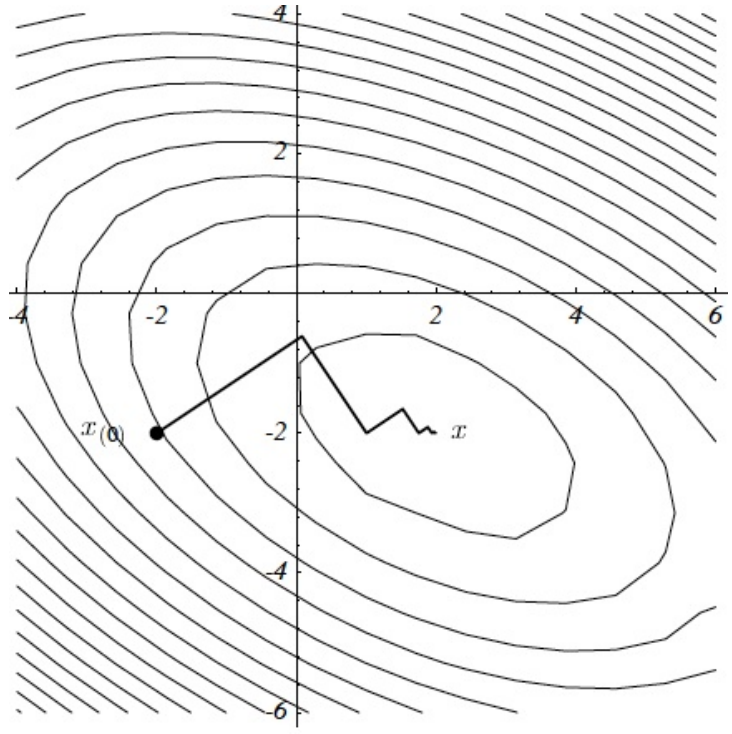


Steepest Descent

- Final iterate is $x^{p+1} = x^p + \alpha_p \rho_p$
- How does the convergence compare to the Jacobi iterate?



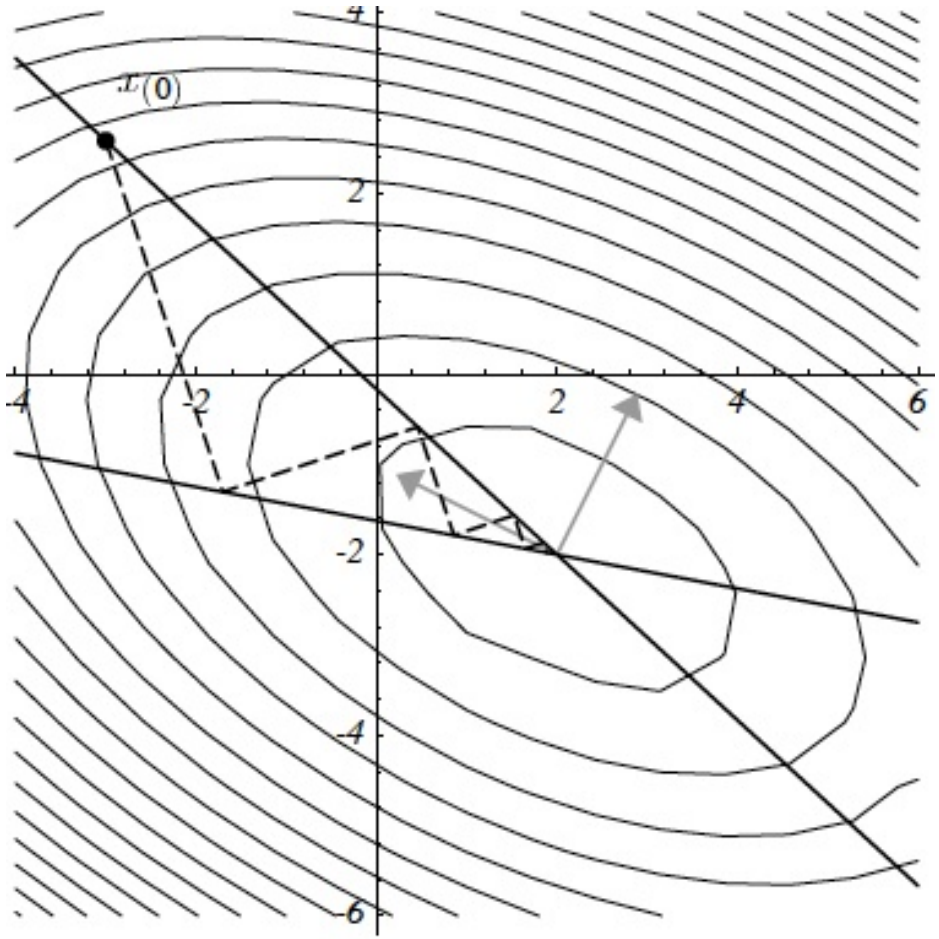
Jacobi



Steepest Descent

Generalized Descent Methods

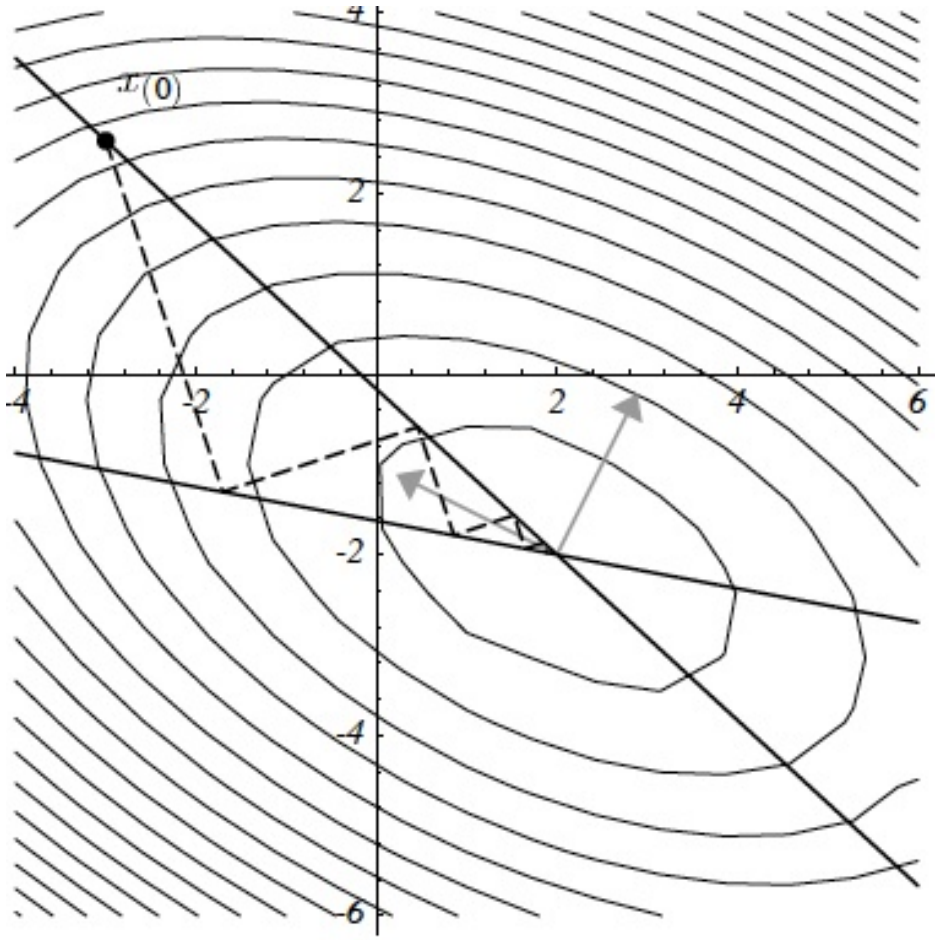
A pathological case for the steepest descent



The directions are uncorrelated!

Generalized Descent Methods

A pathological case for the steepest descent



The directions are uncorrelated!

Conjugate Gradient